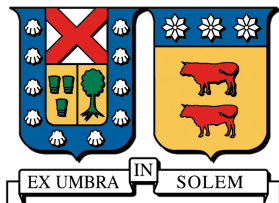


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



**“DESARROLLO DE PLATAFORMA DE
SEÑALIZACIÓN DIGITAL PARA
DISTRIBUCIÓN DE CONTENIDO
MULTIMEDIA: UNIDAD DE ADMINISTRACIÓN
DE CONTENIDOS (UAC)”**

SEBASTIÁN ANDRÉS CORNEJO SOTO

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES**

PROFESOR GUÍA: AGUSTÍN GONZÁLEZ

PROFESOR CORREFERENTE: RUDY MALONNEK

JULIO 2025

Material de referencia, su uso no involucra responsabilidad del autor o de la Institución



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: DESARROLLO DE PLATAFORMA DE SEÑALIZACIÓN DIGITAL PARA DISTRIBUCIÓN DE
CONTENIDO MULTIMEDIA: UNIDAD DE ADMINISTRACIÓN DE CONTENIDOS (UAC)

Nombre del candidato(a): Sebastián Andrés Cornejo Soto

Carrera / Grado: Ingeniería Civil Electrónica

Campus: Casa Central Valparaíso Departamento: Departamento de Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Agustín González Valenzuela, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

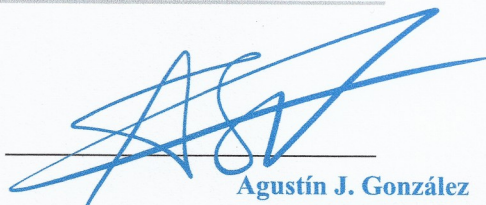
Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 13 / 03 / 2026

Firma: _____


Agustín J. González

Estudiante o Candidato(a):

Fecha: 13/03/2026

Firma: _____

Sebastián

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

Desde lo más recóndito de mi estima, manifiesto mi más profundo y solemne reconocimiento hacia aquellos que, con su sapiencia y cercanía, se han hecho notar en torno a esta obra.

En particular, elevo mis más excelsas muestras de gratitud a los ilustres profesores Agustín González y Rudy Malonnek, cuyas enseñanzas y discernimientos se erigieron como cimientos en la edificación de este trabajo.

Asimismo, consagro mi gratitud a mis consanguíneos y allegados, cuyo abnegado aliento contribuyó a esta obra, para arribar a todo puerto con el estandarte de la perseverancia en alto.

Dedicado a
mi madre, padre y hermano,
por sus alegrías, desafíos y momentos únicos.

**Desarrollo de Plataforma de Señalización Digital para Distribución de Contenido
Multimedia: Unidad de Administración de Contenidos (UAC)**

Sebastián Andrés Cornejo Soto

Memoria para Optar al Título de Ingeniero Civil Electrónico

Mención en Computadores

Profesor guía: Agustín González

Julio-2025

Resumen

La señalización digital es un paradigma que implementa soporte digital para desplegar información. Las entidades que cuentan con equipos, como televisores, y desean aprovecharlos para posicionar estratégicamente sus mensajes, suelen recurrir a soluciones ofrecidas por empresas especializadas en gestión de contenidos digitales.

Un caso particular es el de la Universidad Técnica Federico Santa María, que actualmente depende de un proveedor externo para implementar su señalización digital. Si bien esta solución es funcional, presenta limitaciones significativas, como la falta de personalización y la pérdida de soberanía sobre sus propios activos digitales. Esta dependencia de un servicio tercerizado está presente pese a que la implementación interna de una plataforma propia es una alternativa factible y estratégica para brindar autonomía a la universidad.

La solución propuesta desarrolla un sistema que ofrece una plataforma basada en la web, permitiendo a operarios del área de comunicaciones programar y crear proyectos de señalización digital a través de una interfaz intuitiva, construida con tecnologías como JavaScript, HTML y FFmpeg. Un usuario administrador tiene control centralizado sobre las cuentas de usuario y la configuración general del servicio. Además de la plataforma, se contempla alguna integración con dispositivos digitales finales, como televisores.

Palabras clave: Señalización digital, CMS, Plataforma web, Gestión de usuarios, FFmpeg, HTML, JavaScript.

Development of a Digital Signage Platform for Multimedia Content Distribution:

Content Management Unit (UAC)

Sebastián Andrés Cornejo Soto

Final Project Report towards the fulfillment of the Ingeniero Civil Electrónico

Minor in Computation degree (6-year program)

Advisor Professor: Agustín González

July 2025

Abstract

Digital signage is a paradigm that implements digital support to display information. Entities that have equipment such as televisions and wish to strategically position their messages often turn to solutions offered by companies specialized in digital content management.

A particular case is that of Universidad Técnica Federico Santa María, which currently relies on an external provider to implement its digital signage. Although this solution is functional, it presents significant limitations, such as the lack of customization and the loss of sovereignty over its own digital assets. This dependence on outsourced services persists even though internal implementation of a proprietary platform is a feasible and strategic alternative to provide autonomy to the university.

The proposed solution develops a system that offers a web-based platform, allowing communications department operators to schedule and create digital signage projects through an intuitive interface, built with technologies such as JavaScript, HTML, and FFmpeg. An administrator user has centralized control over user accounts and general service configuration. In addition to the platform, integration with final digital devices, such as televisions, is also considered.

Keywords: Digital signage, CMS, Web platform, User management, FFmpeg, HTML, JavaScript.

Glosario

- **API:** Application Programming Interface.
- **CMS:** Content Management System.
- **CSS:** Cascading Style Sheets.
- **GPIO:** General Purpose Input/Output.
- **HDMI:** High-Definition Multimedia Interface.
- **HTML5:** HyperText Markup Language version 5.
- **HTTP:** HyperText Transfer Protocol.
- **IoT:** Internet of Things.
- **JSON:** JavaScript Object Notation.
- **LED:** Light-Emitting Diode.
- **MIME:** Multipurpose Internet Mail Extensions.
- **OLED:** Organic Light-Emitting Diode.
- **PHP:** Hypertext Preprocessor.
- **RDB:** Relational Database.
- **SD:** Secure Digital.
- **SSH:** Secure Shell.
- **SoC:** System on a Chip.
- **UAC:** Unidad de Administración de Contenido.
- **UTFSM:** Universidad Técnica Federico Santa María.
- **VNC:** Virtual Network Computing.

Índice General

1. INTRODUCCIÓN	1
1.1. Contexto Actual	1
1.2. Descripción del Problema	2
1.3. Objetivos	3
2. TECNOLOGÍAS DE GESTIÓN DE CONTENIDO PARA SEÑALIZACIÓN DIGITAL	4
2.1. Trabajos de otros autores	4
2.2. Señalización digital profesional	8
2.3. Herramientas y tecnologías asociadas.	9
3. DISEÑO DE LA UNIDAD DE ADMINISTRACIÓN	10
3.1. Definición de los requisitos	10
3.2. Análisis del Problema	11
3.3. Arquitectura de la solución	12
3.3.1. Sistema A: Plataforma de Gestión de Contenidos	13
3.3.2. Sistema B: Mecanismo de Despliegue en Tiempo Real	14
3.3.3. Entorno y Consideraciones	15
4. IMPLEMENTACIÓN DE LA UNIDAD DE ADMINISTRACIÓN	18
4.1. Implementación del entorno de trabajo	18
4.2. Implementación de la UAC	21
4.3. Implementación del Mecanismo de Despliegue	31
4.4. Implementación de una utilidad de administración	34
5. RESULTADOS	38
6. CONCLUSIONES Y TRABAJO FUTURO	43
7. REFERENCIAS	45

Índice de Figuras

1.1. Diagrama del esquema actual de funcionamiento	2
1.2. Diagrama del esquema propuesto de funcionamiento	3
2.1. Prototipo del sistema de bajo costo [1]	5
2.2. Diagrama de la arquitectura [1]	6
2.3. Diagrama de la arquitectura evolucionada [2]	7
3.1. Esquema conceptual con la disposición de los sistemas A y B	11
3.2. Modelos de datos de usuarios y monitores	12
4.1. Estructura de directorios del proyecto	19
4.2. Contraseña, límite de almacenamiento y monitores permitidos de cada usuario .	20
4.3. Posición y tipo del "widget"	20
4.4. Página de "login"	22
4.5. Página de "retry login"	22
4.6. Página principal con el "dashboard", en la sección de perfil	23
4.7. Página principal en la sección de contenido	25
4.8. Página principal en la sección de contenido, con ventana de exploración	25
4.9. Página principal en la sección de proyecto, con algunas zonas	27
4.10. Página principal en la sección de proyecto, con ventana emergente	27
4.11. Página principal en la sección de agenda	28
4.12. Página principal en la sección de contenido, seleccionado fecha	29
4.13. Página principal en la sección de contenido, seleccionado "widget"	29
4.14. Pseudocódigo para los módulos de gestión de usuarios y contenido	30
4.15. Pseudocódigo para los módulos de proyecto y agenda	31
4.16. Esquema lógico del servicio levantado por el Sistema B	33
4.17. Ejemplo de la salida del comando uac-util con la opción de ayuda	36
5.1. Página principal, en la sección de contenido	38
5.2. Página principal, en la sección de proyecto	39
5.3. Página principal, en la sección de agenda	39
5.4. Página levantada del Sistema B, para la reproducción asignada a un monitor . .	40
5.5. Una sección de contenido con contenido multimedia elaborado	40
5.6. Una sección de proyecto, con zonas elaboradas (Parte superior)	41
5.7. Una sección de proyecto, con zonas elaboradas (Parte inferior)	41
5.8. Una sección de proyecto, con zonas elaboradas en el lienzo	42
5.9. Un cuadro del video final elaborado, agendado y desplegado por el Sistema B .	42

1 INTRODUCCIÓN

La capacidad de digitalización de la comunicación ha cambiado la manera en que se administra la información, introduciendo herramientas que permiten su organización y presentación al público. La señalización digital surge como uno de estos métodos, empleados para gestionar contenido multimedia digital.

Este trabajo aborda el desafío que enfrenta la Universidad Técnica Federico Santa María (UTFSM) al depender de un proveedor externo para su sistema de señalización, lo que implica limitaciones sobre personalización, control de los activos digitales y escalabilidad. A través del desarrollo de una plataforma web propia, este trabajo explora una alternativa que permitiría a la universidad administrar sus recursos comunicacionales digitales de manera interna y flexible.

Desde un punto de vista global, las instituciones han adoptado sistemas de señalización digital basados en gestores de contenido “Content Management System” (CMS) de ayuda a la hora de administrar pantallas distribuidas en sus instalaciones. Soluciones comerciales como Xibo [3], Mixxer.tv [5] o Scala [4], suelen ser genéricas, costosas y poco flexibles como para adaptarse a los requerimientos específicos de un cliente. Chile no es la excepción, y un caso de especial interés son las universidades, como la UTFSM, cuyo servicio actual está tercerizado a una empresa externa.

1.1 Contexto Actual

Actualmente la solución externa utilizada en la UTFSM opera mediante un sistema compuesto por una plataforma web, servidores administrados por la empresa proveedora y un mecanismo de despliegue que facilita la reproducción de los contenidos en los televisores.

Los operarios del área de comunicaciones acceden a la plataforma web autenticándose para luego gestionar proyectos de contenido digital, mediante un flujo de trabajo que guía al usuario desde la selección de contenido multimedia, elaboración de un proyecto y establecimiento de su programación. La información necesaria es procesada en los servidores del proveedor, desde donde se distribuye hacia el mecanismo de despliegue, responsable de emitir el contenido.

El mecanismo de despliegue opera mediante una infraestructura que conecta el servidor con televisores encargados de la reproducción. Estos dispositivos incluyen ordenadores configurados por la empresa proveedora, vinculados a las pantallas mediante una interfaz “High-Definition Multimedia Interface” (HDMI). Una vez que los servidores están comunicados con los ordenadores, estos últimos realizan la reproducción conforme al esquema definido.

La interacción entre la plataforma web, sus usuarios, la infraestructura de servidores y el mecanismo de despliegue puede representarse gráficamente en el diagrama de la Figura 1.1 ilustrando la relación entre sus elementos. Esta estructura define el funcionamiento del sistema de señalización digital actual en la UTFSM y proporciona una base sobre la cual se puede desarrollar una plataforma alternativa.

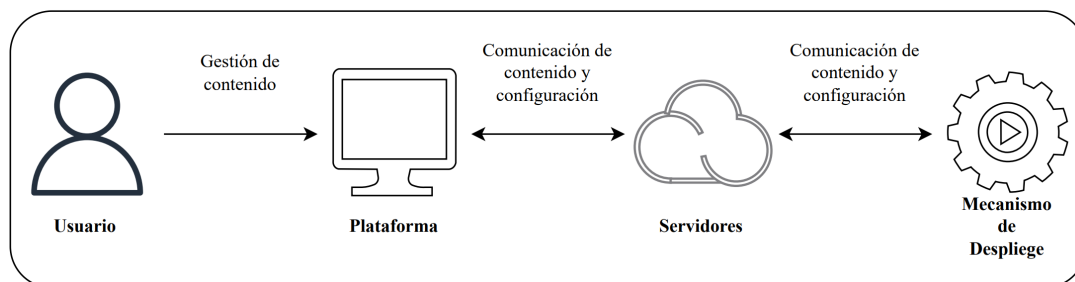


Figura 1.1: Diagrama del esquema actual de funcionamiento

1.2 Descripción del Problema

En la Universidad Técnica Federico Santa María, el sistema de señalización digital actual depende de un proveedor externo, lo que implica que tanto la administración del contenido como su almacenamiento y distribución dependen de una plataforma ajena a la universidad. Si bien esta solución ha permitido mantener un funcionamiento estable, su naturaleza externa introduce ciertas condiciones que afectan aspectos clave de autonomía, integración y costos operativos.

La infraestructura utilizada actualmente se basa en servidores gestionados por la empresa proveedora, donde se almacenan y procesan los contenidos antes de ser enviados a los dispositivos de reproducción. Esto significa que la universidad no tiene control directo sobre sus datos, lo que limita la capacidad de modificar o acceder al sistema sin depender de terceros. Además, la plataforma utilizada presenta restricciones en cuanto a personalización, ya que su diseño no contempla posibles integraciones específicas con otros sistemas internos de la UTFSM. Cualquier adaptación o mejora requiere costos adicionales, lo que reduce la flexibilidad del servicio y dificulta su evolución conforme a nuevas necesidades. A esto se suman los pagos recurrentes por licencias y soporte técnico, lo que representa una inversión continua que podría ser reemplazada por una alternativa gestionada internamente.

Para abordar esta situación, se propone el desarrollo de una plataforma web propia, que permita a los operarios del área de comunicaciones continuar administrando contenido digital sin depender de servicios externos. Esta nueva plataforma ofrecerá las funcionalidades esenciales de un sistema de señalización, pero con mejoras orientadas a la flexibilidad y la autonomía.

También se contempla la migración de la infraestructura de almacenamiento y distribución hacia servidores internos. Como parte de esta transición, se incorpora un mecanismo de despliegue alternativo, basado en el uso de navegadores convencionales para desplegar contenido mediante transmisión en tiempo real. Esta solución permitirá a los dispositivos visualizar el material sin requerir el equipo específico del proveedor.

Elaborar esta solución constituye el problema global, y se ilustra el diagrama del esquema de dicho funcionamiento en la Figura 1.2 donde cabe precisar que el foco de este trabajo será principalmente el desarrollo de la plataforma basada en la web.

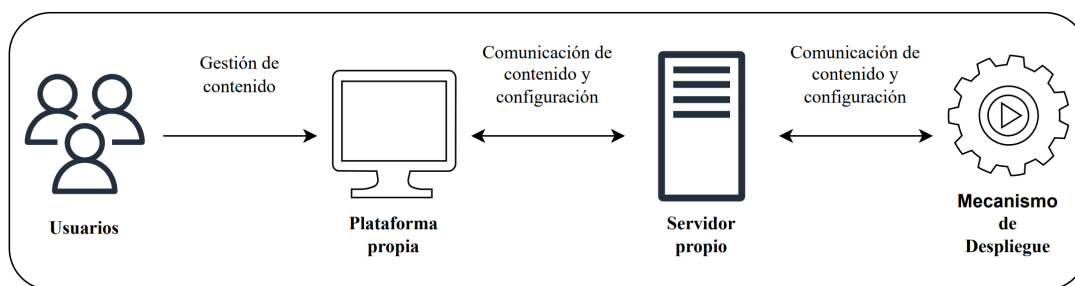


Figura 1.2: Diagrama del esquema propuesto de funcionamiento

1.3 Objetivos

La meta es ofrecer alternativas viables, que perfectamente podrían implementarse en la UTFSM, a servicios por los que normalmente se contrata a terceros. En concreto, para el sistema de señalización digital de la universidad, ofrecer una solución autónoma y flexible, percibida por los operarios del área de comunicaciones como una solución alternativa al sistema actual.

Este trabajo se centra principalmente en desarrollar una alternativa a la plataforma web, para la cual se plantean los siguientes objetivos a alcanzar.

- Desarrollar un sistema de administración de contenido basado en la web de un sistema de señalización digital, el cual permite a usuarios crear proyectos con opciones para ver, agregar y eliminar contenido multimedia.
- Implementar un sistema de autenticación, almacenamiento y permisos mediante cuentas propias para gestionar proyectos independientes por usuario.
- Interpretar proyectos de un usuario para definir su visualización en dispositivos asignados, junto a crear y conceder cuentas (con dispositivos asociados) por un usuario administrador.

2 TECNOLOGÍAS DE GESTIÓN DE CONTENIDO PARA SEÑALIZACIÓN DIGITAL

La señalización digital ha evolucionado desde pantallas estáticas hasta sistemas dinámicos adaptables a diversos entornos. Su desarrollo se ha impulsado con tecnologías como realidad aumentada y computación en la nube, permitiendo ajustar el contenido al ambiente operativo.

Los sistemas de gestión de contenido (CMS) han revolucionado la administración de estas redes, centralizando la creación, edición y distribución del material digital. Ofrecen herramientas para automatización, segmentación y control remoto, facilitando la personalización a gran escala.

La tecnología abarca tanto el software como el hardware para garantizar un rendimiento óptimo en diversas aplicaciones. En el ámbito del hardware, dispositivos como televisores o pantallas destacan por su calidad, resolución y conectividad, aspectos esenciales para ofrecer una experiencia visual envolvente y eficaz. Por otro lado, el software desempeña un papel fundamental en la gestión y optimización de contenidos. Plataformas como Scala [4] o Mixxer.tv [5] permiten administrar redes de pantallas, programar contenido y analizar audiencias, facilitando la personalización y automatización de la información. Asimismo, las soluciones de código abierto, como lo es Xibo [3], democratizan el acceso a herramientas avanzadas, brindando una alternativa flexible y accesible para diferentes tipos de usuarios y sus necesidades.

La señalización digital es una herramienta más en comunicación, integrando innovación en la forma de ofrecer los mensajes. Su evolución continúa, impulsada por la convergencia de soluciones cada vez más accesibles.

2.1 Trabajos de otros autores

Entre las soluciones más relevantes figura la obra realizada por Kazuhiro Mishima, Takeshi Sakurada y Yoichi Hagiwara, titulada Cost Effective Digital Signage System using Low-Cost Information Device [1] de la Universidad de Agricultura y Tecnología de Tokio, que presenta un prototipo económico y flexible de sistema de reproducción y gestión de contenido digital. Su diseño se basa en un computador de placa única, específicamente una Raspberry Pi 3B+ [13], elegida por su bajo costo y compatibilidad con sistemas operativos basados en Linux, lo que permite alta flexibilidad en el desarrollo. El sistema de reproducción implementado en la Raspberry Pi se muestra en la Figura 2.1 dentro de la caja transparente.

Este hardware cuenta con múltiples pines “General Purpose Input/Output” (GPIO) utilizados para equipar la interfaz serial, además de una ranura para tarjeta “Secure Digital” (SD) particionada para el sistema operativo y el contenido multimedia. Por una cotización de 70 dólares para la fecha en la que se realizó el trabajo.



Figura 2.1: Prototipo del sistema de bajo costo [1]

Para compensar las limitaciones de procesamiento local de la Raspberry Pi, se incorpora un servidor de soporte multimedia que se encarga de preparar los archivos más exigentes antes de su envío al dispositivo destino. Este proceso se realiza mediante DirectVNC [9], un cliente “Virtual Network Computing” (VNC) optimizado para Linux, que permite mantener la eficiencia operativa en contextos de reproducción remota. Si bien esta solución reduce la carga de procesamiento en el dispositivo, requiere una conexión de red estable para asegurar una experiencia fluida.

Este enfoque responde a las restricciones propias del hardware utilizado, que no está diseñado para gestionar de forma autónoma contenidos de alta complejidad. En este sentido, el uso de VNC permite delegar parte del esfuerzo computacional, priorizando la ligereza en el dispositivo y trasladando ciertas exigencias al entorno de red. Esta relación entre consumo de recursos locales y demanda de ancho de banda constituye un aspecto relevante a considerar en desarrollos futuros.

Adicionalmente, se implementó una plataforma de administración basada en la web, que permite la gestión remota del sistema mediante funciones como reinicio, control de reproducción y monitoreo del estado operativo. Todo este esquema se ilustra en la Figura 2.2 donde el administrador carga contenido y gestiona el servicio general a través del manejador de sesión, el que a su vez, transfiere el contenido y su configuración al servidor de soporte de contenido en la Raspberry Pi.

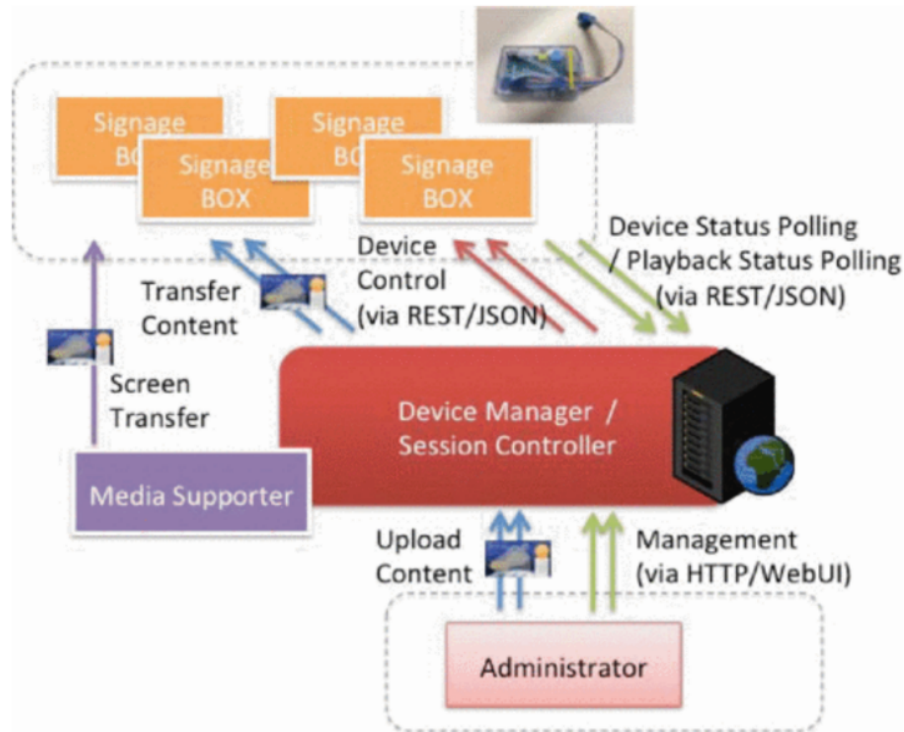


Figura 2.2: Diagrama de la arquitectura [1]

Una evolución del trabajo anterior fue desarrollada por los mismos autores, Kazuhiro Mishima, Takeshi Sakurada y Yoichi Hagiwara, en su publicación Low-Cost Managed Digital Signage System with Signage Service [2] también de la Universidad de Agricultura y Tecnología de Tokio. Esta nueva versión introduce mejoras significativas respecto al sistema original, con especial énfasis en la escalabilidad de la gestión de contenido y el reforzamiento de la seguridad.

Uno de los cambios más significativos es la incorporación de una base de datos “Relational DataBase” (RDB) para almacenar información clave sobre los dispositivos de señalización, la autenticación y el contenido multimedia. Este módulo busca centralizar la administración y permitir una fácil integración con múltiples equipos de forma simultánea.

Además, se incorpora comunicación remota mediante “Secure Shell” (SSH) para el acceso seguro a la Raspberry Pi, complementado con la herramienta rsync [10], que permite la sincronización eficiente de archivos entre múltiples dispositivos a través de la red. Esta integración no solo facilita la transferencia de contenido, sino que habilita funciones avanzadas de administración como asignación remota de archivos multimedia, monitoreo de la reproducción, reinicio, apagado y ajuste de parámetros operativos. Estas capacidades amplían significativamente el control del sistema y la flexibilidad para el administrador, consolidando una solución más robusta y escalable. La arquitectura resultante, representada en la Figura 2.3 mantiene la estructura del prototipo original, pero con mejoras clave: soporte para RDB y una capa de comunicación segura por SSH entre los dispositivos y el manejador de dispositivos.

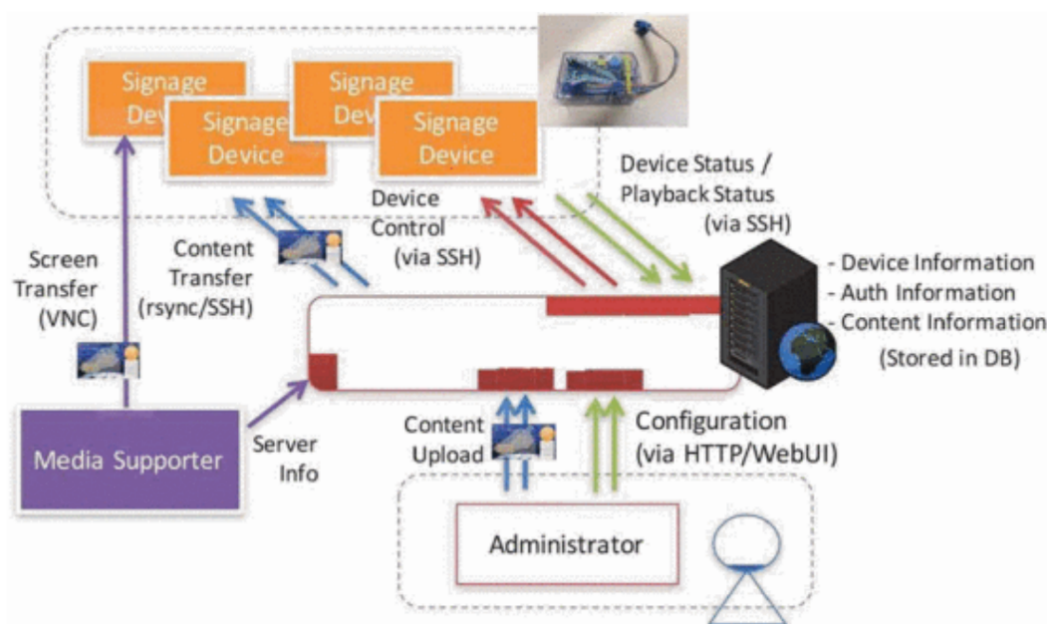


Figura 2.3: Diagrama de la arquitectura evolucionada [2]

En términos de viabilidad, este proyecto es totalmente aplicable en escenarios donde se requiera una solución económica, flexible y escalable para señalización digital, especialmente en instituciones educativas y corporativas que buscan independencia de tecnologías propietarias.

2.2 Señalización digital profesional

La señalización digital combina hardware, software y protocolos para distribuir contenido de forma eficiente. Existen soluciones propietarias como Scala, orientada a redes de gran escala mediante una arquitectura centralizada con Scala Enterprise, que utiliza reproductores dedicados Scala Players [7] o dispositivos compatibles con Windows IoT y Android. Permite operación híbrida en la nube o local. Su integración con sistemas externos se realiza a través de "Application Programming Interface" (API) REST/XML, y se comunica tanto con bases de datos como con un CMS. En infraestructura, ofrece despliegues en servidores y versiones alojadas en AWS o Azure. Su "backend" está desarrollado en Java, con soporte para contenido interactivo en HTML5 y JavaScript, gestionado mediante eventos en tiempo real y sensores "Internet of Things" (IoT). La creación de contenido se realiza en Scala Designer [8], editor basado en plantillas adaptables a diferentes contextos, como kioscos.

En contraste, Xibo es una plataforma de código abierto con arquitectura basada en un CMS auto-hospedado desarrollado en PHP y MySQL. Es compatible con clientes ligeros en Raspberry Pi, Android o Windows, y permite operar en entornos locales o en la nube mediante hosting externos. En contraste con soluciones propietarias como Scala, esta alternativa de código abierto representa un ahorro en licencias y mayor libertad en la implementación del sistema. El "backend" está basado en PHP y la interfaz en HTML5/CSS/JavaScript, que permite el desarrollo de "widgets" interactivos y la programación de contenido mediante listas dinámicas, aunque con integración de APIs externas. Aunque su diseño es más técnico y menos intuitivo que Scala, ofrece una alternativa flexible para entornos que priorizan autonomía, escalabilidad y bajo costo.

Mixxer.tv es el proveedor actual de señalización digital de la UTFSM, operando con una plataforma basada en la web que permite la gestión centralizada de pantallas en cada campus y sede. Su sistema sincroniza contenido en tiempo real y optimiza la comunicación institucional mediante televisores con interfaz HDMI. Su enfoque práctico lo convierte en una opción eficaz para entornos educativos y corporativos.

2.3 Herramientas y tecnologías asociadas.

En el ámbito de la señalización digital, algunas herramientas y tecnologías son fundamentales para garantizar una gestión eficiente y una experiencia visual impactante. Entre los reproductores multimedia más comunes destacan aquellos basados en "System on Chip" (SoC), como los televisores Smart TV con navegadores web integrados o soporte de aplicaciones personalizadas. Estos dispositivos permiten recibir contenido directamente desde una plataforma web sin requerir hardware adicional, facilitando implementaciones más simples y accesibles. Marcas como LG [11] ofrecen soluciones comerciales con SoC capaces de operar en entornos empresariales. En contraste, también es común el uso de ordenadores discretos de bajo costo como la Raspberry Pi, los cuales pueden conectarse mediante HDMI a cualquier pantalla convencional. Este enfoque requiere que el dispositivo tenga instalado un navegador web capaz de recibir contenido en tiempo real, permitiendo una configuración más flexible y controlada desde el servidor. La diferencia entre ambos enfoques radica en la arquitectura, mientras los televisores con SoC integran el procesamiento dentro del mismo equipo visual, los sistemas basados en ordenadores externos separan la visualización del procesamiento, ofreciendo mayor capacidad de personalización pero exigiendo una gestión más detallada para la correcta comunicación.

Las pantallas comerciales para señalización digital se clasifican según luminosidad, entorno e interacción. Las "Light Emitting Diode" (LED) de alto brillo son ideales para exteriores, mientras las "Organic Light-Emitting Diode" (OLED) ofrecen colorimetría para interiores. Las táctiles (capacitivas/infrarrojas) permiten interacción en kioscos, y las certificadas con distintos niveles de protección resisten condiciones extremas de temperatura y lluvia. Además, para el diseño y despliegue interactivo, se utilizan herramientas, como los lenguajes HTML5, JavaScript y CSS, que permiten el desarrollo de "widgets" interactivos y un entorno de trabajo moderno y constantemente actualizado. Finalmente, la conectividad, mediante protocolos como WebSocket o "HyperText Transfer Protocol" (HTTP), permite la comunicación remota y la sincronización de datos en tiempo real, y el soporte de Wi-Fi o 5G ofrece la posibilidad de implementaciones inalámbricas.

3 DISEÑO DE LA UNIDAD DE ADMINISTRACIÓN

El sistema de señalización digital actualmente utilizado por la Universidad Técnica Federico Santa María (UTFSM) opera bajo el modelo descrito en la introducción, representado en la Figura 1.1. Si bien dicho enfoque cumple con los requisitos funcionales básicos, presenta limitaciones sustanciales en términos de soberanía digital, flexibilidad de adaptación y costos operativos a largo plazo.

Para abordar estas deficiencias, se propone una transición hacia un modelo autónomo, representado en la Figura 1.2, que permita a la universidad gestionar internamente todo su ecosistema de señalización digital, reduciendo la dependencia de soluciones externas y fortaleciendo su infraestructura institucional.

3.1 Definición de los requisitos

Con el fin de asegurar la validez técnica del diseño propuesto y facilitar su evaluación, se definen una serie de requisitos funcionales medibles, agrupados según los objetivos principales establecidos en la introducción:

- **R1.1:** La plataforma web debe proporcionar a los usuarios autenticados la capacidad de incorporar, eliminar y visualizar contenido multimedia de manera eficiente y segura.
- **R1.2:** Los proyectos contruidos por cada usuario, utilizando su propio contenido multimedia, deben ser procesados automáticamente en segundo plano, generando un archivo de vídeo final con almacenamiento persistente.
- **R2.1:** La plataforma debe ofrecer una interfaz de autenticación por credenciales que permita el acceso seguro mediante un módulo de inicio de sesión.
- **R2.2:** La plataforma debe gestionar archivos multimedia subidos por cada usuario, controlando sus propiedades técnicas y asegurando acceso exclusivo para su propietario.
- **R2.3:** Desde el módulo de agenda, los usuarios deben poder asignar fechas y horarios específicos a los monitores autorizados para la transmisión programada de contenidos.
- **R3.1:** Los monitores deben estar configurados para recibir contenido transmitido en tiempo real a través del navegador web, sin requerir intervención manual posterior a su configuración inicial.

- **R3.2:** El sistema debe permitir al administrador gestionar usuarios, incluyendo su creación, edición y eliminación, con posibilidad de definir permisos específicos como monitores asignados, límite de almacenamiento y credenciales de acceso.

3.2 Análisis del Problema

Tomando como referencia el modelo autónomo ilustrado en la Figura 1.2, se plantea un esquema de solución compuesto por dos sistemas complementarios que cooperan entre sí y constituyen la base de una propuesta integral. Estos son el Sistema A y el Sistema B.

- **Sistema A:** Corresponde al módulo de gestión de contenido y usuarios. Su función principal es permitir la creación, organización y programación de proyectos multimedia, asegurando que cada usuario disponga de un espacio lógico propio para administrar sus recursos.
- **Sistema B:** Representa el módulo de despliegue en tiempo real. Se encarga de interpretar las programaciones definidas en el Sistema A y de transmitir los contenidos hacia los monitores asociados, manteniendo la sincronización y la actualización dinámica.
- **Solución propuesta:** Se espera que el Sistema A materialice su lógica mediante una plataforma web que se ejecute y sirva desde servidores locales en la universidad, concebidos como una plataforma propia y un servidor propio en el esquema de solución propuesto en la Figura 3.1. Mientras que el Sistema B se proyecta como una interfaz de visualización en una página web a pantalla completa, capaz de reproducir la programación definida, ya sea usando el navegador integrado del televisor o un ordenador (con navegador) conectado al mismo, actuando así como mecanismo de despliegue en el esquema de solución propuesto en la Figura 3.1.

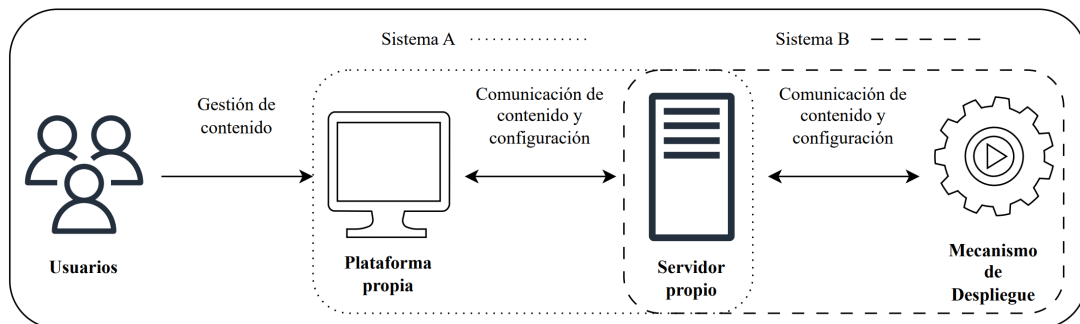


Figura 3.1: Esquema conceptual con la disposición de los sistemas A y B

3.3 Arquitectura de la solución

La arquitectura propuesta para el sistema de señalización digital de la UTFSM se compone de dos subsistemas principales que colaboran para ofrecer una solución integral, eficiente y autónoma. El Sistema A es responsable de la gestión de usuarios y contenidos mediante una plataforma web, mientras que el Sistema B se encarga del despliegue en tiempo real de los contenidos programados hacia los dispositivos finales.

El diseño de la solución se complementa con la definición de los modelos de datos que permiten estructurar y organizar la información gestionada por el sistema. En la Figura 3.2 se presentan los esquemas principales. El EsquemaUsuario concentra los atributos relacionados con la administración de cuentas, permisos y contenidos. El EsquemaMonitor describe la información tanto del proyecto asignado como los parámetros de despliegue en cada dispositivo. Esta representación asegura la coherencia entre la gestión de usuarios y la operación de los monitores, y constituye la base para la correcta interacción entre los subsistemas A y B.

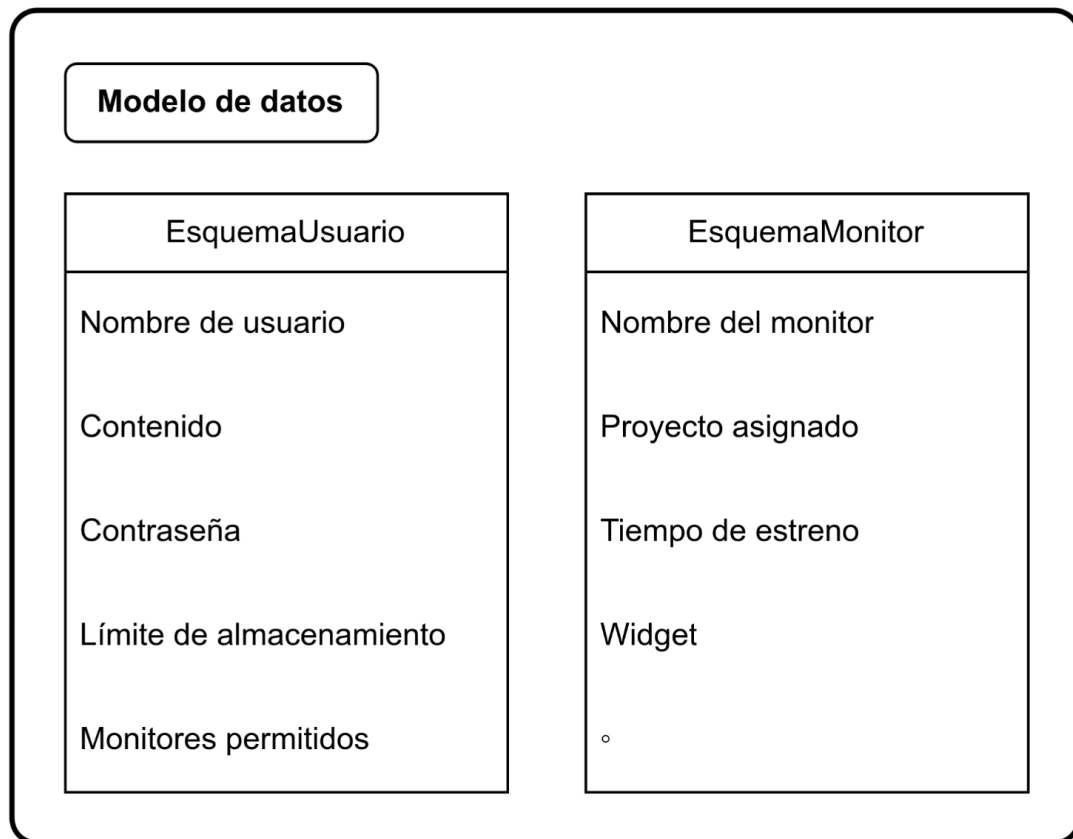


Figura 3.2: Modelos de datos de usuarios y monitores

3.3.1 Sistema A: Plataforma de Gestión de Contenidos

El Sistema A corresponde a la plataforma web encargada de la administración integral de usuarios, contenidos y proyectos de señalización digital. Su diseño se organiza en módulos conceptuales que delimitan responsabilidades específicas y, en conjunto, definen el flujo lógico de interacción del usuario con la plataforma.

Módulos del sistema El diseño contempla cuatro módulos principales, cada uno con funciones claramente diferenciadas que, en conjunto, permiten cubrir las funcionalidades básicas de un sistema de señalización digital:

- **Módulo de gestión de usuarios:** Administra las cuentas de acceso, validando credenciales y controlando los permisos asociados a cada perfil. Además, gestiona la cuota de almacenamiento asignada y la vinculación de monitores autorizados, garantizando un uso seguro y personalizado del sistema.
- **Módulo de contenido:** Permite a cada usuario cargar, visualizar y eliminar archivos multimedia dentro de su espacio personal. Este módulo asegura el aislamiento entre cuentas y aplica validaciones lógicas sobre el tipo y tamaño de los archivos, manteniendo la coherencia de los datos almacenados.
- **Módulo de proyectos:** Elabora composiciones audiovisuales a partir de los contenidos cargados y del esquema definido por el usuario, considerando la duración y la resolución general del proyecto, y estructurando los archivos en una secuencia coherente.
- **Módulo de agenda:** Gestiona la programación temporal de los proyectos, permitiendo asignar fechas y horarios de reproducción en los monitores autorizados.

Flujo de operación El uso de la plataforma sigue una secuencia lógica que refleja la interacción natural del usuario con el Sistema A:

1. El usuario accede a la página web y se autentica.
2. Una vez validado, carga contenido multimedia en su espacio personal.
3. Luego, crea un proyecto utilizando los archivos cargados.
4. Finalmente, programa la reproducción del proyecto en uno de los monitores autorizados.

3.3.2 Sistema B: Mecanismo de Despliegue en Tiempo Real

El Sistema B constituye el componente encargado de materializar las programaciones definidas en el Sistema A, transformándolas en experiencias visuales dinámicas sobre los monitores asignados. Su propósito es garantizar una reproducción precisa, sincronizada y continua de los contenidos multimedia, manteniendo la coherencia del servicio y la autonomía de la institución.

La arquitectura lógica del Sistema B se organiza en dos módulos:

- **Módulo de gestión de monitores:** Responsable de mapear cada monitor físico a un identificador único dentro del sistema, validar la existencia de los directorios correspondientes en mon/ y preparar el entorno de ejecución. Esta etapa asegura que todos los recursos necesarios estén disponibles antes de iniciar el ciclo de operación, previniendo errores en la ejecución posterior.
- **Módulo de reproducción:** Encargado de supervisar de manera continua el estado de cada monitor, verificando si existe un proyecto programado cuyo tiempo de estreno coincida con el tiempo actual del sistema. En caso afirmativo, solicita el proyecto correspondiente y lo reproduce en pantalla completa, incorporando los elementos visuales adicionales definidos (como "widgets"). Si no existe programación activa, el módulo mantiene una interfaz de espera constante, evitando interrupciones o pantallas en blanco.

El flujo de operación del sistema se desarrolla de la siguiente manera:

1. Al iniciar el servicio, el módulo de gestión de monitores asigna un puerto de salida a cada monitor y valida la disponibilidad de sus datos de configuración.
2. El módulo de reproducción supervisa continuamente el estado de cada monitor, verificando si corresponde iniciar la reproducción de un proyecto.
3. Cuando se detecta una programación válida, el proyecto es enviado para su despliegue al monitor correspondiente, incluyendo los "widgets" definidos por el usuario.
4. Si no se encuentra una programación válida, el sistema mantiene visible una interfaz de espera hasta que se habilite una reproducción válida.

Este diseño modular permite una operación robusta y flexible, donde cada monitor funciona como una instancia independiente, facilitando la escalabilidad del sistema y la incorporación de nuevas funcionalidades en el futuro.

3.3.3 Entorno y Consideraciones

Los sistemas A y B interactúan de forma complementaria para conformar una solución integral de señalización digital. Mientras el Sistema A se encarga de la gestión de usuarios, contenidos y programación, el Sistema B se especializa en la reproducción sincronizada de los proyectos definidos.

La coherencia entre ambos subsistemas se asegura mediante los modelos de datos presentados en la Figura 3.2. El `EsquemaUsuario` organiza la información relativa a cada cuenta, incluyendo credenciales, límites de almacenamiento, monitores autorizados y contenidos disponibles. Por su parte, el `EsquemaMonitor` concentra los atributos necesarios para el despliegue, tales como el proyecto asignado, el tiempo de estreno y los elementos adicionales de visualización.

De esta manera, el Sistema A actualiza la información en el `EsquemaMonitor` al momento de programar un proyecto, y el Sistema B accede a dicho esquema para determinar qué contenidos deben reproducirse y en qué momento. Este mecanismo de referencia compartida constituye el vínculo lógico entre la gestión realizada por los usuarios y la ejecución en los dispositivos finales, evitando dependencias externas y garantizando la autonomía de la solución.

Cada sistema contempla interfaces específicas para cumplir sus funciones:

- **Sistema A:**

- Interfaz web para usuarios: Gestión de contenido, proyectos y programación.
- Repositorio común de programación: Espacio lógico donde se registran las configuraciones de despliegue accesibles también por el Sistema B.

- **Sistema B:**

- Interfaz de reproducción: Visualizador web adaptativo para los monitores.
- Repositorio común de programación: Fuente de referencia para interpretar la información definida en el Sistema A.

Además, se incorpora una herramienta exclusiva para el administrador del sistema, orientada a automatizar y agilizar la gestión tanto de usuarios como de monitores. Este módulo opera de forma transversal sobre los sistemas A y B, garantizando la coherencia de los datos de usuarios y monitores.

La implementación de esta solución requiere un conjunto específico de herramientas y tecnologías que abarcan tanto componentes de software como de hardware. A continuación se detallan los elementos necesarios:

- **”Backend” y Procesamiento Multimedia**

- **Node.js:** Plataforma de ejecución para el ”backend”.
- **NPM:** Gestor de módulos de Node.js.
- **JSON:** Principal insumo para la comunión de información web de la plataforma.
- **FFmpeg:** Utilidad para elaborar el proyecto final, compatible con codecs de vídeo H.264 y H.265.

- **”Frontend”**

- **JavaScript / TypeScript:** Lógica de la interfaz y comunicación con el ”backend”.
- **HTML5 / CSS3:** Lenguaje de marcado y estilos para el diseño de la plataforma.
- **Fetch API:** Para la comunicación asíncrona con el ”backend”.
- **WebSockets:** Comunicación en tiempo real entre el servidor y los dispositivos finales.

- **Infraestructura y Requisitos del Servidor**

- **Sistema operativo:** Linux (recomendado) o Windows Server.
- **Procesador:** x64 con al menos 4 núcleos (recomendado).
- **Memoria:** Al menos 4GB de RAM.
- **Almacenamiento:** Al menos 64GB (ajustable según la demanda).
- **Red:** Conexión de red local, con un ancho de banda mínimo de 10 Mbps por dispositivo.

La arquitectura propuesta proporciona ventajas en comparación con soluciones tradicionales, entre ellas:

- Autonomía completa y seguridad robusta gracias a infraestructura universitaria interna con aislamiento y validación centralizada.
- Escalabilidad y bajo mantenimiento mediante estructura modular con directorios simples en /usr o /mon y actualizaciones por subsistema.
- Eficiencia optimizada usando hardware como "Smart TVs", sin necesidad de computadores adicionales.
- Integración desacoplada y basada en eventos a través del sistema de archivos compartido en dónde cambios en un subsistema son detectados automáticamente por el otro permitiendo evolución independiente.
- Extensibilidad futura con integración a sistemas universitarios, más formatos multimedia y caché local para mayor resiliencia.

En conjunto, esta solución no solo aborda los desafíos de dependencia externa detectados en la infraestructura actual, sino que sienta las bases para una plataforma de señalización digital totalmente personalizable y escalable, preparada para acompañar la evolución tecnológica y comunicacional de la UTFSM en los años venideros.

4 IMPLEMENTACIÓN DE LA UNIDAD DE ADMINISTRACIÓN

Tal como se mencionó previamente, la solución propuesta consiste en la integración del Sistema A (plataforma y servidor) y del Sistema B (servidor y mecanismo de despliegue). A partir de este capítulo, se denominará al Sistema A como Unidad de Administración de Contenidos (UAC). Esta unidad constituye el núcleo funcional del presente trabajo, validando su viabilidad tanto como solución operativa como alternativa tecnológica interna para la universidad. En esta sección se detalla la arquitectura de la UAC junto con el Sistema B, los cuales operan en conjunto para ofrecer una solución integral de administración y despliegue de contenido.

4.1 Implementación del entorno de trabajo

La implementación se realizó en el entorno Node.js sobre el sistema operativo Linux, utilizando la herramienta FFmpeg [12] para el procesamiento multimedia. El proyecto se organiza en el directorio memoria/, que contiene los módulos de desarrollo (node_modules/) y la carpeta src/. Esta última concentra los elementos principales, tales como los scripts sa.js y sb.js, las páginas web de la UAC (.html) y las carpetas usr/ y mon/, destinadas al almacenamiento de los datos de usuarios y monitores, respectivamente, como se ilustra en la Figura 4.1.

El script sa.js levanta el servidor de la UAC, sirviendo las páginas de autenticación y el panel principal. Sus funciones abarcan autenticación, permisos, gestión de archivos y programación de proyectos. Las dependencias principales son:

- **express, express-session, body-parser**: servidor y autenticación.
- **Multer, mime-types**: carga y validación de archivos.
- **path, fs, child_process**: manejo de rutas, archivos y ejecución de FFmpeg.

El script sb.js levanta el servidor del Sistema B mediante WebSockets, escaneando periódicamente las carpetas de monitores y transmitiendo el contenido en pantalla completa. Sus dependencias son:

- **express**: servidor de visualización.
- **ws**: comunicación en tiempo real.
- **path, fs, http**: manejo de archivos y fechas de programación.

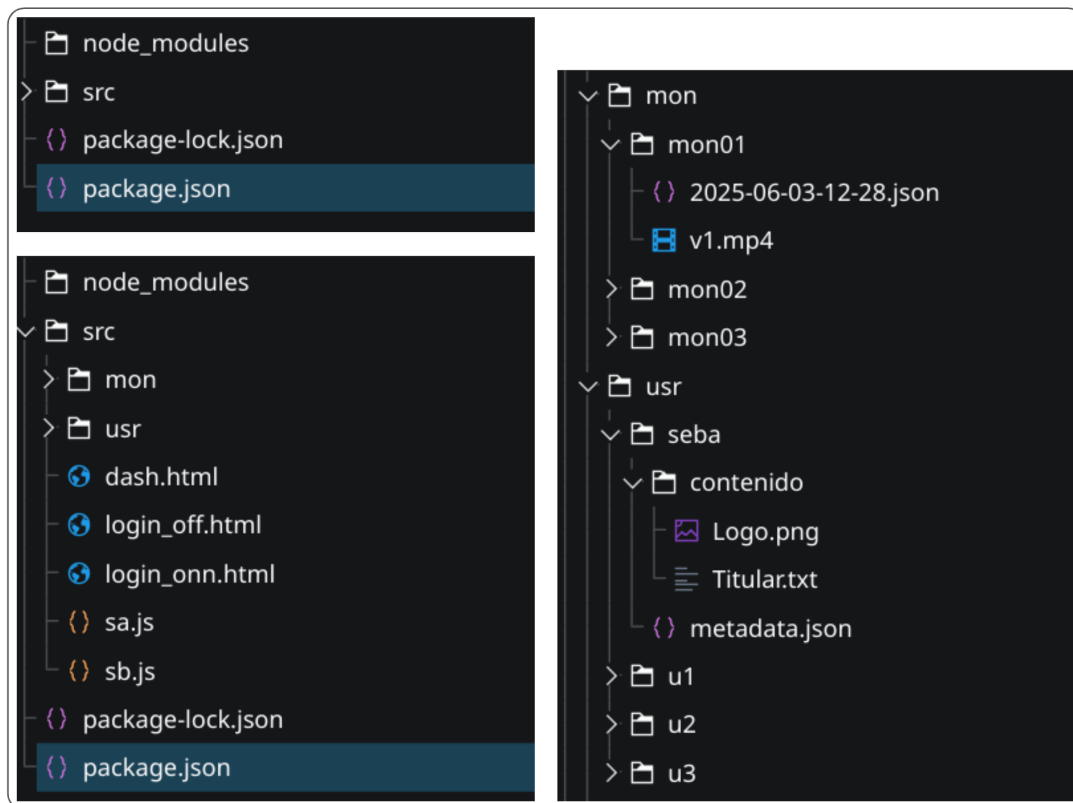


Figura 4.1: Estructura de directorios del proyecto

Dentro de la carpeta `usr/`, cada subcarpeta representa un `EsquemaUsuario`, como se muestra en la Figura 3.2. El nombre de la subcarpeta corresponde al nombre de usuario. En su interior se encuentra un directorio llamado `contenido/`, que materializa el atributo `Contenido`, y un archivo `metadata.json`, donde se registran la `Contraseña`, el `Límite de almacenamiento` y los `Monitores permitidos`. Este archivo, ilustrado en la Figura 4.2, funciona como descriptor de los parámetros del usuario, necesarios para su autenticación y gestión dentro del sistema.

De manera análoga, en la carpeta `mon/`, cada subcarpeta corresponde a un `EsquemaMonitor`, de acuerdo con la Figura 3.2. El nombre de la subcarpeta se asocia directamente al atributo `Nombre del monitor`. En su interior se almacena un archivo de video que constituye el `Proyecto asignado`, acompañado de un archivo en formato JSON cuyo nombre, expresado como `YYYY-MM-DD-HH-mm.json`, define el `Tiempo de estreno`. El contenido de este archivo describe el `Widget` correspondiente, especificando su tipo y su posición en pantalla. Por defecto, el sistema inicia con un archivo base denominado `1970-01-01-00-00.json`, y un ejemplo de esta estructura puede verse en la Figura 4.3.

```

{
  "AUT":{
    "passwd": "1234"
  },
  "ST0":{
    "size": 50
  },
  "ACL":{
    "mon": [1,2,3]
  }
}

```

Figura 4.2: Contraseña, límite de almacenamiento y monitores permitidos de cada usuario

```

{
  "POS": {
    "x": 80,
    "y": 20
  },
  "NAME": {
    "id": "tiempo"
  }
}

```

Figura 4.3: Posición y tipo del "widget"

Esta arquitectura modular permite una administración eficiente, donde las carpetas de usuario y monitor operan de forma independiente pero se integran mediante el sistema compartido de archivos. Como resultado, cada modificación realizada por sa.js en contenidos o programación se refleja automáticamente en la ejecución de sb.js, logrando una operación colaborativa en tiempo real sin necesidad de configuraciones manuales por parte del usuario final.

4.2 Implementación de la UAC

La UAC está compuesta por un servidor web con acceso a datos de usuarios y monitores, y por una interfaz accesible desde el navegador. El servidor atiende las solicitudes, mientras que la interfaz ofrece tres funciones principales. Primero, permite el acceso mediante autenticación por credenciales. Segundo, si la validación falla, se muestra una pantalla de rechazo con opción de reintento. Finalmente, tras autenticarse correctamente, el usuario accede a un panel de control ("dashboard") desde donde gestiona contenidos, proyectos, programaciones y su perfil.

La comunicación entre la interfaz y el servidor se realiza mediante peticiones asincrónicas y dinámicas, lo que permite que las acciones del usuario, como subir archivos, programar contenidos o consultar información, se procesen de inmediato sin interrumpir la navegación. Los cambios se reflejan directamente en la interfaz sin recargar la página. Todas las solicitudes van a rutas específicas o "endpoints" que activan procesos concretos según la operación requerida.

Los principales "endpoints" implementados son los siguientes:

- /generar (POST): Genera un video combinando archivos y aplicando filtros.
- /listarTodo (GET): Lista todos los archivos del usuario.
- /getInfo (GET): Devuelve información del usuario (almacenamiento y permisos).
- /infoM (GET): Muestra la cantidad de monitores disponibles.
- /agendar (POST): Programa la reproducción en un monitor específico.
- /getFilesVid (GET): Lista archivos de video (.mp4) disponibles.
- /getPreviewSimple (GET): Envía un archivo si la extensión es válida.
- /getFiles (GET): Lista los archivos contenidos en la carpeta de contenido.
- /getNombre (GET): Devuelve el nombre de usuario en sesión.
- /upload (POST): Sube archivos con validación de tipo y tamaño.
- /delete (POST): Elimina un archivo del usuario.
- /logout (GET): Cierra la sesión del usuario.

Cada "endpoints" cumple una función específica. Algunos gestionan archivos, otros solici-

tan información sobre usuarios o monitores, y también controlan la programación y creación de proyectos multimedia. De este modo, el servidor responde a cada acción iniciada desde la interfaz.

El usuario accede mediante un navegador convencional y es redirigido a la interfaz de inicio de sesión (login_onn.html, Figura 4.4), donde ingresa sus credenciales. Estas se verifican contra los metadatos almacenados en su carpeta personal. En caso de que los datos sean incorrectos, se muestra la página de "retry login" (login_off.html, Figura 4.5), la cual informa del error y permite volver a intentar.

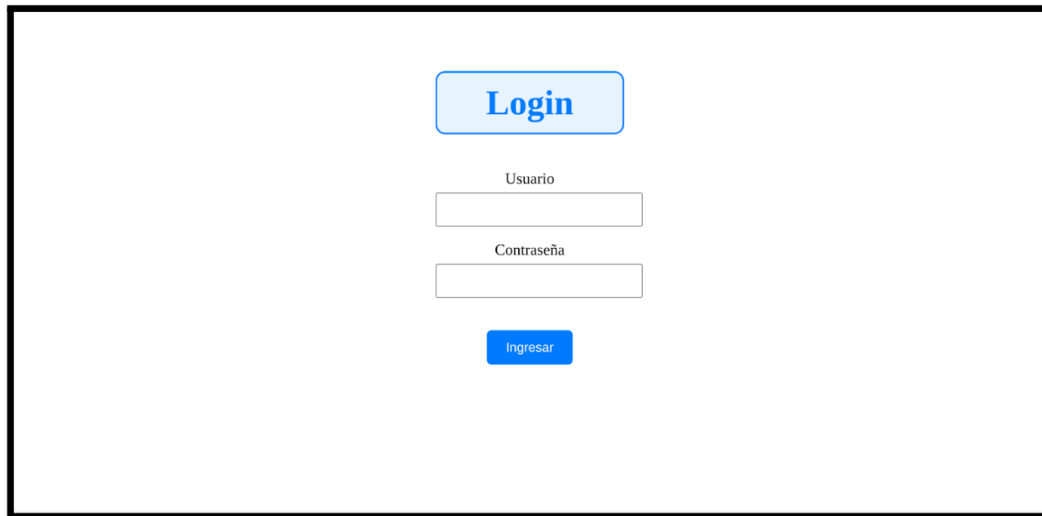
The image shows a login form centered on a white background. At the top is a light blue rounded rectangle with the word "Login" in bold blue text. Below this are two white input fields with thin grey borders. The first field is labeled "Usuario" and the second is labeled "Contraseña". Below the second field is a blue rounded rectangle with the word "Ingresar" in white text.

Figura 4.4: Página de "login"

The image shows a message box centered on a white background. It consists of a light red rounded rectangle with the word "Incorrecto" in bold red text. Below this is a red rounded rectangle with the text "Intentar de nuevo" in white text.

Figura 4.5: Página de "retry login"

Una vez que la autenticación es exitosa, el usuario accede a la página principal, llamada "dashboard" (Figura 4.6). Esta mantiene siempre una barra superior fija con cuatro botones para navegar entre las secciones de contenido, proyecto, agenda y perfil. Bajo esta barra se despliega la sección activa, lo que garantiza una experiencia continua en la que, por defecto, se carga primero el perfil.

La sección inicial es la de perfil y muestra la capacidad de almacenamiento asignada, los monitores autorizados para programación, dos botones para cambiar el tema o cerrar la sesión y un saludo personalizado con el nombre del usuario en el botón que lleva a perfil en la barra de menú, ubicada en la esquina superior derecha. Con esta información básica disponible, el usuario puede comenzar a operar dentro de su espacio personal, mientras que los siguientes "endpoints" gestionan esta sección:

- /getNombre: Invocado al cargar la página principal, para obtener y mostrar el nombre del usuario activo.
- /getInfo: Utilizado para consultar información del usuario activo, como el almacenamiento máximo disponible y los monitores asignados al perfil.
- /logout: Vinculado al botón de cierre de sesión, que finaliza la sesión actual y redirige al usuario al "login".

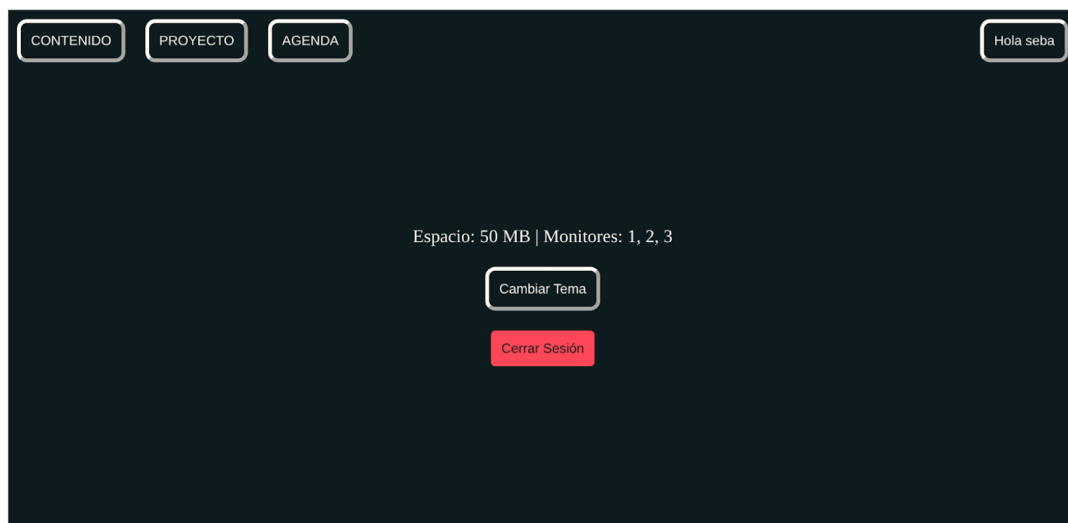


Figura 4.6: Página principal con el "dashboard", en la sección de perfil

En la sección de contenido (figuras 4.7 y 4.8), cargada al presionar el botón de contenido en la barra superior, el usuario administra su contenido multimedia. Las acciones disponibles son:

- Subir archivos multimedia (imágenes, videos, textos) desde el equipo local, siempre que cumplan los criterios de formato y tamaño máximo permitido. Además, dispone de un área de vista previa para revisar archivos compatibles; esta funcionalidad se presenta en el panel derecho.
- Visualizar una lista de los contenidos previamente almacenados. Con la opción de eliminar o ver cada uno. Esto en la parte de la izquierda.

Cada archivo pasa por una validación MIME (permite sólo .png, .jpeg, .mp4 y .txt) y de tamaño (no exceder el límite de almacenamiento definido para el usuario con el nuevo archivo, considerando su total ya almacenado) antes de almacenarse en el directorio del usuario. Esta operación garantiza una experiencia de trabajo segura y aislada, donde cada usuario opera sobre su propio directorio sin interacción externa. Las rutas utilizadas en esta sección son:

- /upload: Vinculado al botón de subida de archivos, realiza validaciones por tipo "Multipurpose Internet Mail Extensions" (MIME) y tamaño antes de almacenar el contenido.
- /delete: Ejecutado al presionar el botón de eliminar, remueve archivos seleccionados del almacenamiento del usuario.
- /getPreviewSimple: Empleado para visualizar una vista rápida de archivos compatibles.
- /listarTodo: Asociado al botón de refresco del listado de contenido, y devuelve todos los archivos disponibles del usuario activo.

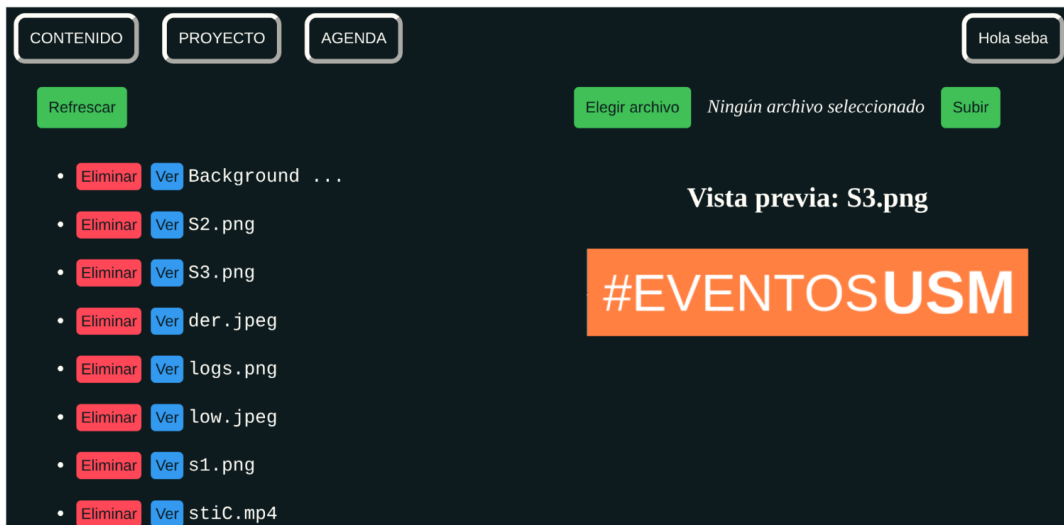


Figura 4.7: Página principal en la sección de contenido

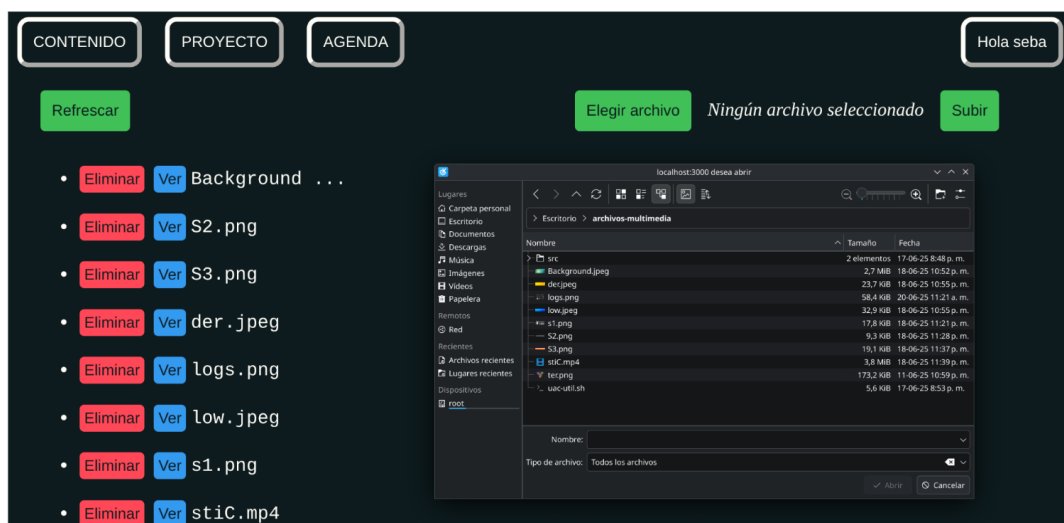


Figura 4.8: Página principal en la sección de contenido, con ventana de exploración

Con archivos disponibles, el usuario accede a la sección de proyecto (figuras 4.9 y 4.10), donde puede construir composiciones de señalización digital, a partir de su material personal. El editor visual permite:

- Seleccionar la duración y resolución del video final como a su vez definir zonas espaciales para la reproducción en la zona de la izquierda.
- Visualizar un lienzo con las zonas que el usuario ubica para facilitar la compresión del

proyecto a elaborar, junto a un cuadro para asignar un nombre al proyecto y un botón de crear. Esto en la parte de la derecha.

- A las zonas se les puede asignar una lista de archivos a reproducir uno tras otro, con la opción de seleccionar el archivo que se desee y el tiempo de aparición de los mismos.

Una vez seleccionados los parámetros de duración y resolución del video final, el usuario puede definir zonas espaciales dentro del área de trabajo, ubicada en el panel izquierdo de la interfaz. Al mismo tiempo, en el panel derecho se presenta un lienzo que refleja visualmente la distribución de dichas zonas. Este entorno de edición incluye además un cuadro para asignar nombre al proyecto y un botón que activa el proceso de generación.

Cada zona definida permite la asignación de múltiples archivos multimedia dispuestos en secuencia. El usuario puede seleccionar el archivo deseado y establecer, en segundos, el instante en que comienza y el instante en que finaliza su reproducción dentro de la composición. Esto otorga gran flexibilidad para diseñar presentaciones audiovisuales complejas.

Una vez completada la estructura visual, la interfaz genera un archivo de instrucciones en formato JSON que se envía al "backend" del sistema. El script `sa.js` interpreta estas instrucciones y emplea FFmpeg para procesar los contenidos y generar un archivo de video (.mp4) que representa el proyecto final. Este archivo se almacena automáticamente en el directorio contenido/ del usuario, quedando disponible para futuras programaciones. Los "endpoints" utilizados en esta sección son:

- `/getFiles`: Invocado desde el editor para recuperar el conjunto de archivos multimedia que el usuario tiene disponibles, permitiendo asignarlos a zonas específicas dentro del lienzo de trabajo.
- `/generar`: Asociado al botón "crear proyecto", recibe instrucciones del diseño visual elaborado por el usuario y desencadena el proceso de renderizado mediante FFmpeg.

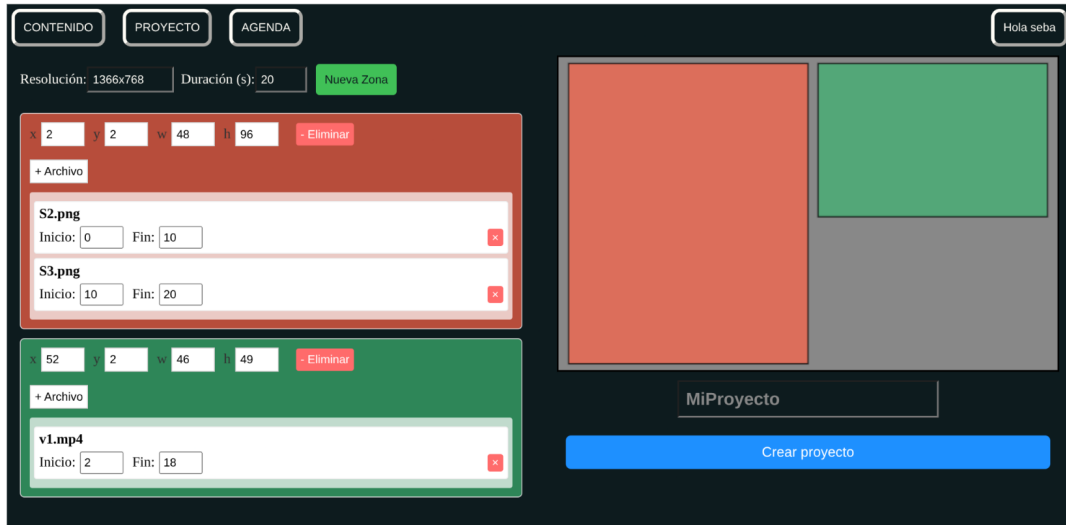


Figura 4.9: Página principal en la sección de proyecto, con algunas zonas

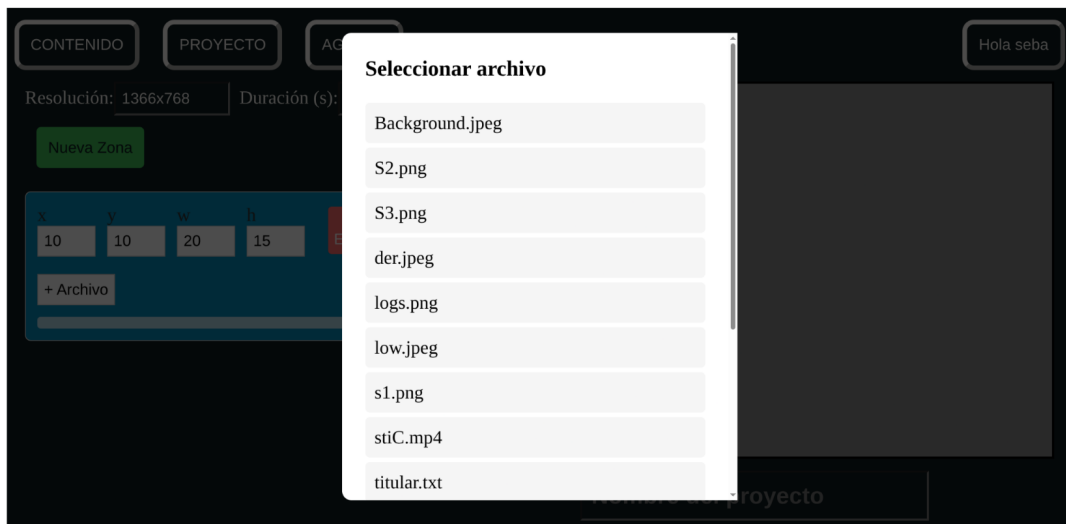


Figura 4.10: Página principal en la sección de proyecto, con ventana emergente

Una vez generado el proyecto final, el usuario puede continuar hacia la sección de agenda (figuras 4.11, 4.12 y 4.13) para definir su programación. Desde este módulo, el operario tiene la posibilidad de:

- Seleccionar el monitor objetivo.
- Elegir la fecha y hora exacta de reproducción (estreno).

- Incorporar un "widget", como un reloj, de forma opcional y ubicarlo mediante coordenadas porcentuales en pantalla.

Una vez configurada la agenda, el sistema genera los archivos necesarios dentro de la carpeta correspondiente al monitor seleccionado. Esta información es utilizada por el Sistema B, que ejecuta un ciclo de revisión cada 2 segundos sobre las carpetas de monitores (mon/), activando la reproducción del contenido cuando llega el momento programado.

El sistema realiza validaciones internas para evitar conflictos de programación, sobrescritura de datos o asignación indebida de monitores, asegurando que los contenidos se desplieguen correctamente y en el horario definido. Los "endpoints" involucrados en esta etapa son:

- /getFilesVid: Invocado para listar solo los archivos .mp4 disponibles en el directorio del usuario, facilitando la selección del proyecto a agendar.
- /agendar: Vinculado al botón de agendar, envía al "backend" la configuración de monitor, fecha de estreno y "widget" seleccionado.
- /infoM: Ejecutado al cargar esta sección, recupera la lista de monitores asignados a cada usuario. El sistema valida que el monitor elegido se encuentre dentro de este conjunto antes de aceptar la programación.

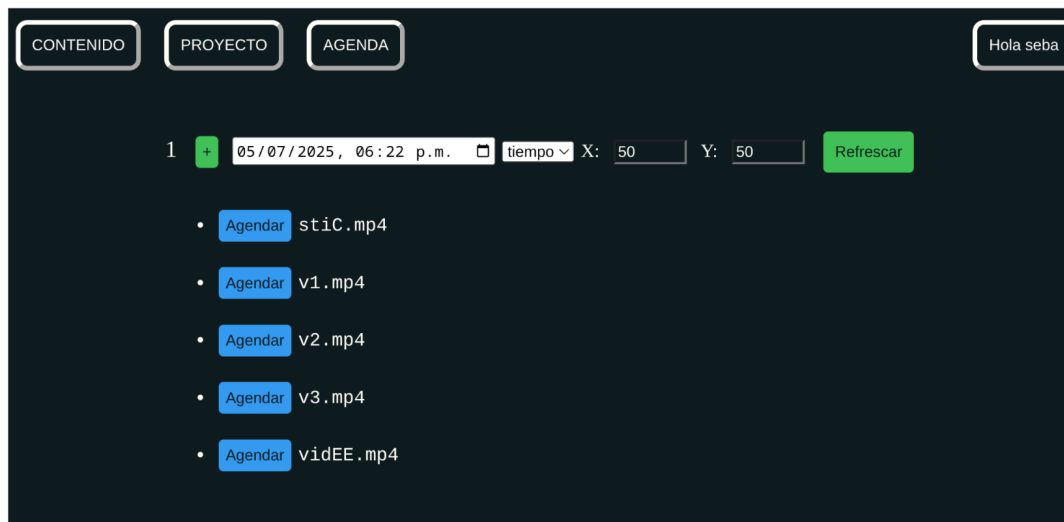


Figura 4.11: Página principal en la sección de agenda

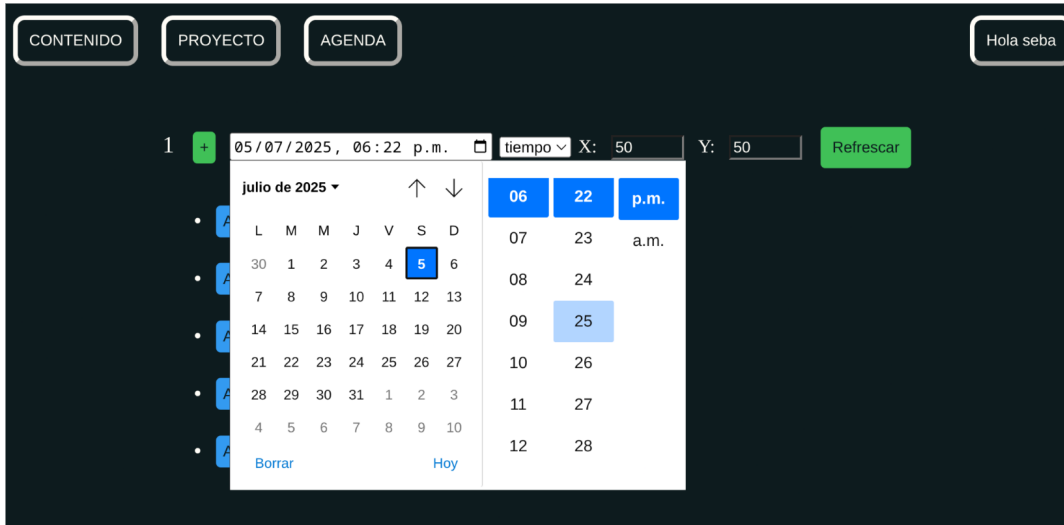


Figura 4.12: Página principal en la sección de contenido, seleccionado fecha

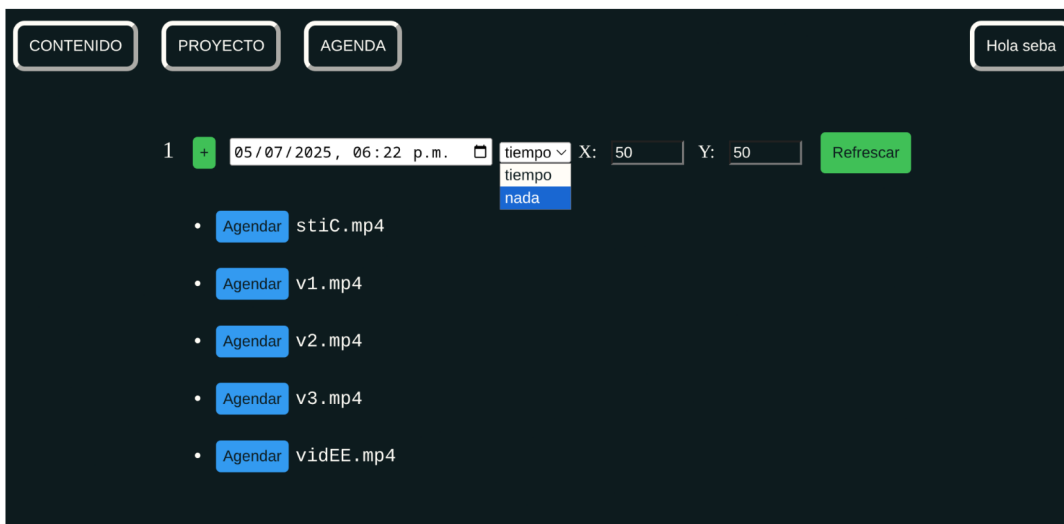


Figura 4.13: Página principal en la sección de contenido, seleccionado "widget"

De este modo, los módulos definidos en la etapa de diseño se implementan a través de los *endpoints* establecidos en el servidor. El **módulo de gestión de usuarios** se concreta mediante las rutas `/getNombre`, `/getInfo` y `/logout`, que permiten autenticar al usuario, recuperar sus datos básicos de perfil y finalizar la sesión de manera segura. El **módulo de contenido** se implementa a través de los "endpoints" `/upload`, `/delete`, `/getPreviewSimple` y `/listarTodo`, los cuales habilitan la carga, validación, previsualización y administración de los archivos multimedia en el espacio personal de cada usuario. A su vez, el **módulo de proyectos** se articula

mediante las rutas /getFiles y /generar, que posibilitan la recuperación de los archivos disponibles y la construcción de composiciones audiovisuales procesadas con FFmpeg, generando un producto final en formato de video. Finalmente, el **módulo de agenda** se implementa con los "endpoints" /getFilesVid, /agendar e /infoM, encargados de listar los proyectos disponibles, registrar la programación en monitores autorizados y validar la coherencia de la asignación.

En las figuras 4.14 y 4.15 se presenta el pseudocódigo correspondiente a cada uno de estos "endpoints", lo que ilustra de forma clara su lógica interna.

```
2
3 // Módulo de Gestión de Usuarios
4
5 function getNombre(request):
6     usuario = request.session.usuario
7     if usuario existe:
8         return usuario.nombre
9     else:
10        return error("Sesión no válida")
11
12 function getInfo(request):
13     usuario = request.session.usuario
14     if usuario existe:
15         return {
16             "almacenamientoMax": usuario.limite,
17             "monitoresAsignados": usuario.monitores
18         }
19     else:
20        return error("Sesión no válida")
21
22 function logout(request):
23     destruirSesion(request.session)
24     redirigir("login_onn.html")
25
26 // Módulo de Contenido
27
28 function upload(request, archivo):
29     if validarMIME(archivo) AND validarTamaño(archivo, usuario):
30         guardarArchivo(usuario.directorio, archivo)
31         return success("Archivo subido")
32     else:
33        return error("Archivo inválido o supera límite")
34
35 function delete(request, archivo):
36     if archivo perteneceA usuario:
37         eliminarArchivo(usuario.directorio, archivo)
38         return success("Archivo eliminado")
39     else:
40        return error("Acceso denegado")
41
42 function getPreviewSimple(request, archivo):
43     if archivo esCompatibleConPreview:
44         return generarVistaPrevia(archivo)
45     else:
46        return error("Formato no soportado")
47
48 function listarTodo(request):
49     return listarArchivos(usuario.directorio)
```

Figura 4.14: Pseudocódigo para los módulos de gestión de usuarios y contenido

```

2
3 // Módulo de Proyectos
4
5 function getFiles(request):
6     return listarArchivos(usuario.directorio)
7
8 function generar(request, jsonInstrucciones):
9     instrucciones = parseJSON(jsonInstrucciones)
10    if validarInstrucciones(instrucciones):
11        videoFinal = FFmpeg.render(instrucciones)
12        guardarArchivo(usuario.directorio + "/contenido", videoFinal)
13        return success("Proyecto generado", videoFinal)
14    else:
15        return error("Instrucciones inválidas")
16
17 // Módulo de Agenda
18
19 function getFilesVid(request):
20     return listarArchivos(usuario.directorio, filtro=".mp4")
21
22 function agendar(request, video, monitor, fecha, widget):
23     if monitor in usuario.monitores:
24         if validarConflictos(monitor, fecha, widget):
25             crearArchivoProgramacion(video, monitor, fecha, widget)
26             return success("Proyecto agendado")
27         else:
28             return error("Conflicto de programación")
29     else:
30         return error("Monitor no autorizado")
31
32 function infoM(request):
33     return usuario.monitores
34

```

Figura 4.15: Pseudocódigo para los módulos de proyecto y agenda

Este flujo de trabajo, que articula las secciones de contenido, proyecto y agenda, fue concebido para dar respuesta a las necesidades de gestión identificadas en la universidad, ofreciendo las funcionalidades básicas de un sistema de señalización digital sólido, escalable y administrado íntegramente por personal interno. La propuesta garantiza al usuario control completo sobre sus activos digitales, desde la carga inicial hasta su publicación final, cumpliendo con el requisito de operar sin depender de servicios externos de terceros.

4.3 Implementación del Mecanismo de Despliegue

El Sistema B actúa como componente ejecutor dentro del ecosistema de señalización digital, siendo el encargado de transformar las programaciones definidas por la UAC en experiencias visuales dinámicas. A diferencia del Sistema A, este módulo se especializa en garantizar una

reproducción precisa, confiable y continua, integrando tecnologías como WebSockets y Express.js en una solución eficiente y robusta.

En su núcleo, el sistema implementa un servicio de verificación periódica, calibrado para ofrecer una alta capacidad de respuesta con un bajo consumo de recursos. El servidor despliega múltiples páginas web, cada una asociada a un monitor específico, y ejecuta dos mecanismos principales:

- **Configuración inicial:** el servidor asigna a cada monitor un puerto dedicado (por ejemplo, 3001, 3002, etc.) y valida la existencia del directorio `mon/` junto con la programación correspondiente. Esta verificación inicial permite prevenir errores y asegura la disponibilidad de los componentes desde el arranque del sistema.
- **Verificación temporal:** el servidor, en su funcionamiento regular, realiza cada dos segundos una comparación entre la hora actual y los archivos JSON de programación, cuyo formato de nombre es `YYYY-MM-DD-HH-mm.json`, comenzando por defecto con `1970-01-01-00-00.json`. Cuando la hora actual coincide con la hora de estreno especificada, se transmite a través de WebSocket el nombre del video correspondiente junto con un valor booleano verdadero. Simultáneamente, se posicionan los "widgets" en pantalla según las coordenadas definidas y se oculta la interfaz de espera. En caso contrario, se envía el valor booleano falso y se mantiene visible la interfaz de espera.

Cada dos segundos, el sistema transmite únicamente un valor booleano y una cadena con el nombre del video, lo que permite optimizar el uso del ancho de banda. El navegador, mediante su sistema estándar de red, ensambla el Document Object Model utilizando esta información, y solicita el video únicamente cuando detecta un cambio en el nombre, evitando así descargas innecesarias. Para que el navegador pueda realizar esta solicitud de forma autónoma, es necesario que el servidor exponga públicamente el contenido de la carpeta `mon/`, lo cual se configura durante el mecanismo de configuración inicial. En consecuencia, dicho mecanismo implementa el módulo de configuración inicial, mientras que el mecanismo de verificación temporal implementa el módulo de reproducción, ambos módulos definidos en la etapa de diseño.

Un ejemplo de "widget" es el reloj, que actualiza la fecha y la hora cada segundo. Este se posiciona en pantalla mediante coordenadas porcentuales, sin interferir con el contenido principal.

Con el fin de garantizar una experiencia de visualización óptima, la página de reproducción incorpora un modo de pantalla completa, el cual se activa mediante un solo "clic". Este modo

elimina los elementos de navegación del navegador y maximiza el área útil de visualización. De esta manera, el administrador puede acceder desde el navegador del televisor, ingresar la URL con el puerto correspondiente, activar el modo de pantalla completa y dejar la página reproduciendo la programación de forma inmersiva y envolvente para los espectadores. Los videos se escalan automáticamente manteniendo su proporción original, y el código HTML/CSS está optimizado para ejecutarse en dispositivos de bajo rendimiento, como la Raspberry Pi. La Figura 4.16 ilustra la lógica de operación del servicio.

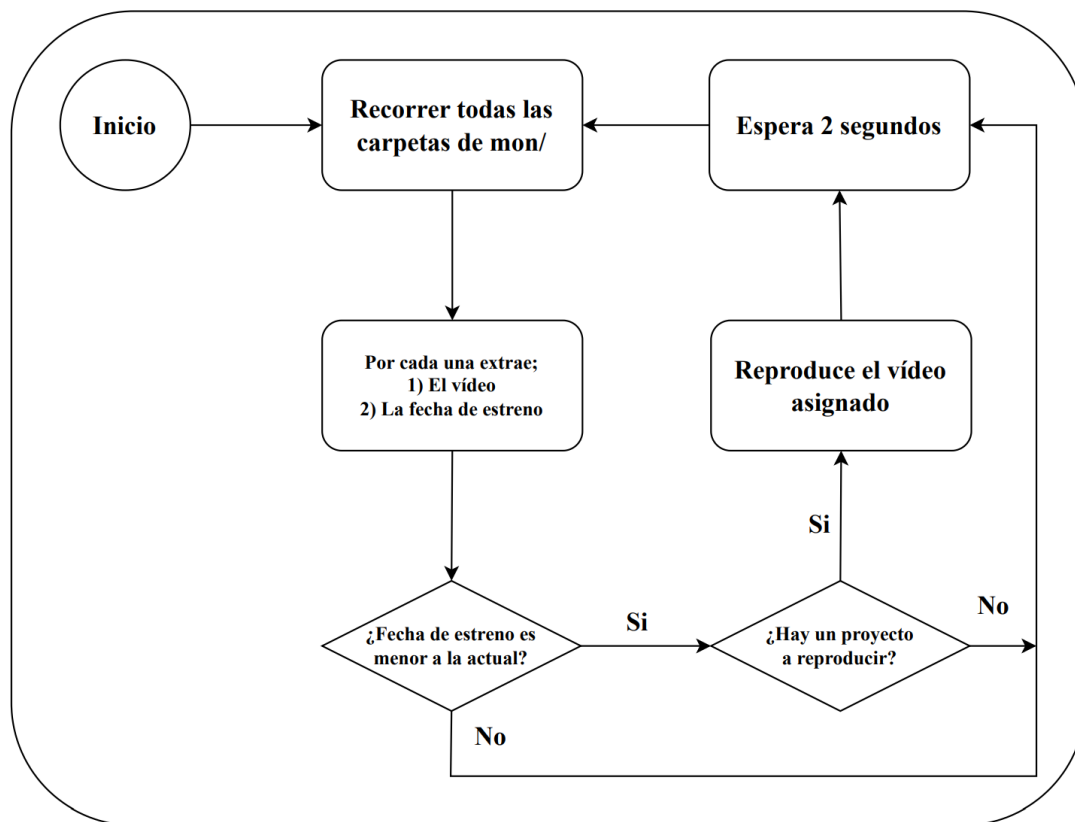


Figura 4.16: Esquema lógico del servicio levantado por el Sistema B

La estabilidad y fiabilidad del sistema se evidencian también en su manejo proactivo de errores. Cuando no se detecta contenido programado, o se encuentran archivos corruptos, el sistema activa automáticamente un estado de espera, mostrando un mensaje informativo mientras continúa escaneando nuevas programaciones disponibles. Este enfoque evita la aparición de pantallas en blanco o fallos visibles, preservando la experiencia del usuario final.

Uno de los pilares tecnológicos más relevantes del Sistema B es la comunicación bidireccional mediante WebSocket. Esta conexión persistente permite actualizaciones en tiempo real sin necesidad de recargar la página, con una latencia mínima que asegura la sincronización efectiva

entre el servidor y los dispositivos. Además, su ligereza permite una operación fluida incluso en redes con ancho de banda reducido.

Desde el punto de vista del despliegue físico, el sistema destaca por su flexibilidad. Cada monitor se comporta como una instancia independiente con su propio puerto de acceso, lo que permite la ejecución simultánea de contenidos diferenciados en pantallas heterogéneas. Esta arquitectura simplifica la escalabilidad, ya que agregar un nuevo monitor implica únicamente asignarle un puerto disponible y crear la carpeta correspondiente dentro de `mon/` para la perspectiva de este sistema. Esta flexibilidad permite acceder a la programación de un monitor únicamente por su URL, por lo que varios televisores pueden presentar el contenido de un monitor específico.

El código fuente del sistema revela una estructura modular bien definida, que combina funcionalidad avanzada con simplicidad de mantenimiento. Esta organización facilita futuras extensiones como la incorporación de nuevos tipos de "widgets", soporte para formatos adicionales o mejoras visuales. Gracias a este equilibrio entre potencia técnica y accesibilidad, el sistema se posiciona como una solución ideal para entornos institucionales donde la confiabilidad y facilidad de operación son esenciales.

Finalmente, desde el punto de vista del espectador, basta con acceder a la página del servidor `sb.js` desde un navegador convencional utilizando el puerto asignado (ej. 3001, 3002, etc.), y el sistema comenzará a reproducir automáticamente el contenido sin necesidad de intervención adicional.

4.4 Implementación de una utilidad de administración

Como parte del enfoque orientado a la autonomía operativa del sistema, se ha desarrollado una herramienta auxiliar que facilita la gestión administrativa, evitando intervenciones manuales sobre la estructura de archivos del sistema. Acciones como la creación de nuevos usuarios, asignación de permisos, establecimiento de contraseñas, modificación de parámetros de almacenamiento o eliminación de directorios asociados pueden conllevar errores cuando se realizan directamente sobre los archivos del sistema, como `metadata.json`.

Para abordar este desafío, se implementó la utilidad `uac-util`, una herramienta de línea de comandos desarrollada en Bash. Esta permite al administrador ejecutar operaciones seguras y estructuradas desde la terminal de sistemas operativos Linux, con dependencias mínimas, destacando el uso de `jq` [6] para la manipulación de archivos JSON. El nombre de la utilidad refiere al acrónimo del sistema principal (UAC) y su propósito específico como interfaz administrativa.

uac-util debe ubicarse en la raíz del proyecto, desde donde se ejecuta. Su conjunto de comandos está organizado en funciones de iniciación, creación, edición, eliminación y diagnóstico de estructuras del sistema, tales como usuarios y monitores. La interacción se realiza mediante flags simples, lo que permite construir comandos claros y reproducibles. Las funciones principales son:

- **Inicialización del entorno:** creación de las carpetas base `src/usr/` y `src/mon/` mediante el comando `--setup`, necesarias para el funcionamiento general del sistema.
- **Listado de carpetas existentes:** visualización del estado de usuarios y monitores actuales con `--list` y `--list m`, respectivamente.
- **Creación de elementos:** nuevos usuarios (`--add u <nombre>`) y monitores (`--add m <nombre>`), generando automáticamente sus respectivas carpetas y archivos de configuración con valores predeterminados.
- **Eliminación de elementos:** usuarios o monitores mediante `--rm u <nombre>` y `--rm m <nombre>`, con remoción completa de sus directorios asociados.
- **Edición de metadatos:** modificación de parámetros críticos de cada usuario utilizando `--edit u <nombre>` acompañado de opciones como:
 - `-p <contraseña>`: para cambiar su contraseña.
 - `-s <tamaño>`: para modificar el límite de almacenamiento permitido.
 - `-m <lista>`: para redefinir el conjunto de monitores asignados, recibiendo una lista del tipo `[1,2,3]`.
- **Comando de ayuda:** con la opción `--help` que despliega un resumen de los comandos y algunos ejemplos como ilustra la Figura 4.17.

```

Uso: ./uac-util [OPCIÓN]...

Opciones:
--help           Mostrar esta ayuda
--setup          Crear estructura de directorios src/usr/ y src/mon/
--list u         Listar carpetas en src/usr/
--list m         Listar carpetas en src/mon/
--add u <nombre> Crear carpeta de usuario en src/usr/
--add m <nombre> Crear carpeta de monitor en src/mon/
--rm u <nombre>  Eliminar carpeta de usuario en src/usr/
--rm m <nombre>  Eliminar carpeta de monitor en src/mon/
--edit u <nombre> [OPCIONES] Editar metadatos de usuario
  Opciones para --edit:
    -p <contraseña> Cambiar contraseña
    -s <tamaño>      Cambiar tamaño de almacenamiento
    -m <lista>       Cambiar lista de monitores

Ejemplos:
./uac-util --edit u pepe -p abc -s 100 -m [1,2,3]
./uac-util --add u juan
./uac-util --list m

```

Figura 4.17: Ejemplo de la salida del comando uac-util con la opción de ayuda

La implementación de uac-util agiliza significativamente la administración del sistema de señalización digital, reduciendo riesgos derivados de la edición manual de archivos y estructuras de directorios. Gracias a su diseño modular y bajo acoplamiento con el sistema principal, esta herramienta puede extenderse fácilmente en el futuro e incorporar nuevas funcionalidades.

Con uac-util, el administrador dispone de una herramienta práctica y eficaz para gestionar la plataforma directamente desde la terminal del servidor, alineándose con el objetivo de autonomía institucional que este proyecto propone.

Es importante señalar que esta utilidad actúa como un complemento al proceso manual, sin sustituirlo. En particular, la creación de un monitor mediante la opción `--add m` habilita una instancia en el sistema de almacenamiento. No obstante, para que esta funcione completamente, es necesario asignarle manualmente un puerto y un identificador de permiso. Dichas asignaciones deben realizarse directamente en las variables globales del sistema, las cuales se estructuran en tres grupos de variables administrativas:

- El primer grupo es una variable global en `sb.js` que mediante un diccionario clave-valor asigna a cada monitor de `mon/` un puerto de salida.
- El segundo grupo es una variable global en `sa.js` que mediante un diccionario clave-valor asigna a cada monitor de `mon/` un identificador de permiso.

- El tercer grupo son 2 variables globales en dash.html llamadas MaxParamDur y MaxParamRes para definir la duración máxima (en segundos) y la resolución máxima (en píxeles) respectivamente. Para que el administrador limite el desempeño según la cantidad de usuarios, monitores, recursos de hardware y ancho de banda disponible para garantizar una operativa estable a lo largo del tiempo. Estas variables no requieren modificarse obligatoriamente al agregar un nuevo monitor, ya que su ajuste solo se recomienda por razones de rendimiento general.

5 RESULTADOS

El sistema desarrollado ha demostrado cumplir de manera satisfactoria con todos los requisitos planteados en la etapa de diseño, validando tanto su funcionalidad técnica como su aptitud para gestionar flujos de trabajo completos en señalización digital. Mediante pruebas exhaustivas y casos de uso realistas, se comprobó que la plataforma ofrece una solución operativa robusta que responde eficazmente a las necesidades comunicacionales de la UTFSM.

El proceso de autenticación, ilustrado en las figuras 4.4 y 4.5, ratifica el cumplimiento del requisito R2.1. Se implementó un sistema seguro de credenciales que restringe el acceso exclusivamente a usuarios autorizados. Tras el ingreso exitoso (Figura 4.6), la interfaz personalizada despliega información relevante como el nombre del usuario y la capacidad de almacenamiento disponible. Además, gestiona adecuadamente los permisos por monitor, satisfaciendo los requisitos R2.2 y R3.3, lo que evidencia la correcta implementación del sistema de gestión de usuarios.

Respecto a la gestión de contenidos (R1.1), la Figura 5.1 muestra una interfaz intuitiva que permite a los usuarios subir, visualizar y eliminar archivos multimedia de forma segura y aislada. El sistema realiza validaciones automáticas sobre el tipo y tamaño de archivo, asegurando que se respeten los límites establecidos. En relación con el requisito R1.2, la sección de proyectos (Figura 5.2) permite diseñar composiciones multimedia complejas mediante la definición de zonas temporales y espaciales, las cuales son procesadas en segundo plano para la generación del video final.

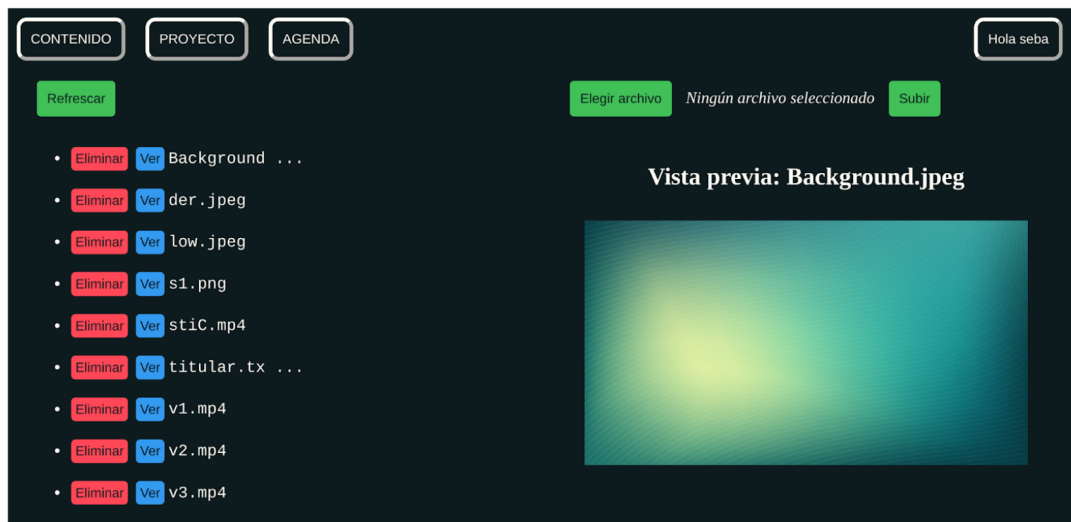


Figura 5.1: Página principal, en la sección de contenido

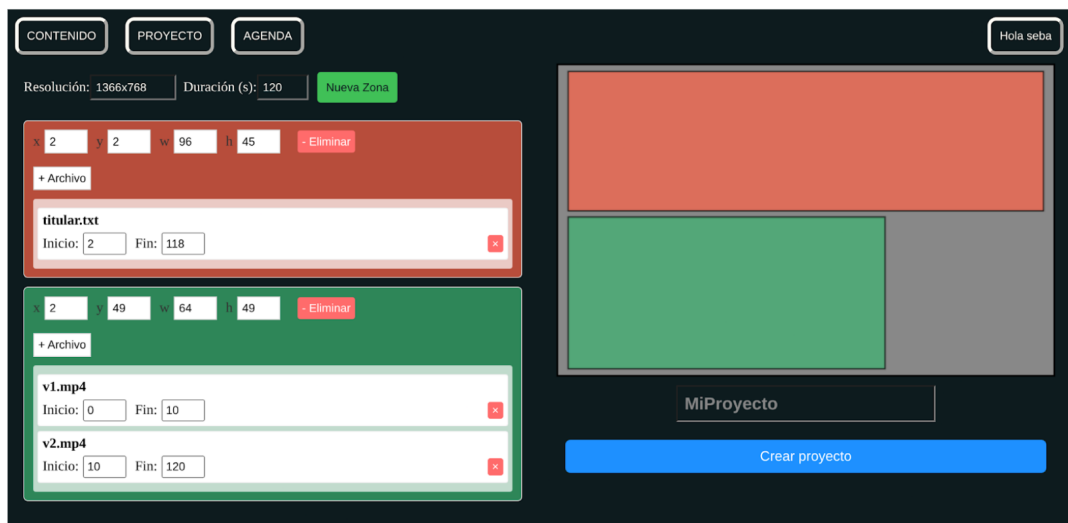


Figura 5.2: Página principal, en la sección de proyecto

La programación de contenidos (Figura 5.3 y Figura 5.4) verifica los requisitos R3.1 y R3.2, mostrando cómo los proyectos pueden asignarse a monitores específicos con fechas de estreno precisas. La inclusión de "widgets" en tiempo real, como el de tiempo, con la fecha y hora, añade valor informativo sin requerir intervención manual en los televisores, que se configuran inicialmente para recibir la transmisión desde el Sistema B.

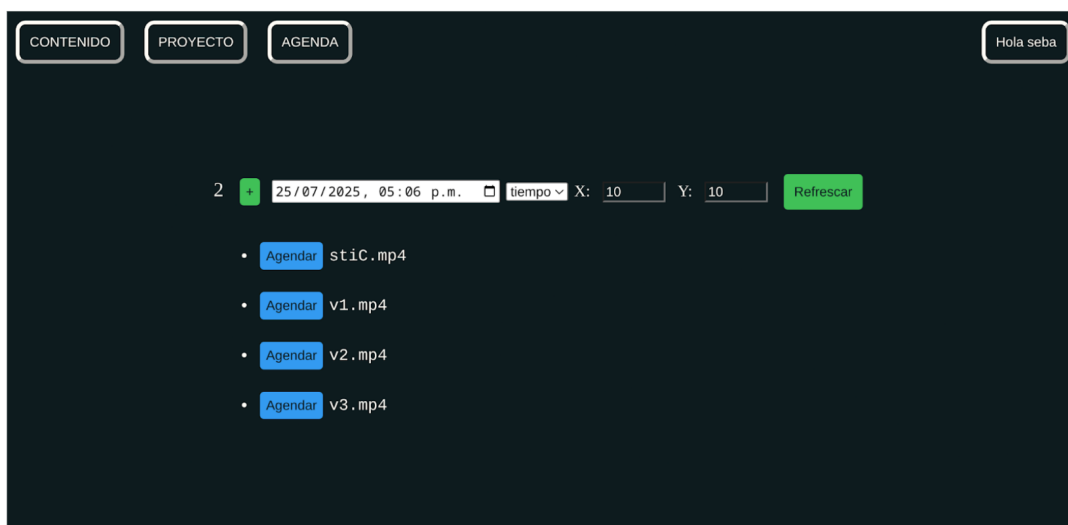


Figura 5.3: Página principal, en la sección de agenda



Figura 5.4: Página levantada del Sistema B, para la reproducción asignada a un monitor

Para demostrar la capacidad integral del sistema, se presenta un caso de uso realista basado en contenido alineado con el esquema actual de visualización institucional. Desde la selección de materiales multimedia de alta calidad (Figura 5.5), pasando por el diseño detallado de zonas y temporizaciones (figuras 5.6, 5.7 y 5.8), hasta llegar al video final renderizado (Figura 5.9), este ejemplo evidencia que la plataforma está en condiciones de generar contenidos que cumplen con los estándares de la UTFSM y de manera totalmente interna.

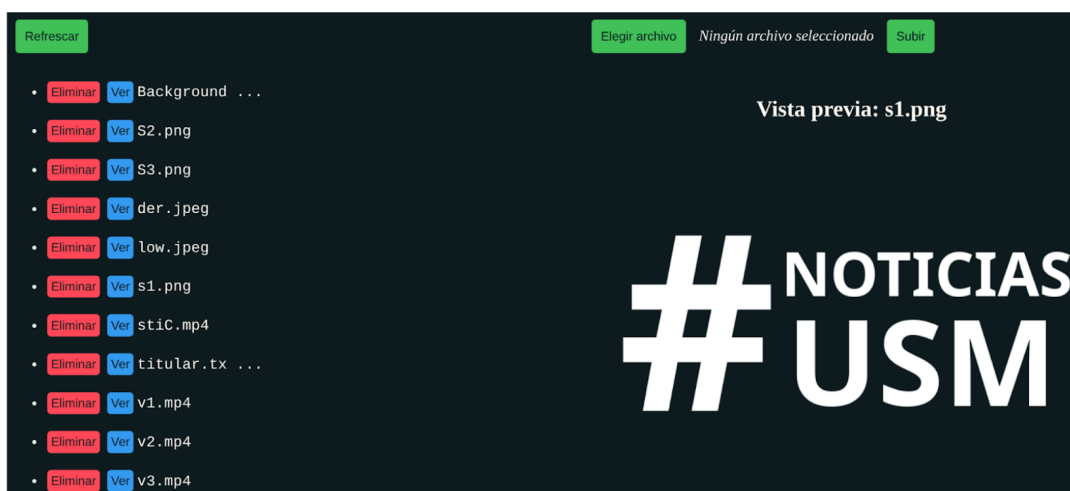


Figura 5.5: Una sección de contenido con contenido multimedia elaborado

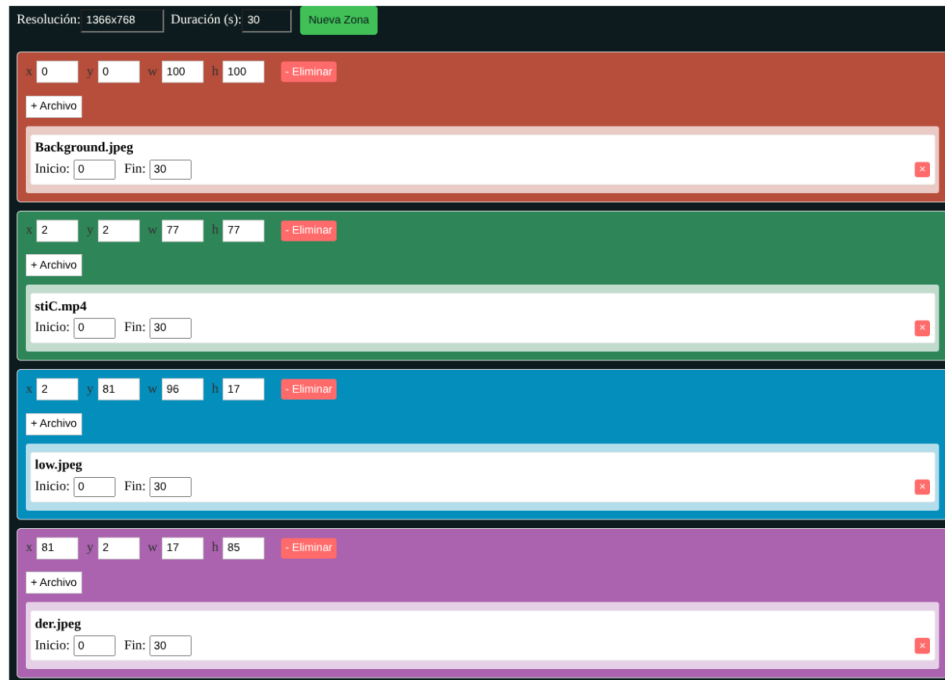


Figura 5.6: Una sección de proyecto, con zonas elaboradas (Parte superior)

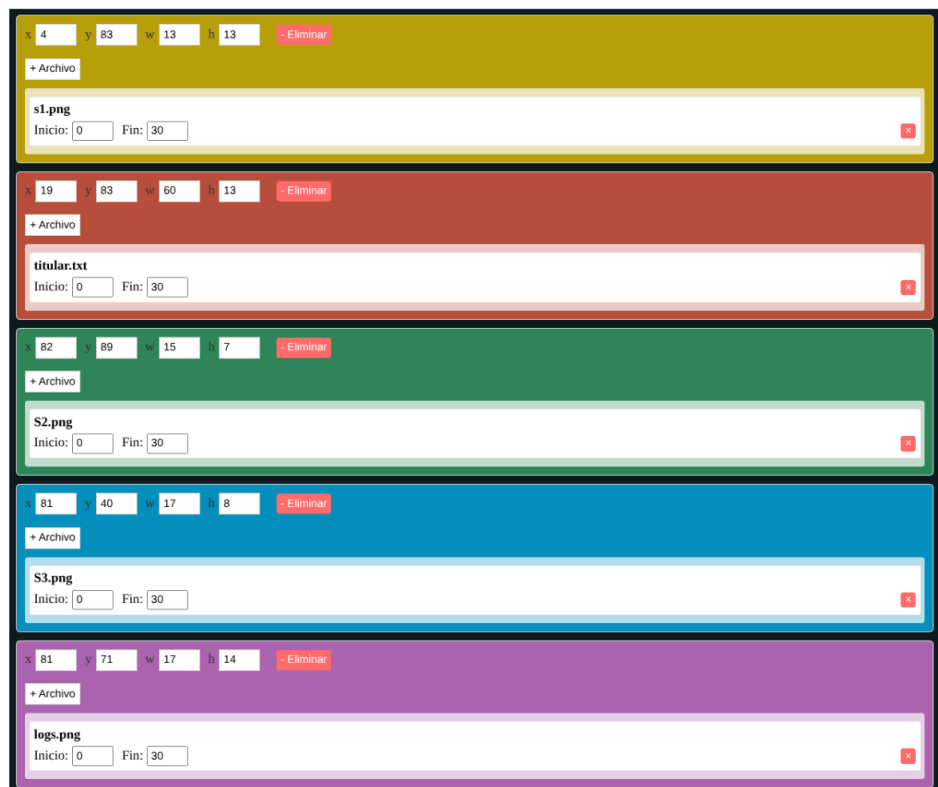


Figura 5.7: Una sección de proyecto, con zonas elaboradas (Parte inferior)

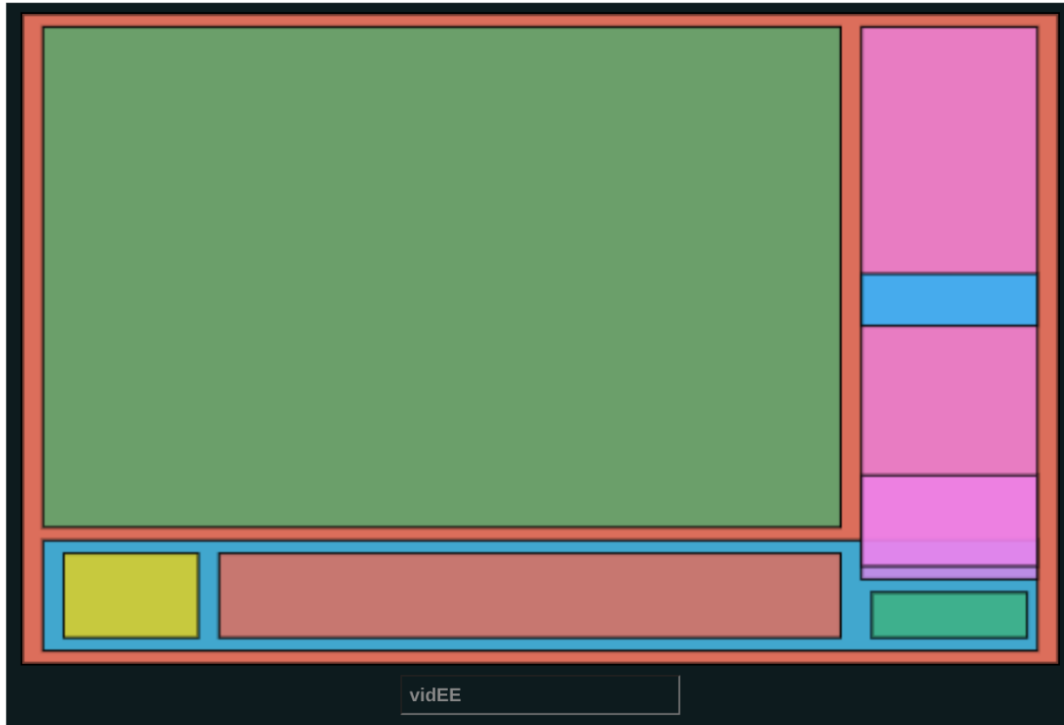


Figura 5.8: Una sección de proyecto, con zonas elaboradas en el lienzo

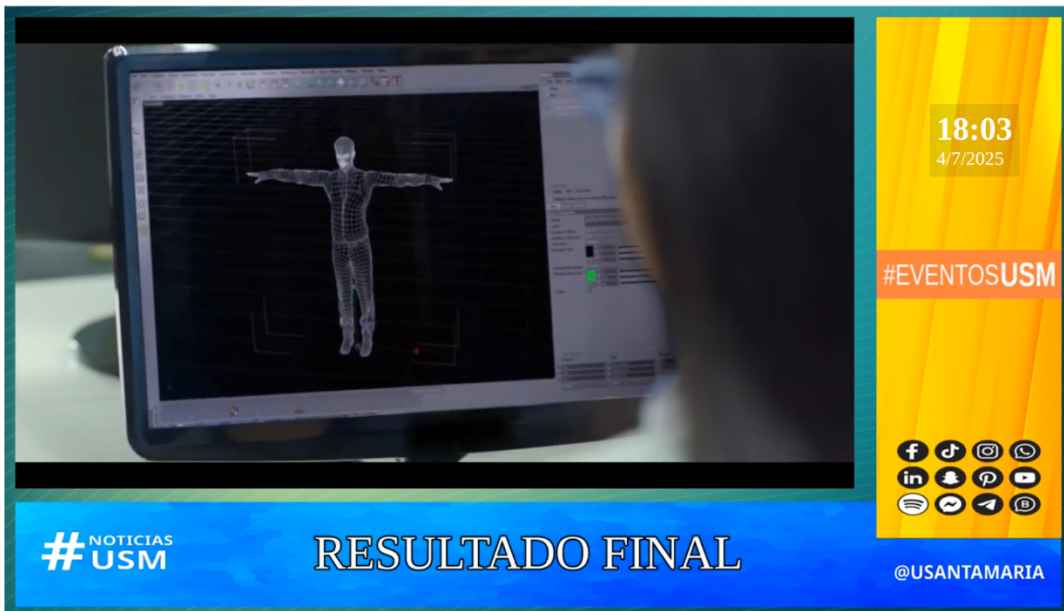


Figura 5.9: Un cuadro del video final elaborado, agendado y desplegado por el Sistema B

6 CONCLUSIONES Y TRABAJO FUTURO

La presente obra demuestra que es técnicamente factible desarrollar internamente un sistema de señalización digital funcional, eliminando la dependencia de proveedores externos que, aunque operativos, limitan la autonomía institucional sobre activos digitales. La implementación propuesta no solo integra funcionalidades básicas de sistemas de señalización digital, sino que establece una base adaptable y escalable bajo control universitario. La señalización digital, como herramienta de comunicación institucional, adquiere mayor valor cuando se gestiona de manera soberana, permitiendo ajustes específicos sin depender de condicionamientos externos.

La principal contribución de este trabajo radica en probar que instituciones como la UTFSM poseen la capacidad técnica para desarrollar alternativas viables a servicios tradicionalmente tercerizados, recuperando así el control sobre sus procesos comunicacionales. Esta autonomía se traduce en ventajas concretas, desde la eliminación de costos recurrentes de licenciamiento hasta la posibilidad de integración con otros sistemas internos, pasando por una gestión más segura y privada de los contenidos. Además, la posibilidad de manejar múltiples grupos de usuarios y monitores permite ofrecer un servicio de gestión autónoma a distintas unidades de la universidad. La arquitectura modular adoptada permite, además, evoluciones futuras sin requerir cambios disruptivos, asegurando que la solución crezca junto con las necesidades de la universidad.

No obstante, el análisis durante el desarrollo ha revelado áreas con margen de mejora. Particularmente en la sección de contenido, si bien el sistema actual permite operaciones básicas, sería deseable ampliar los formatos compatibles e implementar un sistema de etiquetado que facilite la organización avanzada de archivos. Asimismo, una función de búsqueda inteligente por metadatos o nombres agilizaría el trabajo de los operarios cuando el volumen de contenido aumente.

Asimismo, la sección de agenda, siendo fundamental para la señalización digital, presenta oportunidades claras de evolución. Actualmente solo permite programar un proyecto a la vez, lo que limita la creación de listas de reproducción con múltiples fechas de estreno y proyectos. Una mejora en este aspecto permitiría, por ejemplo, rotación automática o programar mensajes urgentes que interrumpan temporalmente la programación, permitiendo su reanudación automática. Además, el sistema de "widgets", actualmente restringido a un solo tipo, de tiempo, podría enriquecerse con opciones personalizables, como la inclusión de datos en tiempo real como eventos académicos, alertas o incluso "widgets" que los usuarios puedan modificar según sus necesidades.

Respecto a la sección de proyecto, si bien no es el núcleo crítico del sistema, más aún con las poderosas herramientas de edición con las que cuentan entidades audiovisuales, comunicaciones o "marketing", su evolución podría optimizar significativamente el flujo de trabajo. La introducción de plantillas predefinidas para distintos tipos de comunicaciones, ya sean eventos, informativas o emergencias, reduciría el tiempo de creación, especialmente para usuarios menos técnicos. Complementariamente, herramientas básicas de edición integradas, como transiciones o una línea de tiempo visual, lograrían un proceso de creación de contenido más eficaz dentro de la plataforma.

Este proyecto ofrece oportunidades estratégicas para la UTFSM, más allá de sus mejoras técnicas. La plataforma puede convertirse en el eje de un ecosistema de comunicación digital integrado, conectándose con sistemas existentes como calendarios académicos o alertas de emergencia. Además, sienta un precedente en la adopción de modelos de desarrollo interno para otras herramientas institucionales, promoviendo la soberanía tecnológica. La capacitación al personal universitario en el mantenimiento y evolución del sistema no solo garantiza su sostenibilidad, sino que también fortalece las competencias técnicas internas.

En síntesis, este trabajo no solo resuelve el problema inicial de contar con una herramienta de señalización digital funcional y autónoma, sino que también demuestra que es posible desarrollar soluciones tecnológicas institucionales con recursos propios. A partir de esta base, se abren nuevas oportunidades para fortalecer la soberanía digital de la UTFSM, integrar sistemas internos y adaptar la plataforma a las necesidades cambiantes de cada unidad. Las mejoras identificadas, como la ampliación de formatos, la gestión avanzada de contenidos y la personalización de la agenda, ofrecen un camino claro para su evolución. Este proyecto marca un punto de partida hacia una comunicación más efectiva dentro del ecosistema universitario, representando un caso de éxito en cómo las instituciones pueden recuperar el control sobre sus herramientas digitales, equilibrando autonomía, flexibilidad y eficiencia para evolucionar según los desafíos venideros.

7 REFERENCIAS

- [1] Kazuhiro Mishima, Takeshi Sakurada, Yoichi Hagiwara. Cost effective digital signage system using low cost information device. En: 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC) (2016, Las Vegas, NV, USA).
- [2] Kazuhiro Mishima, Takeshi Sakurada, Yoichi Hagiwara. Low-cost Managed Digital Signage System with Signage Device using Small-sized and Low-cost Information Device. En: 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC) (2017, Las Vegas, NV, USA).
- [3] Página web oficial de Xibo. [En línea].
<https://xibosignage.com/>
[Consulta: 9 junio 2025].
- [4] Página web oficial de Scala. [En línea].
<https://scala.com/es/>
[Consulta: 9 junio 2025].
- [5] Página web oficial de Mixxer.tv. [En línea].
<https://mixxertv.com/account/login>
[Consulta: 9 junio 2025].
- [6] Página web oficial de Jq. [En línea].
<https://jqlang.org/>
[Consulta: 22 septiembre 2025].
- [7] Página web oficial de Scala Player. [En línea].
<https://scala.com/en/products/software/scala-player/>
[Consulta: 11 junio 2025].
- [8] Página web oficial de Scala Design. [En línea].
<https://latam.scala.com/mx/products/software/scala-designer/>
[Consulta: 11 junio 2025].
- [9] Página web oficial de directvnc. [En línea].
<https://github.com/drinkmilk/directvnc>

[Consulta: 22 abril 2025].

[10] Página web oficial de rsync. [En línea].

<https://rsync.samba.org/>

[Consulta: 22 abril 2025].

[11] Página web oficial de LG. [En línea].

<https://www.lg.com/cl/>

[Consulta: 22 abril 2025].

[12] Página web oficial de FFmpeg. [En línea].

<https://ffmpeg.org/>

[Consulta: 22 abril 2025].

[13] Página web oficial de Raspberry Pi. [En línea].

<https://www.raspberrypi.com/>

[Consulta: 22 abril 2025].

ANEXOS

Anexo A - Códigos de Bash

Código de uac-util.sh

```
#!/bin/bash

show_help() {
    echo "Uso: ./uac-util [OPCIÓN]..."
    echo ""
    echo "Opciones:"
    echo "  --help           Mostrar esta ayuda"
    echo "  --setup          Crear estructura de directorios src/usr/ y src/mon/"
    echo "  --list u         Listar carpetas en src/usr/"
    echo "  --list m         Listar carpetas en src/mon/"
    echo "  --add u <nombre> Crear carpeta de usuario en src/usr/"
    echo "  --add m <nombre> Crear carpeta de monitor en src/mon/"
    echo "  --rm u <nombre>  Eliminar carpeta de usuario en src/usr/"
    echo "  --rm m <nombre>  Eliminar carpeta de monitor en src/mon/"
    echo "  --edit u <nombre> [OPCIONES] Editar metadatos de usuario"
    echo "    Opciones para --edit:"
    echo "      -p <contraseña> Cambiar contraseña"
    echo "      -s <tamaño>      Cambiar tamaño de almacenamiento"
    echo "      -m <lista>       Cambiar lista de monitores"
    echo ""
    echo "Ejemplos:"
    echo "  ./uac-util --edit u pepe -p abc -s 100 -m [1,2,3]"
    echo "  ./uac-util --add u juan"
    echo "  ./uac-util --list m"
}

setup_directories() {
    mkdir -p src/usr
    mkdir -p src/mon
    echo "Estructura de directorios creada en src/usr/ y src/mon/"
}
```

```

list_directories() {
    local type=$1
    if [ "$type" = "u" ]; then
        if [ -d "src/usr" ]; then
            ls src/usr/
        else
            echo "El directorio src/usr/ no existe. Ejecuta --setup."
        fi
    elif [ "$type" = "m" ]; then
        if [ -d "src/mon" ]; then
            ls src/mon/
        else
            echo "El directorio src/mon/ no existe. Ejecuta --setup."
        fi
    else
        echo "Tipo incorrecto para --list. Usa 'u' o 'm'."
    fi
}

add_user() {
    local name=$1
    local dir="src/usr/$name"
    if [ -d "$dir" ]; then
        echo "El usuario $name ya existe."
        return
    fi
    mkdir -p "$dir/contenido"
    cat > "$dir/metadata.json" <<EOF
{
    "AUT":
    {
        "passwd": "1234"
    },
    "STO":
    {
        "size": 50
    },
    "ACL":
    {
        "mon": [0]
    }
}
EOF
    echo "Usuario $name creado con éxito." }

```

```
add_monitor() {
    local name=$1
    local dir="src/mon/$name"

    if [ -d "$dir" ]; then
        echo "El monitor $name ya existe."
        return
    fi

    mkdir -p "$dir"
    echo "Monitor $name creado con éxito."
}

remove_item() {
    local type=$1
    local name=$2

    if [ "$type" = "u" ]; then
        local dir="src/usr/$name"
    elif [ "$type" = "m" ]; then
        local dir="src/mon/$name"
    else
        echo "Tipo incorrecto. Usa 'u' o 'm'."
        return
    fi

    if [ -d "$dir" ]; then
        rm -r "$dir"
        echo "$name eliminado con éxito."
    else
        echo "$name no existe."
    fi
}
```

```

edit_user() {
    local name=$1
    shift
    local dir="src/usr/$name"
    local passwd=""
    local size=""
    local mon=""
    local update_mon=false

    if [ ! -d "$dir" ]; then
        echo "El usuario $name no existe."
        return
    fi

    # Parsear opciones
    while [[ $# -gt 0 ]]; do
        case $1 in
            -p)
                passwd="$2"
                shift 2
                ;;
            -s)
                size="$2"
                shift 2
                ;;
            -m)
                mon="$2"
                update_mon=true
                shift 2
                ;;
            *)
                echo "Opción desconocida: $1"
                return
                ;;
        esac
    done

    local tmpfile=$(mktemp)

    if [ "$update_mon" = true ]; then
        # Si se proporcionó -m, procesar con --argjson

```

```

jq --arg passwd "$passwd" \
   --arg size "$size" \
   --argjson mon "$mon" \
   ,
if ($passwd != "") then .AUT.passwd = $passwd else . end |
if ($size != "") then .STO.size = ($size | tonumber) else . end |
.ACL.mon = $mon
' "$dir/metadata.json" > "$tmpfile" && mv "$tmpfile"
"$dir/metadata.json"
else
# Si no se proporcionó -m, no usar --argjson para mon
jq --arg passwd "$passwd" \
   --arg size "$size" \
   ,
if ($passwd != "") then .AUT.passwd = $passwd else . end |
if ($size != "") then .STO.size = ($size | tonumber) else . end
' "$dir/metadata.json" > "$tmpfile" && mv "$tmpfile"
"$dir/metadata.json"
fi

echo "Metadatos de $name actualizados correctamente."
}

```

```

case "$1" in
    --help)
        show_help
        ;;
    --setup)
        setup_directories
        ;;
    --list)
        list_directories "$2"
        ;;
    --add)
        if [ "$2" = "u" ]; then
            add_user "$3"
        elif [ "$2" = "m" ]; then
            add_monitor "$3"
        else
            echo "Tipo incorrecto para --add. Usa 'u' o 'm'."
        fi
        ;;
    --rm)
        if [ "$2" = "u" ]; then
            remove_item "u" "$3"
        elif [ "$2" = "m" ]; then
            remove_item "m" "$3"
        else
            echo "Tipo incorrecto para --rm. Usa 'u' o 'm'."
        fi
        ;;
    --edit)
        if [ "$2" = "u" ]; then
            shift 2
            edit_user "$@"
        else
            echo "Solo se puede editar usuarios (--edit u <nombre>)."

```