

MEMORIA DEL PROYECTO

Avilés, 06/2025

Autor/a: Carro Da Silva, Javier

INDICE

1. Introducción	3
1.1. Contexto.	3
1.2. Planteamiento del Problema	4
1.3. Se realizará un breve resumen del proyecto	4
2. Objetivos del proyecto	5
2.1. Objetivo general del proyecto	5
2.2. Objetivos específicos	5
3. Análisis y Diseño	6
3.1. Requisitos funcionales	6-8
3.2. Casos de Uso	9-10
3.3. Modelo de datos (Diagrama E/R)	10-11
3.4. Pruebas (Unitarias y de Integración)	12-13
4. Implementación	13
4.1 Tecnologías/Herramientas empleadas	13-15
4.2 Descripción	15-16
5. Planificación y presupuesto	17
5.1. Diagrama Gantt de planificación de tareas	17-20
5.2. Presupuesto	20
6. Futuras mejoras y ampliaciones	21-22
7. Conclusiones sobre el proyecto	23-24
8. Índice de figuras (imágenes / tablas)	24
9. Siglas y acrónimos	25

1. Introducción

Hoy en día, la música forma parte de la vida cotidiana de millones de personas. Ya sea para relajarse, trabajar, entrenar o simplemente desconectar, la mayoría de nosotros utilizamos plataformas digitales para acceder a nuestras canciones favoritas. Sin embargo, muchas de estas plataformas están limitadas por suscripciones, catálogos cerrados o restricciones para subir y compartir música propia.

En este contexto surge la idea de crear una aplicación web que no solo permita escuchar música de forma libre y personalizada, sino que también ofrezca un espacio para que cualquier persona pueda cargar sus propias canciones y artistas emergentes puedan dar a conocer su trabajo.

1.1. Contexto

VibraSync es una aplicación web centrada en la música que busca ofrecer una experiencia diferente, práctica y accesible para cualquier tipo de usuario. La idea principal es que cualquier persona pueda escuchar música desde la plataforma de forma gratuita, sin necesidad de tener una cuenta premium o depender de servicios externos.

Una de las funciones más destacadas de la aplicación es que el usuario puede subir su propia música a través de una carpeta local. Es decir, puede arrastrar o cargar archivos desde su dispositivo y escucharlos directamente desde el navegador. Lo importante es que esos archivos no se almacenan ni en el servidor ni en la base de datos: se cargan de forma temporal, solo para que el propio usuario los escuche mientras tenga la sesión activa. Esto permite disfrutar de la música personal sin comprometer la privacidad ni ocupar espacio en la nube.

Además, VibraSync también está pensada como una plataforma para artistas emergentes. Cualquier creador puede subir su música de forma oficial a través de distribuidoras autorizadas o servicios integrados, lo que permite que sus canciones estén disponibles para todos los usuarios de la plataforma. Esta función abre la posibilidad de descubrir música nueva y apoyar a pequeños artistas de manera directa.

La interfaz está diseñada para ser intuitiva y atractiva. Cuando una canción comienza a reproducirse, parte de la página se transforma visualmente para adaptarse a la música, creando una experiencia más inmersiva. También existe una sección donde se muestran videos o contenido visual relacionado con la canción que está sonando, todo de forma dinámica.

En resumen, VibraSync combina escucha gratuita, personalización y promoción artística en una sola plataforma web, todo sin depender de cuentas premium ni almacenamiento innecesario.

1.2. Planteamiento del Problema

Hoy en día, la mayoría de las plataformas de streaming musical funcionan bajo un modelo cerrado, donde muchas funciones están limitadas a usuarios con suscripción. Escuchar música sin anuncios, elegir canciones libremente o reproducir contenido sin conexión son características que suelen estar bloqueadas tras un pago mensual. Además, estas plataformas no permiten al usuario subir su propia música desde archivos locales, lo que impide una experiencia realmente personalizada.

A esto se suma la dificultad que enfrentan los artistas independientes para publicar su música en estas plataformas. Subir canciones a través de distribuidoras implica procesos técnicos y económicos que no todos pueden asumir, lo que limita la visibilidad de nuevos talentos.

Esta situación plantea dos necesidades claras: por un lado, ofrecer una forma libre y accesible para que cualquier usuario pueda escuchar su música sin restricciones y, por otro, facilitar que artistas emergentes compartan su trabajo sin depender de terceros.

VibraSync surge como respuesta a este problema. La aplicación permite escuchar música gratuita, subir carpetas locales de forma temporal (sin guardarlas en el servidor) y también ofrece la posibilidad de que los artistas publiquen su música fácilmente a través de distribuidoras, todo desde una misma plataforma.

1.3. Se realizará un breve resumen del proyecto

Este proyecto consiste en el desarrollo de una aplicación web llamada VibraSync, orientada a ofrecer una experiencia musical más abierta, personalizada y accesible. La plataforma permitirá a los usuarios escuchar música de forma gratuita, sin necesidad de suscripción, y además podrán subir sus propias carpetas de música directamente desde su dispositivo. Estos archivos se cargarán temporalmente en el navegador, sin almacenarse en el servidor ni en la base de datos, garantizando privacidad y autonomía.

Además, VibraSync contará con una funcionalidad para que artistas independientes puedan distribuir su música a través de distribuidoras oficiales, facilitando su visibilidad en la plataforma. La interfaz será dinámica, adaptándose visualmente al estilo de cada canción y mostrando contenido relacionado para enriquecer la experiencia del usuario.

En resumen, el proyecto busca combinar reproducción gratuita, soporte para música local y una herramienta de difusión musical en una sola aplicación web, funcional, ligera y centrada en el usuario.

2. Objetivos del proyecto

2.1. Objetivo general del proyecto

Desarrollar una aplicación web llamada VibraSync que permita a los usuarios escuchar música de forma gratuita, combinando la reproducción de archivos locales cargados temporalmente desde su dispositivo con la difusión de música de artistas independientes a través de distribuidoras. La plataforma ofrecerá una experiencia visual dinámica que se adapte a la música en reproducción, integrando contenido relacionado para hacerla más inmersiva. Además, se garantizará la privacidad del usuario al no almacenar ni compartir los archivos personales, manteniéndolos solo durante la sesión activa en el navegador.

VibraSync estará diseñada para funcionar en distintos dispositivos y tamaños de pantalla, utilizando tecnologías web modernas para ofrecer una experiencia fluida, accesible y atractiva, que sirva como alternativa flexible y abierta frente a las plataformas tradicionales de streaming musical.

2.2. Objetivos específicos

- Diseñar una interfaz de usuario atractiva, intuitiva y moderna que mejore la experiencia de navegación y se adapte visualmente en función de la canción que se esté reproduciendo.
- Implementar un reproductor de música completo que permita funciones básicas como reproducir, pausar, avanzar, retroceder, controlar el volumen y visualizar el progreso de la canción.
- Desarrollar una función que permita al usuario subir carpetas o archivos de música desde su propio dispositivo, con la particularidad de que estos archivos se cargan únicamente de forma temporal en el navegador, sin guardarse en la base de datos ni en el servidor.
- Asegurar que los archivos cargados por el usuario solo estén disponibles durante su sesión activa y que no se comprometa la privacidad ni la seguridad de su contenido local.
- Integrar una opción para que artistas independientes puedan publicar su música a través de distribuidoras musicales, facilitando así la difusión de nuevos talentos dentro de la propia plataforma.
- Garantizar que la aplicación sea completamente responsive, compatible con distintos tamaños de pantalla y dispositivos (PC, tablet, smartphone), y que funcione fluidamente en todos ellos.
- Utilizar tecnologías modernas del desarrollo web (como HTML5, CSS3, JavaScript, y tecnologías nuevas) para asegurar un rendimiento óptimo y un diseño actual.

3. Análisis y Diseño

3.1. Requisitos funcionales

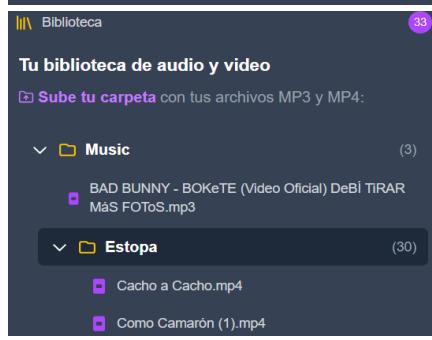
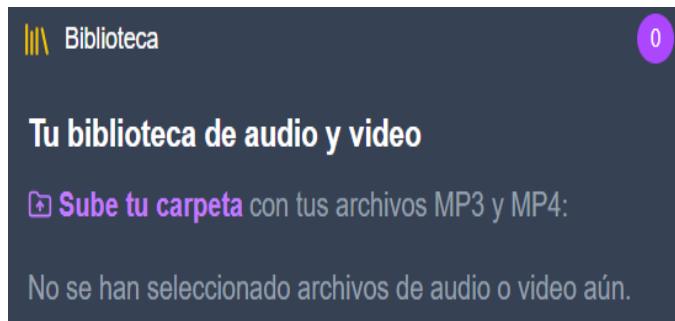
- **Reproducción gratuita de música**

La plataforma debe permitir a cualquier usuario, sin necesidad de registrarse, acceder y reproducir canciones disponibles en la plataforma de manera gratuita. Esto incluye la capacidad de seleccionar pistas, reproducirlas en orden o de forma aleatoria y pausar o detener la reproducción en cualquier momento.



- **Carga temporal de archivos locales**

Los usuarios deben poder cargar carpetas o archivos de música almacenados en sus dispositivos personales para escucharlos directamente en el navegador. Estos archivos no deben ser subidos ni guardados en ningún servidor o base de datos, asegurando que solo se mantengan activos durante la sesión del usuario en el navegador.



- **Control completo del reproductor**

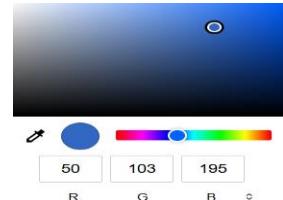
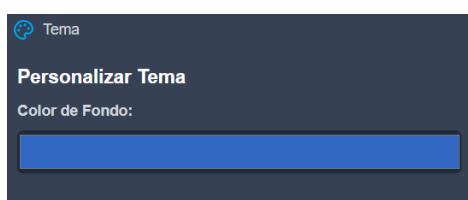
El reproductor integrado debe contar con controles básicos como reproducir, pausar, avanzar a la siguiente canción, retroceder a la anterior, ajustar el volumen y

visualizar el progreso de la pista en reproducción. Además, debe ofrecer una interfaz intuitiva y accesible para facilitar su uso.



- **Interfaz visual adaptable**

La interfaz de usuario pueda ser modificada según sus preferencias. Esto incluye modificar colores, fondos.



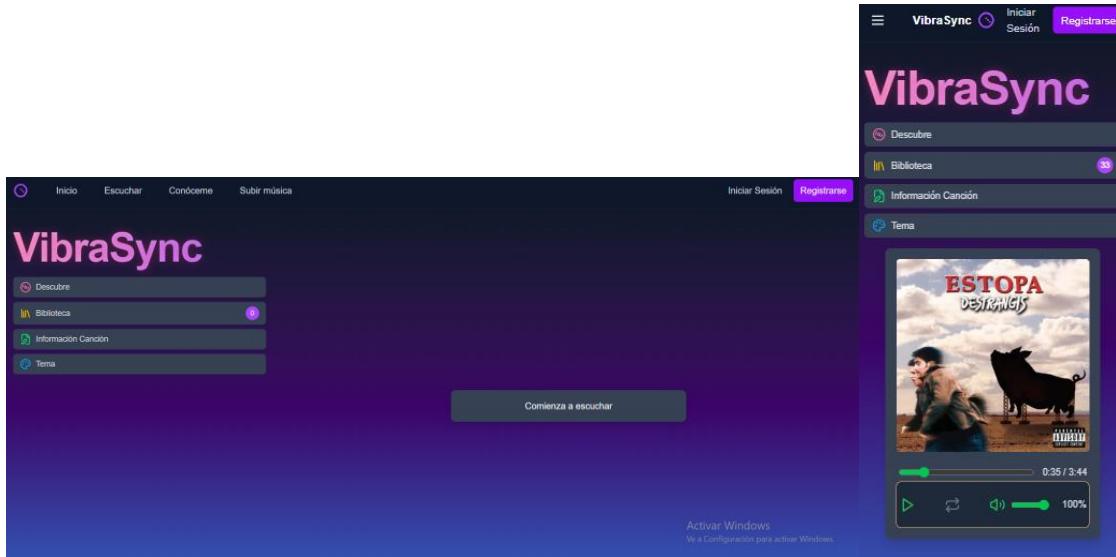
- **Integración con artistas y distribuidoras**

La plataforma debe permitir que artistas independientes puedan subir su música a través de distribuidoras autorizadas, asegurando que su contenido esté disponible para todos los usuarios. Esta función debe incluir un sistema para la gestión y verificación de los archivos enviados.

Two screenshots of a mobile application interface. The left screenshot shows a purple-themed landing page with the text '¿Te gustaría ser creador en nuestra aplicación?' and a 'Quiero unirme' button. The right screenshot shows a 'Sube música' (Upload music) screen with fields for 'Título' (Title) and 'Archivo (Leap o mp3)', and a 'Subir canción' (Upload song) button. Below these are two smaller preview cards for songs titled 'Prueba' and 'aa'.

- **Compatibilidad responsiva**

La aplicación debe ser responsive y funcionar correctamente en dispositivos de diferentes tamaños y características, tales como ordenadores de escritorio, tablets y smartphones, asegurando una experiencia consistente en todos ellos.



- Registro y gestión de usuarios**

Incorporar un sistema de registro y gestión de usuarios para futuras mejoras, como listas personalizadas, favoritos.

Nombre de Usuario	Nombre completo	Correo	Role	Verificado
VibraSync_Oicial	—	vibrasync@gmail.com	admin	No
jcarrada	Javier Carrada	jcarrada@gmail.com	admin	No
Antonio	Antonio Miranda Nine	natalie6305@kobro.com	usuario	Si
jcarrada	Javier Carrada	jcarrada@gmail.com	admin	Si

3.2. Casos de Uso

Caso de uso 1: Escuchar música sin registro y funciones básicas.

Usuario: no registrado.

Como lo consigue: Con entrar en la aplicación ya puede realizar lo siguiente.

Que puede hacer:

- Puede visualizar el inicio.
- Puede reproducir canciones alojadas en el servidor..
- Puede reproducir canciones subidas en una carpeta o por si solas.
- Puede ver los metadatos de estas.
- Puede cambiar el tema del reproductor.
- Puede ir a conócame.
- Puede acceder al registro y log in.

-Puede ver los términos y privacidad.

Caso de uso 2: Escuchar añadir música a favoritos y funciones añadidas

Usuario: registrado (rol usuario).

Como lo consigue: A de crearse una cuenta y verificar el email.

Que puede hacer:

- Puede visualizar el inicio.
- Puede reproducir canciones alojadas en el servidor.
- Puede reproducir canciones subidas en una carpeta o por si solas.
- Puede ver los metadatos de estas.
- Puede cambiar el tema del reproductor.
- Puede ir a conócmeme.
- Puede ver los términos y privacidad.
- Puede acceder al registro y log in.
- Puede guardar canciones en favoritos y borrarlas de favoritos.
- Permanecerá activa la sesión gracias al uso de cookies.
- Permite el acceso al módulo de subir canciones (Panel Informativo)
- Permite acceder al panel de usuario, donde el usuario puede añadir, modificar y borrar sus datos propios, además de poder eliminar su cuenta.
- Podrás solicitar en sube música, que una discográfica suba tu música, contactando con ellos por correo.

Caso de uso 3: Subir canciones al servidor y demás funciones.

Usuario: registrado (rol creador).

Como lo consigue: Una vez ya tienes cuenta podrás solicitar ser creador (en calidad de discográfica, lo que te permitirá subir y administrar las canciones de la base de datos).

Que puede hacer:

- Puede visualizar el inicio.
- Puede reproducir canciones alojadas en el servidor.
- Puede reproducir canciones subidas en una carpeta o por si solas.
- Puede ver los metadatos de estas.
- Puede cambiar el tema del reproductor.
- Puede ir a conócmeme.
- Puede ver los términos y privacidad.
- Puede acceder al registro y log in.
- Puede guardar canciones en favoritos.
- Permanecerá activa la sesión gracias al uso de cookies.
- Permite el acceso al módulo de subir canciones (Panel Informativo).
- Permite acceder al panel de usuario, donde el usuario puede añadir, modificar y borrar sus datos propios, además de poder eliminar su cuenta.
- Puedes insertar y modificar las canciones de la base de datos en la pestaña de subir música que en vez de llevarte a la que ven los demás usuarios sin rol creador o admin te lleva a un sube música creador (un panel de control donde te permite administrar estas canciones).

Caso de uso 4: Administrar usuarios globales de la base de datos y demás funciones.

Usuario: registrado (rol admin).

Como lo consigue: A este rol solo podrán acceder propios trabajadores de la página.

Que puede hacer:

- Puede visualizar el inicio.
- Puede reproducir canciones alojadas en el servidor.
- Puede reproducir canciones subidas en una carpeta o por si solas.
- Puede ver los metadatos de estas.
- Puede cambiar el tema del reproductor.
- Puede ir a conócmeme.
- Puede ver los términos y privacidad.
- Puede acceder al registro y log in.
- Puede guardar canciones en favoritos.
- Permanecerá activa la sesión gracias al uso de cookies.
- Permite el acceso al módulo de subir canciones (Panel Informativo).
- Permite acceder al panel de usuario, donde el usuario puede añadir, modificar y borrar sus datos propios, además de poder eliminar su cuenta.
- Puedes insertar y modificar las canciones de la base de datos en la pestaña de subir música que en vez de llevarte a la que ven los demás usuarios sin rol creador o admin te lleva a un sube música creador (un panel de control donde te permite administrar estas canciones).
- Se te aparecerá un nuevo botón llamado panel de administrador.
- Al acceder a panel administrador, tendrás la opción de ver, añadir y borrar cualquier usuario en la base de datos.

3.3. Modelo de datos (Diagrama E/R)

Mi modelo entidad relación de la base de datos es el siguiente:

- **usuarios**
 - d_usuario (PK)
 - nombre_usuario
 - nombre_completo
 - correo
 - contraseña
 - rol (usuario, admin, creador)
 - fecha_registro
 - verificado
- **canciones**
 - id_cancion (PK)
 - fecha_publicacion
 - archivo_audio (URL o ruta local)
 - id_usuario (FK)
 - titulo
- **favoritos**
 - id_usuario (PK, FK)
 - id_cancion (PK, FK)

fecha_agregado

Relaciones:

Un usuario puede subir muchas canciones (1:N).

Un usuario puede tener muchas canciones marcadas como favoritas (N:M), lo cual se gestiona a través de la tabla intermedia favoritos.

Una canción puede estar en los favoritos de varios usuarios (N:M).

Este modelo permite:

- Control de acceso por roles.
- Gestión segura de contraseñas (encriptadas).
- Relación entre usuarios y las canciones que suben.
- Seguimiento de las canciones favoritas de cada usuario.
- Potencial escalabilidad para futuras mejoras como listas de reproducción o comentarios.

3.4. Pruebas (Unitarias y de Integración)

Pruebas Unitarias:

Estas pruebas se realizaron para verificar individualmente el comportamiento de cada funcionalidad crítica del sistema. Algunos ejemplos:

- Validación del registro de usuarios, comprobando que no se acepten correos duplicados ni contraseñas vacías.
- Encriptación y verificación de contraseñas con hashing (bcrypt).
- Comprobación del formulario de subida de canciones, asegurando que solo se acepten archivos multimedia válidos.
- Verificación de integridad en la carga de favoritos (que no se repita la misma canción más de una vez por usuario).
- Visualización dinámica de canciones en el dashboard según el usuario.

Todas las funciones se testearon desde el backend y el frontend, incluyendo la respuesta del servidor, la inserción en base de datos y la visualización en la interfaz del usuario.

Pruebas de integración:

Estas pruebas se enfocaron en el flujo completo de funcionalidades combinadas, asegurando que el sistema respondiera correctamente cuando se ejecutaban procesos en conjunto. Ejemplos:

- Registro de nuevo usuario → Cierre de sesión → Reingreso con las mismas credenciales.
- Subida de canción no válida (formato incorrecto) → Comprobación de mensaje de error.
- Eliminación de cuenta de usuario → Comprobación de eliminación de sus canciones asociadas.
- Marcado de canción como favorita → Eliminación de canción → Eliminación automática de esa entrada en favoritos.
- Subida simultánea de canciones por dos usuarios diferentes → Comprobación de integridad en base de datos.
- Cambio de rol a creador → Visualización del formulario exclusivo de subida de canciones.
- Acceso a perfil de otro usuario → Comprobación de visualización de sus canciones públicas.
- Carga de canción temporal desde carpeta del usuario → Reproducción sin almacenamiento en el servidor.
- Login con usuario no verificado → Acceso denegado hasta verificación.
- Inserción de canción con metadatos editados → Comprobación de correcta visualización del título y fecha.
-
- Comprobación del login de administradores y de la visibilidad de opciones exclusivas de gestión.

Se usaron herramientas como el navegador para verificar las rutas HTTP, así como el comportamiento de la base de datos mediante phpMyAdmin.

El sistema demostró estabilidad general y un correcto cumplimiento de las restricciones establecidas en el modelo de datos.

4. Implementación

4.1 Tecnologías/Herramientas empleadas

Frontend (Cliente)

Lenguajes y tecnologías:

- HTML5: Estructura semántica de la aplicación.
- CSS3: Estilos básicos y personalización visual.
- JavaScript (ES6+): Lógica del cliente, interacción y dinámicas de interfaz.
- Next.js (v15.3.2): Framework React con renderizado híbrido (SSR/CSR) utilizado como núcleo del frontend.

- Tailwind CSS: Framework de utilidades CSS para maquetación rápida y responsive.
- React (v19.0.0): Librería base para la construcción de interfaces de usuario dinámicas.
- ReactDOM (v19.0.0): Motor de renderizado de React.

Dependencias y librerías:

- axios: Comunicación HTTP entre cliente y servidor.
- framer-motion: Animaciones fluidas y transiciones en componentes.
- gsap: Biblioteca para animaciones avanzadas.
- js-cookie: Gestión de cookies en el navegador (autenticación y preferencias).
- lucide-react: Iconografía moderna y ligera integrada en React.
- music-metadata: Análisis de metadatos musicales (título, duración, artista) desde archivos subidos.

Backend (Servidor)

Tecnologías y frameworks:

- Node.js: Entorno de ejecución JavaScript en el servidor.
- Express.js (v5.1.0): Framework web para construir la API REST.
- MySQL: Sistema de gestión de base de datos relacional utilizado para el almacenamiento persistente.

Dependencias y paquetes:

- bcryptjs: Encriptación de contraseñas con hash seguro.
- cookie-parser: Parseo y gestión de cookies desde el lado del servidor.
- cors: Permisos de acceso entre dominios durante el desarrollo.
- dotenv: Gestión de variables de entorno para configuración segura.
- jsonwebtoken (JWT): Generación y validación de tokens para control de acceso.
- multer: Middleware para subir archivos de audio (almacenamiento temporal y persistente).
- mysql2: Cliente optimizado para conexión y consultas MySQL.
- nodemailer: Envío de correos electrónicos automatizados (por ejemplo, para verificación de cuenta).

Otros herramientas:

- phpMyAdmin: Gestión visual de la base de datos durante el desarrollo.
- Git: Control de versiones del código fuente.
- Visual Studio Code: Editor principal del entorno de desarrollo.

4.2 Descripción

El proyecto VibraSync ha sido desarrollado con un enfoque full stack moderno, aplicando un conjunto de tecnologías y herramientas que reflejan las mejores prácticas del desarrollo web

actual. La aplicación se estructura en dos grandes capas: el frontend (interfaz de usuario) y el backend (lógica de negocio y base de datos), conectadas mediante una API RESTful construida con Node.js y Express.

En el lado del frontend se ha optado por Next.js (v15.3.2) como framework principal. Esta decisión se debe a su capacidad para ofrecer renderizado híbrido (estático y del lado del servidor), lo que mejora notablemente el rendimiento inicial de carga, el SEO y la organización del código. Junto a Next.js se ha utilizado React 19, aprovechando su sistema de componentes funcionales y su nueva arquitectura concurrente, que mejora la fluidez de la interfaz.

Para la maquetación y el diseño visual, se ha incorporado Tailwind CSS, una biblioteca de utilidades CSS altamente optimizada para el desarrollo rápido de interfaces responsive y limpias. Esta elección elimina la necesidad de archivos CSS tradicionales y permite mantener los estilos directamente en el marcado JSX, haciendo que el código sea más predecible y mantenible.

En términos de interactividad avanzada y animaciones, el proyecto integra Framer Motion y GSAP. Framer Motion se encarga de las transiciones de componentes, efectos al hacer scroll y animaciones condicionales dentro de React, mientras que GSAP se ha empleado para efectos personalizados más complejos o continuos. Ambas librerías trabajan de forma eficiente sin bloquear el renderizado ni sobrecargar el DOM.

Se ha hecho uso de Axios para la gestión de peticiones HTTP entre frontend y backend, ya que proporciona una API sencilla y flexible para interceptores, manejo de errores y configuración global. Para la gestión de cookies (por ejemplo, el token JWT o la preferencia de sesión del usuario), se ha usado js-cookie, lo cual facilita tanto el acceso como la manipulación de cookies desde el navegador de forma segura.

Para el reconocimiento de metadatos musicales (duración, nombre del artista, etc.), se integró la dependencia music-metadata. Esta librería permite extraer información útil de los archivos de audio cargados directamente desde el navegador o al subirlos al backend, sin necesidad de utilizar herramientas externas.

En cuanto al backend, se ha utilizado Node.js con Express (v5.1.0) para construir una API modular, ligera y escalable. Express permite definir rutas, middlewares y controladores de forma limpia y sencilla, adaptándose a las necesidades tanto del usuario final como del panel de administración. Se ha seguido una estructura MVC lógica, separando los controladores, rutas y lógica de base de datos para facilitar el mantenimiento.

Para la autenticación, se han empleado jsonwebtoken para generar y verificar tokens JWT, y bcryptjs para el hashing seguro de contraseñas antes de almacenarlas en la base de datos. Esto garantiza un sistema de inicio de sesión seguro y conforme a las buenas prácticas.

El manejo de archivos ha sido resuelto mediante la dependencia multer. Esta herramienta se ha configurado para aceptar archivos de audio, almacenarlos temporalmente en carpetas específicas del servidor, y generar rutas accesibles para su posterior reproducción. Se ha configurado de forma que se limita el tamaño y tipo de archivo aceptado, previniendo usos maliciosos.

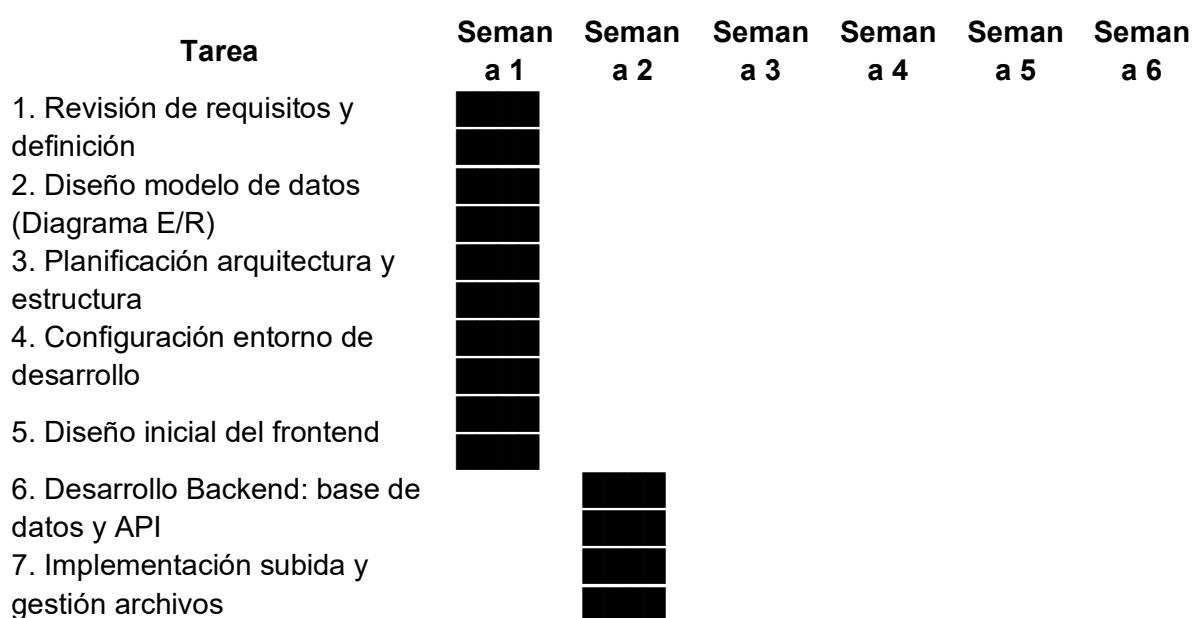
La base de datos utilizada es MySQL (v10.4.32-MariaDB), accedida mediante el cliente mysql2, que permite ejecutar consultas SQL de forma asíncrona y segura. Las tablas han sido modeladas con claves primarias, relaciones foráneas y restricciones de integridad referencial, permitiendo una gestión robusta de usuarios, canciones y favoritos.

Otras herramientas utilizadas incluyen dotenv para la gestión de variables de entorno sensibles (como contraseñas y claves secretas), cors para habilitar peticiones entre dominios en desarrollo y producción, y cookie-parser para facilitar la lectura de cookies desde Express.

En resumen, este proyecto representa una arquitectura full stack basada en tecnologías modernas como React, Next.js y Node.js, con un enfoque en modularidad, seguridad, rendimiento y mantenibilidad. El uso cuidadoso de dependencias específicas ha permitido resolver problemas concretos, mejorar la experiencia de desarrollo y garantizar un producto final optimizado.

5. Planificación y presupuesto

5.1. Diagrama Gantt de planificación de tareas



8. Desarrollo API canciones y favoritos
9. Pruebas básicas API y corrección errores
10. Integración frontend-backend básica
11. Desarrollo frontend: reproductor música
12. Implementación gestión sesiones y cookies
13. Diseño interfaz subida carpetas locales
14. Integración animaciones (Framer Motion)
15. Implementación visual favoritos
16. Optimización rutas archivos y audio
17. Subida música distribuidoras y validación
18. Desarrollo panel administración
19. Integración seguridad adicional
20. Pruebas funcionales backend y frontend
21. Pruebas unitarias backend
22. Pruebas unitarias frontend
23. Pruebas integración flujo completo
24. Ajustes y corrección de bugs
25. Optimización rendimiento y seguridad
26. Documentación técnica y manual usuario
27. Preparación entorno despliegue y deployment
28. Presentación y revisión final
29. Implementación feedback y ajustes finales
30. Entrega oficial y backup final



Descripción del Diagrama de Gantt (6 semanas)

Este diagrama de Gantt refleja la planificación detallada del proyecto web musical desarrollado a lo largo de seis semanas. Cada fila representa una tarea clave del desarrollo, y cada columna corresponde a una semana del cronograma. Los bloques “██████████” indican las semanas específicas en las que se ejecuta cada actividad.

Resumen por fases:

Semana 1 (Inicio y base del proyecto)

- Se llevan a cabo las tareas iniciales del proyecto: definición de requisitos, planificación de la estructura y diseño del modelo de datos (E/R).
- Se configura el entorno de desarrollo, tanto para el frontend como el backend.
- Se comienza el diseño básico del frontend, asegurando una base sólida para las interfaces futuras.

Semana 2 (Desarrollo del backend)

- Se construye la base de datos y se desarrolla la API principal, incluyendo la subida y gestión de archivos.
- Se implementan funcionalidades esenciales como la gestión de canciones y favoritos.
- Se realizan las primeras pruebas y ajustes de la API.
- Se comienza la integración básica entre frontend y backend.

Semana 3 (Desarrollo del frontend)

- El foco se traslada al desarrollo de interfaces más específicas como el reproductor de música, la gestión de sesiones y cookies, y la interfaz de subida de carpetas locales.
- Se añade dinamismo con animaciones usando Framer Motion.
- También se trabaja en la interfaz de favoritos y en aspectos de usabilidad visual.

Semana 4 (Funcionalidades avanzadas)

- Se abordan tareas más complejas como la integración de música desde distribuidoras, la creación del panel de administración y la implementación de medidas de seguridad adicionales.
- Se realizan pruebas funcionales del sistema en su conjunto.

Semana 5 (Pruebas y optimización)

- Se realizan pruebas unitarias tanto del backend como del frontend, así como pruebas de integración completas del flujo de la app.
- Se corrigen errores detectados y se optimizan aspectos clave de rendimiento, seguridad y rutas de acceso a los archivos.

Semana 6 (Cierre y entrega)

- Esta semana se dedica a la documentación técnica y del usuario, la preparación del entorno de despliegue, la entrega oficial y la presentación del proyecto.
- Se recogen los últimos ajustes basados en feedback, se crea un backup completo y se finaliza el proceso de desarrollo.

5.2. Presupuesto:

Fase del proyecto	Tareas incluidas	Horas estimadas	Tarifa/ hora (€)	Coste total (€)
1. Análisis y definición de requisitos	Definición de funciones, estructura, base de datos, usuarios	10 horas	14 €/h ora	140 €
2. Diseño de la aplicación	Maquetación, estructura visual, interfaz responsive (Figma/HTML/CSS)	14 horas	14 €/h ora	196 €
3. Desarrollo del backend	API REST, autenticación, subida de archivos, conexión a MySQL	24 horas	14 €/h ora	336 €
4. Desarrollo del frontend	Next.js, integración con API, visualización dinámica, integración multimedia	28 horas	14 €/h ora	392 €
5. Integración y conexión Front-Back	Pruebas de integración, ajustes de rutas, comprobación de sesiones y flujos	10 horas	14 €/h ora	140 €
6. Sistema de subida de música local	Carga de carpetas locales, sin persistencia, reproductor con rutas temporales	12 horas	14 €/h ora	168 €
7. Pruebas (unitarias e integración)	Testing de endpoints, carga de datos, accesos por rol	10 horas	14 €/h ora	140 €
8. Documentación y entrega final	Redacción de memoria, capturas, pruebas funcionales y explicación técnica	8 horas	14 €/h ora	112 €
■ Total estimado del proyecto		116 horas	14 €/h ora	1.624 €

El servidor por el momento me saldría gratis porque estoy usando una capa gratuita de oracle

6. Futuras mejoras y ampliaciones

Aunque la plataforma desarrollada ofrece una experiencia completa en términos de reproducción musical, subida de canciones y gestión de usuarios, se identifican varias líneas de mejora y expansión que podrían implementarse en futuras versiones para aumentar el valor funcional del sistema y su escalabilidad:

- **Personalización avanzada del interfaz del usuario**
Se implementará un sistema de personalización visual mediante temas o skins, permitiendo que cada usuario configure colores, tipo de interfaz, modo claro/oscuro, e incluso layouts preferidos. Esto se podrá gestionar mediante persistencia en cookies o base de datos para mantener el diseño personalizado entre sesiones.
- **Mayor dinamismo visual en el reproductor musical**
Se mejorará el componente visual de la reproducción musical con animaciones reactivas (ondas de audio, barras dinámicas, efectos de transición entre canciones) utilizando librerías como Framer Motion o GSAP para ofrecer una experiencia estética más inmersiva y moderna.
- **Reproductor flotante accesible desde cualquier parte de la web**
En versiones posteriores se añadirá un reproductor musical persistente y flotante que acompañará al usuario mientras navega por otras secciones de la aplicación. Esto permitirá seguir escuchando música sin necesidad de permanecer en la página principal de escucha, similar a la experiencia de plataformas profesionales como Spotify o SoundCloud.
- **Perfiles públicos para creadores de contenido**
Se implementará una sección visible para cada creador donde se muestre su información personal (nombre artístico, descripción, redes sociales), así como todas las canciones que ha subido y sus estadísticas (reproducciones, seguidores, etc.). Esto fomentará la creación de marca personal dentro de la plataforma.
- **Almacenamiento en la nube y streaming escalable**
Se plantea migrar el sistema actual de almacenamiento local a uno basado en servicios cloud como AWS S3 o Google Cloud, permitiendo mayor capacidad, disponibilidad y distribución geográfica del contenido, junto con la implementación de streaming adaptativo.
- **Playlists personalizadas y colaborativas**
Se ampliará la funcionalidad del sistema para permitir la creación de listas de reproducción por parte del usuario, así como compartirlas o editarlas en colaboración con otros, mejorando la organización y descubrimiento musical.
- **Integración con APIs externas de metadatos musicales**
Se integrarán servicios como Last.fm, MusicBrainz o Deezer API para completar

automáticamente los metadatos de las canciones subidas, como género, portada, duración exacta, artista, etc.

- **Sistema de seguidores y notificaciones**

Se incorporará una funcionalidad para seguir artistas, recibir notificaciones cuando suban nuevo contenido y mantenerse al tanto de novedades. Esto fomenta una comunidad más activa y conectada.

- **Aplicación como Progressive Web App (PWA)**

Se adaptará la aplicación para poder ser instalada como una PWA en dispositivos móviles y ordenadores, permitiendo reproducción offline de música almacenada localmente mediante caché o IndexedDB.

- **Panel de administración mejorado**

Se ampliará el actual panel de administración para incluir control de estadísticas generales, acceso a logs de usuario, gestión de canciones reportadas, y análisis de uso por segmentos.

- **Moderación y control de contenido subido**

Se añadirá un sistema de revisión automática/manual de las canciones subidas para asegurar el cumplimiento de las normas de uso, derechos de autor y evitar contenido no deseado.

- **Multiformato de audio y conversión automática**

El sistema admitirá más tipos de archivo (como FLAC, OGG o AAC) y realizará la conversión automática a MP3 para reproducción estándar mediante herramientas como FFmpeg.

- **Multilenguaje y localización**

Se preparará el sistema para soportar varios idiomas, permitiendo a los usuarios seleccionar el idioma desde el frontend. Esto se gestionará mediante librerías como i18next o Next-i18next.

7. Conclusiones sobre el proyecto

El desarrollo de este proyecto ha supuesto una experiencia integral en la construcción de una aplicación web full-stack, abordando tanto la parte técnica como organizativa. Desde la concepción de la idea hasta su implementación final, se han puesto en práctica conocimientos adquiridos a lo largo del ciclo formativo, combinando tecnologías modernas de frontend, backend y bases de datos, y enfrentando retos reales de desarrollo.

A nivel técnico, la utilización de herramientas como Next.js y React ha permitido crear una interfaz dinámica y moderna, mientras que la integración de Tailwind CSS ha facilitado una maquetación responsive y profesional. En el lado del servidor, Express.js y Node.js han ofrecido una base sólida para la creación de APIs seguras y escalables, junto a una base de datos MySQL estructurada y optimizada para el almacenamiento de usuarios, canciones y favoritos. La gestión de sesiones, subida de archivos, protección de rutas y tratamiento de datos han sido implementadas con dependencias especializadas como multer, jsonwebtoken, bcryptjs y dotenv.

Durante el proceso se han llevado a cabo pruebas unitarias y de integración que permitieron identificar errores, mejorar el rendimiento del sistema y garantizar la cohesión entre módulos. La planificación del trabajo por semanas, junto con el diagrama de Gantt y el presupuesto detallado, ha sido clave para organizar correctamente las fases del desarrollo, y cumplir con los objetivos dentro del tiempo estimado.

Entre los principales logros destacan:

- El funcionamiento completo del sistema de autenticación y subida de canciones.
- La reproducción directa de archivos sin almacenarlos en el servidor.
- La implementación de favoritos y perfil de usuario funcional.
- La gestión segura de datos con roles diferenciados (usuario, creador y administrador).

Asimismo, se identificaron posibles líneas de mejora orientadas a optimizar la experiencia del usuario, dotar a la app de una interfaz más personalizable, incorporar perfiles públicos de artistas y escalar el sistema con almacenamiento cloud o reproducción offline. Esto deja abierta la puerta a seguir evolucionando el proyecto más allá del ámbito académico.

En conclusión, este proyecto ha sido una oportunidad para demostrar autonomía técnica, pensamiento crítico, resolución de problemas reales y capacidad para desarrollar una aplicación web moderna que cubra una necesidad concreta, como lo es el acceso gratuito, personalizado y funcional a la música digital. El resultado final refleja no solo conocimientos técnicos, sino también el compromiso y la planificación necesaria para llevar una idea desde su concepto hasta su implementación funcional y presentable.

8. Índice de figuras (imágenes / tablas)

Tablas:

- Tabla 1. Diagrama de planificación de tareas (Gantt) – pág. 16-17
- Tabla 2. Presupuesto detallado del proyecto – pág. 19

Figuras:

- Figura 1. Sección descubre – pág. 5

- Figura 2. Sección biblioteca sin cargar archivos – pág. 5
- Figura 3. Sección biblioteca con archivos cargados – pág. 5
- Figura 4. Reproductor– pág. 6
- Figura 5. Sección de temas – pág. 6
- Figura 6. Selector de color – pág. 6
- Figura 7. Pestaña de contacto (Sube Música)– pág. 6
- Figura 8. Panel creador para administrar canciones– pág. 6
- Figura 9. Imagen de la vista de escritorio de la pestaña Escuchar – pág. 7
- Figura 10. Vista de escritorio de la pestaña Escuchar – pág. 7
- Figura 11. Vista de móvil de la pestaña Escuchar – pág. 7
- Figura 12. Vista del login – pág. 7
- Figura 13. Vista del registro – pág. 7
- Figura 14. Vista del panel de control personal de usuario – pág. 7
- Figura 15. Vista del panel de control global de usuarios (administración) – pág. 7

9. Siglas y acrónimos

- **BBDD (Base de Datos):** Sistema organizado para almacenar, gestionar y recuperar información de forma eficiente.
- **CSS (Cascading Style Sheets):** Lenguaje utilizado para definir el estilo visual de los documentos HTML.
- **E/R (Entidad/Relación):** Modelo utilizado para representar gráficamente la estructura de una base de datos.
- **GIT (Global Information Tracker):** Sistema de control de versiones distribuido utilizado para el seguimiento de cambios en el código fuente.
- **HTML (HyperText Markup Language):** Lenguaje de marcado utilizado para estructurar contenidos en la web.
- **JS (JavaScript):** Lenguaje de programación interpretado ampliamente utilizado para añadir interactividad a las páginas web.
- **JWT (JSON Web Token):** Estándar abierto que permite transmitir datos de forma segura entre partes como un objeto JSON.
- **MySQL:** Sistema de gestión de bases de datos relacional basado en SQL.
- **Node.js:** Entorno de ejecución para JavaScript del lado del servidor.
- **SQL (Structured Query Language):** Lenguaje estructurado para gestionar y manipular bases de datos relacionales.
- **UI (User Interface):** Interfaz de Usuario. Hace referencia a la parte visible y con la que el usuario interactúa dentro de una aplicación.
- **UX (User Experience):** Experiencia de Usuario. Hace referencia al conjunto de factores que determinan el grado de satisfacción del usuario con un sistema.

- **DOM (Document Object Model)**: Modelo que representa la estructura de un documento HTML o XML como un conjunto de objetos.
- **CRUD (Create, Read, Update, Delete)**: Operaciones básicas que se pueden realizar sobre los datos en una base de datos.

10. Bibliografía

- **Next.js**
URL: <https://nextjs.org/docs>
- **React**
URL: <https://react.dev/>
- **Tailwind CSS**
URL: <https://tailwindcss.com/docs>
- **Node.js**
URL: <https://nodejs.org/en/docs>
- **Framer Motion**
URL: <https://www.framer.com/motion/>
- **music-metadata**
URL: <https://www.npmjs.com/package/music-metadata>
- **js-cookie**
URL: <https://www.npmjs.com/package/js-cookie>