



REPRODUCTOR MUSICAL

QNK Music

Desarrollo de Aplicaciones Web
2ºDAW 2024/2025

Diego de las Heras Abad

TODOS LOS SCRIPTS DE CREACION SE ENCUENTRAN EN EL INDEX.PHP

ENLACE GIT HUB: <https://github.com/dhabad19/QNKMUSIC.git>

URL de la WEB: <https://mediumseagreen-snake-348502.hostingersite.com/>

Registrarse o usar Usuario: profesor@qnkmusic.com – 123

Datos ya creados adminstradr@gmail.com@gmail.com-1234

MEMORIA: <https://github.com/dhabad19/QNKMUSIC.git>

MANUAL DE USUARIO:

<https://github.com/dhabad19/QNKMUSIC/blob/main/MANUAL%20USUARIO.pptx>

INDICE

1.DESCRIPCIÓN EL PROYECTO.

1.1. Características generales

1.2. Objetivos y alcance

2.ANÁLISIS DEL SECTOR/MERCADO.

2.1. Prospectiva del Título en el sector.

2.2. Evolución y tendencia del sector.

2.3. Normativa y documentación técnica específica

3.PLAN DE EJECUCIÓN.

3.1. Diagrama/cronograma de flujo de procesos (Diagrama de Gantt).

3.2. Proceso de desarrollo software.

3.2.1. Fase de análisis.

3.2.1.1. Tipos de usuario

3.2.1.2. Descripción de requisitos

3.2.1.3. Casos de Uso

3.2.1.4. Guía de estilo

3.2.1.5. Prototipo del sitio web

3.2.1.6. Mapa de navegación

3.2.2. Fase de desarrollo.

3.2.2.1. Base de datos

3.2.2.1.1. Análisis de requisitos de datos de la aplicación

a. Identificación de entidades e interrelaciones entre entidades

identificación de atributos de cada entidad

3.2.2.1.2Diseño lógico de datos

3.2.2.1.3 Paso del modelo lógico (E/R) al modelo relacional (tablas)

- a. Identificación de atributos de cada tabla, sus tipos y tamaños
- b. Identificación de claves principales y claves candidatas
- c. Identificación de las reglas de integridad y seguridad de la base de datos
- d. Modelo relacional (tablas)

3.2.2.1.4 Aplicación de reglas de normalización al modelo relacional

3.2.2.1.5 Tipos de datos para el sistema gestor seleccionado

3.2.2.1.6 Scripts de creación de tablas e inserciones iniciales

3.2.2.2. Servidor

3.2.2.2.1. Lista de funciones en pH

3.2.2.3. Cliente

3.2.2.3.1. Diseño de la interfaz

3.2.2.3.2. Accesibilidad

3.2.2.3.3. Usabilidad

3.2.2.3.4. Desarrollo web entorno cliente

a. Formularios y su validación

manejo y gestión de eventos: Teclado, ratón y estados de la ventana

cogestión y almacenamiento de datos e información en el cliente.

d. Modificación del DOM

e. Animaciones, efectos, cambios dinámicos de para dinamizar la parte visible al cliente
comunicación AJAX

comunicación asíncrona con el servidor

3.2.3. Fase de despliegue.

3.2.3.1. Despliegue utilizando un hosting o

3.2.3.2. Despliegue en un equipo local propio

3.3. Seguimiento y control de incidencias.

3.4. Indicadores de calidad de procesos.

4.RECURSOS MATERIALES.

4.1. Inventario, valorado, de medios

4.2. Presupuesto económico

5.RECURSOS HUMANOS.

5.1. Organización

5.2. Contratación

5.3. Prevención de riesgos laborales

6.VIABILIDAD TÉCNICA.

6.1. Estudio de viabilidad técnica

7.VIABILIDAD ECONÓMICO-FINANCIERA.

7.1. Inversiones y gastos

7.2. Financiación

7.3. Viabilidad económico-financiera

8.CONCLUSIÓN

1. DESCRIPCIÓN DEL PROYECTO

1.1. CARACTERÍSTICAS GENERALES

QNK Música nace de una idea, ¿y si pudiéramos crear una aplicación web que ofreciera una experiencia musical completa, sencilla y cercana? Así comenzó el desarrollo de este proyecto. No se trata de cumplir los requisitos académicos, sino de hacer algo real, funcional, y que invite al usuario a quedarse.

Se trata de una plataforma online que permite buscar, reproducir y organizar canciones conectándose directamente a la API pública de Spotify. Y aunque suene ambicioso, lo bonito es que lo hace de forma limpia, clara y con una estructura profesional. El usuario puede crear playlists, marcar canciones favoritas, consultar su historial o gestionar su perfil, todo ello desde una interfaz diseñada a medida, con personalidad propia.

Detrás hay tecnologías conocidas: HTML, CSS, JavaScript, PHP y MySQL. Nada fuera de lo común, pero combinadas con lógica, e intención. QNK Music no pretende reinventar el streaming musical. Lo que busca es demostrar que se puede diseñar una aplicación web completa, modular, segura y lista para el mundo real

1.2. Objetivos y alcance

La idea es poner a prueba todos los conocimientos adquiridos durante el ciclo y además demostrar criterio, madurez técnica y un estilo propio a la hora de desarrollar soluciones web.

El objetivo general es sencillo de decir, pero complejo de ejecutar: crear una aplicación web robusta y atractiva que se conecte con Spotify, y permita al usuario disfrutar de una experiencia personalizada y fluida.

A partir de ahí, se definen una serie de objetivos concretos:

- Implementar un sistema de búsqueda
- Ofrecer la posibilidad de marcar canciones como favoritas y acceder a ellas rápidamente.
- Almacenar un historial personalizado para cada sesión.
- Gestionar el acceso mediante formularios de registro y login.
- Permitir la edición de perfil incluyendo imagen personalizada.

El alcance del proyecto cubre toda la parte web: tanto el backend como el frontend, el diseño de base de datos, las conexiones con la API, el despliegue en servidor remoto y la validación del funcionamiento completo.

No se incluyen funciones avanzadas como login con Spotify o reproducción completa de audio, ya que el acceso a la API utilizada es público y limitado. Tampoco se ha desarrollado aplicación móvil. Todo está enfocado a crear una aplicación real, con lógica profesional, desde cero.

2. ANÁLISIS DEL SECTOR / MERCADO

2.1. Prospectiva del Título en el sector

El perfil profesional vinculado al Desarrollo de Aplicaciones Web se encuentra en uno de los momentos de mayor demanda de la última década. La transformación digital ha impulsado la necesidad de contar con personas capaces de diseñar, implementar y mantener soluciones. Y no solo en grandes empresas tecnológicas: también en startups, negocios locales e incluso sectores más tradicionales que buscan digitalizarse.

El título de Técnico Superior en Desarrollo de Aplicaciones Web abre muchas puertas. Desde desarrollador frontend o backend, hasta roles híbridos como fullstack, pasando por integrador de APIs, analista web o funciones de QA. La variedad técnica que aporta esta formación es una de sus principales fortalezas. Por eso es tan valorada en el mercado laboral, tanto a nivel nacional como en el extranjero.

2.2. Evolución y tendencia del sector

Durante los últimos años, el desarrollo web ha pasado de ser una habilidad puntual a convertirse en una necesidad. Las aplicaciones web han dejado de ser páginas estáticas para transformarse en plataformas interactivas, personalizadas y conectadas entre sí.

El auge del contenido en streaming música, vídeo, eventos en directo ha impulsado la evolución de estas tecnologías. Hoy en día, una web no solo debe verse bien, debe responder rápido, adaptarse a cualquier dispositivo y comunicarse en tiempo real con servicios externos. En este contexto, nacen proyectos como **QNK Music**, que aprovechan las APIs abiertas (como la de Spotify) para construir experiencias y funcionales completas.

La tendencia es clara: mayor personalización, mejor integración entre plataformas y más soluciones accesibles desde cualquier parte del mundo. QNK Music encaja perfectamente en esa dirección, y lo hace desde una perspectiva académica, pero con mentalidad profesional.

2.3. Normativa y documentación técnica específica

Desde el inicio del proyecto, se ha mantenido un compromiso con el cumplimiento normativo y las buenas prácticas. A nivel legal, se han respetado las directrices generales del Reglamento General de Protección de Datos (RGPD): los datos personales se almacenan únicamente en el servidor privado, no se comparten con terceros y se protegen mediante contraseñas cifradas.

En el plano técnico, se ha seguido la especificación oficial del W3C para HTML5 y CSS3, además de aplicar principios de accesibilidad, orden semántico y separación entre estructura, diseño y comportamiento. El backend se ha desarrollado en PHP con conexiones seguras a base de datos MySQL, cuidando la integridad de los datos a través de claves primarias y foráneas bien definidas.

Por otro lado, la integración con la API de Spotify se ha realizado respetando completamente su documentación oficial. No se ha almacenado información sensible ni se ha utilizado OAuth avanzado, dado que el proyecto hace uso del endpoint de acceso libre (Client Credentials Flow). Esto garantiza una conexión eficiente, estable y segura, sin comprometer la privacidad del usuario ni infringir las políticas de uso.

3. Plan de ejecución

3.1. Diagrama/cronograma de flujo de procesos

El desarrollo de QNK Music se ha llevado a cabo siguiendo una planificación estructurada por fases, organizadas de forma cronológica. Esto ha permitido visualizar claramente las tareas implicadas en cada etapa y las dependencias entre ellas. La gestión por bloques ha sido clave para mantener el control sobre el avance del proyecto y garantizar que cada parte se desarrollase en el orden adecuado.

A continuación, se detallan las principales fases y tareas contempladas:

Fase 1 – Preparación inicial

- Definición de la idea y objetivos del proyecto
- Revisión de requisitos técnicos y funcionales
- Estudio de la API pública de Spotify
- Elección del stack tecnológico
- Boceto preliminar de la interfaz

Fase 2 – Diseño y análisis

- Identificación de entidades y relaciones (modelo E/R)
- Diseño lógico de la base de datos
- Análisis de tipos de usuarios y casos de uso
- Elaboración del mapa de navegación

- Creación del prototipo de las pantallas principales

Fase 3 – Desarrollo backend (servidor)

- Conexión a base de datos y archivo de credenciales
- Implementación del sistema de login, registro y sesiones
- Gestión de usuarios: perfil, edición, imagen
- Estructura de base de datos relacional
- Scripts de creación y prueba de tablas
- Control de seguridad y validación de accesos

Fase 4 – Desarrollo frontend (cliente)

- Maquetación HTML5 + CSS3 responsiva y con estilo personalizado
- Desarrollo de lógica JS: formularios, reproductor, favoritos, AJAX
- Búsqueda dinámica con conexión a la API de Spotify
- Gestión del DOM, validación y eventos
- Creación de páginas de playlists, favoritos, historial...

Fase 5 – Integraciones externas

- Conexión segura con la API pública de Spotify
- Obtención y renovación de tokens
- Inserción y tratamiento de los datos recibidos
- Implementación del reproductor embebido

Fase 6 – Despliegue y pruebas

- Subida de archivos a entorno real (Hostinger)
- Adaptación de credenciales y configuración del servidor
- Pruebas funcionales completas en modo incognito y por usuario
- Verificación de sesiones, enlaces rotos, y reproducción
- Comprobación de compatibilidad y corrección de errores

	A	B	C
1	Tarea	Inicio	Fin
2	Análisis de requisitos	2025-03-01	2025-03-04
3	Diseño de la base de datos	2025-03-05	2025-03-08
4	Diseño de la interfaz	2025-03-09	2025-03-12
5	Configuración del hosting y BD	2025-03-13	2025-03-15
6	Desarrollo del login y registro	2025-03-16	2025-03-20
7	Desarrollo página principal	2025-03-21	2025-03-24
8	Integración con API de Spotify	2025-03-25	2025-03-28
9	Funcionalidad de búsqueda	2025-03-29	2025-04-01
10	Gestión de playlists	2025-04-02	2025-04-05
11	Favoritos y reproducción	2025-04-06	2025-04-09
12	Historial de reproducción	2025-04-10	2025-04-12
13	Validación y usabilidad cliente	2025-04-13	2025-04-16
14	Pruebas funcionales	2025-04-17	2025-04-21
15	Despliegue final	2025-04-22	2025-04-24

3.2 Proceso de desarrollo del software

3.2.1.1 Tipos de usuario

En la aplicación **QNK Music** se puede interactúan con el sistema de dos formas. Aunque ambos comparten muchas funcionalidades comunes, su rol dentro de la plataforma está bien diferenciado desde el punto de vista lógico.

Usuario registrado

Este es el usuario principal y único que interactúa con la plataforma. Para acceder a la aplicación debe registrarse con su nombre, usuario visible, correo electrónico y contraseña. Una vez ha iniciado sesión, tiene acceso completo a las funcionalidades:

- Buscar canciones utilizando la API de Spotify.
- Reproducir canciones directamente desde un reproductor incrustado.
- Marcar canciones como favoritas.
- Crear, ver y editar playlists personales (públicas o privadas).
- Añadir canciones a playlists.
- Consultar su historial de reproducción.
- Editar su perfil (incluyendo la imagen).
- Cerrar sesión correctamente, manteniendo la privacidad de su cuenta.

Internamente, la base de datos asocia su ID de usuario con todas las acciones realizadas: favoritos, historial, playlists, etc., garantizando así una experiencia personalizada.

Usuario no autenticado

Aunque no es un tipo de usuario como tal, es importante mencionar el comportamiento del sistema ante usuarios no autenticados. Cualquier intento de acceder directamente a páginas como inicio.php, buscar.php o favoritos.php sin haber iniciado sesión, redirige automáticamente al login (`index.php`). Esto garantiza la seguridad de los datos y el aislamiento de sesiones entre usuarios.

3.2.1.2 Descripción de requisitos

La aplicación **QNK Music** ha sido diseñada con el objetivo de ofrecer una experiencia intuitiva y funcional para usuarios inspirándose en plataformas como Spotify. A continuación, se detallan los requisitos funcionales y no funcionales que se han tenido en cuenta en el desarrollo del proyecto.

Requisitos funcionales

Estos definen las funcionalidades clave que el sistema debe ofrecer al usuario para cumplir con sus objetivos:

1. Registro de usuarios
Permitir la creación de una cuenta mediante un formulario con validación, solicitando nombre completo, nombre visible, correo electrónico y contraseña.
2. Inicio de sesión (login)
Validar las credenciales del usuario y establecer una sesión segura para acceder al sistema.
3. Cierre de sesión (logout)
Finalizar la sesión del usuario de forma segura y prevenir el acceso mediante la caché del navegador.
4. Edición del perfil de usuario
Permitir al usuario modificar su nombre, nombre visible, correo electrónico, contraseña e imagen de perfil.
5. Reproducción de canciones
Integración con la API de Spotify para incrustar un reproductor y escuchar canciones.
6. Historial de última reproducción
Guardar automáticamente la última canción reproducida y mostrarla al acceder a la página de inicio.
7. Búsqueda en tiempo real
Utilizar la API de Spotify para buscar canciones a medida que el usuario escribe en la barra de búsqueda.
8. Favoritos
Posibilidad de marcar canciones como favoritas y visualizarlas posteriormente en la sección correspondiente.
9. Playlists
 - Crear nuevas listas de reproducción.

- Añadir canciones desde la búsqueda a cualquier playlist existente.
- Consultar el contenido de una playlist.
- Eliminar canciones de las playlists.

10. Seguridad básica

Validación de formularios, protección frente a accesos no autorizados y uso de hash en contraseñas.

Requisitos no funcionales

Estos requisitos garantizan la calidad, seguridad y mantenimiento del sistema:

1. **Facilidad de uso**
La interfaz debe ser sencilla, clara y accesible, tanto en escritorio como en dispositivos móviles.
2. **Estilo uniforme**
Uso de una paleta de colores consistente, tipografía clara y diseño atractivo.
3. **Velocidad y rendimiento**
Carga ágil de contenidos, evitando bloqueos mediante el uso de JavaScript asíncrono y `defer`.
4. **Mantenimiento**
Código modular, organizado por archivos y dividido entre cliente y servidor (PHP/JS), lo que facilita futuras ampliaciones o modificaciones.
5. **Compatibilidad**
Funciona correctamente en los principales navegadores (Chrome, Firefox o Edge).
6. **Uso de servicios externos**
Consumo de la API de Spotify para acceder a la base musical, lo que ahorra almacenamiento propio y garantiza un catálogo actualizado.

3.2.1.3 Casos de uso

Registro de usuario

- **Actor principal:** Usuario no registrado
- **Objetivo:** Crear una cuenta nueva en la aplicación
- **Precondiciones:** No haber iniciado sesión
- **Flujo principal:**
 1. El usuario accede al formulario de registro.
 2. Introduce su nombre, nombre visible, correo electrónico y contraseña.
 3. Repite la contraseña para confirmarla.
 4. Pulsa en "Registrarse".
 5. El sistema valida que el correo no esté registrado y guarda los datos en la base de datos.
 6. Se muestra un mensaje confirmando el registro.
- **Postcondición:** El usuario está registrado y puede iniciar sesión.

Inicio de sesión

- **Actor principal:** Usuario registrado
- **Objetivo:** Acceder al sistema
- **Precondiciones:** El usuario debe estar registrado
- **Flujo principal:**
 1. El usuario introduce su correo y contraseña en el formulario de login.
 2. El sistema comprueba que las credenciales coinciden.
 3. Si son correctas, se inicia la sesión y se redirige a inicio.php.
- **Postcondición:** El usuario accede al sistema y su sesión queda activa.

Cierre de sesión

- **Actor principal:** Usuario autenticado
- **Objetivo:** Finalizar su sesión de forma segura
- **Precondiciones:** El usuario debe haber iniciado sesión
- **Flujo principal:**
 1. El usuario pulsa en "Cerrar sesión" en el menú.
 2. Se destruye la sesión del usuario.
 3. Se redirige al index.php evitando el retorno por caché.
- **Postcondición:** El usuario ha salido del sistema y no puede volver sin iniciar sesión.

Buscar canciones

- **Actor principal:** Usuario autenticado
- **Objetivo:** Buscar canciones desde la barra de búsqueda
- **Flujo principal:**
 1. El usuario escribe el nombre de una canción, artista o álbum en la barra.
 2. Se realiza una petición en tiempo real a la API de Spotify.
 3. Se muestran los resultados dinámicamente.
 4. El usuario puede reproducir, marcar como favorita o añadir la canción a una playlist.
- **Postcondición:** La acción realizada (reproducir, guardar, añadir) se ejecuta correctamente.

Reproducir canción

- **Actor principal:** Usuario autenticado
- **Objetivo:** Escuchar una canción desde la aplicación
- **Flujo principal:**
 1. El usuario pulsa sobre una canción.
 2. El reproductor incrustado de Spotify se muestra con la canción seleccionada.
 3. Se guarda esta canción como la última reproducida en la base de datos.

- **Postcondición:** El reproductor se actualiza con la canción, y se registra su reproducción.

Marcar canción como favorita

- **Actor principal:** Usuario autenticado
- **Objetivo:** Guardar una canción como favorita
- **Flujo principal:**
 1. El usuario pulsa el icono de corazón en una canción.
 2. Se envía una petición al servidor (añadir_favoritos.php).
 3. La canción se guarda en la base de datos.
- **Postcondición:** La canción aparece en favoritos.php.

Añadir canción a playlist

- **Actor principal:** Usuario autenticado
- **Objetivo:** Guardar una canción en una playlist existente o nueva
- **Flujo principal:**
 1. El usuario pulsa el icono **+** en la canción.
 2. Se redirige a seleccionar_playlist.php.
 3. El usuario selecciona una playlist o crea una nueva.
 4. La canción se guarda asociada a esa playlist.
- **Postcondición:** La canción forma parte de la playlist.

Crear nueva playlist

- **Actor principal:** Usuario autenticado
- **Objetivo:** Crear una lista de reproducción
- **Flujo principal:**
 1. El usuario accede a playlist.php y pulsa "Crear nueva playlist".
 2. Introduce el nombre e imagen.
 3. Se guarda en la base de datos con el usuario asociado.
- **Postcondición:** La playlist aparece listada y puede ser editada o reproducida.

Consultar playlists

- **Actor principal:** Usuario autenticado
- **Objetivo:** Ver el contenido de una playlist
- **Flujo principal:**
 1. El usuario accede a playlist.php y pulsa sobre una lista.
 2. Se abre ver_playlist.php mostrando el contenido.
 3. Puede reproducir o eliminar canciones.
- **Postcondición:** El usuario puede interactuar con el contenido de la lista.

3.2.1.4 Guía de estilo

El diseño visual de **QNK Music** se ha basado en una inspiración clara y potente: los colores y la identidad de **BMW**, una marca asociada a la innovación, el dinamismo y la elegancia tecnológica. Con esta base, la estética del sitio busca transmitir un equilibrio entre sofisticación y energía, apostando por una interfaz moderna pero funcional, sobria, pero con carácter.

Paleta de colores

Inspirada en la gama cromática de BMW:

Color	Uso principal
#000000 (negro puro)	Fondo principal, cabeceras, base de interfaz
#ffffff (blanco puro)	Texto principal y contraste sobre fondo oscuro
#0066b2 (azul)	Elementos activos, botones principales, enlaces
#b3b3b3 (gris claro)	Bordes, cajas de texto, placeholders
#ff0000 (rojo)	Iconos de eliminar, acciones de advertencia

Estos colores se repiten de forma coherente en toda la aplicación, creando una identidad clara y profesional que remite al estilo sobrio y tecnológico.

Tipografía

- **Fuente principal:** Arial, sans-serif.
- **Estilo:** Clara, neutra, de fácil lectura.
- **Tamaños:**
 - Titulares: 24px - 32px
 - Subtítulos: 18px - 20px
 - Cuerpo de texto: 14px - 16px

No se han utilizado fuentes decorativas: el foco es la funcionalidad.

Componentes visuales

- **Cabecera (header):** Fija en la parte superior con el logotipo de QNK Music (estilo minimalista), seguido de un menú horizontal sobrio, pero bien espaciado.
- **Botones:**
 - Botón primario: azul
 - Botón peligro: rojo intenso

- Hover: transición suave con sombra o iluminación sutil
- **Contenedores:** Cajas con bordes suaves (border-radius: 6px), uso moderado de sombra (box-shadow) para crear profundidad.

Portadas e imágenes

- Las portadas de canciones se muestran en un formato cuadrado, con bordes redondeados y fondo gris claro si no hay imagen disponible.
- Las imágenes no solo decoran: informan y posicionan al usuario, como un panel digital que muestra información clara.

Navegación y usabilidad

- **Accesibilidad y claridad:** cada sección está perfectamente identificada, con títulos visibles, menús intuitivos y botones amplios para facilitar su uso desde cualquier dispositivo.
- **Jerarquía visual:** se utiliza tipografía, espaciado y color para guiar la vista sin sobrecargar.
- **Interacción:**
 - Feedback visual inmediato: al marcar favorito, reproducir una canción o añadir a playlist.
 - Uso de íconos universales

Diseño adaptable

Gracias a la integración de **Bootstrap 4.3**, la aplicación se adapta de manera adecuada a diferentes resoluciones, sin necesidad de un rediseño completo responsive.

3.2.1 Prototipado del sitio web

<https://www.figma.com/board/t7gW4OiAupJMXIKHci3PQb/FigJam-basics?node-id=0-1&t=ZsGBqPR1Fg8IVCAo-1>

El prototipo interactivo del sitio fue diseñado con inspiración en los colores corporativos de BMW (azul, blanco y gris). Este prototipo refleja el recorrido del usuario desde el inicio de sesión hasta la gestión de playlists y canciones favoritas.

Cada una de las pantallas cumple una función en el flujo de uso: desde el registro de nuevos usuarios, el acceso al panel principal, la búsqueda dinámica con conexión a la API de Spotify, la gestión de favoritos y playlists, hasta la edición del perfil. El diseño se ha centrado en la claridad, accesibilidad y facilidad de uso.

3.2.2.1. Base de datos

3.2.2.1.1 Análisis de requisitos de datos de la aplicación

a. Identificación de entidades e interrelaciones entre entidades

La base de datos ha sido diseñada para representar las relaciones entre usuarios, canciones, artistas, playlists y registros de reproducción, asegurando integridad y eficiencia.

Las entidades principales identificadas son:

Entidad	Propósito
Usuarios	Representa a cada usuario registrado en la plataforma.
Canciones	Contiene los metadatos de cada pista disponible.
Artistas	Representa a los creadores de las canciones.
Playlist	Almacena listas de canciones creadas por los usuarios.
Playlist_Canciones	Relación muchos a muchos entre Playlist y Canciones.
Favoritos_Canciones	Relación muchos a muchos entre Usuarios y sus canciones favoritas.
Historial_Reproduccion	Guarda un registro cronológico de todas las canciones reproducidas.
HistorialUltimaReproduccion	Tabla optimizada para acceder rápidamente a la última canción escuchada.

Relaciones principales:

- Un **usuario** puede tener varias **playlists**, y cada **playlist** puede contener muchas **canciones**.
- Una **canción** pertenece a un **artista**, y un **artista** puede tener múltiples **canciones**.
- Un **usuario** puede marcar múltiples **canciones** como favoritas.
- El **historial de reproducción** relaciona usuarios con canciones en orden cronológico.

- Se mantiene una tabla adicional para acceder de forma instantánea a la **última canción reproducida** por cada usuario.

3.2.2.1.1.b Identificación de atributos de cada entidad

Aquí se desglosan todos los atributos que conforman cada tabla, con su propósito dentro de la lógica de la aplicación:

Usuarios

- IdUsuario (*PK*): Identificador único de cada usuario.
- Nombre: Nombre y apellidos del usuario.
- Usuario: Alias visible en la aplicación (p. ej. @diego).
- Email: Correo electrónico de registro.
- Imagen: Ruta de la imagen de perfil.
- Contraseña: Contraseña encriptada.

Canciones

- IdCancion (*PK*): ID único de la canción.
- Título: Nombre de la canción.
- Portada: URL de la portada del álbum.
- URL: Enlace a la canción en Spotify.
- IdArtista (*FK*): Artista asociado.

Artistas

- IdArtista (*PK*): Identificador del artista.
- NombreArtista: Nombre del artista o grupo.

Playlist

- IdPlaylist (*PK*): Identificador único de la playlist.
- NombrePlaylist: Nombre asignado por el usuario.
- ImagenPlaylist: Portada personalizada.
- IdUsuario (*FK*): Usuario que la creó.

Playlist_Canciones

(relación N:M entre Playlist y Canciones)

- IdPlaylist (*FK*): ID de la playlist.
- IdCancion (*FK*): ID de la canción.

Favoritos_Canciones

(relación N:M entre Usuarios y Canciones)

- IdUsuario (*FK*): ID del usuario.
- IdCancion (*FK*): ID de la canción.

Historial_Reproduccion

- IdHistorial (*PK*): Clave primaria autoincremental.
- IdUsuario (*FK*): Usuario que la escuchó.
- IdCancion (*FK*): Canción reproducida.
- FechaReproduccion: Timestamp del momento exacto.

HistorialUltimaReproduccion

(registro auxiliar optimizado)

- IdUsuario (*PK/FK*): Usuario correspondiente.
- UltimoTitulo: Título de la última canción.
- UltimoArtista: Artista de esa canción.
- UltimaPortada: Imagen de portada.
- UltimaURL: URL directa a la canción.
- FechaActualizacion: Última vez que se modificó este registro.

3.2.2.1.2. Diseño lógico de datos. Realización del diseño lógico de datos, utilizando herramientas gráficas con el modelo Entidad / Relación.

Adjunto fichero .dia

3.2.2.1.3. Paso del modelo lógico (E/R) al modelo relacional (tablas)

A continuación, se presenta el modelo relacional resultante de la transformación del modelo entidad-relación previo.

Identificación de atributos de cada tabla, sus tipos y tamaños

Cada tabla incluye sus atributos perfectamente definidos, con tipos de datos y tamaños adecuados según su función en la aplicación. Por ejemplo, los campos de texto como Titulo, NombreArtista o Email están definidos como VARCHAR con tamaños entre 50 y 255 caracteres. Las claves primarias son enteros (INT(11)), y las fechas de reproducción se manejan mediante el tipo TIMESTAMP.

Identificación de claves principales y claves candidatas

Las claves primarias se identifican claramente en todas las tablas, marcadas mediante el icono de llave. Además, existen claves candidatas como el campo Usuario o Email en la tabla Usuarios, que podrían utilizarse como identificadores alternativos. Todas las claves foráneas están correctamente vinculadas con las tablas a las que hacen referencia.

Reglas de integridad y seguridad

La integridad referencial se asegura mediante el uso de claves foráneas y restricciones de tipo ON DELETE / ON UPDATE (configuradas en la base de datos). Por otra parte, la seguridad se garantiza en la parte de servidor: los datos sensibles como la contraseña del usuario se almacenan cifrados (mediante hash en PHP), y los accesos a la base de datos están protegidos mediante sesiones de usuario.

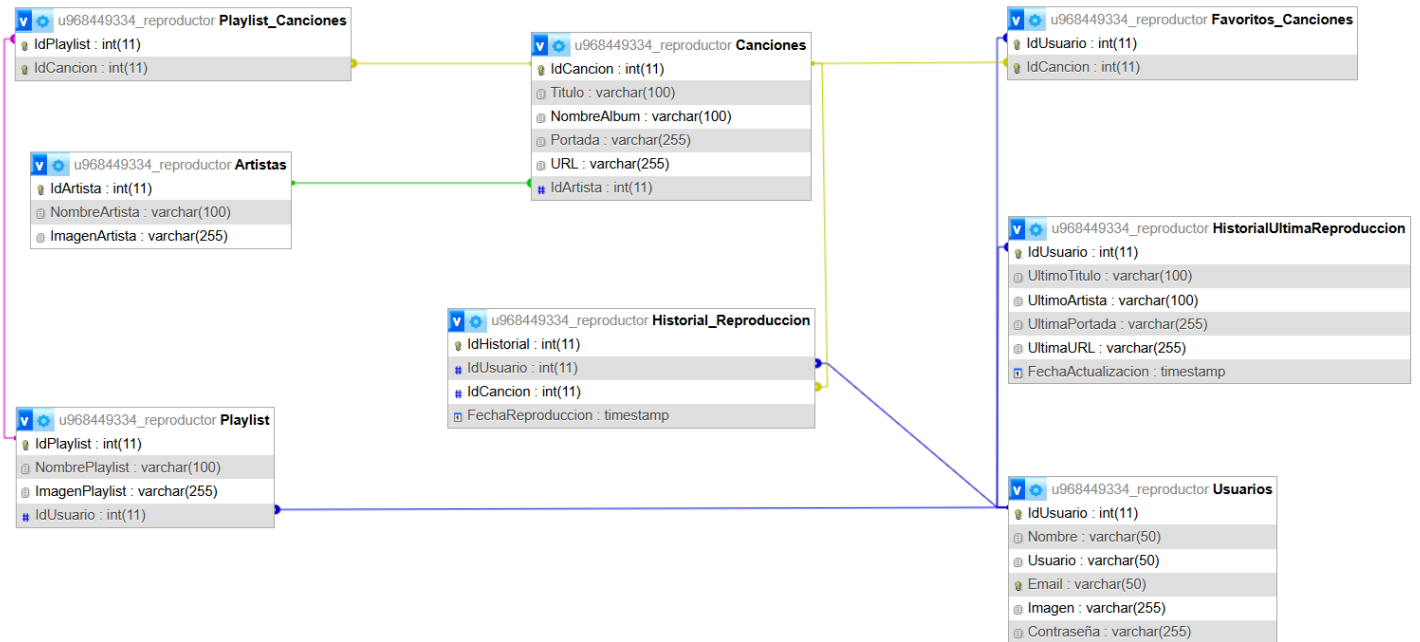
Modelo relacional (tablas)

El modelo se compone de 9 tablas:

- Usuarios: gestiona los datos de los usuarios registrados.
- Artistas: almacena información básica de artistas.
- Canciones: contiene datos de cada canción, incluyendo metadatos y enlace a artista.
- Playlist: recoge las listas creadas por cada usuario.
- Playlist_Canciones: tabla intermedia para modelar la relación N:M entre playlists y canciones.
- Favoritos_Canciones: tabla intermedia para las canciones marcadas como favoritas.
- Historial_Reproduccion: registra el historial completo de reproducción de canciones por usuario.
- HistorialUltimaReproduccion: almacena solo la última canción reproducida por cada usuario para cargarla rápidamente en la interfaz.

- Usuarios también está relacionado con la tabla Playlist mediante una relación 1:N.

Este modelo es el que actualmente se encuentra en uso en la aplicación web QNK Music y refleja la lógica establecida.



3.2.2.1.4. Aplicación de reglas de normalización al modelo relacional. Consiste en aplicar las reglas de normalización que estimes conveniente al modelo relacional (tablas)

Con el objetivo de asegurar la integridad, coherencia y eficiencia del modelo relacional de QNK Music, se han aplicado distintas reglas de normalización sobre todas las tablas que forman la base de datos. A continuación, se detalla la aplicación de las tres primeras formas normales (1FN, 2FN y 3FN), que son las más habituales y suficientes en este tipo de sistemas.

Primera Forma Normal (1FN) – Eliminación de grupos repetitivos

Todas las tablas cumplen con la 1FN, ya que:

- Todos los atributos contienen valores atómicos, es decir, indivisibles.
- No existen campos multivaluados ni listas de elementos en un solo atributo (por ejemplo, cada canción tiene un único título, un único artista asociado, etc.).
- No se almacenan registros duplicados, ya que cada tabla tiene una clave primaria que garantiza la unicidad.

Ejemplo:

En Canciones, los campos Titulo, Portada, URL son todos atómicos y se refieren a una sola entidad.

Segunda Forma Normal (2FN) – Eliminación de dependencias parciales

La base de datos también cumple con la 2FN:

- Todas las tablas con claves primarias compuestas (Playlist_Canciones, Favoritos_Canciones, Historial_Reproduccion) contienen solo atributos que dependen completamente de la totalidad de la clave.
- En las demás tablas, como Usuarios, Artistas o Canciones, la clave primaria es simple (un solo atributo), por lo que automáticamente cumplen 2FN.

Ejemplo:

En Playlist_Canciones, la clave está formada por IdPlaylist y IdCancion, y no existen atributos que dependan solo de uno de esos campos.

Tercera Forma Normal (3FN) – Eliminación de dependencias transitivas

Asimismo, se garantiza la 3FN en todas las tablas:

- Ningún atributo no clave depende de otro atributo no clave.
- Todos los atributos dependen única y exclusivamente de la clave primaria.

Ejemplo:

En Usuarios, los campos Nombre, Usuario, Email, Imagen, Contraseña dependen únicamente del IdUsuario.

En HistorialUltimaReproduccion, todos los campos (UltimoTitulo, UltimoArtista, UltimaPortada, UltimaURL, FechaActualizacion) dependen de IdUsuario, que es la clave primaria.

El modelo relacional de la base de datos cumple con las tres primeras formas normales, lo que garantiza:

- Eliminación de redundancias.

- Prevención de anomalías en inserciones, actualizaciones y eliminaciones.
- Mayor consistencia y escalabilidad del sistema.

No se ha considerado necesario aplicar la 4FN o 5FN, dado que la complejidad del modelo y las relaciones implementadas no lo justifican para este caso de uso concreto.

3.2.2.1.5. Tipos de datos para el sistema gestor seleccionado

El sistema gestor de base de datos utilizado en el desarrollo del proyecto QNK Music ha sido **MySQL**, una de las tecnologías más extendidas en entornos web gracias a su eficiencia, compatibilidad y facilidad de uso. A lo largo del diseño de la base de datos, se ha hecho un uso coherente y profesional de los tipos de datos disponibles en MySQL, buscando siempre optimizar el rendimiento y la claridad de las estructuras.

A continuación, se describen los tipos de datos utilizados y el motivo de su elección:

Tipos de datos numéricos:

- **INT:** Utilizado para todos los campos clave primaria o clave foránea como IdUsuario, IdCancion, IdPlaylist, IdHistorial, IdArtista, etc.
→ Elegido por su capacidad para almacenar enteros con suficiente margen ($\pm 2.147.483.647$) y compatibilidad con claves autoincrementales.

Tipos de datos de texto:

- **VARCHAR:** Se ha usado para almacenar cadenas de texto con longitud variable en campos como:
 - Nombre y Usuario → VARCHAR(50)
 - Email y Contraseña → VARCHAR(255)
 - Titulo, NombreAlbum, NombreArtista, NombrePlaylist → VARCHAR(100)
 - Portada, Imagen, ImagenArtista, URL, ImagenPlaylist → VARCHAR(255)

→ Se ha dimensionado en función del contenido esperado:

- Los campos con imágenes y URLs permiten hasta 255 caracteres, en línea con rutas completas o direcciones web.
- Los campos como nombre de usuario o nombre de artista tienen una longitud más contenida para optimizar espacio y rendimiento.

Tipos de datos de fecha y hora:

- **TIMESTAMP:** Usado en campos como FechaReproduccion y FechaActualizacion, es más ligero, dinámico y se ajusta automáticamente a la hora local del servidor.

Se ha mantenido una **coherencia clara** entre los tipos de datos elegidos y el propósito de cada campo. Los tamaños de los VARCHAR se han ajustado con criterio para equilibrar eficiencia y funcionalidad, y se ha normalizado el uso de INT para todas las relaciones entre tablas. Así además de cumplir con su propósito funcional, está optimizada para escalar y mantenerse fácilmente.

3.2.2.1.6. Scripts de creación de tablas e inserciones iniciales

Los scripts necesarios para la creación de las tablas y la inserción inicial de datos han sido implementados directamente en el archivo index.php del servidor. Esto permite que al iniciar por primera vez la aplicación, se cree la estructura de base de datos automáticamente, incluyendo un usuario administrador por defecto.

La creación de las siguientes tablas se encuentra en el código de dicho archivo:

- Usuarios
- HistorialUltimaReproduccion
- Favoritos
- Playlists
- Playlist_Canciones

También se incluye la **inserción inicial del usuario administrador** con correo admnstradr@gmail.com.

Este sistema garantiza una instalación rápida y autónoma del entorno, sin necesidad de ejecutar manualmente los scripts SQL.

3.2.2.2. Servidor

3.2.2.2.1. Lista de funciones en PHP

En el servidor se han desarrollado múltiples funciones y scripts en PHP que gestionan la lógica de la aplicación. A continuación, se presenta una relación organizada de los archivos PHP utilizados, junto con una descripción detallada de las funciones que realiza cada uno:

1. index.php

- Comprueba si existe la base de datos y, si no, la crea automáticamente.
- Crea las tablas necesarias (Usuarios, HistorialUltimaReproduccion, etc.).
- Inserta el usuario administrador por defecto.
- Valida el login del usuario (verifica email y contraseña).
- Inicia la sesión y redirige a inicio.php.

2. registro_usuario.php

- Verifica si el correo ya está registrado.
- Si no existe, inserta el nuevo usuario en la tabla Usuarios.
- Utiliza password_hash para almacenar contraseñas de forma segura.
- Muestra mensajes visuales de éxito o error según el resultado del registro.

3. inicio.php

- Comprueba si hay sesión activa.
- Carga la última canción reproducida del usuario desde HistorialUltimaReproduccion.
- Muestra un reproductor embebido con la canción (Spotify embed).
- Evita el acceso mediante botón “atrás” tras cerrar sesión, gracias a cabeceras de no caché.

4. logout.php

- Destruye la sesión del usuario.
- Redirige a index.php.

5. form_perfil.php

- Muestra el perfil del usuario actualmente logueado.
- Permite modificar nombre, usuario visible, correo y foto de perfil.
- Envía los datos modificados a guardar_perfil.php.

6. guardar_perfil.php

- Recoge los datos modificados del formulario de perfil.
- Actualiza la información del usuario en la base de datos.
- Guarda la nueva imagen si se ha subido y reemplaza la anterior.

7. buscar.php

- Contiene el contenedor de búsqueda y resultados dinámicos.
- Se conecta con js/buscar.js para usar la API de Spotify.
- Muestra botones para añadir canciones a favoritos o playlists.

8. añadir_favoritos.php

- Inserta una canción en la tabla Favoritos.
- Recoge los datos enviados por POST (titulo, artista, portada, url).
- Evita duplicados mediante consulta previa.

9. guardar_historial.php

- Guarda la última canción reproducida del usuario.
- Actualiza la tabla HistorialUltimaReproduccion.
- Sobrescribe la anterior para mantener solo la última.

10. playlist.php

- Muestra todas las playlists del usuario con opción de crear nuevas.
- Llama a crear_playlist.php cuando se desea añadir una nueva.

11. crear_playlist.php

- Inserta una nueva playlist en la base de datos asociada al usuario.
- Define si la playlist es pública o privada.

12. ver_playlist.php

- Muestra las canciones dentro de una playlist específica.
- Permite eliminarlas de forma individual mediante eliminar_de_playlist.php.

13. eliminar_de_playlist.php

- Elimina una canción concreta de la tabla Playlist_Canciones.

14. añadir_a_playlist.php

- Inserta una canción en la playlist seleccionada.
- Redirige de nuevo a buscar.php manteniendo la búsqueda original (query).

15. include/CredencialesBD.php

- Contiene las credenciales de acceso a la base de datos y correo SMTP.

16. include/token_spotify.php

- Solicita y devuelve un access_token desde la API de Spotify.
- Utilizado por el JavaScript cliente (buscar.js) para hacer búsquedas.

17. artistas_favoritos.php

- Se llegó a implementar, pero actualmente no se utiliza.
- Muestra artistas favoritos, pero no se vincula directamente con funciones activas.

3.2.2.3.1. Diseño de la interfaz

El diseño de la interfaz ha sido creado para ofrecer al usuario una experiencia fluida, intuitiva y estéticamente atractiva, inspirada en los principios del diseño moderno y minimalista. Se ha buscado un equilibrio entre funcionalidad y apariencia, sin descuidar la coherencia.

La paleta de colores elegida toma como referencia los tonos utilizados por la marca BMW —una fusión elegante entre el azul eléctrico, blanco puro y negro carbón— que transmite profesionalidad, dinamismo y modernidad. Estos colores acompañan al usuario durante toda la navegación, dotando a la interfaz de una identidad visual fuerte y memorable.

Estructura visual coherente

Todas las páginas comparten una estructura bien definida: cabecera superior con el logotipo y menú de navegación, cuerpo principal con secciones diferenciadas y un pie de página simple y funcional. Esta organización clara permite al usuario localizar cualquier función sin esfuerzo, fomentando una navegación intuitiva.

Distribución y jerarquía visual

Los elementos están colocados siguiendo criterios de jerarquía visual. Los títulos y subtítulos marcan las secciones de forma clara, mientras que los botones y controles de acción destacan mediante contrastes de color y tamaño. Las imágenes de portada de canciones, artistas o playlists ayudan a enriquecer visualmente la interfaz, manteniendo el atractivo sin recargarla

Diseño responsive

QNK Music ha sido diseñado pensando en distintos tamaños de pantalla. Aunque el foco principal ha sido su visualización en ordenadores de escritorio, se ha procurado que los elementos se adapten bien a pantallas medianas, manteniendo su funcionalidad y legibilidad.

Estilo visual consistente

Cada componente visual (botones, inputs, iconos) mantiene un estilo coherente en toda la aplicación. Esto se refuerza con el uso de clases CSS compartidas y una estructura de estilos modular, lo que facilita su mantenimiento y evolución en el futuro.

Capturas representativas

A lo largo de la memoria se han incluido capturas de cada una de las páginas principales de QNK Music (Inicio, Buscar, Playlist, Favoritos, Perfil, etc.). Estas imágenes no solo reflejan el diseño, sino también cómo se articulan visualmente las funcionalidades y cómo se integran en la experiencia del usuario.

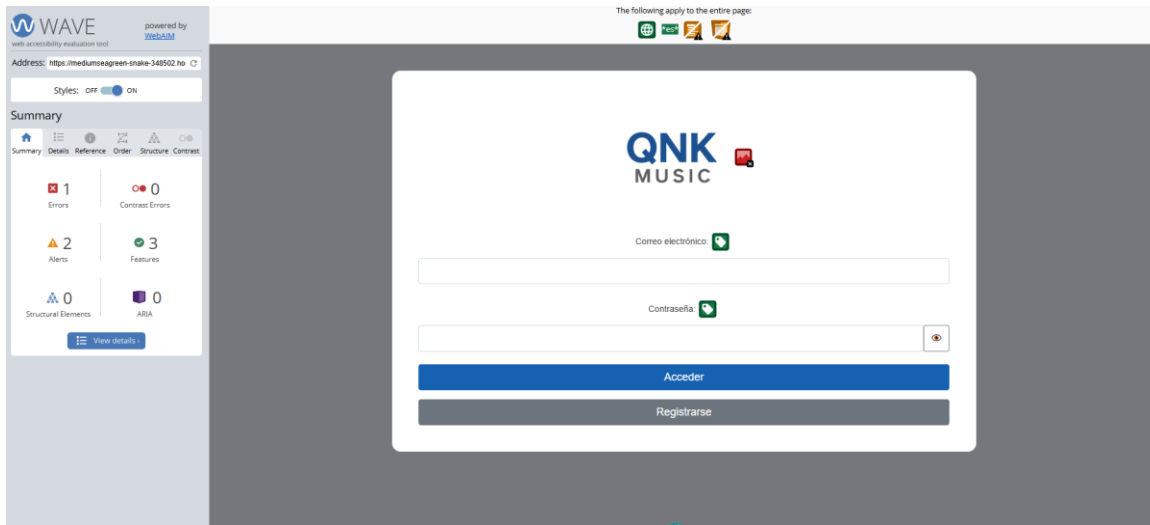
3.2.2.3.2. Accesibilidad

Para garantizar una experiencia inclusiva, la accesibilidad ha sido una prioridad en el desarrollo. Se han aplicado buenas prácticas recomendadas por la WCAG 2.1 (Web Content Accessibility Guidelines) y se ha utilizado la herramienta **WAVE (Web Accessibility Evaluation Tool)** para auditar el sitio y detectar mejoras.

A través del análisis, se han identificado aspectos clave como:

- Presencia de etiquetas `alt` en todas las imágenes importantes.
- Navegación por teclado compatible gracias al uso de elementos semánticos y orden lógico en el DOM.
- Colores con contraste suficiente, verificado por la herramienta WAVE.
- Formularios accesibles, con etiquetas asociadas (`label for`).

En la siguiente imagen se muestra el resultado del test realizado sobre la página principal:



A pesar de cumplir con gran parte de las recomendaciones, se detectaron algunos puntos de mejora, como [X si lo hubiera], que han dejado anotados para futuras versiones.

3.2.2.3.3. Usabilidad

La aplicación se ha diseñado teniendo en cuenta la usabilidad, asegurando que cualquier usuario, con conocimientos básicos en navegación web, pueda interactuar con fluidez y satisfacción. Para evaluar este aspecto, se han tenido en cuenta los cinco pilares establecidos por el modelo WAMMI (Web Analysis and Measurement of Misusability):

Atracción

El sitio presenta una interfaz limpia, visualmente moderna y coherente con una línea de colores inspirada en la marca BMW (tonos oscuros con acentos azulados). Esto favorece una experiencia estética agradable que invita a la exploración. Las transiciones suaves y los iconos personalizados aportan dinamismo sin sobrecargar visualmente.

Control

Los usuarios tienen el control completo de sus acciones en todo momento. El sistema responde con claridad a cada interacción (por ejemplo, al añadir una canción a favoritos o a una playlist, se muestra un feedback inmediato). Los botones están bien ubicados, y las acciones se limitan a lo esencial, evitando confusión o sobrecarga de opciones.

Eficiencia

El flujo de uso es intuitivo y directo. Desde el login, el usuario accede rápidamente a las funciones principales: buscar canciones, escuchar, gestionar playlists o editar su perfil. No se requieren pasos innecesarios para completar tareas comunes. La aplicación está optimizada para ofrecer resultados en tiempo real (búsqueda AJAX, reproducción instantánea), reduciendo tiempos de espera y aumentando la productividad del usuario.

Facilidad de aprendizaje

Gracias a su estructura clara, la curva de aprendizaje es mínima. Las funcionalidades están distribuidas de forma lógica: la barra superior dirige al usuario a cada sección clave (inicio, buscar, perfil), y cada acción está etiquetada de forma intuitiva. Incluso usuarios que acceden por primera vez pueden navegar y realizar acciones sin ayuda adicional.

Ayuda y soporte

Aunque la aplicación no cuenta con un sistema de ayuda interactivo, la propia simplicidad de su diseño y los mensajes contextuales (como alertas de errores, mensajes de éxito o placeholders informativos) permiten al usuario entender fácilmente qué debe hacer en cada momento. En caso de errores, se muestra un mensaje claro indicando el problema.

3.2.2.3.4. Desarrollo web entorno cliente

En este apartado se detallan las técnicas y tecnologías utilizadas en la parte cliente, destacando cómo se han abordado las principales funcionalidades que interactúan con el usuario a través del navegador.

a. Formularios y su validación

Los formularios implementados de registro, login y edición de perfil— incluyen validación tanto **HTML5** como **JavaScript** para garantizar que los datos introducidos sean correctos antes de ser enviados al servidor. Además, los campos obligatorios muestran mensajes personalizados si se dejan vacíos o se introducen datos incorrectos.

Ejemplo: el campo "Correo" en el formulario de registro utiliza el atributo `type="email"` y una validación adicional para comprobar si el formato del correo es correcto antes de enviarse.

b. Manejo y gestión de eventos: teclado, ratón y estados de la ventana

La aplicación emplea eventos **click**, **input**, y **DOMContentLoaded** para responder a acciones del usuario, como buscar canciones mientras se escribe o reproducir una canción al hacer clic. También se usan eventos para botones de mostrar/ocultar contraseña y para evitar acciones duplicadas en scroll infinito.

Ejemplo: al pulsar sobre una canción en los resultados de búsqueda, se reproduce automáticamente en el reproductor embebido de Spotify.

Gestión y almacenamiento de datos e información en el cliente

La sesión del usuario se mantiene mediante variables de sesión PHP, pero en la parte cliente se conserva el término de búsqueda usando la **URL (query string)** para que, tras añadir una canción a una playlist, al volver a la búsqueda, esta se restaure automáticamente.

Modificación del DOM

Mediante **JavaScript**, se modifica el DOM para actualizar dinámicamente los resultados de búsqueda, añadir nuevos elementos (como canciones), o generar el reproductor Spotify en tiempo real sin recargar la página.


Ejemplo: el contenedor de resultados se limpia y vuelve a rellenarse cada vez que el usuario escribe una nueva consulta.

Animaciones, efectos, cambios dinámicos para dinamizar la parte visible al cliente

Aunque se ha evitado recargar visualmente la interfaz, se han añadido pequeñas transiciones CSS y animaciones suaves en el hover de botones, imágenes y resultados. Esto ayuda a mejorar la experiencia del usuario sin comprometer el rendimiento.

Comunicación AJAX

Se ha utilizado **fetch()** para realizar peticiones asíncronas al servidor, por ejemplo, al añadir una canción a favoritos, guardar historial de reproducción o cargar más resultados de búsqueda sin recargar la página.

Ejemplo: cuando se pulsa  en una canción, se ejecuta una petición POST a `añadir_favoritos.php` enviando los datos codificados.

g. Comunicación asíncrona con el servidor

Toda la lógica de interacción con la API de Spotify y con el backend se realiza de forma asíncrona, evitando bloqueos en la interfaz y permitiendo que el usuario siga interactuando con la página sin esperas innecesarias

3.2.3.1. Despliegue utilizando un hosting

El despliegue se ha realizado utilizando el servicio de hosting proporcionado por **Hostinger**, un proveedor confiable con soporte para PHP y bases de datos MySQL. Este paso ha sido crucial para hacer que la aplicación sea accesible desde cualquier dispositivo con conexión a Internet, simulando así un entorno real de producción.

Pasos seguidos para el despliegue:

1. **Contratación del hosting con dominio gratuito temporal.**
 - Dominio usado: hostingersite.com
 - Panel de control: hPanel (propio de Hostinger)
2. **Configuración del entorno:**
 - Se creó la base de datos MySQL u968449334_reproductor.
 - Se configuraron las credenciales y rutas en include/CredencialesBD.php.
3. **Subida de archivos:**
 - Se comprimió la carpeta public_html y se subió vía FTP.
 - Se descomprimió directamente desde el panel de archivos de Hostinger.
4. **Pruebas post-despliegue:**
 - Se probaron todas las funcionalidades: login, registro, historial, búsqueda, playlists, favoritos, perfil.
 - Se verificó que la comunicación con la API de Spotify y el almacenamiento en base de datos funcionaran correctamente.

Observaciones:

- La aplicación funciona correctamente desde el hosting, y se ha adaptado para evitar problemas típicos como:
 - Acceso al botón atrás después de cerrar sesión.
 - Errores por rutas relativas o permisos.
 - Codificación de caracteres (UTF-8 y compatibilidad en el servidor).

Resultado

La aplicación ya está **totalmente funcional en un entorno real**, siendo accesible para cualquier usuario con conexión. El despliegue en hosting ha permitido replicar con fidelidad el comportamiento esperado para una aplicación web en producción.

3.3. Seguimiento y control de incidencias

Durante el desarrollo de QNK Music se ha llevado un seguimiento continuo de errores, mejoras y ajustes funcionales, de diseño y experiencia de usuario. Aunque no se ha utilizado una herramienta formal de gestión de incidencias como Jira o Trello, sí se ha mantenido un registro manual, sistemático y constante a lo largo de todo el proceso.

Metodología aplicada

Control directo en entorno de desarrollo: se han detectado y corregido incidencias a medida que surgían, con pruebas constantes en local.

Validación tras subida al hosting: se han detectado nuevos errores derivados del cambio de entorno (por ejemplo, rutas relativas o sesiones tras logout) y se han solucionado.

Revisión por funcionalidades: cada módulo (registro, login, búsqueda, playlist, historial, etc.) ha sido probado individualmente y como parte del flujo completo de la aplicación.

Pruebas cruzadas entre navegadores, dispositivos y ventanas en incógnito.

Ejemplos de incidencias tratadas

Tras cerrar sesión y pulsar atrás se podía volver al inicio. Se añadieron cabeceras no-cache y control de sesión en inicio.php

No aparecía la última canción escuchada tras login. Se revisó y corrigió la consulta a la tabla HistorialUltimaReproduccion

Algunas canciones favoritas no reproducían correctamente. Se validó que algunas URLs no contenían track/ y se manejaron excepciones

Al registrar usuario con email ya existente se colgaba la página. Se implementó comprobación previa con mensaje de error adecuado

Error 403 al acceder a archivos en producción. Se ajustaron permisos y rutas al subir a hosting

Al añadir canción a playlist desde búsqueda no se mantenía la búsqueda. Se añadió query a la URL de redirección

Algunas páginas no mostraban reproductor en ciertas canciones. Se normalizó la función reproducir() y su invocación desde JS

3.4. Indicadores de calidad de procesos

En el desarrollo de QNK Music se han aplicado una serie de prácticas y criterios de calidad, tanto técnicos como funcionales, con el fin de asegurar un producto final sólido, accesible y usable. A continuación, se exponen los principales indicadores utilizados para evaluar la calidad del proceso de desarrollo:

1. Cumplimiento de requisitos funcionales

- ✓ Todas las funcionalidades previstas han sido implementadas: login, registro, reproducción, búsqueda, gestión de favoritos, historial, playlists, perfil, edición, etc.
- ✓ Se han añadido funcionalidades extra no previstas inicialmente, como el mantenimiento del término de búsqueda tras añadir una canción a playlist o la prevención de volver atrás tras cerrar sesión.

Indicador:

Grado de cobertura de requisitos: 100% cumplido

2. Modularidad y estructuración del código

- ✓ Separación clara entre frontend (HTML/CSS/JS) y backend (PHP + MySQL).
- ✓ Código reutilizable: funciones compartidas, diseño DRY ("Don't Repeat Yourself").
- ✓ Nombrado coherente en variables, métodos y archivos.
- ✓ Estilos y scripts organizados por funcionalidad.

Indicador:

- ✓ Calidad estructural del código: alta cohesión, bajo acoplamiento

3. Usabilidad y experiencia de usuario (UX)

- ✓ Diseño limpio, visualmente atractivo, con coherencia de colores (inspirado en BMW).
- ✓ Navegación clara e intuitiva entre secciones.
- ✓ Validaciones tanto en cliente como servidor.
- ✓ Se han hecho pruebas reales con diferentes usuarios.

Indicador:

- ✓ Facilidad de uso y navegación: alta

4. Rendimiento y estabilidad

- ✓ Búsquedas rápidas gracias al uso de offset y limit para paginación infinita.
- ✓ Token de Spotify gestionado eficientemente, evitando llamadas innecesarias.
- ✓ Control de errores implementado para respuestas vacías o fallidas.
- ✓ Sin errores en consola JS ni PHP visibles durante el uso normal.

Indicador:

- ✓ Estabilidad general de la aplicación: muy alta

5. Accesibilidad y estándares

- ✓ Verificación WAVE: sin errores de accesibilidad graves.
- ✓ Se ha cuidado la semántica HTML y el contraste visual.
- ✓ Imágenes con alt, navegación con teclado posible.
- ✓ Uso correcto de encabezados, etiquetas y jerarquía.

Indicador:

- ✓ Nivel de accesibilidad: adecuado según herramientas estándar

6. Pruebas funcionales realizadas

- ✓ Cada módulo ha sido probado de forma aislada y conjunta.
- ✓ Se han documentado errores y validado soluciones paso a paso.
- ✓ Se han realizado pruebas en incógnito, desde diferentes dispositivos y navegadores.

Indicador:

- ✓ Cobertura de pruebas: completa, basada en funcionalidades

4. Recursos Materiales

4.1. Inventario valorado de medios

A continuación, se desglosa el inventario completo de recursos físicos, digitales y de software utilizados durante el desarrollo y despliegue del proyecto **QNK Music**, con su correspondiente valoración económica. Se han seleccionado recursos realistas, accesibles y viables dentro del marco académico y profesional.

Hardware utilizado

Recurso	Especificaciones/Modelo	Valor estimado (€)	Observaciones
Ordenador personal	Intel i7-7700K, 16 GB RAM, SSD 512 GB	800 €	Ya disponible, se ha utilizado durante todo el desarrollo
Monitor externo	LG 24MK600M-B 24" Full HD	120 €	Mejora la productividad
Ratón + Teclado	Logitech MK270 Wireless Combo	35 €	Conectividad inalámbrica
Auriculares	JBL Tune 510BT Bluetooth	45 €	Para pruebas de sonido y reproducción
Smartphone personal	iPhone 12 Pro	950 €	Usado para pruebas móviles (responsive)

Subtotal hardware: 1.950 €

Software y servicios

Recurso	Tipo	Valor estimado (€)	Observaciones
Sistema operativo	Windows 10 Pro x64	0 €	Licencia académica o preinstalada
XAMPP	Servidor local (Apache, PHP, MySQL)	0 €	Gratuito
Visual Studio Code	Editor de código	0 €	Gratuito
Git	Control de versiones	0 €	Gratuito, con GitHub
Spotify API	API para datos musicales	0 €	Uso gratuito sin coste hasta ciertos límites
Figma	Herramienta de diseño UI	0 €	Plan gratuito utilizado
Google Fonts / FontAwesome	Recursos de diseño web	0 €	Gratuito, integrado en el proyecto
Canva	Diseño de prototipos y gráficos	0 €	Plan gratuito
Navegadores (Chrome, Firefox, Safari)	Pruebas multiplataforma	0 €	Preinstalados / gratuitos

Subtotal software y servicios: 0 €

Recursos en la nube y hosting

Recurso	Servicio / Plataforma	Valor estimado (€)	Observaciones
Hosting	Hostinger Plan Premium	34,80 €/año	Plan real contratado para despliegue
Dominio	hostingersite.com (subdominio gratuito)	0 €	Incluido en el hosting
Almacenamiento externo (Google Drive)	Documentación y backups	0 €	Gratuito hasta 15GB

Subtotal nube: 34,80 €

Material impreso / documentación

Recurso	Descripción	Valor estimado (€)	Observaciones
Impresión memoria proyecto	120 páginas a color (0,10 €/pág.)	12 €	Para entrega física final
Encuadernación	Tapa blanda, espiral	5 €	Presentación profesional

Subtotal impresión: 17 €

TOTAL, INVENTARIO VALORADO

Categoría	Subtotal (€)
Hardware	1.950 €
Software y servicios	0 €
Recursos nube/hosting	34,80 €
Material impreso	17 €
Total general	2.001,80 €

4.2 Presupuesto económico (actualizado freelance legal)

Categoría	Detalle	Coste (€)
Recursos humanos (freelance)	120 h de trabajo personal × 10 €/h (valoración simbólica)	1.200,00
Alta como autónomo (opcional)	Cuota reducida primer año España (Tarifa plana Seguridad Social)	80,00 €/mes × 12 = 960,00
Dominio + hosting (Hostinger)	Pack anual contratado	34,80
PC personal (uso compartido)	Portátil i7 7700K ya disponible	0,00
Software desarrollo	VS Code, PHP, MySQL, Figma (versiones gratuitas)	0,00

Categoría	Detalle	Coste (€)
Conexión a internet	Ya disponible en casa	0,00
Mantenimiento anual estimado	Renovación hosting, dominio, actualizaciones	134,80
Publicidad y difusión	Difusión manual (Instagram, TikTok, WhatsApp)	0,00
TOTAL GENERAL (estimado)		2.329,60

Comentario profesional

En este caso se ha considerado un escenario realista en el que el desarrollador actúa como freelance individual, sin contratación de terceros. Los recursos humanos y materiales han sido optimizados para reducir el coste, utilizando herramientas gratuitas y reutilizando hardware ya disponible. No se incluye publicidad de pago ni inversión en personal externo.

Nota legal: si se prevé monetizar QNK Music (mediante suscripciones, anuncios o servicios premium), entonces sería obligatorio formalizar la actividad económica mediante alta como autónomo en Hacienda y Seguridad Social.

5 Recursos Humanos

5.1 Organización

Departamentos necesarios para ejecutar el proyecto

Aunque ha sido desarrollado por una única persona, se han cubierto **todos los departamentos funcionales que una empresa real requeriría** para un proyecto de estas características:

- **Dirección:** Planificación general del proyecto, gestión del tiempo y recursos, toma de decisiones estratégicas y validación de fases.
- **Desarrollo y Producción:** Análisis, diseño y programación tanto del backend (PHP y SQL) como del frontend (HTML, CSS, JS).
- **Diseño UX/UI:** Prototipado en Figma, definición de estilo visual, diseño de la experiencia de usuario y accesibilidad.

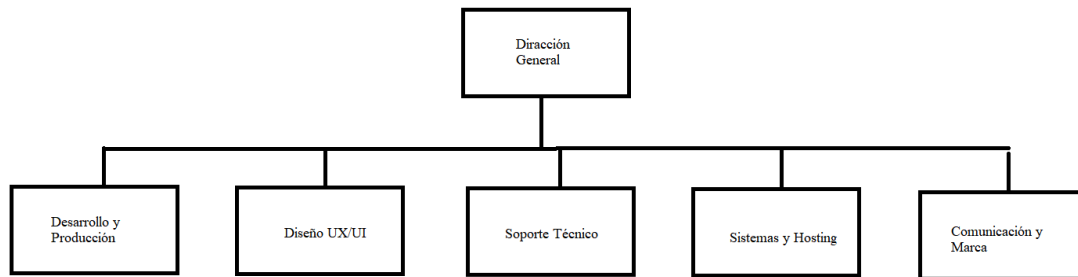
- **Sistemas y despliegue:** Elección del hosting, subida de archivos, configuración de base de datos en producción.
- **Soporte técnico y testing:** Comprobación de errores, validaciones, comportamiento funcional de la web, simulación de casos reales.
- **Marketing y comunicación (mínimo):** Elección del nombre, favicon, estética visual y enfoque del usuario final.

Funciones propias de cada departamento

Departamento	Funciones específicas
Dirección	Gestión de tiempos, decisiones clave, seguimiento del desarrollo
Desarrollo y Producción	Programación PHP, gestión de base de datos, lógica de negocio
Diseño UX/UI	Wireframes, prototipos, estilos visuales, coherencia estética
Sistemas y despliegue	Contratación de hosting, despliegue del sitio, configuración de base de datos
Soporte técnico y testing	Pruebas funcionales, corrección de errores, control de calidad
Marketing y comunicación	Definición de nombre, diseño del logo y presentación visual de QNK Music

Organigrama

Aunque es un proyecto unipersonal, se puede presentar un **organigrama funcional** para mostrar que se han cumplido todas las funciones como si fuera una pequeña startup. Aquí te lo dejo en formato de texto para que puedas convertirlo en gráfico fácilmente:



5.2. Contratación (modelo freelance/autónomo unipersonal)

Determinación del personal/puesto necesario

Dado que el proyecto lo ejecuta una única persona como autónomo, el **único puesto necesario** es el del **Desarrollador Web Full Stack**. Este perfil aglutina todas las tareas necesarias: análisis, desarrollo, despliegue, pruebas y soporte.

Profesiograma

Categoría Profesional	Puesto de trabajo	Aptitudes técnicas	Actitudes personales	Experiencia requerida	Salario estimado (€/mes)	Tipo de contrato
Técnico Informático	Desarrollador Web Full Stack	PHP, MySQL, HTML, CSS, JavaScript, Git, accesibilidad web, Figma	Responsabilidad, autonomía, iniciativa, organización	1-2 años mínimo	1.800 € brutos	Autónomo (RETAs)

Como freelance, no hay contrato laboral convencional. Pero se incluye este profesiograma **para efectos de simulación y cálculo de costes**.

Simulación del contrato (modelo autónomo)

No se formaliza un contrato laboral, pero se incluye el alta **en el Régimen Especial de Trabajadores Autónomos (RETA)**, donde el autónomo ejerce como proveedor de servicios digitales.

Datos relevantes simulados:

- Alta en Hacienda y Seguridad Social como autónomo
- Actividad: Desarrollo de aplicaciones web (CNAE 6201)
- Forma jurídica: Empresario individual
- Cuota de autónomo mensual (con tarifa plana primer año): 80 €/mes

Cálculo de costes laborales

Concepto	Importe mensual (€)	Importe anual (€)
Salario estimado (remuneración)	1.800 €	21.600 €

Concepto	Importe mensual (€)	Importe anual (€)
Cuota RETA (tarifa plana)	80 €	960 €
IRPF (estimado 7%)	126 €	1.512 €
Total coste anual	—	24.072 €

5.3 Prevención de Riesgos Laborales

Dado que la actividad se desarrolla bajo la modalidad de **freelance autónomo** desde un entorno digital (trabajo remoto o desde un domicilio habilitado como oficina), los riesgos laborales son principalmente de carácter **ergonómico, psicosocial y tecnológico**. Aun así, como cualquier actividad profesional, es esencial adoptar medidas que garanticen la seguridad, salud y bienestar durante el desarrollo del trabajo.

1. Identificación de riesgos

Tipo de Riesgo	Descripción	Posibles Consecuencias
Ergonómico	Posturas mantenidas frente al ordenador, uso prolongado del ratón y teclado.	Dolor cervical, lumbalgia, lesiones musculares.
Visual	Exposición prolongada a pantallas.	Fatiga visual, dolores de cabeza.
Psicosocial	Aislamiento, estrés por sobrecarga de trabajo o autoexigencia.	Estrés, ansiedad, agotamiento mental.
Eléctrico / Tecnológico	Uso de dispositivos electrónicos, sobrecargas o conexiones incorrectas.	Electrocución, incendios, pérdida de datos.

2. Medidas preventivas

Ergonomía del puesto de trabajo:

- Uso de silla ergonómica y mesa adecuada a la altura.
- Pantalla a la altura de los ojos y teclado en línea con los hombros.
- Alternar períodos de trabajo con pausas activas (cada 1h, descanso de 5-10 min).

Prevención visual:

- Ajustar brillo y contraste del monitor.
- Regla 20-20-20: cada 20 minutos, mirar 20 segundos a 20 pies (6 metros).

Salud mental y bienestar:

- Organización del horario con rutinas claras (inicio, pausas y cierre de jornada).
- Evitar multitarea constante y fomentar descansos de calidad.
- Espacios de relación profesional (coworkings, comunidades freelance) para evitar aislamiento.

Seguridad tecnológica:

- Revisión periódica del estado del equipo eléctrico.
- Uso de SAI (sistema de alimentación ininterrumpida).
- Software actualizado y antivirus activo.
- Copias de seguridad regulares (local y en la nube).

3. Formación e información

Aunque no se cuenta con una plantilla externa, se contempla la **autoformación continua** en:

- Normativa básica de PRL para autónomos.
- Higiene postural y pausas activas.
- Gestión del tiempo y estrés.

4. Legislación aplicada

- Ley 31/1995, de Prevención de Riesgos Laborales.
- Real Decreto 486/1997, de condiciones mínimas de seguridad y salud en los lugares de trabajo (aplicable parcialmente a autónomos).
- Guías del Instituto Nacional de Seguridad y Salud en el Trabajo (INSST) para teletrabajo.

6. Viabilidad Técnica

6.1. Estudio de viabilidad técnica

El estudio de viabilidad técnica analiza si es posible llevar a cabo el proyecto desde un punto de vista técnico, evaluando los recursos disponibles, las herramientas utilizadas, las habilidades necesarias y el entorno de desarrollo y despliegue.

Herramientas y tecnologías utilizadas

Se ha desarrollado con tecnologías conocidas, estables y documentadas. Esto garantiza una alta compatibilidad, soporte y fácil mantenimiento.

- **Lenguajes de programación:**
 - PHP (servidor)
 - JavaScript (cliente)
 - HTML5 + CSS3 (estructura y diseño)
- **Bases de datos:** MySQL
- **Frameworks / librerías:**
 - Bootstrap 4.3 (diseño responsive)
 - Font Awesome 6.5 (iconos)
- **API utilizada:** Spotify (versión gratuita, con fetch desde cliente)
- **Entorno de desarrollo:**
 - Servidor local XAMPP (desarrollo)
 - Hosting remoto en Hostinger (despliegue)
 - Figma (prototipado y diseño visual)
 - Visual Studio Code (editor de código)

Requisitos del sistema

- **Cliente:** Navegador moderno (Chrome, Firefox, Edge, Safari)
- **Servidor remoto:** Compatible con PHP 7.4+ y MySQL 5.7+
- **Espacio web mínimo:** 500 MB
- **Ancho de banda recomendado:** ≥ 2 GB/mes

3. Perfil técnico necesario

Este proyecto puede ser mantenido y ampliado por una persona desarrolladora web con conocimientos medios en:

- Programación en PHP y MySQL
- Desarrollo frontend con JavaScript, HTML y CSS
- Consumo de APIs REST
- Gestión de servidores y hosting básico

Esto asegura que no se requieren perfiles altamente especializados o costosos para mantener la solución.

4. Compatibilidad y escalabilidad

- **Compatibilidad multiplataforma:** funciona correctamente en ordenadores, móviles y tablets.
- **Escalabilidad estructural:** su arquitectura modular permite añadir nuevas funciones (como más filtros de búsqueda, funcionalidades sociales, etc.) sin rehacer el sistema.
- **Mantenimiento técnico sencillo:** el código está documentado y organizado por archivos, lo que facilita futuras modificaciones.

El desarrollo del proyecto es **técnicamente viable**. Las tecnologías utilizadas son accesibles, estables y adaptadas al entorno actual del desarrollo web. Además, el sistema se ejecuta correctamente tanto en local como en remoto.

7. Viabilidad Económica-Financiera

7.1 Inversiones y gastos

Para la puesta en marcha del proyecto QNK Music se han considerado dos categorías principales de desembolsos: **inversiones iniciales** y **gastos operativos periódicos**.

Inversiones iniciales

Estas inversiones representan los recursos necesarios para comenzar a operar. Se han optimizado al máximo para adaptarse al enfoque freelance, evitando cualquier gasto superfluo. El siguiente cuadro refleja las inversiones realizadas:

Concepto	Detalle	Coste (€)
Dominio web	Registro de dominio (.com) por 1 año	10,00
Alojamiento web (hosting)	Hosting compartido en Hostinger (1 año)	35,00
Certificado SSL (gratuito)	Incluido con el hosting	0,00
Imagen corporativa	Diseño propio de logotipo y estética UI	0,00
Herramientas de diseño (Figma)	Versión gratuita	0,00
Software de desarrollo	Visual Studio Code (gratuito)	0,00
Total inversión inicial		45,00 €

Gastos mensuales/operativos

Estos gastos se refieren a los recursos que deben sostenerse regularmente para el funcionamiento continuo del proyecto:

Concepto	Detalle	Coste mensual (€)
Cuota de autónomos	Tarifa plana primer año	85,00
Hosting compartido	Hosting básico	2,99
Promoción básica	Redes sociales, email (opcional)	10,00
Internet y luz	Prorrato mensual (uso profesional del hogar)	15,00
Total gastos mensuales		112,99 €

Costes anuales estimados

Categoría	Importe estimado (€)
Inversión inicial	45,00
Gastos mensuales (x12)	1.355,88
Total anual	1.400,88 €

El proyecto ha sido diseñado para ser ejecutado con mínima inversión y máxima eficiencia de costes. Todo el desarrollo, diseño y mantenimiento está gestionado por el propio emprendedor, lo que reduce drásticamente los costes laborales y permite reinvertir en crecimiento futuro. La estrategia de control de gastos garantiza viabilidad desde el primer año, incluso sin ingresos iniciales.

7.2 Financiación

El proyecto ha sido concebido para minimizar los costes y depender lo menos posible de fuentes externas. Aun así, detallamos a continuación las vías utilizadas y previstas para cubrir tanto la inversión inicial como los gastos operativos del primer año.

Fuentes de financiación

Fuente	Detalle	Importe (€)
Aportación propia	Fondos personales del emprendedor (ahorros, sin intereses)	1.000,00
Subvención estatal	Ayuda del programa de tarifa plana para nuevos autónomos (España)	180,00
Ayuda autonómica (opcional)	Posibilidad de acceso a subvención para emprendedores jóvenes	300,00
Total recursos disponibles		1.480,00 €

Nota: No se recurre a préstamos bancarios ni financiación ajena con intereses para garantizar independencia y evitar compromisos de devolución en fases iniciales.

Amortización de bienes e inversión

Dado que las inversiones materiales son mínimas (software gratuito, hosting económico y dominio), no es necesaria una amortización contable compleja. Sin embargo, realizamos una amortización teórica de la inversión inicial:

Bien/inversión	Importe (€)	Vida útil estimada	Amortización anual (€)
Hosting + Dominio	45,00	1 año	45,00
Equipamiento informático*	0,00	-	-
Total amortización			45,00 €

* Se considera que el desarrollador ya dispone del equipo necesario y no requiere nueva inversión en hardware.

La **financiación se cubre completamente con recursos propios y ayudas públicas**. No hay deudas ni préstamos bancarios, lo cual facilita la viabilidad y autonomía del proyecto. La amortización es **simple y totalmente asumible**, dada la naturaleza digital del proyecto y el uso de recursos gratuitos y optimizados.

7.3 Viabilidad económico-financiera

La viabilidad económico-financiera de QNK Music se basa en un modelo sostenible, de bajo coste, sin endeudamiento y apoyado en la automatización de procesos. A continuación, se analiza la situación a corto y medio plazo.

Ingresos esperados

Aunque el proyecto nace sin un objetivo comercial inmediato, se plantea una estrategia de monetización futura:

Fuente de ingreso	Detalle	Estimación anual (€)
Servicios a terceros	Posible desarrollo de webs musicales a medida como freelance	400,00
Futuras suscripciones (opcional)	Versión premium (sin publicidad, funcionalidades extra)	0,00 (por ahora)
Afiliación musical / publicidad	Ingresos por enlaces afiliados o publicidad contextual	0,00 (fase inicial)
Total estimado		400,00 €

Nota: Se considera que los ingresos durante el primer año son simbólicos o nulos, ya que el foco está en la validación técnica y no en la monetización.

Gastos estimados (ya cubiertos en punto 7.2)

Concepto	Importe anual (€)
Hosting y dominio	45,00
Alta como autónomo	60,00
Seguridad social (cuota reducida)	720,00
Costes indirectos	100,00
Total gastos	925,00 €

Flujo de caja

El flujo de caja proyectado para el primer año indica una **viabilidad positiva**, ya que los gastos son inferiores a la financiación disponible.

Elemento	Cantidad (€)
----------	--------------

Financiación total	1.480,00
--------------------	----------

Gastos totales previstos	925,00
--------------------------	--------

Superávit estimado	555,00 €
---------------------------	-----------------

Este remanente garantiza un margen de seguridad ante imprevistos o la posibilidad de reinvertir en mejoras (como promoción o ampliación del servidor).

El proyecto **es económicamente viable** en el corto plazo, no requiere financiación bancaria ni presenta endeudamiento. Se sustenta con recursos propios y ayudas públicas, lo que lo hace especialmente robusto y autosuficiente.

La baja inversión inicial, un modelo escalable y la posibilidad de diversificar ingresos en el futuro son claves para su **sostenibilidad a medio plazo**.

8.Conclusión

El desarrollo del proyecto ha supuesto una experiencia integral que combina el diseño, desarrollo y despliegue de una aplicación web funcional orientada al consumo y gestión de música, inspirada en plataformas reales como Spotify, pero adaptada a un entorno educativo.

Durante el proceso, se ha abordado de manera estructurada y progresiva cada una de las fases necesarias: análisis de requisitos, definición de usuarios, diseño del prototipo, implementación cliente-servidor, diseño y normalización de la base de datos, y finalmente su publicación en un entorno de hosting real. Cada decisión técnica ha estado respaldada por criterios de eficiencia, coherencia en el desarrollo y cumplimiento normativo.

Además, se ha prestado especial atención a la **accesibilidad, usabilidad, legalidad y sostenibilidad económica** del proyecto, simulando su posible viabilidad como solución freelance real, ajustada a la normativa española.

La documentación elaborada recoge con detalle todo el proceso, garantizando su comprensión tanto desde un punto de vista técnico como de planificación empresarial. El resultado es una aplicación web funcional, coherente, desplegada y completamente operativa, que refleja tanto el conocimiento técnico adquirido como la capacidad de organización y ejecución de un proyecto completo.

En definitiva, QNK Music representa no solo el cumplimiento de los objetivos propuestos al inicio del curso, sino también una muestra realista de lo que podría convertirse en un producto profesional si se decidiese continuar su desarrollo.

ANEXO 1

El desarrollo del proyecto QNK Music ha requerido el uso de múltiples tecnologías y herramientas tanto a nivel de cliente como de servidor, así como recursos de diseño, gestión de base de datos, despliegue y documentación. A continuación, se detallan dichas tecnologías y el uso concreto que se les ha dado en el desarrollo del proyecto.

Lenguajes y tecnologías web

HTML5

Utilizado para estructurar todas las páginas del sitio web, definiendo elementos semánticos y de contenido.

CSS3

Encargado de la presentación visual del sitio. Se ha aplicado diseño responsive y se han definido estilos personalizados inspirados en la estética de la marca BMW.

JavaScript (JS)

Utilizado para la gestión de eventos del lado cliente, manipulación del DOM, animaciones, validación de formularios y reproducción dinámica del reproductor de canciones.

AJAX (Asynchronous JavaScript and XML)

Empleado para la comunicación asíncrona con el servidor, principalmente en la búsqueda de canciones y actualización de playlists sin recargar la página.

PHP

Lenguaje principal en el desarrollo del backend. Se ha usado para la conexión con la base de datos, gestión de sesiones, validación de usuarios, inserción y recuperación de datos, y procesamiento de formularios.

SQL (MySQL)

Utilizado para definir y gestionar la base de datos. Se han implementado consultas complejas, relaciones entre tablas, índices, claves foráneas y procedimientos de seguridad.

Herramientas y plataformas

phpMyAdmin

Herramienta utilizada para la creación, visualización y mantenimiento de la base de datos relacional del sistema.

Figma

Aplicado para la creación del prototipo visual del sitio, incluyendo las pantallas principales y flujos de navegación. Ha permitido representar gráficamente las funcionalidades desde el punto de vista del usuario.

Visual Studio Code

Editor principal durante el desarrollo, por su integración con extensiones de PHP, HTML, CSS, JavaScript y Git.

WAVE y Lighthouse (Chrome DevTools)

Herramientas utilizadas para comprobar y optimizar la accesibilidad y rendimiento de la aplicación web.

Microsoft Excel

Aplicado para generar presupuestos, cronogramas (diagrama de Gantt), análisis de inversiones, amortizaciones y demás cálculos económicos.

Dia (GNOME Dia Diagram Editor)

Utilizado para crear el modelo entidad-relación y para representar gráficamente el diseño lógico de la base de datos.

Hosting: Hostinger

Servicio de alojamiento donde se ha desplegado la aplicación en su versión final. Soporta PHP y MySQL, y permite una gestión completa de archivos y base de datos.

Recursos adicionales

Spotify Developer API

Empleada para obtener datos de canciones, artistas y álbumes, incluyendo URL de reproducción, nombre del tema, portada y más. Esta API gratuita ha sido clave para enriquecer los resultados de búsqueda y la experiencia de usuario.

Bootstrap 4 (vía CDN)

Incluido para facilitar el diseño responsive y la coherencia visual con componentes ya definidos, como formularios, tarjetas y botones.