



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Aplicación web para streaming de música

Autor:

Francisco Jesús Palomares Alabarce

Director:

José Manuel Benítez Sánchez

ETSIIT
Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación



Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación
Granada, Junio 2018



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA

Aplicación web para streaming de música

Autor:

Francisco Jesús Palomares Alabarce

Director:

José Manuel Benítez Sánchez

Aplicación web para streaming de música

Francisco Jesús Palomares Alabarce

Palabras clave: streaming, música, JavaScript

Resumen:

El desarrollo de este Trabajo Fin de Grado (TFG) tiene como objetivo principal la transmisión de música en streaming.

Además del objetivo principal, tiene otras funcionalidades como pueden ser el manejo de usuarios, la gestión de artistas, junto con sus álbumes y canciones. También se dispone de funcionalidades para crear playlists, que es una lista de canciones que un usuario crea y no tiene que ser del mismo artista, de esta manera escuchar diferentes artistas sin ir navegando entre ellos.

Este desarrollo ofrecerá de una interfaz responsiva vía web, para que el usuario pueda acceder a la plataforma de una manera más usable y sencilla. También a los dispositivos móviles se les ofrecerá una aplicación móvil.

Este proyecto está desarrollado en diferentes lenguajes de programación como Javascript, HTML y CSS.

Para ello se ha utilizado Node JS junto Express JS para la creación de la API Rest (Backend), el conjunto de funciones y protocolos que pueden acceder a la base de datos para consultar, eliminar y modificar información.

Para la interfaz web se ha utilizado Angular, con ella se puede crear webs SPA (Single Page App), de esta manera no se recarga la web entera, solamente se cargan las partes que nos haga falta para hacer la navegación más eficiente. Esto funciona con HTML y CSS y JQUERY, también se ha utilizado Bootstrap para hacer la página responsiva.

Para la aplicación móvil se ha utilizado IONIC que junto con Angular hace que se puedan hacer aplicaciones móviles para diversas plataformas como Android, IOS y Windows Mobile de una manera fácil.

Para el despliegue de este proyecto se utilizará Docker, una herramienta que automatiza el despliegue dentro de contenedores de software.

Web Application for music streaming

Francisco Jesús Palomares Alabarce

Keywords: streaming, music, JavaScript

Abstract:

The main objective of this Final Degree Project (TFG) is to stream music.

In addition to the main objective, it has other features such as user management, artist management, along with their albums and songs. There are also functions to create playlists, which is a list of songs that a user creates and does not have to be the same artist, in this way, listening to different artists without going to browse among them.

This development offers a responsive interface via web, for the user can access the platform in a more usable and simple way. A mobile application is also offered to mobile devices.

This project is developed in different programming languages such as Javascript, HTML and CSS.

To do this, JS Node was used together with Express JS for the creation of the Rest (Backend) API, the set of functions and the protocols that can access the database to consult, delete and modify information.

Angular has been used for the web interface, with it you can create SPA websites (single-page application), in this way the whole web is not reloaded, only the parts that you do not need are loaded to make the navigation more efficient. This works with HTML and CSS and JQUERY, Bootstrap has also been used to make the page responsive.

For the mobile application IONIC has been used, which together with Angular makes it possible to make applications for Android, iOS and Windows Mobile platforms in an easy way.

For the deployment of this project Docker is used, a tool that automates the deployment within software containers.

Yo, **Francisco Jesús Palomares**, alumno de la titulación *Grado en Ingeniería Informática* de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75940353B, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Francisco Jesús Palomares Alabarce

Granada a 12 de Junio de 2018 .

D. **José Manuel Benítez Sánchez** Profesor área de Ciencia de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Aplicación web para streaming de música***, ha sido realizado bajo su supervisión por **Francisco Jesús Palomares Alabarce**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a Junio de 2018.

El director: **José Manuel Benítez Sánchez**

Agradecimientos:

A mi familia y amigos que siempre han estado apoyándome para seguir mejorando, en especial a mis padres por haber creído en mí todo este tiempo y sin los cuales nada de esto hubiese sido posible.

También a mis compañeros de facultad que me han sugerido mejoras en el desarrollo de este proyecto.

Además, también a mi tutor **José Manuel Benítez Sánchez** por haber confiado en mí para la realización de este proyecto.

ÍNDICE

Aplicación web para streaming de música	6
Resumen:	6
Web Application for music streaming	8
Abstract:	8
1. Introducción	24
1.1 Motivación	24
1.2 Objetivos	24
1.3 Estructura	25
2. Planificación	26
2.1 Metodología	26
2.2 Etapas	26
2.2.1 Etapa inicial	26
2.2.2 Etapa de análisis	26
2.2.2.1 Análisis de requerimientos	26
2.2.2.2 Análisis de recursos	27
2.2.3 Etapa de desarrollo	27
2.2.4 Etapa de pruebas y corrección de errores	27
2.2.5 Entrega del sistema	27
2.2.6 Redacción de la memoria	27
2.3 Temporización	28
2.4 Presupuesto	28
2.4.1 Recursos	28
2.4.1.1 Recursos hardware	28
2.4.1.2 Recursos software	29
2.4.2 Costes	29
2.4.2.1 Licencias	29
2.4.2.2 Recursos hardware	29

2.4.2.3 Recursos humanos	29
2.4.2.4 Coste total	29
3. Análisis	30
3.1 Análisis de mercado	30
3.1.1 Spotify [1]	30
3.1.2 Apple Music [2]	30
3.1.3 Soundcloud [3]	30
3.2 Especificación de requisitos	31
3.2.1 Identificar implicados	31
3.2.2 Requisitos funcionales	33
3.2.3 Requisitos no funcionales	35
3.2.4 Restricciones semánticas	35
3.2.5 Requisitos de rendimiento	36
3.2.6 Requisitos de información	36
3.3 Modelos de casos de uso	36
3.3.1 Descripción de sus actores y sus objetivos	36
3.3.2 Clasificación de objetivos	41
3.3.3 Diagramas de casos de uso	41
3.3.4 Descripción de casos de uso	45
GESTIÓN DE ADMINISTRADORES:	45
GESTIÓN DE USUARIOS:	50
GESTIÓN DE ARTISTAS:	56
GESTIÓN DE ÁLBUMES:	62
GESTIÓN DE CANCIONES:	69
GESTIÓN DE PLAYLISTS:	75
4. Diseño	86
4.1 Patrón arquitectónico	86
4.2 Diagrama entidad-relación	86
4.3 Lenguajes y entornos a utilizar	87
4.3.1 Lenguajes utilizados	87

4.3.1.1 JavaScript [4]	87
4.3.1.2 HTML (HyperText Markup Language) [5]	87
4.3.1.3 CSS (Cascading Stylesheets) [6]	87
4.3.2 Sistema de gestión de base de datos, PostgreSQL [7]	87
4.3.3 Entorno del servidor Backend, Node JS [8]	88
4.3.4 Entorno del frontend, Angular [9]	88
4.3.5 Entorno del desarrollo para aplicaciones móviles, Ionic [10]	89
4.4 Herramientas a utilizar	89
4.4.1 IDE - Visual Studio Code	89
4.4.2 Bibliotecas y frameworks utilizados	89
4.4.2.1 Para Node JS	89
4.4.2.2 Para Angular	90
4.4.2.3 Para Ionic	90
4.5 Diseño de la plataforma	91
4.5.1 Diseño de página web	91
4.5.1.1 Inicio de sesión	91
4.5.1.2 Registro de usuario	92
4.5.1.3 Página principal de la plataforma	92
4.5.2 Diseño de aplicación móvil	93
4.5.2.1 Inicio de sesión	93
4.5.2.2 Registro	94
4.6 Diseño del objetivo principal	95
4.7 Despliegue: Docker [11]	95
5. Implementación:	96
5.1 Fases del desarrollo	96
5.1.1 Desarrollo del Backend	96
5.1.1.1 Creación de tablas	96
5.1.1.2 Creación de controladores	97
5.1.1.3 Autenticación	99
5.1.1.4 Middlewares	99

5.1.1.5 Creación de rutas	100
5.1.1.6 Implementación streaming	101
5.1.2 Desarrollo del Frontend	102
5.1.2.1 Creación de modelos	102
5.1.2.2 Creación de login y registro	103
5.1.2.3 Creación del componente principal	103
5.1.2.4 Creación del componente reproductor	104
5.1.2.5 Control de expiración del token	104
5.1.3 Desarrollo de la aplicación móvil	104
5.1.3.1 Reproductor de música	104
5.1.3.2 Subida de imágenes	105
5.1.3.3 Control de música en notificación	105
5.1.3.4 Control de expiración del token	105
6. Conclusiones	106
6.1 Objetivos alcanzados	106
6.2 Lecciones aprendidas	106
7. Bibliografía	107
Bibliografía	107
8. Anexos	109
8.1 Código fuente	109
8.2 Versión final	109
8.2.1 Aplicación web	110
8.2.1.1 Iniciar sesión	110
8.2.1.2 Registro	110
8.2.1.3 Página inicial	111
8.2.1.4 Actualizar mis datos	111
8.2.1.5 Página de artistas	112
8.2.1.6 Página de artista	112
8.2.1.6 Página de álbum con reproducción de canción	113
8.2.1.6 Página de álbum con vista de cola de canciones a reproducir	113

8.2.1.7	Página de playlists	114
8.2.1.8	Página de una playlist	114
8.2.1.9	Buscador	115
8.2.1.9	Playlist de otro usuario	115
8.2.2	Aplicación móvil	116
8.2.2.1	Inicio	116
8.2.2.2	Registro	116
8.2.2.3	Principal	117
8.2.2.4	Artistas	117
8.2.2.5	Artista	118
8.2.2.6	Álbum	118
8.2.2.7	Álbum con canción reproduciendo	119
8.2.2.8	Opciones de canción	119
8.2.2.9	Playlists	120
8.2.2.10	Crear playlist	120
8.2.2.11	Mi cuenta	121
8.2.2.12	Cambio de imagen de usuario	121
8.2.2.13	Buscar	122
8.2.2.15	Playlist	122
8.2.2.16	Opciones de playlist	123
8.2.2.17	Playlist otro usuario	123
8.2.2.18	Notificación móvil para controles	124
8.2.2.19	Reproductor	124
8.2.2.20	Cola de canciones	125
8.3	Glosario de términos	125
8.3.1	Términos	125
8.3.2	Acrónimos	126

ÍNDICE DE IMÁGENES

Imagen 1: Diagrama de Gantt	24
Imagen 2: Esquema entidad relación	83
Imagen 3: Página inicio sesión diseño página web	88
Imagen 4: Página registro diseño página web	88
Imagen 5: Página principal diseño página web	89
Imagen 6: Página inicio sesión diseño aplicación móvil	89
Imagen 7: Página registro diseño aplicación móvil	90
Imagen 8: Página principal diseño aplicación móvil	90
Imagen 9: Página reproductor diseño aplicación móvil	91
Imagen 10: Página inicio sesión página web	106
Imagen 11: Página registro página web	106
Imagen 12: Página principal página web	107
Imagen 13: Página mis datos página web	107
Imagen 14: Página artistas página web	108
Imagen 15: Página artista página web	108
Imagen 16: Página álbum sesión página web	109
Imagen 17: Acceso a cola de canciones página web	109
Imagen 18: Página playlists página web	110
Imagen 19: Página playlist página web	110
Imagen 20: Página búsqueda página web	111
Imagen 21: Página playlist otro usuario página web	111
Imagen 22: Inicio sesión aplicación móvil	112
Imagen 23: Registro aplicación móvil	112
Imagen 24: Principal aplicación móvil	113
Imagen 25: Artistas aplicación móvil	113
Imagen 26: Artista aplicación móvil	114
Imagen 27: Álbum aplicación móvil	114
Imagen 28: Álbum con canción reproduciendo aplicación móvil	115
Imagen 29: Opciones canción aplicación móvil	115
Imagen 30: Playlists aplicación móvil	116
Imagen 31: Crear playlist aplicación móvil	116
Imagen 32: Mi cuenta aplicación móvil	117
Imagen 33: Cambiar foto perfil aplicación móvil	117
Imagen 34: Buscar aplicación móvil	118
Imagen 35: Playlist aplicación móvil	118
Imagen 36: Opciones playlist aplicación móvil	119
Imagen 37: Playlist otro usuario aplicación móvil	119
Imagen 38: Notificación con controles aplicación móvil	120
Imagen 39: Reproductor aplicación móvil	120
Imagen 40: Cola de canciones aplicación móvil	121

1. Introducción

La aplicación que se va a desarrollar se describe como una aplicación cuyo objetivo fundamental es la transmisión de música en streaming tanto para web como dispositivos móviles, esta aplicación es similar a Spotify, Apple Music, Google Play Music, Deezer... y muchas más.

Entre las funcionalidades se deben cumplir: gestión de usuarios, canciones y del servicio de streaming.

Descripción inicial del problema

- Se subirán canciones para que el usuario que quiera escucharlas pueda, de manera que se almacenará el artista, el álbum y otros campos más relacionados con la canción
- Para poder consultar estas canciones el usuario debería estar registrado y logueado en la aplicación, en las que el usuario se podrá loguear a través de un usuario y una contraseña.
- Las canciones tienen que estar asociadas a un álbum del artista, este álbum tendrá una imagen y un listado de canciones enumeradas.
- Los usuarios podrán disfrutar de las canciones de la aplicación, pero el resto de funciones como la subida de estas tiene que hacerlas el administrador.

Se desarrollará una aplicación que cumplirá con diversas funcionalidades como la gestión de usuarios, canciones y álbumes. Además se podrá actualizar la información de cada parte.

1.1 Motivación

La industria musical ha cambiado drásticamente, ahora no se venden tantas copias físicas como antes. La música actualmente se escucha en dispositivos electrónicos como pueden ser móviles o ordenadores, haciendo que la escucha de las canciones sea más fácil de hacer.

Con las tecnologías actuales hacen que este proyecto sea más fácil de desarrollar haciendo que los usuarios cumplan sus necesidades.

Mi sistema propone una solución para estos usuarios que quieran escuchar música de una manera fácil y haciendo usable este sistema.

El desarrollo de este proyecto es tecnológicamente desafiante y apetecible para mejorar y adentrarse en un proyecto grande.

1.2 Objetivos

Este proyecto tiene como principal objetivo desarrollar un sistema informático para la escucha y disfrute de canciones en streaming, de una manera rápida y eficiente, haciendo que la experiencia de usuario sea la mejor posible.

1.3 Estructura

En esta parte se describirá brevemente que va a contener esta memoria, en primer lugar se hablará de la planificación del proyecto y de las diversas etapas que hay en el desarrollo, además se explicará los costes asociados a este proyecto.

En el tercer capítulo, se explicará el análisis del proyecto, una etapa bastante importante en el desarrollo del proyecto, obteniendo las necesidades y funcionalidades que desean los usuarios.

En el siguiente capítulo, mostraremos el diseño previo de la aplicación, explicando los lenguajes a utilizar y mostrando el diseño de la base de datos a implementar.

En el quinto capítulo, se presentará algunos aspectos de la implementación de la plataforma y las fases de desarrollo del proyecto.

En el sexto capítulo, se manifestarán diversas conclusiones del proyecto.

En el siguiente capítulo, se mostrará la bibliografía utilizada para la realización de este proyecto.

En el octavo y último capítulo, se ofrece el anexo de la memoria, mostrando una versión final del desarrollo, también se indicará dónde se puede ver el código final del proyecto.

2. Planificación

2.1 Metodología

La metodología de ingeniería del software a aplicar es la de metodología en cascada. Esta metodología tiene diversas etapas a cumplir como son: planteamiento, iniciación, análisis, construcción, pruebas, implementación y mantenimiento.

Esta metodología tiene diversas ventajas como: estimar costes con mayor precisión, comenzar con el desarrollo del software con mayor rapidez y también se obtiene una mayor satisfacción del cliente.

Aunque también tiene diversas desventajas como son: es difícil alterar el diseño en cada etapa, es más complicado hacer cambios en las diferentes fases y es obligatorio obtener todos los requisitos iniciales.

En conclusión, veo que esta metodología es conveniente para esta aplicación.

2.2 Etapas

Este proyecto se dividió en las siguientes etapas:

2.2.1 Etapa inicial

El punto inicial es la necesidad de crear un sistema donde todo tipo de usuarios puedan disfrutar de escuchar canciones online, haciendo este de un fácil uso y manejo.

Para construir todo esto, se hizo múltiples reuniones con mi tutor para abarcarnos en este proyecto y se extraen los requisitos funcionales y los objetivos principales del sistema para cumplir las necesidades de los usuarios.

Tras terminar de hacer un análisis del sistema, se llegaría a un acuerdo sobre qué tecnologías usar. Primero se planteó la base de datos y de qué tipo desarrollarla, se planteó una base de datos de tipo relacional, en este caso PostgreSQL. También se pensó qué tipo de servidor utilizar y de qué manera hacer la interfaz del usuario.

2.2.2 Etapa de análisis

Aquí se estudia aquellos requisitos del sistema y las diversas funcionalidades.

2.2.2.1 Análisis de requerimientos

Es la primera etapa en el desarrollo del sistema, aquí se obtiene las necesidades de los usuarios implicados. Es una etapa fundamental para el desarrollo del sistema.

Se realizaron las siguientes tareas:

- Identificar los implicados
- Obtener los requisitos funcionales y no funcionales
- Descripción de actores y sus objetivos
- Clasificación de objetivos
- Diagramas de casos de uso
- Descripción de los casos de uso

2.2.2.2 Análisis de recursos

Aquí se realizó un estudio de las diversas tecnologías a necesitar.

2.2.3 Etapa de desarrollo

La parte más intensa y extensa de todas, en el que se lleva el desarrollo del sistema. Esta etapa se dividió en los siguientes objetivos:

- Diseño del diagrama entidad-relación
- Instalar Node JS junto a Express
- Instalar PostgreSQL
- Crear la base de datos
- Instalar las diversas bibliotecas
- Crear las funciones de la API que interactúan con la base de datos
- Instalar Angular y crear el frontend del sistema
- Instalar IONIC y crear la aplicación para móvil

2.2.4 Etapa de pruebas y corrección de errores

Tras la finalización de cada uno de los objetivos, se realizaron pruebas y correcciones de errores para que el sistema funcione en su totalidad.

2.2.5 Entrega del sistema

En esta fase se entrega las versiones del sistema y si se necesita corregir o añadir nuevas funcionalidades en el sistema. Es una etapa fundamental para que el producto tenga éxito.

2.2.6 Redacción de la memoria

Durante el tiempo de desarrollo del proyecto en las diversas etapas se estuvo recogiendo información y plasmando el desarrollo de este para la creación de esta memoria.

El proyecto comenzó en Enero, haciendo un total de 6 meses hasta terminarlo.

2.3 Temporización

Para poder visualizar mejor la planificación de una manera más visual, se creará un diagrama de Gantt:

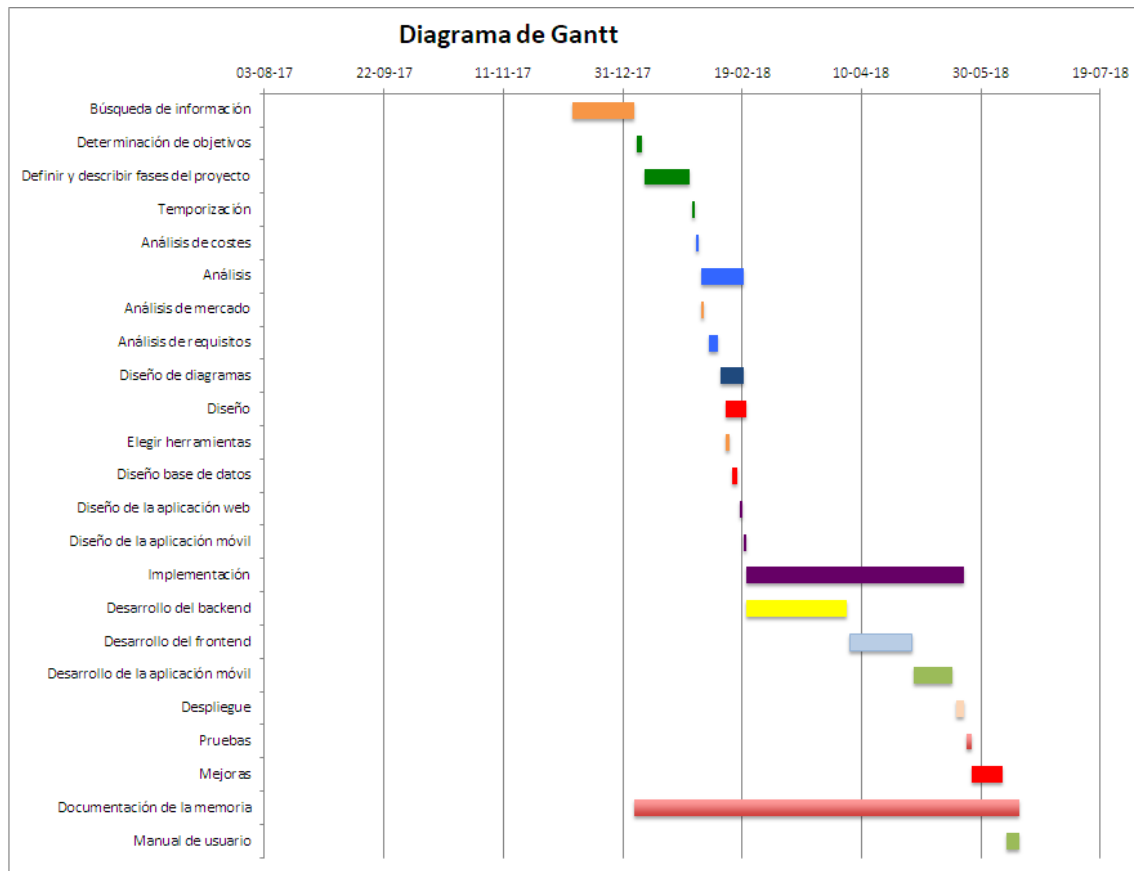


Imagen 1: Diagrama de Gantt

2.4 Presupuesto

En esta parte se describe los recursos utilizados, tanto hardware como software:

2.4.1 Recursos

2.4.1.1 Recursos hardware

Para el desarrollo de este sistema se ha utilizado un ordenador personal, el Lenovo Z50-70, con las siguientes características:

- Intel Core i7 4500-U
- 16GB RAM
- Disco duro Samsung EVO 850 de 250 GB

2.4.1.2 Recursos software

- Sistema Operativo: Windows 10
- IDE: Visual Studio Code
- Lenguajes de programación:
 - JavaScript
 - HTML
 - CSS

2.4.2 Costes

2.4.2.1 Licencias

Aquí no se ha pagado nada de licencias, de tipo libre.

2.4.2.2 Recursos hardware

Se ha utilizado mi ordenador personal como se ha mencionado anteriormente, el Lenovo Z50-70, adquirido en 2014 con un valor de 1000 €.

2.4.2.3 Recursos humanos

Para este proyecto se miró el perfil de desarrollador Full-Stack, se encarga tanto del Backend como del Frontend:

- Desarrollador Full-Stack:
 - Coste: entre 34000 € / Año , 2833 € / Mes

El tiempo del desarrollo del proyecto ha durado un total de 6 meses, el gasto en salarios sería de un total de 16998 € .

2.4.2.4 Coste total

El coste total del desarrollo del proyecto, teniendo en cuenta tanto el hardware y el software, además del salario del desarrollador, tiene un coste total de:

Descripción	Explicación	Coste total
Licencias y software	Coste licencias	0 €
Coste de Hardware	1000 € (Equipo personal)	1000 €
Desarrollador Full-Stack	34000 € / Año , 2833 € / Mes * 6 meses	16998 €
Coste total		17998 €

3. Análisis

En esta fase se hará un análisis previo a la implementación del proyecto, se empezará por hacer un análisis de las diferentes alternativas similares de esta aplicación, terminando con un análisis de requisitos.

3.1 Análisis de mercado

Como ya se sabe, este producto no es nuevo. Esto no es una tarea fácil. Por eso, hacer un análisis de mercado previo sobre los diversos competidores actuales; de esta manera podemos mejorar las características que ellos ofrecen. Podemos destacar:

3.1.1 Spotify [1]

Spotify es un servicio multiplataforma que ofrece la reproducción vía streaming de un gran catálogo de canciones. En él puedes escuchar canciones ya sea buscando por artista, álbum o listas de reproducción. Además dispone permite escuchar música modo radio.

Este servicio se ofrece de dos maneras: la versión gratuita, en ella se escuchan las canciones con publicidad; y la de premium, de pago, quita esta limitación de la publicidad y además ofrece una mejor calidad de sonido.

3.1.2 Apple Music [2]

Al igual que Spotify, es un servicio multiplataforma que ofrece la reproducción vía streaming de un gran catálogo de canciones, en este caso de más de 40 millones de canciones.

Dispone de diversas funciones como Spotify, pero también dispone de una llamada Connect, en ella los usuarios pueden interactuar sobre videos y fotos de los artistas.

Apple Music solo dispone de la versión de pago.

3.1.3 Soundcloud [3]

Soundcloud es una plataforma que permite la distribución de audio en línea. En esta plataforma, los usuarios pueden grabar, subir, promocionar y compartir música. Está enfocada a músicos. Además cualquier usuario, puede escuchar las canciones y almacenarlas en listas de reproducción.

Dispone de diversos servicios premium de dos modos diferentes: la versión Pro, permite subir hasta seis horas de audio y tener diversas estadísticas, y la versión Pro Unlimited, el usuario puede subir horas ilimitadas de audio.

3.2 Especificación de requisitos

En esta etapa, se hará la especificación de requisitos, primeramente se identificará los implicados en el sistema, ofreciendo una breve descripción de ellos. También se hará una descripción de los requisitos más importantes a nivel de funciones que debe incluir el sistema, realizando una clasificación en categorías, a cada uno de los requisitos se le ha asignado un código y un nombre, con el fin de identificarlo fácilmente a lo largo de todo el proyecto.

3.2.1 Identificar implicados

Los usuarios directos de la aplicación son: administrador y cualquier usuario. El primero tiene como misión principal la subida de archivos musicales a la plataforma y de introducir información en la base de datos información respecto a canciones, artistas y álbumes. Cualquier usuario registrado y logueado podrá consultar estos datos y escuchar estos archivos musicales.

Resumen de los implicados

Nombre	Descripción	Tipo	Responsabilidad
Usuario	Persona interesada en música	Usuario Producto	Consultar las canciones del sistemas para su escucha
Administrador	Administrador del sistema a desarrollar	Usuario Producto	Gestiona el sistema e introducirá la información necesaria en la base de datos y subirán los archivos musicales
Artista	Persona que tiene sus canciones junto con la información de álbum y sus detalles	Usuario Sistema	Envía la información respecto a canciones y su información relacionada. No lo introduce en el sistema a desarrollar

Perfiles de los implicados

Usuario identificado

Representante	Pepe Pérez Ortega
---------------	-------------------

Descripción	Persona interesada
Tipo	Utiliza el sistema a nivel de usuario
Responsabilidades	Consultar canciones Consultar artistas Consultar álbumes Reproducir música
Criterios de éxito	Que el usuario obtenga respuesta a las consultas que hace al sistema
Implicación	A nivel usuario de la aplicación, puede ser de manera asidua o esporádica, pero sin ninguna implicación mayor al sistema
Comentarios/Cuestiones	No se sabe si conoce sistemas informáticos, pero puede tener conocimiento sobre ello

Administrador

Representante	Francisco Palomares
Descripción	Administrador del sistema a desarrollar
Tipo	Entendido del sistema y sobre la gestión de las diversas partes
Responsabilidades	Gestionar el sistema para que trabaje perfectamente e introducir información en la base de datos
Criterios de éxito	Que el sistema guarde perfectamente toda la información y suministre las consultas sin errores
Implicación	Total, es el responsable de que el sistema funcione correctamente
Comentarios/Cuestiones	Está familiarizado a los sistemas informáticos y a la información relacionada sobre canciones y artistas

Artista

Representante	Los Informáticos
Descripción	Autor de las canciones
Tipo	No utiliza el sistema de forma directa, sino

	que da información suya, de sus canciones y de sus álbumes que introduce el administrador
Responsabilidades	Dar información al administrador para registrar una canción
Criterios de éxito	
Implicación	
Comentarios/Cuestiones	

3.2.2 Requisitos funcionales

RF-1. Gestión de administradores: Se podrá dar de alta/baja a cualquier administrador del sistema, también se podrá consultar/modificar cualquiera de sus datos.

RF-1.1. Alta de administrador: Se registrará un nuevo administrador, con sus respectivos datos.

RF-1.2. Baja de administrador. Se podrá eliminar todos los datos asociados al administrador.

RF-1.3. Consultar datos del administrador. Se mostrarán los datos asociados al administrador.

RF-1.4. Modificar datos del administrador. Se podrá modificar los datos correspondientes de un administrador.

RF-2 Gestión de artistas. Se podrá dar de alta/baja a artistas, así como consultar y modificar los datos asociados.

RF-2.1. Alta de artista. El administrador registrará un nuevo artista, con sus datos asociados.

RF-2.2. Baja de un artista. El administrador podrá eliminar los datos asociados a un artista.

RF-2.3. Consultar datos de un artista. Sólo serán disponibles algunos datos según el usuario que quiera consultar los datos del respectivo artista.

RF-2.4. Modificar datos de un artista. Se podrán modificar los datos de un artista.

RF-3. Gestión de usuarios: Se podrá dar de alta/baja a cualquier usuario del sistema, también se podrá consultar/modificar cualquiera de sus datos.

RF-3.1. Alta de usuario: Se registrará un nuevo usuario, con sus respectivos datos.

RF-3.2. Baja de usuario. Se podrá eliminar todos los datos asociados al usuario.

RF-3.3. Consultar datos del usuario. Se mostrarán los datos asociados al usuario.

RF-3.4. Modificar datos del usuario. Se podrá modificar los datos correspondientes de un usuario.

RF-4. Gestión de canciones: Se podrá subir/eliminar cualquier canción en el sistema, también se podrá consultar/modificar cualquiera de sus datos.

RF-4.1. Subida de canción: El administrador podrá subir una canción y registrar sus datos asociados

RF-4.2. Eliminar canción. El administrador podrá eliminar la canción y eliminar sus datos asociados

RF-4.3. Consultar datos de la canción. Se mostrarán los datos asociados de la canción, y la escucha de la canción se hará de manera streaming.

RF-4.4. Modificar datos de la canción. Se podrá modificar los datos correspondientes de una canción.

RF-5. Gestión de álbumes: Se podrá dar de alta/baja cualquier álbum asociado a una canción en el sistema, también se podrá consultar/modificar cualquiera de sus datos.

RF-5.1. Alta de álbum: El administrador podrá dar de alta un álbum y registrar sus datos asociados.

RF-5.2. Eliminar álbum. El administrador podrá eliminar los datos asociados a un álbum.

RF-5.3. Consultar datos de un álbum. Se mostrarán los datos asociados del álbum.

RF-5.4. Modificar datos del álbum. Se podrá modificar los datos correspondientes de un álbum.

RF-6. Gestión de playlists: Cualquier usuario podrá dar de alta/baja cualquier playlist con un conjunto de canciones en el sistema, también se podrá consultar/modificar cualquiera de sus datos.

RF-6.1. Alta de playlist: El usuario identificado podrá dar de alta una playlisty registrar sus datos asociados junto con sus canciones asociadas.

RF-6.2. Eliminar playlist. El usuario identificado podrá eliminar los datos asociados a una playlist.

RF-6.3. Consultar datos de una playlist. Se mostrarán los datos asociados de la playlist.

RF-6.4. Modificar datos de la playlist. Se podrá modificar los datos correspondientes de una playlist.

RF-6.5. Seguir playlist. Se podrá seguir una playlist de otro usuario para poder escucharla y tener un acceso rápido.

RF-6.6. Dejar de seguir playlist. Se podrá dejar de seguir una playlist de otro usuario a la que ha sido previamente seguida.

RF-6.7. Añadir canción a playlist. Se podrá añadir una canción a la playlist para su posterior escucha.

3.2.3 Requisitos no funcionales

Aquí se incluyen algunas restricciones que afectan a los requisitos anteriores:

RNF-1. Concurrencia. Los administradores podrán trabajar simultáneamente, ya sea para introducir información en el sistema, para que los usuarios puedan consultar la misma información que es almacenada.

RNF-2. Autenticación. Se tendrá que controlar las funciones de gestión de sistema para que puedan acceder solamente las personas autorizadas (administradores).

RNF-3. El sistema proporcionará un acceso rápido al catálogo de canciones y sus datos relaciones como son los artistas y los álbumes de la base de datos, no tardando más de 2 segundos. El sistema debe soportar un volumen de datos de más de 5000 canciones y 5000 usuarios.

RNF-4. El usuario no deberá de tener un tiempo de aprendizaje mayor de 30 minutos.

RNF-5. El sistema debe tener interfaces gráficas bien integradas.

RNF-6. El sistema debe facilitar mensajes de error y de información al usuario.

RNF-7. El sistema tiene que proporcionar protección a los datos cuyo acceso sea no autorizado.

RNF-8. El sistema debe funcionar en todos los navegadores de internet y para todas las plataformas.

3.2.4 Restricciones semánticas

Aquí se especificarán las restricciones semánticas que afectan a los requisitos funcionales explicados anteriormente:

- No puede introducirse datos del artista no válidos, esto se comprobará anteriormente, y en caso de ser incorrectos, se finalizará la función.
- No puede introducirse datos del álbum no válidos, esto se comprobará anteriormente, y en caso de ser incorrectos, se finalizará la función.
- No puede introducirse datos de la canción no válidos, esto se comprobará anteriormente, y en caso de ser incorrectos, se finalizará la función.

3.2.5 Requisitos de rendimiento

La aplicación debe tener un rendimiento que no dificulte el manejo del usuario. Puede haber momentos que el sistema puede relentizarse unos pocos segundos hasta que el proceso finalice.

3.2.6 Requisitos de información

En este apartado se mostrará la información que es necesaria almacenar en el sistema:

- **RI-1 Usuarios:** Información relacionada a las personas que usan el sistema:
 - Contenido: nombre, apellidos, contraseña(encryptada), email, rol e imagen.
 - Requisitos asociados: todos los correspondientes a RF-3.

- **RI-2 Artistas:** Información relacionada a los artistas que pertenecen al sistema:
 - Contenido: nombre, descripción e imagen.
 - Requisitos asociados: todos los correspondientes a RF-2.

- **RI-3 Álbumes:** Información relacionada a los álbumes que pertenecen a los artistas:
 - Contenido: título, descripción, imagen, imagen, año y artista relacionado.
 - Requisitos asociados: todos los correspondientes a RF-5.

- **RI-4 Canciones:** Información relacionada a las canciones que pertenecen a los artistas:
 - Contenido: número de canción en el álbum, nombre, duración, archivo y canciones que pertenecen a él.
 - Requisitos asociados: todos los correspondientes a RF-4.

- **RI-5 Playlists:** Información relacionada a las playlists (listas de reproducción) creada por un usuario:
 - Contenido: nombre, usuario creador, canciones que pertenecen a ella y usuarios que siguen la playlist.
 - Requisitos asociados: todos los correspondientes a RF-6.

3.3 Modelos de casos de uso

Para describir los diferentes comportamientos que tiene el sistema, se usará el lenguaje de modelado de UML, aquí se figuran los requisitos funcionales del sistema:

3.3.1 Descripción de sus actores y sus objetivos

Aquí se explican los diferentes actores que ejecutan las diferentes funciones del sistema, mediante el uso de una interfaz manejable.

Actor	1 Administrador				AC-01
Descripción	Actor que se encarga de la gestión del sistema como pueden ser la actualización, borrado e inserción de datos relacionados con los artistas, álbumes y playlists.				
Características	Es el actor más importante del sistema y es el encargado de que los datos del sistema estén actualizados.				
Relaciones	Relacionado con los artistas, álbumes y canciones				
Referencias					
Autor	Francisco Jesús Palomares	Fecha	17/02/2018	Versión	1.0

Atributos		
Nombre	Descripción	Tipo
nombre	Nombre de la persona encargada de ese rol	Cadena
apellidos	apellidos de la persona encargada de ese rol	Cadena
Archivo foto	Ruta de la imagen de la persona encargada	Cadena
contraseña	Cadena para que la persona pueda identificarse en el sistema	Cadena
email	Usado para la autenticación de la persona	Cadena

Comentarios:
Es necesario que el administrador se identifique para hacer operaciones sobre el mantenimiento y gestión de los datos.

Objetivos del administrador:

- Altas, bajas y modificaciones de los artistas.

- Altas, bajas y modificaciones de los álbumes de los artistas.
- Altas, bajas y modificaciones de las canciones de los álbumes.

Actor	1 Usuario no registrado				AC-01
Descripción	Actor que no puede consultar los datos del sistema hasta que no se registre e identifique				
Características	Es el actor que necesita escuchar música				
Relaciones	Relacionado con los administradores, artistas, álbumes y canciones				
Referencias					
Autor	Francisco Jesús Palomares	Fecha	17/02/2018	Versión	1.0

Atributos		
Nombre	Descripción	Tipo

Comentarios:
Es necesario que se registre para acceder a la información de la plataforma

Objetivos del usuario no registrado:

- Registrarse para acceder a la plataforma y disfrutar de la música.

Actor	2 Usuario identificado	AC-02
Descripción	Actor principal, cualquier persona que quiere acceder a la información del sistema	

Características	Actor que realiza todas las consultas que el sistema puede ofrecer				
Relaciones	Relacionado con los administrador, artistas, canciones y álbumes				
Referencias					
Autor	Francisco Jesús Palomares	Fecha	17/02/2018	Versión	1.0

Atributos		
Nombre	Descripción	Tipo
nombre	Nombre de la persona	Cadena
apellidos	apellidos de la persona	Cadena
Archivo foto	Ruta de la imagen de la persona	Cadena
contraseña	Cadena para que la persona pueda identificarse en el sistema	Cadena
email	Usado para la autenticación de la persona	Cadena

Comentarios:
Es necesario que el usuario se identifique para hacer las consultas que el sistema ofrece

Objetivos del usuario identificado:

- Realizar consultas de los artistas.
- Realizar consultas álbumes de los artistas.
- Realizar consultas de las canciones de los álbumes.
- Alta, modificación, baja y consulta de playlists de canciones cuyo creador sea el usuario identificado.

Actor	3 Artista				AC-03
Descripción	Actor secundario, cuyo papel es dar información suya al administrador				
Características	Actor que proporciona información sobre él y sobre sus álbumes y canciones y que el administrador tiene que introducir en el sistema				
Relaciones	Relacionado con los administrador				
Referencias					
Autor	Francisco Jesús Palomares	Fecha	17/02/2018	Versión	1.0

Atributos		
Nombre	Descripción	Tipo
nombre	Nombre de la persona	Cadena
descripción	apellidos de la persona	Cadena
Archivo foto	Ruta de la imagen de la persona	Cadena

Comentarios:
Es necesario que el artista mande la información al administrador para introducirla en el sistema de la mejor manera.

Objetivos del usuario identificado:

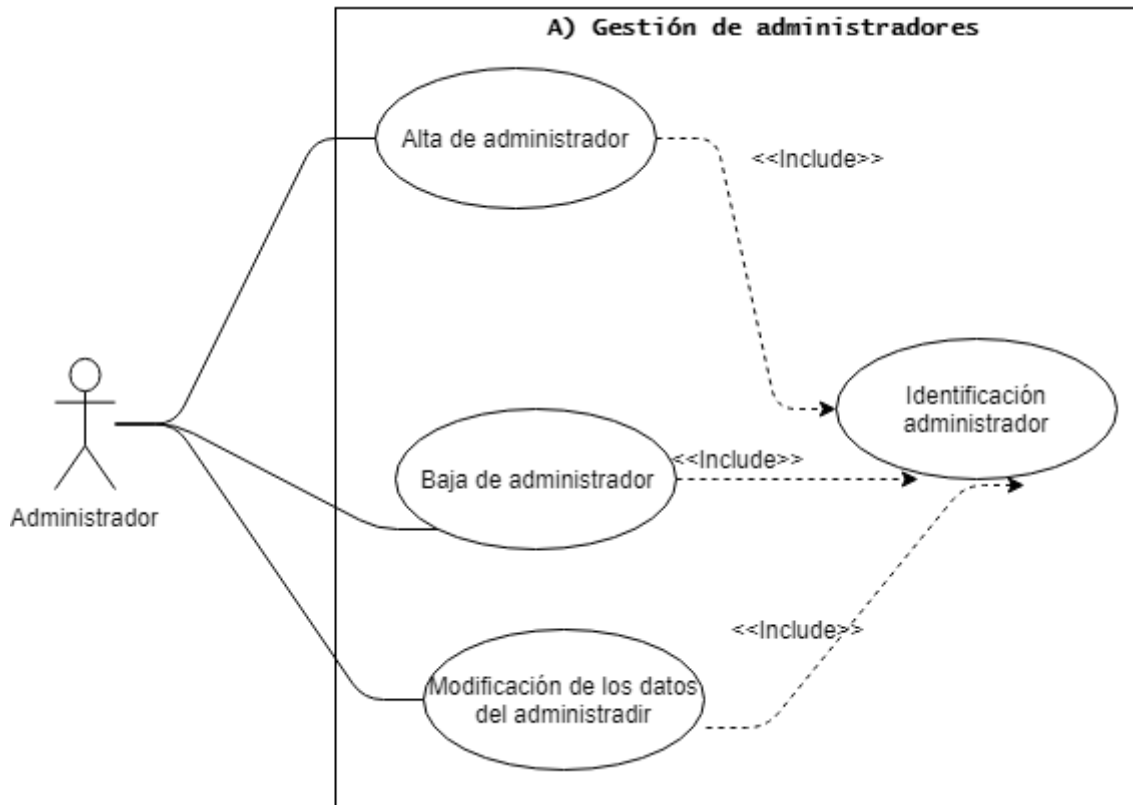
- Enviar información personal.
- Enviar datos sobre sus álbumes.
- Enviar datos y archivos sonoros sobre las canciones de los álbumes.

3.3.2 Clasificación de objetivos

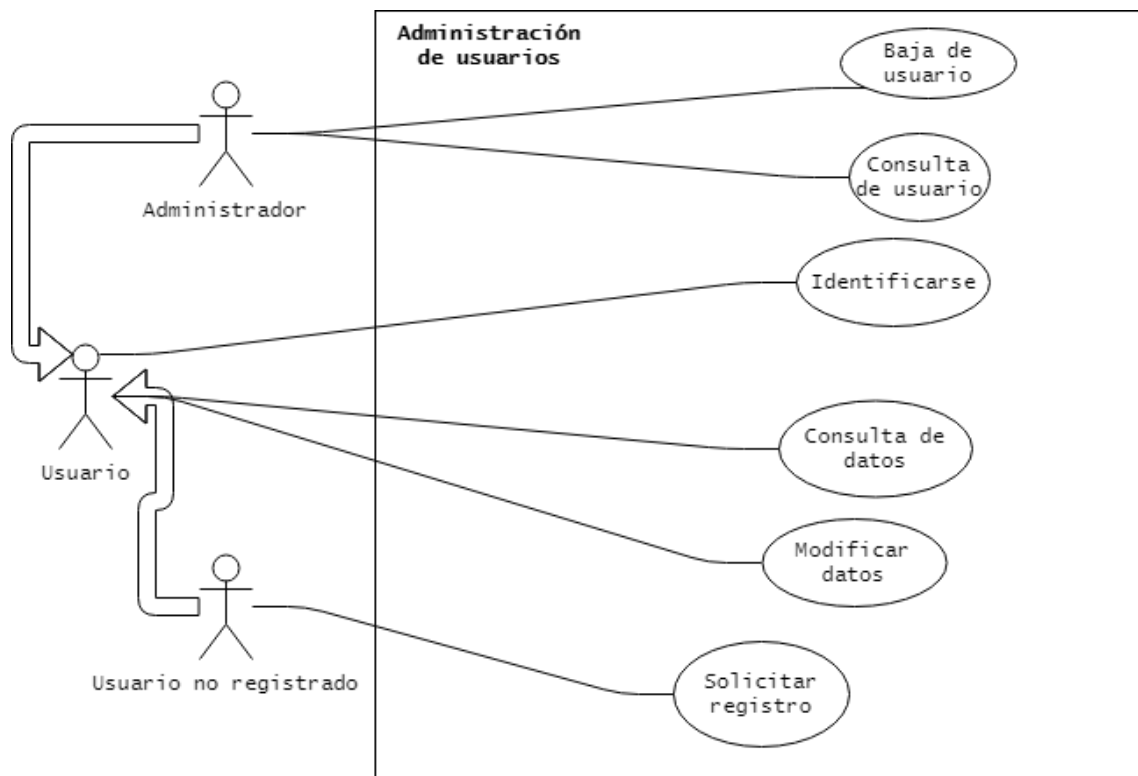
- Relacionados con la gestión de los administradores.
- Relacionados con la administración de usuarios.
- Relacionados con la gestión de artistas.
- Relacionados con la gestión de álbumes.
- Relacionados con la gestión de canciones.
- Relacionados con la gestión de playlists.

3.3.3 Diagramas de casos de uso

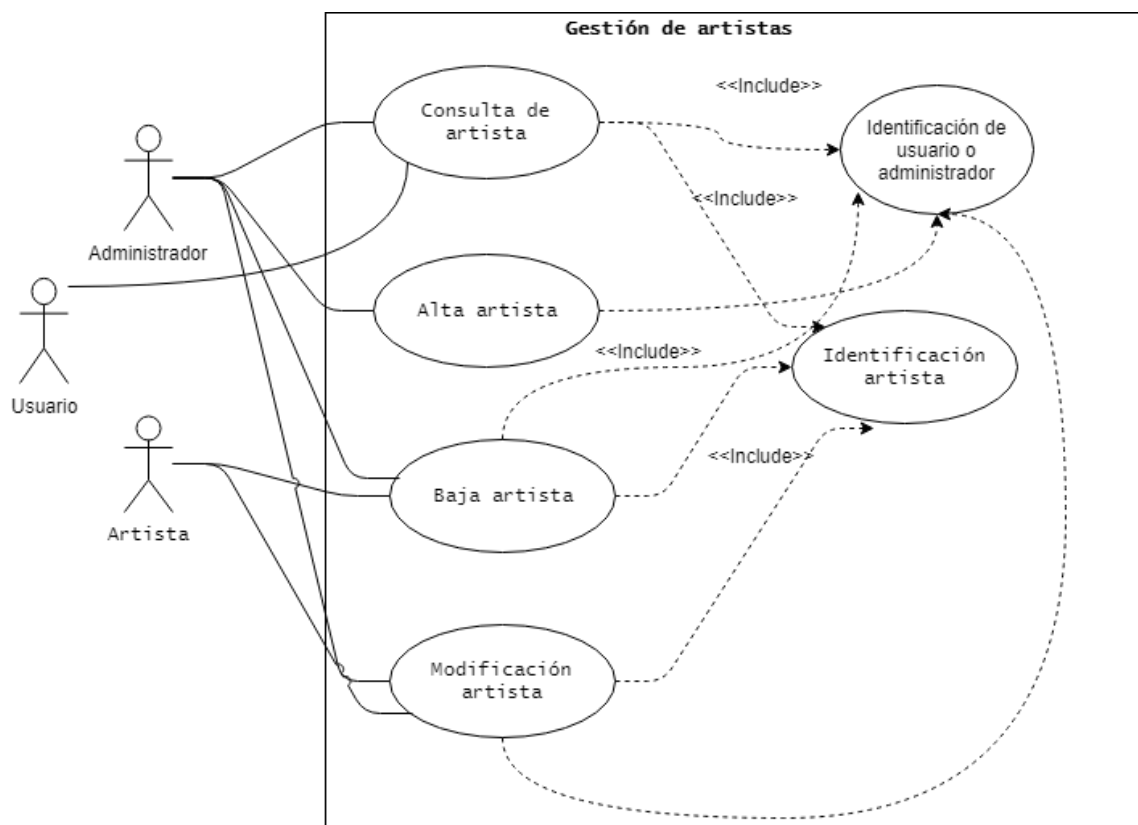
A) Gestión de administradores



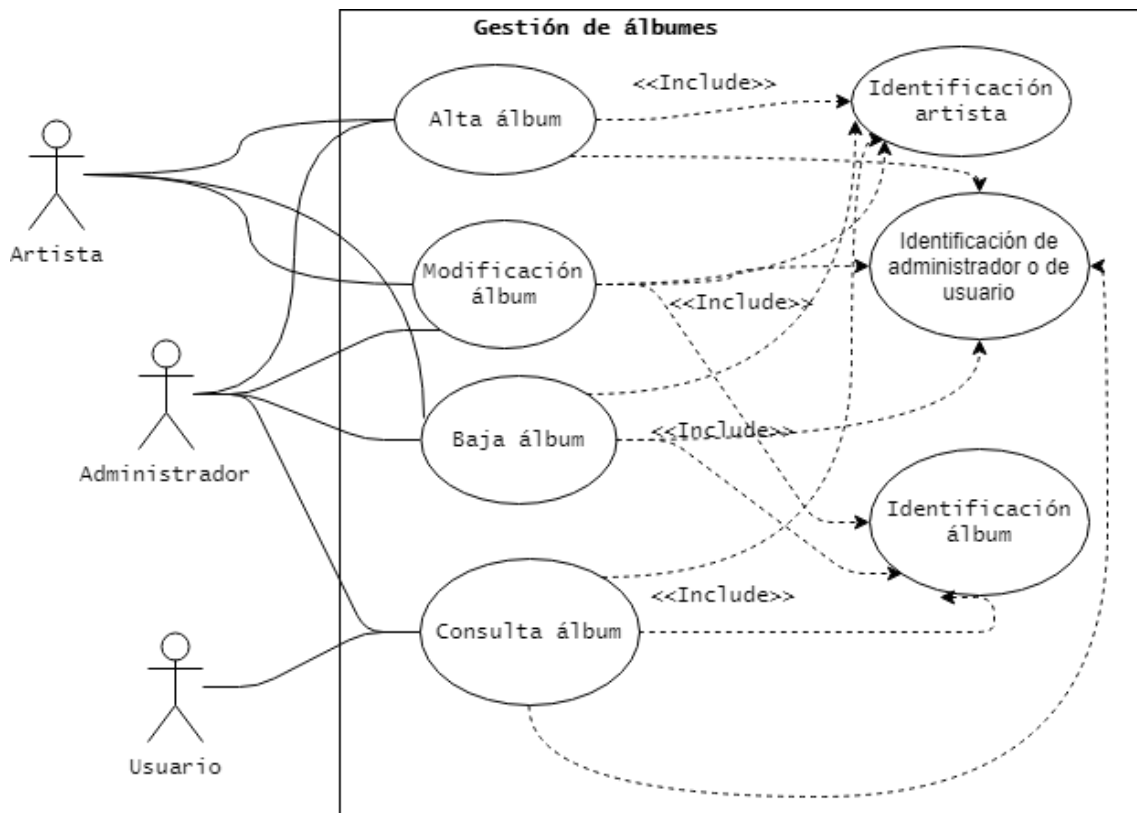
B) Gestión de usuarios



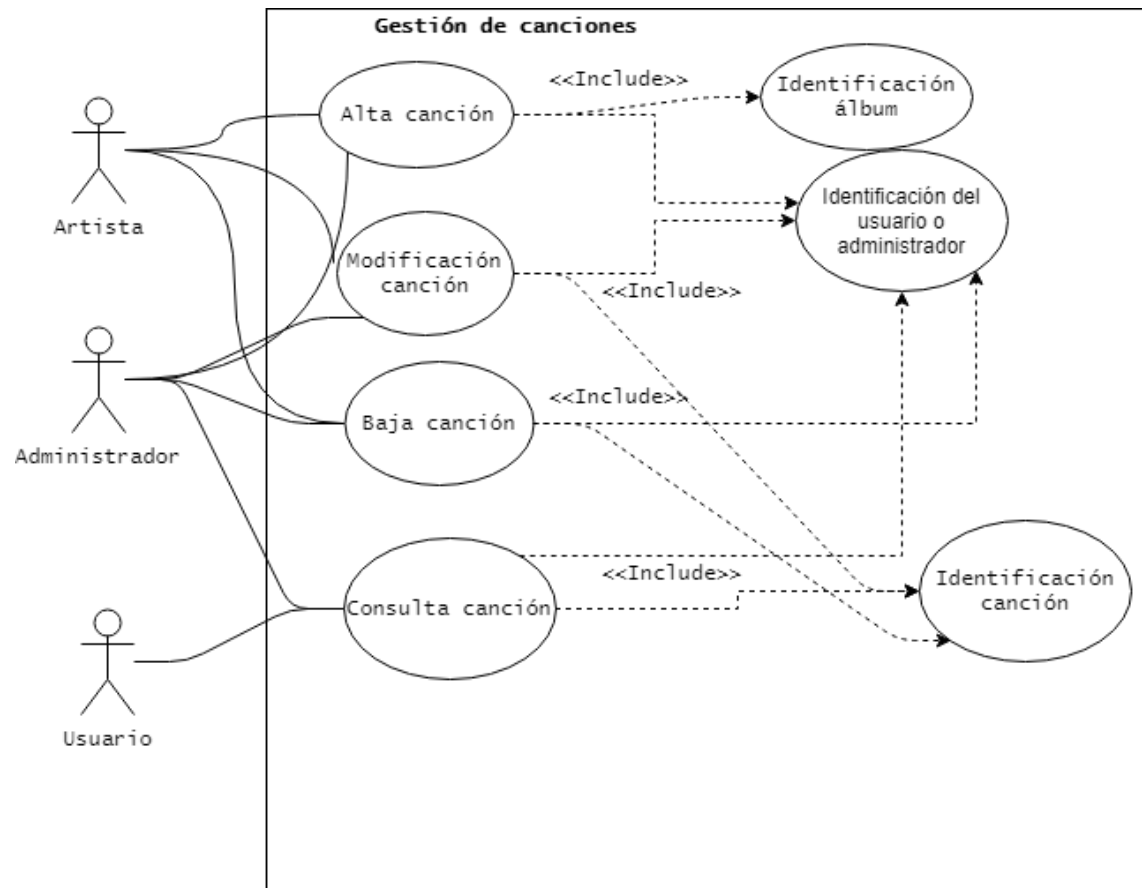
C) Gestión de artistas



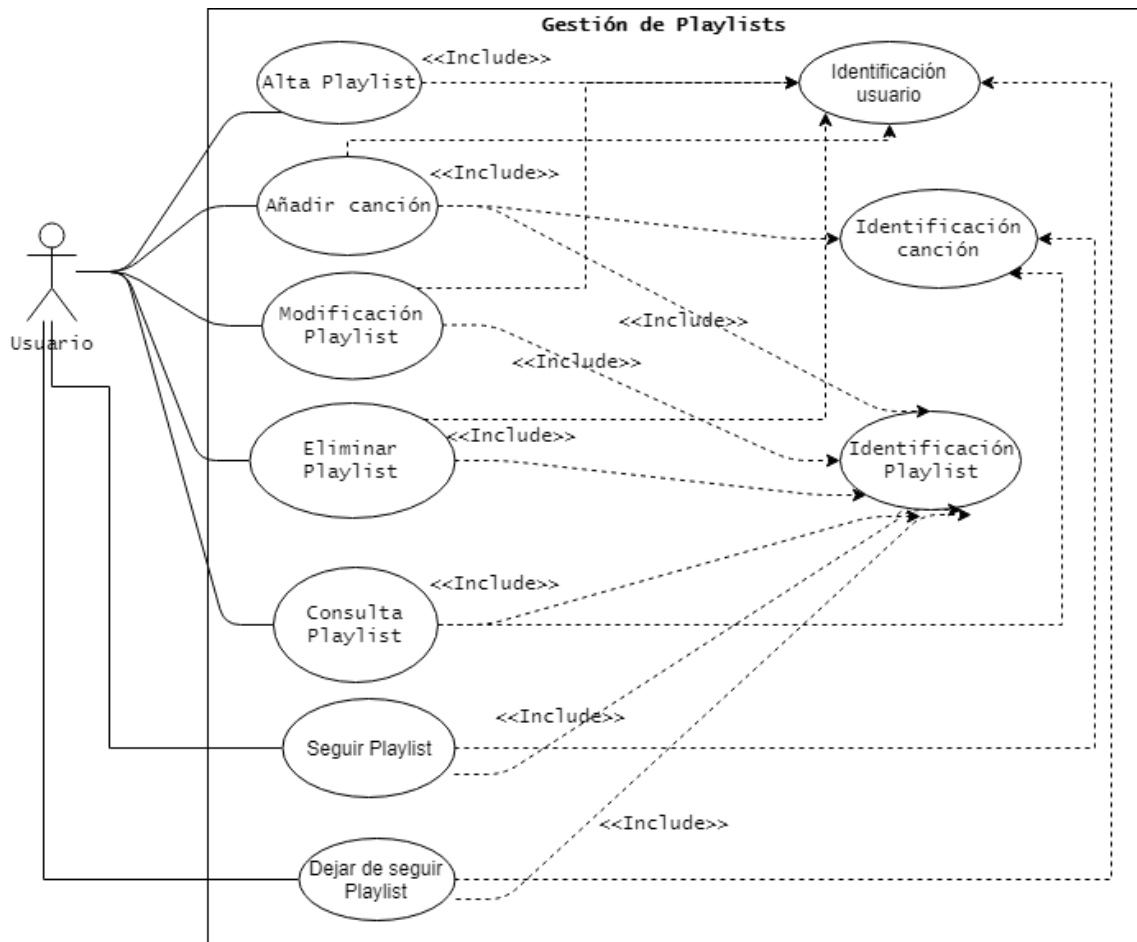
D) Gestión de álbumes



E) Gestión de canciones



F) Gestión de Playlists



3.3.4 Descripción de casos de uso

A) GESTIÓN DE ADMINISTRADORES:

Caso de Uso	Alta de Administrador	CU-0.1
Actores	Administrador (P)	
Tipo	Primario, Esencial	
Precondición		
Poscondición	Se creará un nuevo administrador en el sistema	
Propósito		
Crear un nuevo administrador		
Descripción		
Crear un nuevo administrador en el sistema, con la información adecuada para que lleve sus tareas de administrador pudiéndose identificar		

Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0
--------------	---------------------------	-----------	--------------	-----------------	-----

Curso Normal			
1	Administrador: Pide ser usuario de la plataforma		
		2	Solicita los datos personales del cliente
4	Administrador: Introduce los datos correspondientes		
5	Administrador: Comprueba los datos presentados		
6	Administrador: Solicita al sistema que almacene los datos		
		7	Comprueba si su email corresponde a unos de los emails para ser administrador
		8	Almacena los datos correspondientes e informa de que se ha terminado el proceso
9	Recibe un aviso de que el proceso ha terminado		

Cursos Alternos	
5a	Sistema: Si los datos del usuario son incorrectos se le piden los datos necesarios, si no son aportados por el cliente se finaliza el proceso de alta y se termina el CU
7a	Sistema: Si el administrador ya es usuario de la plataforma se le informa de ello y se termina el proceso de alta de administrador.

Caso de Uso	Baja de Administrador	CU-0.2
Actores	Administrador (P)	
Tipo	Primario, Esencial	
Precondición	El administrador pertenece al sistema	

Poscondición		Se eliminará el administrador del sistema			
Propósito					
Eliminar administrador registrado anteriormente					
Descripción					
Se elimina un administrador del sistema, en el que no formará parte de él					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador: Indica que quiere darse de baja en la plataforma		
		2	Identificación del administrador. Incluir (identificación del administrador)
		3	Eliminar la información asociada al administrador
		4	Informar al administrador que se ha dado de baja
5	Recibe aviso de dado de baja		

Cursos Alternos	

Caso de Uso	Modificación datos de Administrador	CU-0.3
Actores	Administrador (P)	
Tipo	Primario, Esencial	

Precondición	El administrador pertenece en el sistema				
Poscondición	Se modificarán los datos de un administrador				
Propósito					
Modificar datos de un administrador					
Descripción					
Se modificará los datos de un administrador previamente identificado en el sistema, se asegurará que la información quede bien almacenada					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador: Indica que han cambiado algunos de sus datos personales		
		2	Identificación del administrador. Incluir (identificación del administrador)
		3	Presenta sus datos personales actuales almacenados
4	Aporta e introduce sus nuevos datos personales		
		5	Se almacenan los nuevos datos del administrador
		6	Se informa de que el proceso ha acabado correctamente
7	Recibe el aviso de que el proceso ha terminado correctamente		

Cursos Alternos	
5a	Sistema: Si los datos del usuario son incorrectos se le piden los datos necesarios, si no son aportados por el administrador se finaliza el proceso de modificación de datos y se termina el CU

Caso de Uso	Identificación de Administrador	CU-0.4			
Actores	Administrador (P)				
Tipo	Primario, Esencial				
Precondición					
Poscondición					
Propósito					
Identificar a un administrador					
Descripción					
Se identificará un administrador a través de su email y su contraseña					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador: Solicita la identificación		
2	Introduce los datos de identificación, correo y contraseña		
		3	Se buscan los datos
		4	Se identifica al administrador y se devuelve un aviso de que el proceso se ha finalizado correctamente
5	Recibe el aviso y entra en la plataforma		

Cursos Alternos

2a	El administrador no recuerda su contraseña de acceso: 1. Administrador: No recuerda su contraseña, y pincha en se ha olvidado la contraseña 2. Sistema: Envía email con acceso a cambiar su contraseña 3. Administrador: Accede al enlace y cambia la contraseña actual
4a	El administrador no pertenece al sistema: 1. Sistema: Informa de ello

B) GESTIÓN DE USUARIOS:

Caso de Uso	Solicitar registro	CU-1.1			
Actores	Usuario no registrado (P)				
Tipo	Primario, Esencial				
Precondición	El usuario no se ha registrado nunca, registro único por email				
Poscondición	Se creará un nuevo usuario				
Propósito					
Crear un nuevo usuario en el sistema					
Descripción					
Se creará un nuevo usuario, introduciendo sus datos personales, más sus datos de identificación como son el email y la contraseña					
Autor	Francisco Jesús Palomares	Fec ha	15-03-18	Ver sió n	1.0

Curso Normal			
1	Usuario no registrado: Solicita ser registrado en el sistema		
		2	Solicita datos personales
3	Usuario no registrado: aporta los datos pedidos para ser registrado (Nombre,apellidos,email,contraseña)		
		4	Comprueba que los datos son correctos

		5	Almacena en el sistema y da un mensaje de que el proceso ha ido con éxito
6	Recibe un aviso y un email		

Cursos Alternos	
4a	Los datos no pueden ser correctos: <ul style="list-style-type: none"> El usuario escribe mal los datos: se piden de nuevo y si no los introduce se acaba el CU. El email está registrado, se pide un nuevo email

Caso de Uso	Identificación de usuario	CU-1.2			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	El usuario tiene que estar registrado en el sistema				
Poscondición					
Propósito					
Identificar a un usuario					
Descripción					
Se identificará un usuario a través de su correo y contraseña					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario registrado: Solicita identificarse en el sistema		

		2	Solicita email y contraseña.
3	Usuario registrado: aporta el email y la contraseña		
		4	Comprueba que los datos son correctos
		5	Manda un aviso de cómo ha finalizado el proceso
6	Recibe un aviso y entra en el sistema		

Cursos Alternos	
4a	Los datos no pueden ser correctos: <ul style="list-style-type: none"> El usuario escribe mal los datos: se piden de nuevo y si no los introduce se acaba el CU. El email no concuerda con la contraseña, se piden de nuevo los datos

Caso de Uso	Modificar datos de usuario	CU-1.3			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	El usuario pertenece al sistema y tiene que estar identificado				
Poscondición	Se modificarán los dato del usuario				
Propósito					
Modificar datos de un usuario					
Descripción					
Se modificarán los datos de un usuario identificado a través de su correo y contraseña, estos datos se almacenarán correctamente					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió	1.0

				n	
--	--	--	--	---	--

Curso Normal			
1	Usuario: Solicita la modificación de sus datos		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Presenta los datos actuales del usuario
4	Aporta los nuevos datos		
		5	Se comprueban los datos y se almacenan en el sistema
		6	Se informa de cómo el proceso ha finalizado
7	El usuario recibe el aviso		

Cursos Alternos	
5a	Si los datos nuevos no son correctos se informa al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Consulta de datos	CU-1.4
Actores	Usuario (P)	
Tipo	Primario, Esencial	
Precondición	El usuario tiene que estar identificado a través de su correo y contraseña	
Poscondición		
Propósito		

Un usuario podrá consultar sus datos					
Descripción					
Se consultarán los datos de un usuario identificado					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario: Solicita la consulta de sus datos		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Presenta los datos actuales del usuario
4	Recibe los datos actuales		

Cursos Alternos	

Caso de Uso	Baja de un usuario	CU-1.5
Actores	Administrador (P)	
Tipo	Primario, Esencial	
Precondición	El usuario pertenece al sistema, al igual que el usuario a eliminar	
Poscondición	Se eliminará un usuario del sistema	
Propósito		
Se eliminará un usuario a través de un administrador		
Descripción		
Se identificará un administrador a través de su email y su contraseña, y podrá eliminar		

los datos de un usuario					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario: Solicita la baja en el sistema		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Eliminar toda la información relacionada con el usuario
		4	Se informa al antiguo usuario de que se ha eliminado su información
5	Recibe el aviso		

Cursos Alternos	

Caso de Uso	Consulta de un usuario	CU-1.6
Actores	Administrador (P)	
Tipo	Primario, Esencial	
Precondición	El administrador tiene que estar identificado correctamente	
Poscondición		
Propósito		
Consultar los datos de un usuario a través de una administrador		
Descripción		
Un administrador identificado correctamente podrá consultar los datos de un usuario específico.		

Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0
--------------	---------------------------	-----------	--------------	-----------------	-----

Curso Normal			
1	Administrador: Solicita consulta de datos de un usuario		
		2	Realiza identificación del usuario. Incluir (identificación de usuario) es la misma que de administrador
		3	Pide que usuario quiere consultar
4	Aporta el email, nombre y/o apellidos del usuario		
		5	Se comprueban los datos que sean correctos
		6	Ofrece los datos del usuario a consultar
7	El administrador recibe los datos		

Cursos Alternos	
5a	Si los datos no son correctos se informa al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

C) GESTIÓN DE ARTISTAS:

Caso de Uso	Alta de artista	CU-2.1
Actores	Administrador (P), Artista (S)	
Tipo	Primario, Esencial	
Precondición	El artista tiene que tener todos sus papeles en regla	
Poscondición	Se creará un nuevo artista	

Propósito					
Crear un nuevo artista en el sistema					
Descripción					
Se creará un nuevo artista con los datos apropiados					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita ser añadido en el sistema		
2	Administrador: Recibe los datos del artista a dar de alta		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
4	Administrador: solicita dar de alta un artista		
		5	Ofrece formulario de artista
6	Introduce los datos del artista		
		7	Se comprueban que son correctos y se almacenan en el sistema
		8	Da aviso de que el proceso ha finalizado correctamente
9	Recibe el aviso del proceso		

Cursos Alternos	
7a	Si los datos nuevos no son correctos se informa al administrador, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Baja de artista	CU-2.2			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	El artista tiene que pertenecer al sistema				
Poscondición	Se eliminará el artista				
Propósito					
Eliminar los datos de un artista					
Descripción					
Eliminar los datos de un artista que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita ser dado de baja en el sistema		
2	Administrador: Recibe el aviso del artista		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
4	Administrador: solicita dar de baja un artista		
		5	Ofrece formulario de artista a dar de baja
6	Introduce los datos del artista		
		7	Se comprueban que son correctos y se identifica al artista. Incluir (identificación de artista)
		8	Eliminar los datos asociados a un artista y dar un aviso sobre la finalización del proceso
9	Recibe el aviso del proceso		

Cursos Alternos	
7a	Si los datos nuevos no son correctos se informa al administrador, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Modificación datos artista	CU-2.3			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	El artista tiene que pertenecer al sistema				
Poscondición	Se modificarán los datos de un artista				
Propósito					
Modificar datos de un artista					
Descripción					
Se modificarán los datos de un artista que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita modificar datos suyos en el sistema y aporta los nuevos datos		
2	Administrador: Recibe el aviso del artista y recibe los nuevos datos		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
4	Administrador: solicita modificar a un		

	artista		
		5	Ofrece formulario de artista a modificar
6	Introduce los nuevos datos del artista		
		7	Se comprueban que son correctos y se identifica al artista. Incluir(identificación de artista)
		8	Almacenar los nuevos datos asociados a un artista y dar un aviso sobre la finalización del proceso
9	Recibe el aviso del proceso		

Cursos Alternos	
7a	Si los datos nuevos no son correctos se informa al administrador, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Consulta de datos de un artista	CU-2.4			
Actores	Administrador (P), Usuario (P)				
Tipo	Primario, Esencial				
Precondición	El artista tiene que pertenecer al sistema				
Poscondición					
Propósito					
Se consultarán los datos relacionados a un artista					
Descripción					
Consultar los datos relacionados a un artista que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita consultar datos de un usuario		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Ofrece formulario para pedir el artista a consultar
4	Administrador o usuario: aporta algún dato del artista		
		7	Se comprueban que son correctos y se identifica al artista. Incluir(identificación de artista)
		8	Se ofrecen los datos del artista y dar un aviso sobre la finalización del proceso
9	Recibe los datos del artista y el aviso del proceso		

Cursos Alternos	
7a	Si los datos no son correctos se informa al administrador o al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Identificación de un artista	CU-2.5
Actores	Administrador (P), Usuario (P)	
Tipo	Primario, Esencial	
Precondición	El artista tiene que pertenecer al sistema	
Poscondición		
Propósito		
Se identificará un artista del sistema		
Descripción		

Identificar un artista que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fecha	15-03-18	Versión	1.0

Curso Normal			
1	Administrador o usuario: Solicita consultar datos de un artista		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Pide el id del artista
4	Introduce el id del artista (puede ser vía url)		
		5	Se comprueban que el id es correcto y que pertenece a un artista
		6	Se ofrecen los datos del artista y dar un aviso sobre la finalización del proceso
7	Recibe los datos del artista y el aviso del proceso		

Cursos Alternos	
5a	Si los el id del artista se informa al administrador o al usuario, y se le pide nuevos un id correcto. Si no es proporcionado, se acaba el CU.

D) GESTIÓN DE ÁLBUMES:

Caso de Uso	Alta Álbum	CU-3.1
Actores	Administrador (P), Artista (S)	
Tipo	Primario, Esencial	

Precondición					
Poscondición	Crear un álbum				
Propósito					
Crear un álbum					
Descripción					
Se creará un álbum con su información relacionada					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita la creación de un nuevo álbum y aporta los datos de él		
2	Administrador: Recibe los datos del nuevo álbum de un artista y solicita darlo de alta		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
		4	Comprueba los datos del artista y Identifica al artista (Incluir identificación de artista) y ofrece el formulario de creación de álbum
5	Administrador: Recibe el formulario y aporta los datos del nuevo		
		6	Recibe los datos del nuevo álbum y lo comprueba, y los almacena en el sistema
		7	Ofrece un aviso de que el proceso ha finalizado correctamente
8	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del artista no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
6a	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Modificación Álbum	CU-3.2			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	El álbum tiene que pertenecer al sistema				
Poscondición	Se modificarán los datos de un álbum				
Propósito					
Modificar los datos de un álbum					
Descripción					
Se modificará los datos de un álbum que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita la modificación de uno de sus álbumes y aporta los datos de él		
2	Administrador: Recibe los datos del álbum de un artista y solicita modificarlo		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
		4	Comprueba los datos del artista y Identifica al artista (Incluir identificación de artista) y ofrece el

			formulario de modificación de álbum, identifica el álbum. Incluir (identificación del álbum).
5	Administrador: Recibe el formulario y aporta los datos del nuevo		
		6	Recibe los datos del álbum y lo comprueba, y los almacena en el sistema
		7	Ofrece un aviso de que el proceso ha finalizado correctamente
8	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del artista no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
6a	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Baja de Álbum	CU-3.3			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	El álbum a eliminar tiene que pertenecer al sistema				
Poscondición	Se eliminarán los datos relacionados a un álbum				
Propósito					
Dar de baja un álbum					
Descripción					
Se eliminará la información relacionada a un álbum					
Autor	Francisco Jesús Palomares	Fec ha	15-03-18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita dar de baja un álbum y aporta los datos de él		
2	Administrador: Recibe los datos del álbum de un artista y solicita darlo de baja		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
		4	Comprueba los datos del artista y Identifica al artista (Incluir identificación de artista) e identifica el álbum (Incluir identificación de álbum)
		5	Elimina los datos asociados al álbum
		6	Ofrece un aviso de que el proceso ha finalizado correctamente
7	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del artista no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
4b	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Consulta álbum	CU-3.4
Actores	Administrador (P), Usuario (P)	
Tipo	Primario, Esencial	
Precondición	El álbum tiene que pertenecer al sistema	

Poscondición					
Propósito					
Consultar datos de un álbum					
Descripción					
Se consultará los datos relacionados a un álbum que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita consultar datos de un álbum		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Ofrece formulario para pedir el álbum a consultar
4	Administrador o usuario: aporta algún dato del álbum		
		7	Se comprueban que son correctos y se identifica al álbum. Incluir(identificación de artista)
		8	Se ofrecen los datos del álbum y dar un aviso sobre la finalización del proceso
9	Recibe los datos del álbum y el aviso del proceso		

Cursos Alternos	
7a	Si los datos no son correctos se informa al administrador o al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Identificación álbum	CU-3.5			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	El álbum pertenece al sistema				
Poscondición					
Propósito					
Identificar un álbum					
Descripción					
Identificar un álbum que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita consultar datos de un álbum		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Pide el id del álbum
4	Introduce el id del álbum (puede ser vía url)		
		5	Se comprueban que el id es correcto y que pertenece a un álbum
		6	Se ofrecen los datos del álbum y dar un aviso sobre la finalización del proceso
7	Recibe los datos del álbum y el aviso del proceso		

Cursos Alternos	
5a	Si el id del álbum no es correcto, se informa al administrador o al usuario, y se le pide nuevos un id correcto. Si no es proporcionado, se acaba el CU.

E) GESTIÓN DE CANCIONES:

Caso de Uso	Alta canción	CU-4-1			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición					
Poscondición	Crear una canción				
Propósito					
Crear canción					
Descripción					
Dar de alta una canción y guardar su información relacionada en el sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita la creación de una nueva canción y aporta los datos de ella		
2	Administrador: Recibe los datos de una nueva canción de un artista y solicita darla de alta		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
		4	Comprueba los datos del álbum y

			Identifica al álbum (Incluir identificación de álbum) y ofrece el formulario de creación de creación
5	Administrador: Recibe el formulario y aporta los datos del nuevo		
		6	Recibe los datos de la nueva canción y lo comprueba, y los almacena en el sistema
		7	Ofrece un aviso de que el proceso ha finalizado correctamente
8	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
6a	Si los datos de la canción no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Modificación de canción	CU-4.2
Actores	Administrador (P), Artista (S)	
Tipo	Primario, Esencial	
Precondición	La canción tiene que pertenecer al sistema	
Poscondición	Modificación datos de una canción	
Propósito		
Modificar datos de una canción		

Descripción					
Modificación la información relacionada a una canción que pertenece al sistema					
Autor	<i>Francisco Jesús Palomares</i>	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Artista: Solicita la modificación de una de sus canciones y aporta los datos de ella		
2	Administrador: Recibe los datos de la canción de un álbum y solicita modificarlo		
		3	Realiza identificación del administrador. Incluir (identificación de administrador)
		4	Comprueba los datos del álbum y Identifica al álbum (Incluir identificación de álbum) y ofrece el formulario de modificación de canción, identifica la canción. Incluir (identificación de la canción).
5	Administrador: Recibe el formulario y aporta los datos del nuevo		
		6	Recibe los datos de la canción y lo comprueba, y los almacena en el sistema
		7	Ofrece un aviso de que el proceso ha finalizado correctamente
8	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

6a	Si los datos de la canción no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
-----------	---

Caso de Uso	Baja canción	CU-4.3			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	La canción tiene que pertenecer al sistema				
Poscondición	Eliminar datos de una canción				
Propósito					
Eliminar una canción					
Descripción					
Eliminar la información relacionada a una canción que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal					
1	Artista: Solicita dar de baja una canción y aporta los datos de él				
2	Administrador: Recibe los datos del álbum de una canción y solicita darlo de baja				
		3	Realiza identificación del administrador. Incluir (identificación de administrador)		
		4	Comprueba los datos del álbum y Identifica al álbum (Incluir identificación de álbum) e identifica la canción (Incluir identificación de canción)		
		5	Elimina los datos asociados a la canción		
		6	Ofrece un aviso de que el proceso		

			ha finalizado correctamente
7	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos del álbum no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.
4b	Si los datos de la canción no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Consulta de una canción	CU-4.3			
Actores	Administrador (P), Usuario (P)				
Tipo	Primario, Esencial				
Precondición	La canción pertenece al sistema				
Poscondición					
Propósito					
Consultar una canción					
Descripción					
Consultar los datos relacionados de una canción que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita		

	consultar datos de una canción		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Ofrece formulario para pedir la canción a consultar
4	Administrador o usuario: aporta algún dato de la canción		
		7	Se comprueban que son correctos y se identifica la canción. Incluir (identificación de canción)
		8	Se ofrecen los datos de la canción y dar un aviso sobre la finalización del proceso
9	Recibe los datos de la canción y el aviso del proceso		

Cursos Alternos	
7a	Si los datos no son correctos se informa al administrador o al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Identificación canción	CU-4.5			
Actores	Administrador (P), Artista (S)				
Tipo	Primario, Esencial				
Precondición	La canción pertenece al sistema				
Poscondición					
Propósito					
Identificar una canción					
Descripción					
Identificar una canción que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03-18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita consultar datos de una canción		
		2	Realiza identificación del administrador o de usuario. Incluir (identificación de administrador o usuario)
		3	Pide el id de la canción
4	Introduce el id de la canción (puede ser vía url)		
		5	Se comprueban que el id es correcto y que pertenece a una canción
		6	Se ofrecen los datos de la canción y dar un aviso sobre la finalización del proceso
7	Recibe los datos del canción y el aviso del proceso		

Cursos Alternos	
5a	Si el id de la canción no es correcto, se informa al administrador o al usuario, y se le pide nuevos un id correcto. Si no es proporcionado, se acaba el CU.

F) GESTIÓN DE PLAYLISTS:

Caso de Uso	Alta Playlist	CU-5.1
Actores	Usuario (P)	
Tipo	Primario, Esencial	
Precondición		
Poscondición	Crear una playlist	

Propósito					
Crear una playlist					
Descripción					
Crear una playlist y almacenar sus datos correctamente					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario: Solicita la creación de una nueva playlist		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Ofrece el formulario de creación de playlist
4	Administrador: Recibe el formulario y aporta los datos de la nueva playlist		
		5	Recibe los datos de la nueva playlist y lo comprueba, y los almacena en el sistema
		6	Ofrece un aviso de que el proceso ha finalizado correctamente
7	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
5a	Si los datos de la playlist no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Añadir canción a una playlist	CU-5.2
-------------	-------------------------------	--------

Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	La canción pertenece al sistema y la playlist pertenece al usuario				
Poscondición	Se añadirá una canción a la playlist				
Propósito					
Añadir canción a una playlist					
Descripción					
Añadir una canción que pertenece al sistema a una playlist que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal					
1	Usuario: Solicita añadir una canción a una playlist creada por él				
		2	Realiza identificación del usuario. Incluir (identificación de usuario)		
3	Consulta los datos de la canción				
		4	Incluir identificación de canción		
5	Recibe datos de la canción y pincha en añadr canción				
		6	Comprueba las playlists creadas por el usuario y ofrece un listado		
7	Recibe un listado con botones para añadir en una playlist u otra, pincha en una (si no hay ninguna creada, no envía nada)				
		7	Recibe información de la playlist a añadir la canción, lo comprueba y lo almacena en el sistema		
		8	Crea el aviso de cómo ha terminado el proceso y lo envía		
9	Recibe el aviso de finalización de				

	proceso		

Cursos Alternos	
7a	Si los datos de la playlist no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Modificación de playlist	CU-5.3			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	El usuario es creador de la playlist que pertenece al sistema				
Poscondición	Se modificarán los datos relacionados a una playlist				
Propósito					
Modificar una playlist					
Descripción					
Modificar los datos de una playlist que pertenece al sistema, guardando correctamente su nueva información					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sio n	1.0

Curso Normal			
1	Usuario: Solicita modificar una playlist		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)

		3	Identifica la playlist (Incluir identificación de playlist), comprueba que es su playlist y ofrece el formulario de modificación de playlist
4	Usuario: Recibe el formulario y aporta los datos de modificación		
		5	Recibe los datos de la playlist y lo comprueba, y los almacena en el sistema
		6	Ofrece un aviso de que el proceso ha finalizado correctamente
7	Usuario: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
5a	Si los datos de la playlist no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Eliminar playlist	CU-5.4			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	El usuario es creador de la playlist que pertenece al sistema				
Poscondición	Se eliminará una playlist				
Propósito					
Eliminar una playlist					
Descripción					
Modificar los datos de una playlist que pertenece al sistema, guardando correctamente su nueva información					
Autor	Francisco Jesús Palomares	Fec ha	15-03-18	Ver sió n	1.0

Curso Normal			
2	Usuario: solicita dar de baja una de sus playlists		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Identifica la playlist (Incluir identificación de playlist) y comprueba que es creador de la playlist
		4	Elimina los datos asociados a la playlist
		5	Ofrece un aviso de que el proceso ha finalizado correctamente
6	Administrador: Recibe el aviso de que el proceso ha finalizado		

Cursos Alternos	
4a	Si los datos de la playlist no son aportados correctamente, se piden de nuevo los datos. Si no se aportan de nuevo, se finaliza el CU.

Caso de Uso	Consulta de playlist	CU-5.5
Actores	Usuario (P)	
Tipo	Primario, Esencial	
Precondición	La playlist pertenece al sistema	
Poscondición		
Propósito		
Consultar playlist		

Descripción					
Consultar la información relacionada a una playlist que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario: Solicita consultar datos de una playlist		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Ofrece formulario para pedir la playlist a consultar
4	Administrador o usuario: aporta algún dato de la playlist		
		7	Se comprueban que son correctos y se identifica la playlist. Incluir (identificación de playlist)
		8	Se ofrecen los datos de la playlist y dar un aviso sobre la finalización del proceso
9	Recibe los datos de la playlist y el aviso del proceso		

Cursos Alternos	
7a	Si los datos no son correctos se informa al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Seguir playlist	CU-5.5
Actores	Usuario (P)	

Tipo	Primario, Esencial				
Precondición	La playlist pertenece al sistema y no es propietario de ella				
Poscondición					
Propósito					
Seguir Playlist					
Descripción					
Seguir Playlist para que su acceso sea más fácil para el usuario					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Administrador o usuario: Solicita seguir una playlist		
		2	Autenticación de usuario. Incluir (identificación de usuario)
		3	Ofrece formulario para pedir la playlist
4	Usuario: busca una playlist y selecciona seguir playlist		
		7	Se comprueban que son correctos y se identifica la playlist. Incluir (identificación de playlist) y comprueba que no sigue el usuario a esta playlist
		8	Almacena los datos (datos seguir playlist) en el sistema y da un aviso sobre la finalización del proceso
9	Recibe el aviso del proceso		

Cursos Alternos	
7a	Si los datos no son correctos se informa al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Dejar de seguir playlist	CU-5.5			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	La playlist pertenece al sistema y la sigue el usuario				
Poscondición					
Propósito					
Dejar de seguir Playlist					
Descripción					
Dejar de seguir Playlist para que su quitar su acceso rápido al usuario					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal					
1	Administrador o usuario: Solicita dejar de seguir una playlist que sigue el usuario				
		2	Autenticación de usuario. Incluir (identificación de usuario)		
		3	Ofrece formulario para pedir la playlist		
4	Usuario: busca una playlist y selecciona dejar de seguir playlist que sigue el usuario				
		7	Se comprueban que son correctos y se identifica la playlist. Incluir (identificación de playlist) y comprueba que sigue el usuario a esta playlist		
		8	Elimina los datos en el sistema (datos de seguir playlist) y da un aviso sobre la finalización del proceso		
9	Recibe el aviso del proceso				

Cursos Alternos	
7a	Si los datos no son correctos se informa al usuario, y se le piden nuevos datos correctos. Si no son proporcionados, se acaba el CU.

Caso de Uso	Identificación Playlist	CU-5.7			
Actores	Usuario (P)				
Tipo	Primario, Esencial				
Precondición	La playlist pertenece al sistema				
Poscondición					
Propósito					
Identificar una playlist					
Descripción					
Identificar una playlist que pertenece al sistema					
Autor	Francisco Jesús Palomares	Fec ha	15-03- 18	Ver sió n	1.0

Curso Normal			
1	Usuario: Solicita consultar datos de una canción		
		2	Realiza identificación del usuario. Incluir (identificación de usuario)
		3	Pide el id de la playlist
4	Introduce el id de la playlist (puede ser vía url)		
		5	Se comprueban que el id es correcto y que pertenece a una playlist

		6	Se ofrecen los datos de la playlist y dar un aviso sobre la finalización del proceso
7	Recibe los datos de la playlist y el aviso del proceso		

Cursos Alternos	
5a	Si el id de la playlist no es correcto, se informa al usuario, y se le pide nuevos un id correcto. Si no es proporcionado, se acaba el CU.

4. Diseño

En esta parte de la memoria se especificarán diversos aspectos para el desarrollo del sistema: arquitectura, tecnologías a utilizar, diseños de las diferentes páginas; también se va a hablar de qué se va a utilizar para desplegar la aplicación web.

4.1 Patrón arquitectónico

El patrón arquitectónico elegido en el desarrollo de este proyecto ha sido el Modelo-Vista-Controlador (MVC), este patrón tiene como objetivo separar la lógica de la interfaz, haciendo más fácil su desarrollo y su mantenimiento.

- Modelo: se encarga de los datos, normalmente es consulta la base de datos, con modificaciones, actualizaciones etc..
- Vista: es la representación de los datos, en este caso el frontend.
- Controlador: recibe las órdenes del usuario. Se encarga de pedir los datos al modelo y enviárselos a la vista.

Después de presentar el patrón, tenemos que pensar qué tecnologías utilizar que se adapte a este. Finalmente, se decide por Node JS como backend y para el frontend Angular. Para la aplicación móvil se piensa en Ionic.

4.2 Diagrama entidad-relación

En este apartado, se mostrará el diseño de la base de datos. La relación que guardan las distintas entidades. A partir de este diagrama podemos obtener todas las tablas que va a tener el sistema. Estas tablas tienen relaciones entre sí, de tipo 1:N y de tipo N:M.

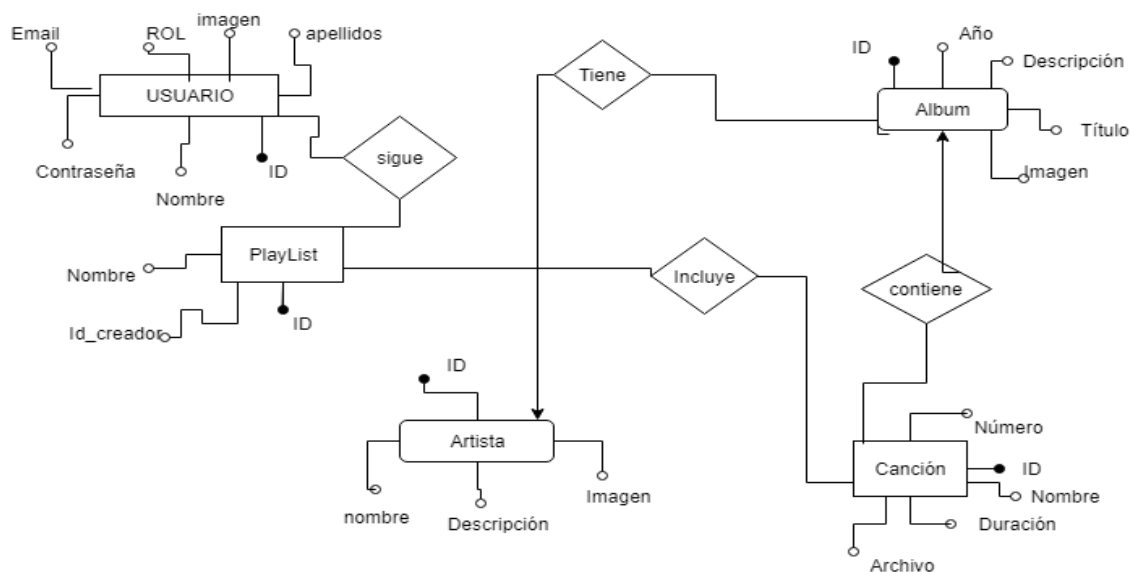


Imagen 2: Esquema entidad relación

Disponemos de las siguientes entidades con sus atributos:

- **Usuario:** id, nombre, apellidos, email, rol e imagen.
- **Playlist:** nombre, id e id_creador.
- **Artista:** id, nombre, descripción e imagen.
- **Álbum:** id, título, descripción, año e imagen.
- **Canción:** id, nombre, número, duración y archivo.

Y de las siguientes relaciones:

- Sigue: relación entre usuario y playlist de tipo N:M .
- Tiene: relación entre artista y álbum de tipo 1:N . Un artista tiene muchos álbumes, pero un álbum pertenece a un solo artista.
- Incluye: relación entre canción y playlist de tipo N:M .
- Contiene: relación entre álbum y canción de tipo 1:N . Un álbum tiene muchas canciones, pero una canción pertenece a un único álbum.

4.3 Lenguajes y entornos a utilizar

Primeramente, se expondrá los lenguajes y herramientas utilizadas a la hora de hacer el desarrollo de este proyecto:

4.3.1 Lenguajes utilizados

4.3.1.1 JavaScript [4]

Es un lenguaje de programación interpretado usado para la creación de aplicaciones web, tanto en frontend como en el backend. Es el lenguaje principal de este proyecto. Últimamente está en auge, ya que su eficiencia y su fácil curva de aprendizaje, hace que últimamente sea muy usado. Ahora está en el puesto número 7 del ranking de TIOBE.

4.3.1.2 HTML (HyperText Markup Language) [5]

Es un lenguaje de etiquetas usado para desarrollar páginas webs, en él se puede incrustar tanto texto, imágenes, vídeos y juegos entre otros. Es un estándar de World Wide Web Consortium (W3C) .

4.3.1.3 CSS (Cascading Stylesheets) [6]

Es un lenguaje de estilos para la definición de documentos HTML. En él se pueden definir los colores, márgenes y otros aspectos más para mejorar el aspecto de las páginas webs.

4.3.2 Sistema de gestión de base de datos, PostgreSQL [7]

Es un sistema de bases de datos relacional libre y está orientado a objetos. Es el sistema usado para el desarrollo de este proyecto. Destaca por su alta concurrencia.

¿Por qué usar PostgreSQL?

PostgreSQL es un gestor de bases de datos relacionales open-source más usado en todo el mundo. Una gran característica que tiene es su alta concurrencia ya que no requiere usar bloqueos de lectura y así dar una mayor escalabilidad. Se puede administrar fácilmente y es multiplataforma.

En conclusión, lo he elegido por su alta concurrencia y por su fácil administración.

4.3.3 Entorno del servidor Backend, Node JS [8]

Es un entorno de servidor en código abierto, utiliza JavaScript en el servidor. Es gratis y se puede utilizar en varias plataformas (Windows, Linux, Unix, Mac OS X, etc..). Node puede abrir, crear, escribir y borrar archivos en el servidor. Node utiliza la programación asíncrona. Lo he utilizado para el desarrollo del backend de la aplicación web. Utilizado para crear la API de la aplicación web, manda JSON o recibe JSON y lo consulta, inserta o edita en la base de datos.

¿Por qué usar Node JS?

Node se caracteriza por ser muy rápido gracias a su bucle de eventos (Event Loop) en el que se puede escalar grandes volúmenes de clientes de manera asíncrona. Node no genera un nuevo hilo por cada cliente, cada conexión dispara una ejecución de evento dentro del proceso motor de Node. Esto hace que los costes de infraestructura sean menores. Lo mejor es que se escribe en Javascript y esto hace que su curva de aprendizaje sea menor, ya que no tienes que aprender un lenguaje nuevo. También destaca por su popularidad y por su gran cantidad de paquetes instalables a través del gestor NPM mencionado anteriormente.

En conclusión, se ha usado Node porque hace que el desarrollo de aplicaciones webs para grandes volúmenes de clientes sea fácil y escalable, y también porque su curva de aprendizaje es baja ya que no tienes que aprender un nuevo lenguaje de programación.

4.3.4 Entorno del frontend, Angular [9]

Angular: es un framework para JavaScript utilizado para el desarrollo de aplicaciones web, su misión es separar el frontend con el backend, de esta manera hacer más fácil su mantenimiento. Se usa para el desarrollo frontend y tiene una gran característica, crear páginas webs SPA. (Single Page App). SPA es una web en una sola página sin refrescar la página, sólo carga el contenido que deseamos. Es creado por Google y tiene gran popularidad. También utiliza NPM para la instalación de paquetes.

¿Por qué usar Angular?

Como se ha mencionado antes, Angular hace que las páginas webs sean mucho más rápidas que anteriormente, cargando sólo los componentes que nos hagan falta. Facilita mucho la creación de aplicaciones webs y hace que el mantenimiento de ellas sea mucho más fácil.

Junto con HTML, Bootstrap y JQuery hace que la creación de aplicaciones sea más fácil, ya que ofrecen estilos y conjunto amplio de herramientas.

4.3.5 Entorno del desarrollo para aplicaciones móviles, Ionic [10]

Ionic: es un framework utilizado para la creación de aplicaciones híbridas en móviles, aunque también se pueden crear aplicaciones webs. Tiene una gran característica y es que utiliza Angular para su desarrollo, esto hace que sea mucho más fácil desarrollarlas. Además utiliza CSS y HTML para la creación de ellas.

¿Por qué he usado Ionic?

Ionic se basa en Angular, con él puedo reutilizar el código desarrollado en Angular en la aplicación web para hacer la aplicación móvil. Junto con sus componentes, el uso de HTML y CSS hacen que el desarrollo de aplicaciones sea bastante fácil.

4.4 Herramientas a utilizar

Primeramente, se expondrá el IDE, bibliotecas y frameworks a utilizar en la implementación de este proyecto:

4.4.1 IDE - Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft. Está creado para múltiples plataformas. Es gratuito y de código abierto, por eso hace que el presupuesto del proyecto no tenga un coste más.

Lo he utilizado porque para JavaScript me facilita mucho las cosas y su interfaz lo hace más atractivo para su uso.

4.4.2 Bibliotecas y frameworks utilizados

4.4.2.1 Para Node JS

NPM [11]: es un gestor de paquetes para Node JS, utilizado para instalar diversos paquetes en Node.

Sequelize [12]: es un paquete para Node JS utilizado para conectar el servidor Node JS con gestores de bases de datos relacionales como MySQL y PostgreSQL . Utiliza ORM (Object Relational Mapping) , de esta manera se puede utilizar la base de datos de una manera más fácil, teniendo clases y objetos de las diferentes tablas de las bases de datos.

Express JS [13]: es un framework para Node JS para la creación de páginas webs. Es rápido y muy flexible. También se instala vía NPM.

Body-Parser [14]: es una paquete de NPM utilizado como parseador JSON en las peticiones HTTP.

Compression [15]: es un paquete para Node JS utilizado para usar la compresión GZIP, de esta manera se mandan datos comprimidos.

Multiparty [16]: es un paquete para Node JS utilizado para gestionar las peticiones HTTP cuyo tipo de contenido es multipart/form-data así como las subidas de archivos.

JWT-simple [17]: es un paquete para Node JS utilizado para la creación de Tokens a través de una clave secreta, utilizado para la autenticación de los usuarios. Se explicará más adelante.

MediaServer [18]: es un paquete para Node JS utilizado para hacer el streaming de archivos estáticos del servidor. Utilizado para el streaming de canciones.

Moment JS [19]: es un paquete para Node JS utilizado para gestionar fechas , hace que sea muy fácil el manejo de estas.

Mp3-duration [20]: es un paquete para Node JS utilizado para obtener la duración de archivos mp3.

Music-metadata [21]: es un paquete para Node JS utilizado para obtener los metadatos de los archivos mp3.

Nodemailer [22]: es un paquete para Node JS utilizado para el envío de emails, usado para enviar emails de registro y de olvidar contraseña.

Nodemon [23]: es un paquete para Node JS usado para que no haga falta reiniciar el servidor cada vez que hagamos un cambio en el código de él, simple pero efectivo.

4.4.2.2 Para Angular

Jquery [24]: es una biblioteca multiplataforma de JavaScript utilizado para interactuar de una manera simple los documentos HTML, manejar eventos y manipular el árbol DOM (Document Object Model).

Bootstrap [25]: es un conjunto de herramientas usado para el diseño de páginas webs. Contiene plantillas de diseño de formularios, botones, menús etc...

Angular-notifier [26]: es un paquete de Angular utilizado para la creación de notificaciones altamente personalizables en Angular de una manera sencilla.

Ngx-drag-scroll [27]: es un paquete de Angular usado para la creación de carruseles de imágenes.

Angular-sortablejs [28]: utilizado para manejar arrays en listas drag and drop, de esta manera hacer interactiva y manejables arrays al usuario.

Typescript [29]: es un lenguaje de programación basado en JavaScript creado por Microsoft para crear clases y objetos en Javascript de manera simple.

4.4.2.3 Para Ionic

Cordova-plugin-camera [30]: plugin para ionic para usar la cámara o tener acceso a la galería del teléfono.

Cordova-file-transfer [31]: plugin para ionic para subir o descargar archivos, en este caso usado para subir las imágenes de un usuario.

Cordova-music-controls [32]: plugin para ionic para acceder al audio reproduciéndose desde una notificación en el móvil, teniendo botones de pausa/reproducir , siguiente canción y canción previa.

4.5 Diseño de la plataforma

Se diseñaron inicialmente mockups para facilitar la maquetación de la plataforma, estos al paso del tiempo y del desarrollo de la plataforma se pueden ver alterados. Aquí se mostrarán los diseños iniciales tanto para web como para la aplicación móvil.

4.5.1 Diseño de página web

4.5.1.1 Inicio de sesión

En esta página se mostrará el login inicial del usuario, también se mostrará un enlace para registrarte si no pertenece a la plataforma:

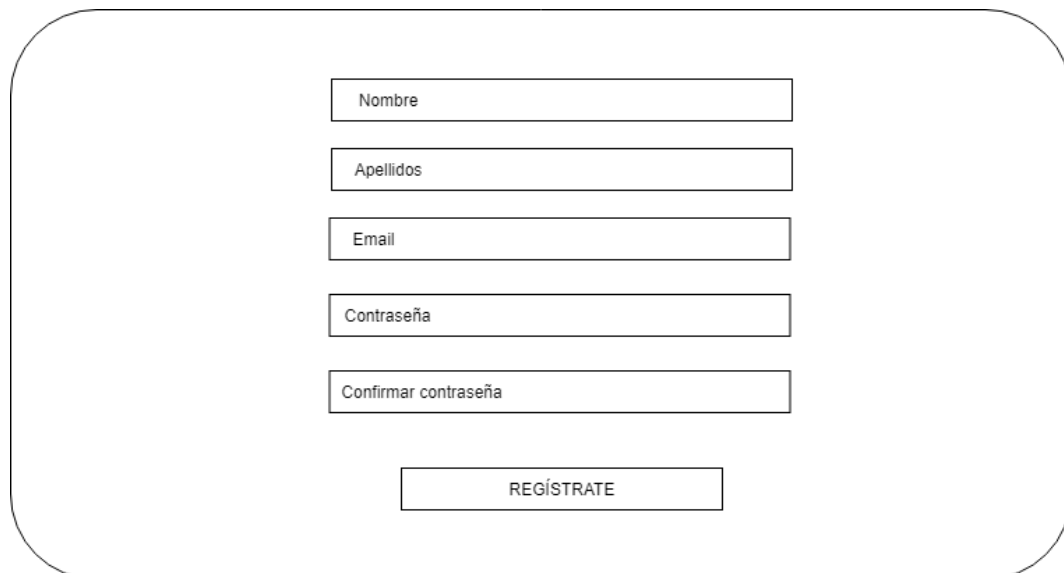


A login form design within a rounded rectangle. It features two input fields: the top one is labeled "Email" and the bottom one is labeled "Contraseña". Below these fields is a button labeled "INICIA SESIÓN". At the bottom center of the form is a link labeled "REGÍSTRATE".

Imagen 3: Página inicio sesión diseño página web

4.5.1.2 Registro de usuario

En esta página se muestra el acceso al registro del usuario:



A registration form design within a rounded rectangle. It features five input fields stacked vertically: "Nombre", "Apellidos", "Email", "Contraseña", and "Confirmar contraseña". Below these fields is a button labeled "REGÍSTRATE".

Imagen 4: Página registro diseño página web

4.5.1.3 Página principal de la plataforma

En esta página se muestra el primer contenido de la plataforma, también se muestra el reproductor de las canciones:

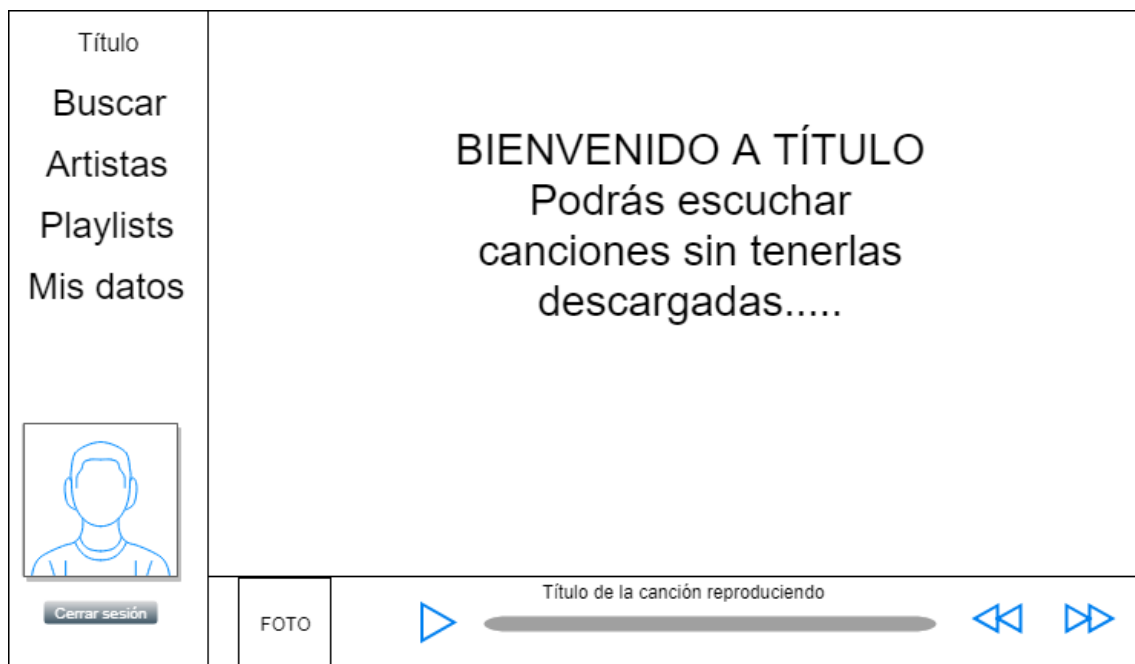


Imagen 5: Página principal diseño página web

4.5.2 Diseño de aplicación móvil

4.5.2.1 Inicio de sesión

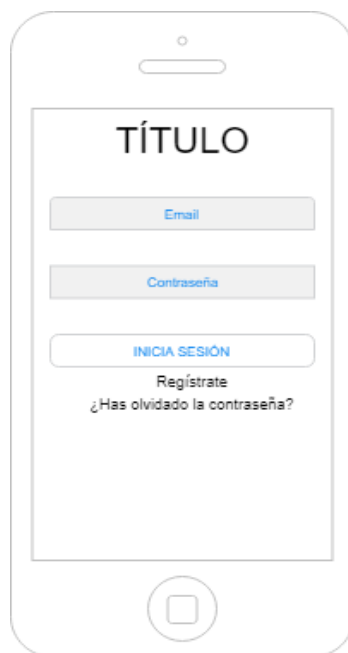


Imagen 6: Página inicio sesión diseño aplicación móvil

4.5.2.2 Registro



A wireframe of a mobile registration screen. At the top, the word "TÍTULO" is centered. Below it are five input fields, each with a placeholder label: "Nombre", "Apellidos", "Email", "Contraseña", and "Confirmar contraseña". Below the input fields is a button labeled "REGÍSTRATE" and a link labeled "Ir al login". The screen is framed by a rounded rectangle representing a mobile device with a home button at the bottom.

Imagen 7: Página registro diseño aplicación móvil

4.5.2.3 Página principal



Imagen 8: Página principal diseño aplicación móvil

4.5.2.4 Página reproductor

Al pulsar en el contenedor de título canción, se abrirá esta página que contiene el reproductor:



Imagen 9: Página reproductor diseño aplicación móvil

4.6 Diseño del objetivo principal

Antes de implementar el proyecto, se necesita saber cómo se va abordar la solución del objetivo principal, la transmisión de música en streaming. Como se ha nombrado anteriormente, para resolver este problema se va a utilizar el módulo de Node JS llamado mediaserver, un módulo bastante utilizado para dar solución a este tipo de aplicaciones. Un módulo bastante fácil de utilizar usado para hacer streaming de contenido estático de un servidor de Node JS mediante HTTP y HTTPS. Da un resultado bastante bueno según las opiniones de los distintos desarrolladores.

Este módulo es compatible con los navegadores populares como son: Edge, Internet Explorer, Chrome y Safari.

4.7 Despliegue: Docker [11]

Para el despliegue de la aplicación, se consideró usar Docker, una herramienta de código abierto que automatiza el despliegue de aplicaciones en contenedores de software a nivel de sistema operativo de Linux.

Según me han comentado, es una herramienta bastante fácil de utilizar y muy eficiente, aportando seguridad y portabilidad.

5. Implementación:

En este capítulo, se explicará las fases más importantes de la implementación de este proyecto, partiendo del backend hasta llegar a la aplicación móvil.

5.1 Fases del desarrollo

5.1.1 Desarrollo del Backend

Como se ha explicado anteriormente, el desarrollo del Backend se ha utilizado Node JS, aquí se ha hecho una API REST, que es una interfaz entre sistemas que utilizan HTTP para obtener y enviar información, en el se llama a través de rutas y se dispone de diversas operaciones de HTTP como POST (crear), GET (obtener), PUT (editar) y DELETE (borrar).

5.1.1.1 Creación de tablas

Gracias al módulo de Sequelize anteriormente explicado, la creación de tablas y el manejo de ellas es bastante intuitivo y muy fácil de manejar, para la creación de ellas, pondré un ejemplo de una de las tablas más importantes:

```
module.exports = function(sequelize, DataTypes) {  
  
    var Usuario = sequelize.define("Usuario", {  
        nombre:DataTypes.STRING,  
        apellidos: {type:DataTypes.STRING,allowNull: false},  
        password: DataTypes.STRING,  
        email: {type:DataTypes.STRING, unique: true,allowNull: false},  
        rol:DataTypes.STRING,  
        imagen: DataTypes.STRING  
    });  
};
```

Aquí vemos la facilidad de crear esta tabla y cómo crear los atributos de un usuario.

Otro código importante sería cómo crear la relación entre tablas:

```
Album.belongsTo(sequelize.models.Artista, {  
    onDelete: "CASCADE",
```

```
    foreignKey: {
      allowNull: false
    }
  });
```

Aquí se ve la relación de la tabla álbum con el Artista, con belongsTo se hace la relación 1:N .

Otro importante es la creación de N:M :

```
sequelize.models.Usuario.belongsToMany(sequelize.models.Playlist,{through:sequelize.models.Sigue,foreignKey:'usuario',as:'playlists'});
sequelize.models.Playlist.belongsToMany(sequelize.models.Usuario,{through:sequelize.models.Sigue,foreignKey:'playlist',as:'usuarios'});
```

Aquí podemos ver la relación N:M de las tablas Usuario y Playlist, esto significa que un usuario puede seguir a muchas playlists y que una playlist puede seguir por más de un usuario, esto significa que un usuario puede acceder a las playlists seguidas de una manera fácil y poder escuchar la música relacionada de una manera muy fácil.

5.1.1.2 Creación de controladores

Después de haber creado las diversas tablas que componen la API, se tiene que crear los diversos controladores que acceden a estas tablas. Los controladores son las funciones que se ejecutan al llamar a las urls de la API. Estos controladores divididos según a qué modelo van a actuar. Estos actúan mediante ORM a los modelos creados con el módulo de Sequelize. Estos controles pueden servir datos en JSON, o errores en HTTP: 404 si es un error de lógica o 500 si es un error del servidor.

Están divididos en:

- **Usuario:** cuyas funciones principales son relacionadas con la gestión de usuarios.
 - Registrar usuario.
 - Loguear usuario.
 - Actualizar usuario.
 - Buscar usuario.
 - Subir imagen de usuario.
 - Obtener imagen de usuario.
 - Filtrar usuarios.
- **Artista:** cuyas funciones principales son relacionadas con la gestión de artistas.

- Obtener artista por id.
 - Obtener todos los artistas.
 - Guardar artista.
 - Actualizar artista.
 - Eliminar artista.
 - Subir imagen de un artista.
 - Obtener imagen de artista.
 - Filtrar artistas.
 - Obtener últimos artistas, se devuelven los últimos 6 artistas creados.
- **Álbum:** cuyas funciones principales son relacionadas con la gestión de álbumes.
 - Obtener álbumes de un artista.
 - Guardar álbum.
 - Obtener información de un álbum.
 - Eliminar álbum.
 - Subir imagen de un álbum.
 - Obtener imagen de un álbum.
 - Filtrar álbumes.
 - Obtener últimos álbumes, se devuelven los últimos 6 álbumes creados.
- **Canción:** cuyas funciones principales son relacionadas con la gestión de canciones.
 - Guardar canción.
 - Obtener información de una canción.
 - Obtener canciones de un álbum.
 - Actualizar una canción.
 - Eliminar canción.
 - Subir archivo de canción.
 - Obtener archivo de canción, usado para la reproducción de canciones.
 - Obtener artista y álbum a partir del id de una canción.
 - Filtrar canciones.
- **Playlist:** cuyas funciones principales son relacionadas con la gestión de playlists.
 - Crear playlist.
 - Eliminar playlist.
 - Obtener información de playlist.
 - Seguir playlist.
 - Dejar de seguir playlist.
 - Obtener playlists creadas de un usuario.
 - Obtener playlists seguidas de un usuario.
 - Modificar playlist.
 - Añadir canción de una playlist creada por el usuario.
 - Eliminar canción de una playlist creada por el usuario.
 - Filtrar playlists.

5.1.1.3 Autenticación

Para la autenticación se utilizará un token JWT (en el que se utiliza JWT-SIMPLE) para detectar el usuario, este Token tiene la siguiente información:

```
id:usuario.id,  
nombre: usuario.nombre,  
apellidos: usuario.apellidos,  
rol:usuario.rol,  
imagen:usuario.imagen,  
iat:moment().unix(),  
exp:moment().add(15,'days').unix()
```

Guardará el id del usuario, nombre, apellidos, ruta de la imagen, el tiempo en el que se ha creado, tiempo final ya que el token tiene un tiempo para que caduque y se cree uno nuevo; esto puede tener esta forma:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MTESlm5vbWJyZSI6IkZyYW5jaXNjbylzImFwZW
xsaWRvcyl6ImlBbG9tYXJlcylzInJvbmCI6IlJPTeVfQURNSU4iLCJpbWFnZW40IjIjVXFMcTUzcmdLMW
FvS2dqaFNOcThrSm4ucG5nliwiaWF0IjoxNTIyOTE5NjYxLjE4aHAIbG9E1MjQyMTU2NjF9.sbvsF2m7
uGOWSTdakIY8xqX37Z2FscALFh4ibo6sasY
```

Esto está codificado con una clave.

Este token se obtiene una vez que el usuario se ha logueado en el sistema y se guarda en el `localStorage`.

Cada vez que se hace una llamada a la API, este token se enviará en una cabecera llamada Authorization, de esta manera el usuario estará identificado y no habría una variable sesión en el servidor y así no saturarlo, sería más eficiente.

5.1.1.4 Middlewares

Antes de proceder a la creación de las rutas, se crearán los middlewares para establecer una seguridad en la API, estos son funciones que actúan antes de que actúen las funciones de los controladores. Todos actúan para comprobar los tokens de autenticación. Cuando da un error al comprobar el token se da un error 403 cuando el token no es válido o 401 si ha pasado el tiempo de expiración.

A continuación se especificarán los diversos middlewares desarrollados:

- **Comprobar Token:** utilizado para comprobar la expiración de un usuario.
- **Comprobar rol token:** utilizado para comprobar si el rol del usuario es de administrador.
- **Comprobar id Usuario:** utilizado para comprobar que las acciones de un usuario pertenecen a ese usuario.

5.1.1.5 Creación de rutas

Por último, queda la creación de rutas, éstas serán llamadas por Angular para servir los datos recibidos de la API. Estas rutas pueden ser llamadas por ejemplo con id que pertenece a una tabla del sistema. Estas pueden ser llamadas por los diferentes métodos HTTP como GET (obtener datos), POST (enviar datos), PUT (actualizar datos), DELETE (eliminar datos). A continuación se mostrará cómo se han dividido las diferentes rutas y qué métodos HTTP se han utilizado:

- **Usuario:**
 - Registrarse con POST
 - Login con POST
 - Actualizar-usuario/:id con PUT
 - Subir-foto-perfil/:id con POST
 - Obtener-foto-perfil/:archivoimagen con GET
 - Filtrar-usuarios/:usuario con GET
 - Info-usuario/:id con GET
 - Refrescar-token con GET

- **Artista:**
 - Artista/:id con GET
 - Get-artistas con GET
 - Guardar-artista con POST
 - Actualizar-artista/:id con PUT
 - Eliminar-artista/:id con DELETE
 - Subir-foto-artista/:id con POST
 - Obtener-foto-artista/:archivoimagen con GET
 - Obtener-foto-artista/:id con GET
 - Filtrar-artistas/:artista con GET
 - Ultimos-artistas con GET

- **Álbum:**
 - Get-albums con GET
 - Guardar-album con POST
 - Get-album/:id con GET
 - Get-albums-artista/:id con GET , el id es de artista
 - Actualizar-album/:id con PUT
 - Eliminar-album/:id con DELETE
 - Subir-foto-album/:id con POST
 - Obtener-foto-album/:archivoimagen con GET
 - Filtrar-albumes/:album con GET
 - Ultimos-albumes con GET

- **Canción:**

- Get-canciones con GET
 - Guardar-cancion con POST
 - Actualizar-cancion/:id con PUT
 - Eliminar-cancion/:id con DELETE
 - Subir-fichero-cancion/:id con POST
 - Obtener-fichero-cancion/:id con GET
 - Get-canciones-album/:id con GET, el id es del álbum
 - Get-album-artista/:id con GET
 - Filtrar-canciones/:cancion con GET
- **Playlist:**
 - Crear-playlist con POST
 - Eliminar-playlist con DELETE
 - Get-playlist/:id con GET
 - Get-playlists con GET
 - Seguir-playlist/:id con GET
 - Dejar-seguir-playlist/:id con GET
 - Get-playlists-creadas con GET
 - Get-playlists-seguidas con GET
 - Modificar-playlist/:id con PUT
 - Add-cancion-playlist/:id con POST
 - Borrar-cancion-playlist/:id con POST
 - Filtrar-playlists/:playlist con GET
 - Ver-playlists-seguidas-usuario/:id con GET, el id es de usuario
 - Ver-playlists-creadas-usuario/:id con GET, el id es de usuario

5.1.1.6 Implementación streaming

Como ya se ha detallado anteriormente en el diseño, para solucionar el problema del streaming de canciones, se usará el módulo mediaserver. Para la implementación de éste, se usará un controlador, el controlador de canción cuya función obtenerFichero llama a este módulo para ofrecer el streaming, y éste es llamado por una ruta de la API:

```
var ms = require('mediaserver');
function obtenerFichero(req,res){

    var archivo_cancion = req.params.cancionFile;
    var ruta_archivo = './subidas/canciones/' + archivo_cancion;
    console.log(ruta_archivo);

    fs.exists(ruta_archivo,function(exits){
```

```

        if(exits){
            //Enviamos el archivo
            ms.pipe(req, res, ruta_archivo);

        }
        else{
            res.status(404).send({message: 'No existe la cancion'});
        }

    });
}

```

Como vemos, primero se comprueba que existe la canción recibida como parámetro, esta canción debería estar alojada en la carpeta de subidas de canciones del servidor. Una vez que se encuentra la canción en el servidor, exits sería verdadero y transmitiría por streaming la canción encontrada mediante el módulo de mediaserver; si no se encontrara la canción en el servidor, daría un error HTTP de 404.

Gracias al módulo mediaserver, la solución del objetivo principal ha sido muy fácil de implementar.

5.1.2 Desarrollo del Frontend

Como se ha explicado anteriormente, en el desarrollo del frontend se va a utilizar en su última versión, la versión 6. Este trabaja con Javascript, HTML y CSS para el desarrollo de páginas webs. Su característica principal es que sólo se cargan las partes que se van a utilizar, de esta manera, por ejemplo, el menú de navegación solo se cargaría una vez. La creación del frontend con Angular se hace a través de componentes en los que el usuario interactúa. Estos componentes están constituidos de la funcionalidad (en JavaScript) y de la vista (en HTML). El usuario no puede acceder a la funcionalidad de ellos, interactúa con la vista y ésta llama a las funcionalidades. Las funcionalidades llamarían a los servicios, estos interactúan directamente con la API del Backend.

En esta parte se explicarán las diversas etapas del desarrollo del frontend.

5.1.2.1 Creación de modelos

Primeramente se crearán los modelos que la aplicación web va a necesitar. Estos modelos van a ser clases que Angular va a llamar para que la creación de objetos sea más fácil.

Se dividirá de la siguiente manera:

- **Usuario:** clase relacionada con la información de los usuarios. Se dispone de los siguientes datos: id, nombre, apellidos, email, rol, contraseña e imagen.

- **Artista:** clase relacionada con la información de los artistas. Se dispone de los siguientes datos: id, nombre, descripción e imagen.
- **Álbum:** clase relacionada con la información de los álbumes. Se dispone de los siguientes datos: id, título, descripción, imagen, año e id_artista.
- **Canción:** clase relacionada con la información de las canciones. Se dispone de los siguientes datos: id, número, nombre, duración, archivo e id_album.
- **Playlist:** clase relacionada con la información de las playlists. Se dispone de los siguientes datos: id y nombre.

5.1.2.2 Creación de login y registro

Antes de crear el componente relacionado con el login y el registro, se creará el servicio que gestiona las llamadas a la API relacionado con la gestión de usuarios. En el primer servicio, se crearán las diversas funciones de registro y de login. La función de registro llamará a la ruta del API de registro; la función de login, obtendrá el token mencionado anteriormente y se guardará en el localStorage, memoria que almacena datos en el navegador del cliente de un mismo origen.

Después de haber creado el servicio, se creará la funcionalidad y la vista del componente. Este componente no tendrá subcomponentes.

5.1.2.3 Creación del componente principal

En este componente se creará el menú de navegación del componente principal. Este menú de navegación tendrá la facilidad de llamar a ciertos componentes: playlists, buscar, artistas y mis datos. Este tendrá subcomponentes que se llamarán a través del router-outlet, esto es usado para cargar solamente el componente que haga falta. A continuación se mostrarán las diferentes rutas o componentes de los que dispone el componente principal:

- mis-datos: Contiene el componente que actualiza los datos de un usuario propio.
- artistas: Contiene el componente que lista a todos los artistas de la plataforma.
- artista/:id : Contiene el componente que dispone de la información de un artista según el id.
- crear-artista: Contiene el componente relacionado con la creación de un artista. Este componente es solamente accesible para los usuarios que tienen el rol de administrador.
- editar-artista/:id : Contiene el componente relacionado con editar a un artista que es referenciado a través del id. Es accesible únicamente por el usuario que tiene el rol de administrador.
- album/:id : Contiene el componente que dispone la información de un álbum según el id.
- crear-album/:artista : Contiene el componente relacionado con la creación de un álbum de un artista. Es accesible solamente por el usuario que tiene el rol de administrador.
- editar-album/:id : Contiene el componente relacionado con editar un álbum referenciado por el id. Es accesible solamente por el usuario que tiene el rol de administrador.

- crear-cancion/:id : Contiene el componente relacionado con la creación de una canción asociado a un álbum por el id. Es accesible solamente por el usuario que tiene el rol de administrador.
- editar-cancion/:id : Contiene el componente relacionado con editar una canción referenciada por el id. Es accesible solamente por el usuario que tiene el rol de administrador.
- crear-playlist : Contiene el componente relacionado con la creación de una playlist.
- playlist/:id : Contiene el componente que ofrece la información relacionada a una playlist.
- playlists : Contiene el componente que ofrece las playlists creadas y seguidas de un usuario.
- buscar: Contiene el componente en el que ofrece la funcionalidad de buscar artistas, canciones, playlists, usuarios y álbumes de la plataforma.

5.1.2.4 Creación del componente reproductor

Componente alojado en la página principal, este componente dispone de diversas funcionalidades para la interacción con el audio que está sonando. En la vista trabajo con la etiqueta de audio de HTML, este ofrece un reproductor, con éste llamo a la API que ofrece el archivo de la canción en streaming. En el anexo ([Página inicial](#)) se dispondrá de como queda el conjunto de componente principal con el menú de navegación y el componente de reproductor.

5.1.2.5 Control de expiración del token

Cuando se llama a la API y ésta comprueba el token, si el token ha expirado, envía un error HTTP de 401 o 403 , para manejar esto se utiliza una herramienta de Angular llamada HTTPInceptor [34]. Esta es una forma de interceptar y modificar las solicitudes HTTP, también maneja cuando se dan errores una vez finalizadas.

Cuando da un error de los mencionados anteriormente, llamo al router, utilizado para moverse por los componentes y rutas de la aplicación; elimino el token Localstorage y voy con el router al componente de login. De esta manera el usuario tiene que volver a loguearse para acceder de nuevo a la plataforma.

5.1.3 Desarrollo de la aplicación móvil

El desarrollo de la aplicación móvil con IONIC ha sido relativamente fácil, ya que IONIC utiliza Angular, se ha podido reutilizar el código del frontend para desarrollar la aplicación móvil, pero también hay algunas diferencias con respecto al frontend:

5.1.3.1 Reproductor de música

Para el audio de la aplicación se ha utilizado el objeto de audio de HTML, con él se pueden ejecutar funciones de reproducir y pausar el audio, también se puede utilizar funciones que devuelven los segundos reproducidos (usado para el progress). Todas estas funciones han facilitado la creación del reproductor de la aplicación.

5.1.3.2 Subida de imágenes

Esta etapa también es diferente que en el frontend, para obtener imágenes del móvil se ha utilizado el plugin mencionado anteriormente Cordova-plugin-camera, con él se puede acceder a las imágenes almacenadas en la galería o acceder a la cámara directamente.+

Para enviar la imagen a la API se ha utilizado el plugin mencionado anteriormente, Cordova-file-transfer, un plugin bastante fácil de usar.

5.1.3.3 Control de música en notificación

Otro aspecto diferente al del frontend, es poder controlar la reproducción de canciones sin abrir la aplicación, para esto se ha usado un plugin mencionado anteriormente, Cordova-music-controls, con este plugin se puede manejar diversos eventos como reproducir/pausar el audio, y otras como siguiente canción e ir a la canción anterior. Además es customizable y es muy fácil de obtener el objetivo descrito.

5.1.3.4 Control de expiración del token

Este aspecto es parecido al del frontend, se usa HTTPInceptor, pero, a diferencia de este, se llama a la API para refrescar el token, el nuevo token se guarda en el LocalStorage, y se hace de nuevo la llamada a la API en la que ha comprobado el servidor que el token se ha expirado, de esta manera el usuario no necesita salirse de la aplicación para seguirla utilizando.

6. Conclusiones

6.1 Objetivos alcanzados

El objetivo principal planteado al principio de la memoria, transmisión de música en streaming, se ha alcanzado con éxito gracias a las herramientas utilizadas, así también como las diversas funcionalidades presentadas.

6.2 Lecciones aprendidas

La creación de este proyecto me ha ayudado a aprender y a mejorar en un muchos aspectos del desarrollo de aplicaciones web. El lenguaje de programación principal, en este caso JavaScript ya lo sabía de antes, aunque no sabía los diversos entornos en donde se podía trabajar con él, en este caso con Angular y Node JS, ya que es un lenguaje único tanto para el Frontend y el Backend, me ha agilizado el tiempo para este gran proyecto, aumentando el nivel de manejo de este lenguaje de programación.

También he mejorado a la hora de desarrollo de aplicaciones móviles, ya que no sabía nada acerca de Ionic, y es una herramienta muy potente y fácil para hacer aplicaciones para diversas plataformas.

Además he podido mejorar mis habilidades en diseño web, ya que mi uso en HTML y CSS era bastante limitado.

También, la extensión y la complejidad del proyecto, me ha ayudado a mejorar mi organización del tiempo y también para mejorar las habilidades para crear un análisis, una documentación de una manera clara para desarrollar este proyecto.

Al haber terminado este proyecto, me veo bastante capacitado para llevar a cabo otros proyectos iguales o mayores con un resultado mucho mejor antes de haberme embarcado en este.

7. Bibliografía

Bibliografía

- [1] Spotify. [Online]. <https://www.spotify.com/>
- [2] Apple Music. [Online]. <https://www.apple.com/la/music/>
- [3] Soundcloud. [Online]. <https://soundcloud.com/>
- [4] JavaScript. [Online]. <https://es.wikipedia.org/wiki/JavaScript>
- [5] HTML. [Online]. <https://www.w3schools.com/html/>
- [6] CSS. [Online]. <https://www.w3schools.com/Css/>
- [7] PostgreSQL. [Online]. <https://www.postgresql.org/>
- [8] Node JS. [Online]. <https://nodejs.org/es/>
- [9] Angular. [Online]. <https://angular.io/>
- [10] IONIC. [Online]. <https://ionicframework.com/>
- [11] NPM. [Online]. <https://www.npmjs.com/>
- [12] Sequelize. [Online]. <http://docs.sequelizejs.com/>
- [13] Express JS. [Online]. <https://www.npmjs.com/package/express>
- [14] Body-parser. [Online]. <https://www.npmjs.com/package/body-parser>
- [15] Compression. [Online]. <https://www.npmjs.com/package/compression>
- [16] Multiparty. [Online]. <https://www.npmjs.com/package/multiparty>
- [17] JWT-simple. [Online]. <https://www.npmjs.com/package/jwt-simple>
- [18] Mediaserver. [Online]. <https://www.npmjs.com/package/mediaserver>
- [19] Moment JS. [Online]. <https://momentjs.com/>
- [20] MP3-Duration. [Online]. <https://www.npmjs.com/package/mp3-duration>
- [21] Music-metadata. [Online]. <https://www.npmjs.com/package/music-metadata>
- [22] Nodemailer. [Online]. <https://www.nodemailer.com/>
- [23] Nodemon. [Online]. <https://nodemon.io/>

- [24] JQuery. [Online]. <https://jquery.com/>
- [25] BootStrap. [Online]. <https://getbootstrap.com/>
- [26] Angular-notifier. [Online]. <https://www.npmjs.com/package/angular-notifier>
- [27] Ngx-drag-scroll. [Online]. <https://www.npmjs.com/package/ngx-drag-scroll>
- [28] Angular-sortablejs. [Online]. <https://www.npmjs.com/package/angular-sortablejs>
- [29] Typescript. [Online]. <https://www.typescriptlang.org/>
- [30] Cordova-plugin-camera. [Online]. <https://ionicframework.com/docs/native/camera/>
- [31] Cordova-file-transfer. [Online]. <https://ionicframework.com/docs/native/file-transfer/>
- [32] Cordova-music-controls. [Online]. <https://ionicframework.com/docs/native/music-controls/>
- [33] Docker. [Online]. <https://www.docker.com/>
- [34] HTTPInceptor. [Online]. <https://angular.io/api/common/http/HttpInterceptor>
- [35] google. [Online]. www.google.es

8. Anexos

8.1 Código fuente

El código fuente, tanto como la documentación se puede encontrar:

- GitHub:
 - <https://github.com/iPhaco96/TFG>

Para ejecutarlo en local se dispone de las instrucciones en el propio repositorio, pero se necesita establecer bien la base de datos con PostgreSQL.

Este proyecto dispone de la cantidad siguiente de líneas de código aproximadas:

Etapas de implementación	Cantidad de líneas de código
Backend	2200
Frontend	5000
Aplicación móvil	3500
TOTAL	10700 líneas de código aproximadas

8.2 Versión final

En esta parte de la memoria se expondrá diversas imágenes de cómo ha quedado la plataforma, se dividirá en dos: aplicación web (frontend), y en la aplicación móvil:

8.2.1 Aplicación web

8.2.1.1 Iniciar sesión

Iniciar sesión

Registrarse

Email:

email

Contraseña:

Contraseña

INICIAR SESIÓN

[¿Has olvidado la contraseña?](#)

Imagen 10: Página inicio sesión página web

8.2.1.2 Registro

Iniciar sesión

Registrarse

Nombre:

Nombre

Apellidos:

Apellidos

Email:

Email

Contraseña:

Contraseña

Confirmar contraseña:

Confirma contraseña

REGISTRARSE

Imagen 11: Página registro página web

8.2.1.3 Página inicial

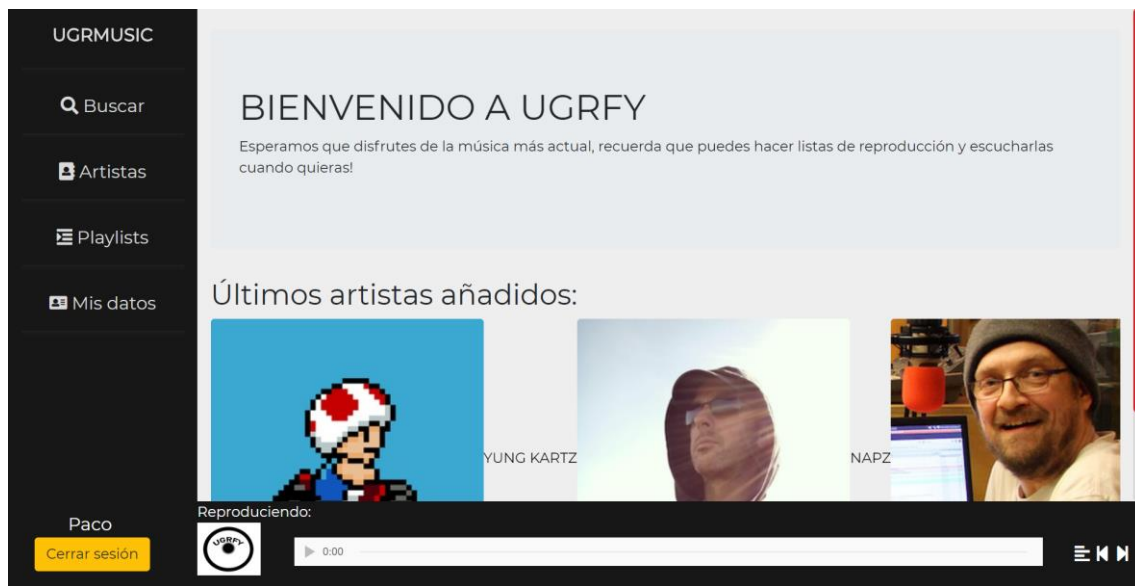


Imagen 12: Página principal página web

8.2.1.4 Actualizar mis datos

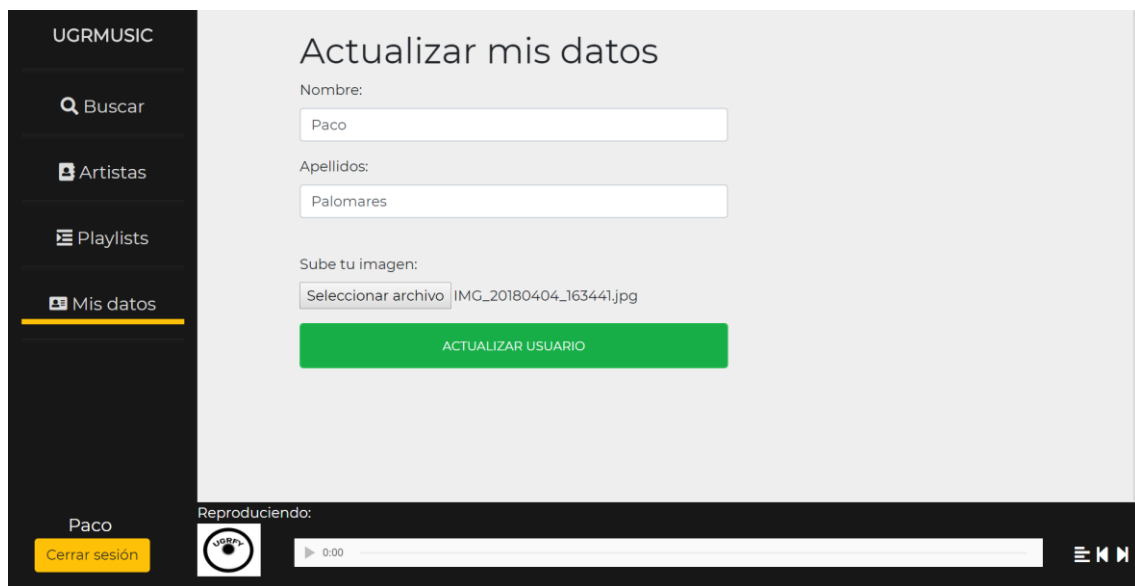


Imagen 13: Página mis datos página web

8.2.1.5 Página de artistas

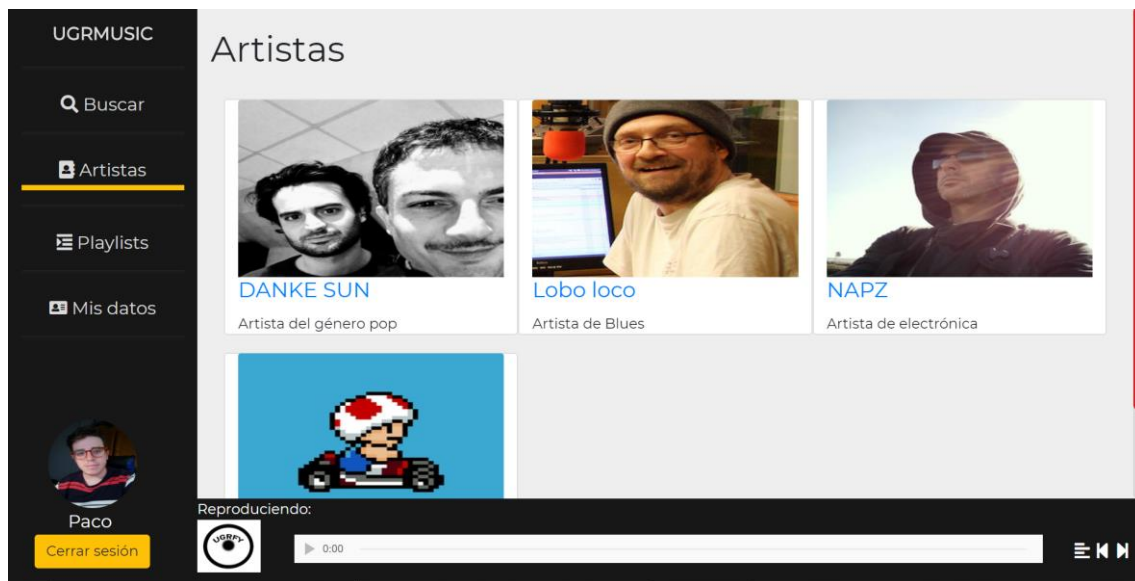


Imagen 14: Página artistas página web

8.2.1.6 Página de artista

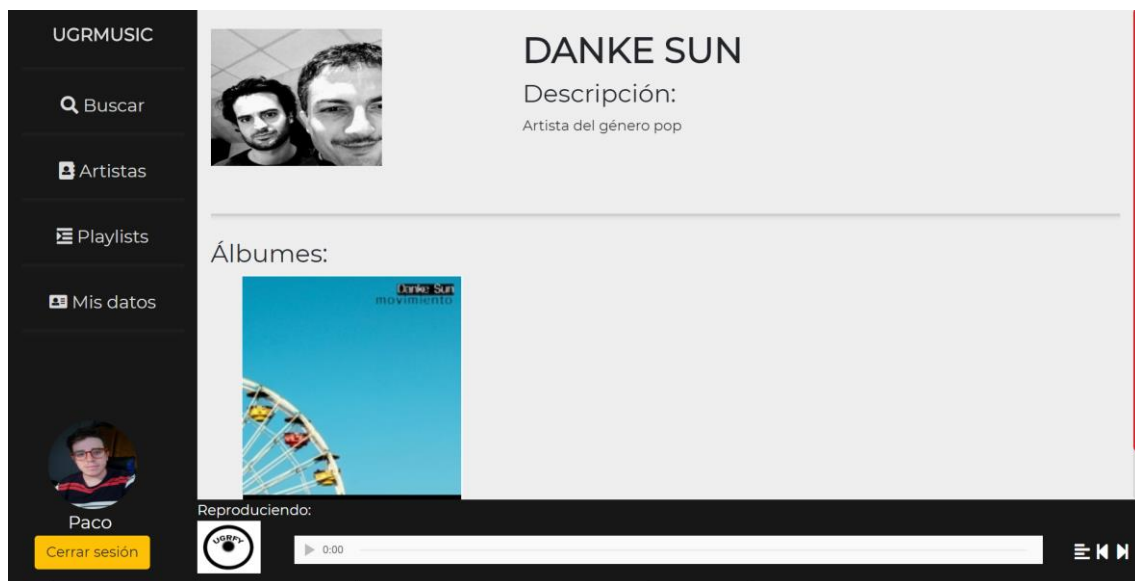


Imagen 15: Página artista página web

8.2.1.6 Página de álbum con reproducción de canción

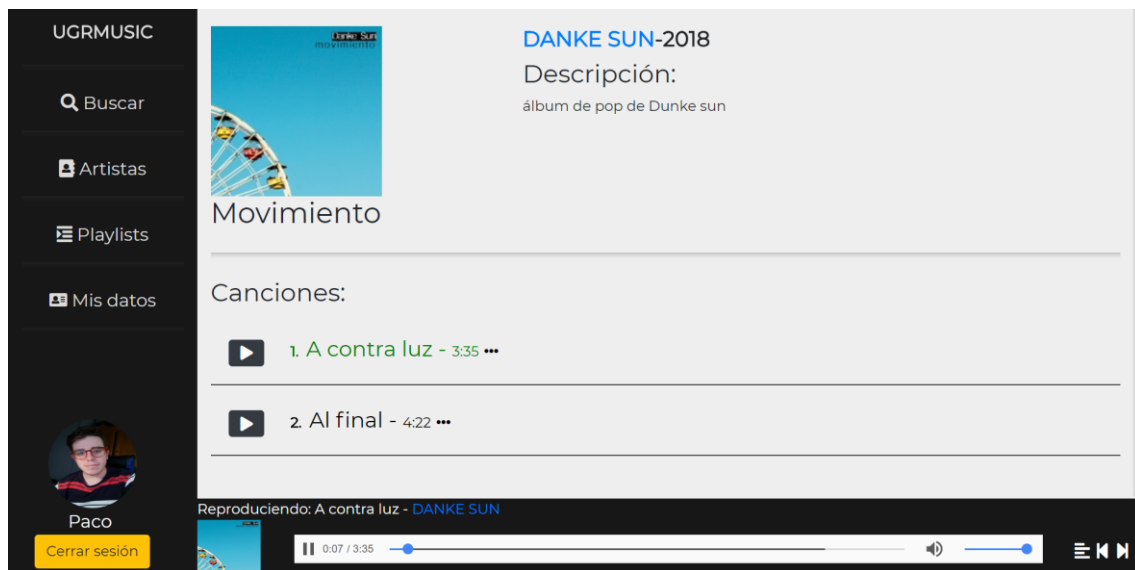


Imagen 16: Página álbum sesión página web

8.2.1.6 Página de álbum con vista de cola de canciones a reproducir

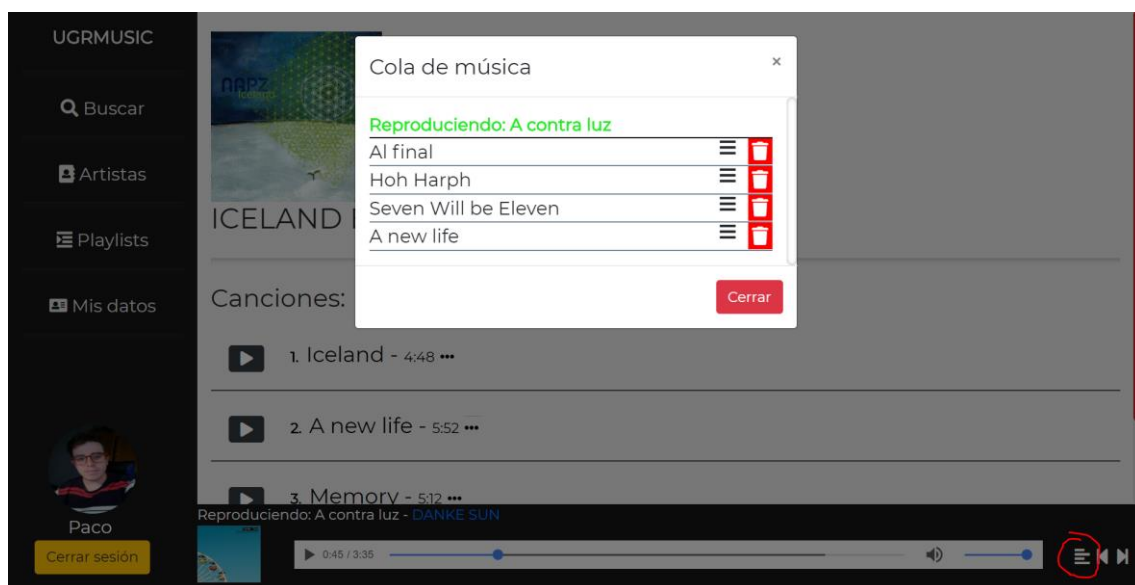


Imagen 17: Acceso a cola de canciones página web

8.2.1.7 Página de playlists

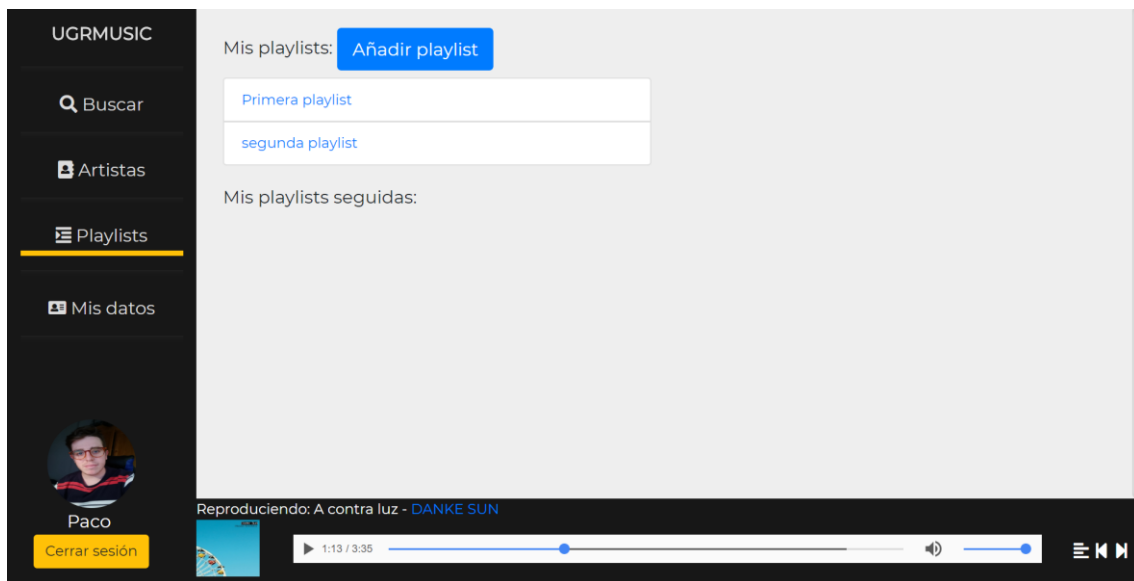


Imagen 18: Página playlists página web

8.2.1.8 Página de una playlist

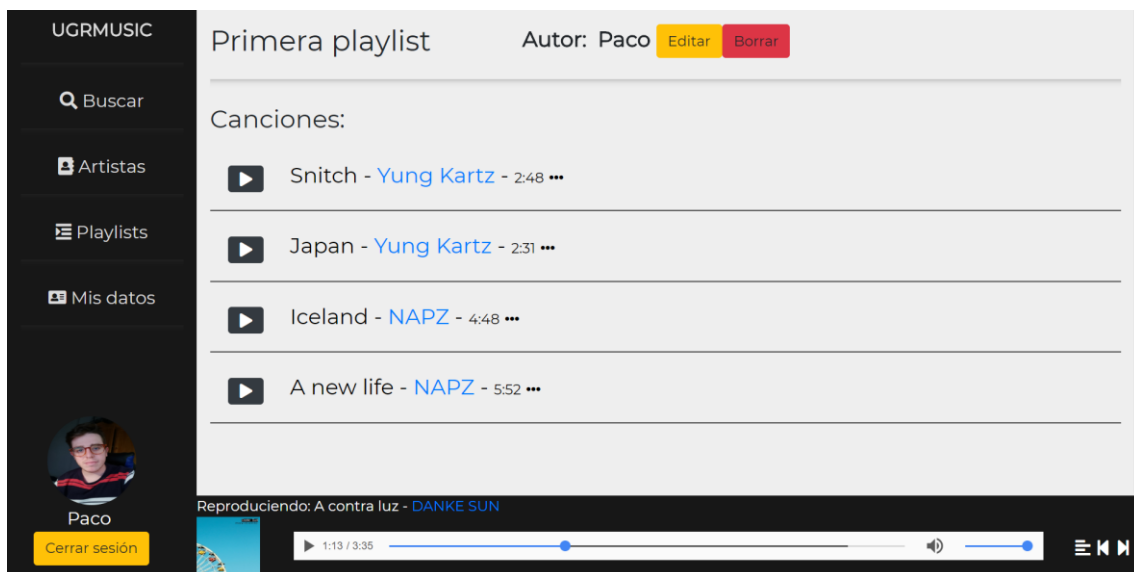


Imagen 19: Página playlist página web

8.2.1.9 Buscador

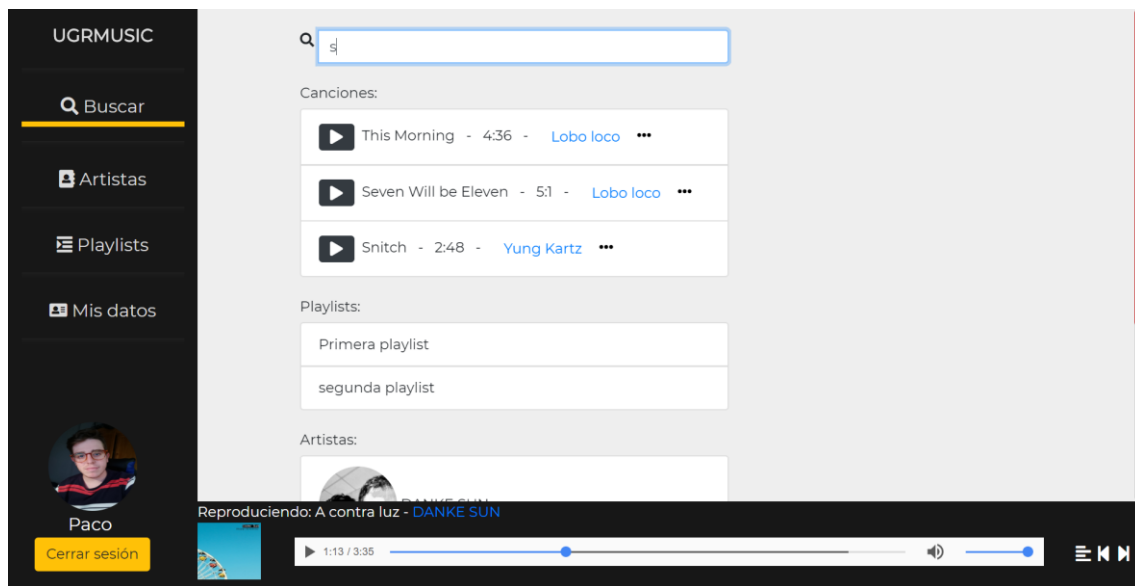


Imagen 20: Página búsqueda página web

8.2.1.9 Playlist de otro usuario

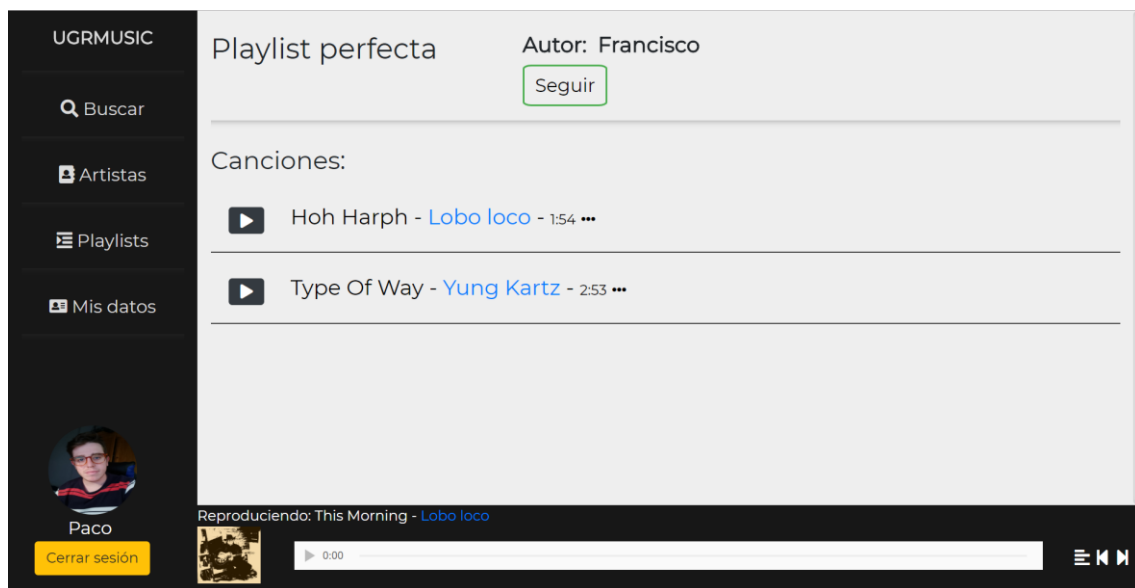


Imagen 21: Página playlist otro usuario página web

8.2.2 Aplicación móvil

8.2.2.1 Inicio



The screenshot shows the login interface of the UGRFY mobile application. At the top, the status bar displays signal strength, 78% battery, and the time 6:07. The app's logo, "UGRFY", is centered at the top in white. Below it, there are two white input fields with rounded corners: "Email:" and "Contraseña:". Under these fields is a grey button with the text "INICIA SESIÓN". Below the button is a link that says "CREAR UNA CUENTA". At the bottom, there is a link that says "¿Has olvidado contraseña?" followed by the word "Recordar". The background is a gradient from purple at the top to red at the bottom.

Imagen 22: Inicio sesión aplicación móvil

8.2.2.2 Registro



The screenshot shows the registration interface of the UGRFY mobile application. At the top, the status bar displays signal strength, 78% battery, and the time 6:07. A back arrow is in the top left corner. The title "Registro" is centered at the top in white. Below it, there are five white input fields with rounded corners: "Nombre:", "Apellidos:", "Email", "Contraseña", and "Repita contraseña". At the bottom, there is a grey button with the text "CREAR CUENTA". The background is a gradient from purple at the top to red at the bottom.

Imagen 23: Registro aplicación móvil

8.2.2.3 Principal

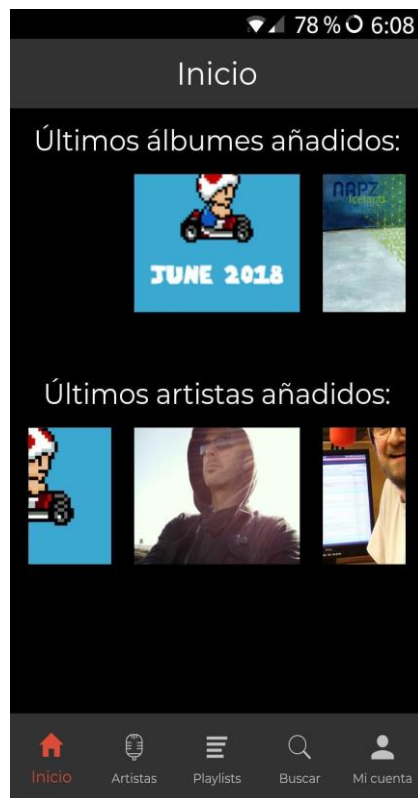


Imagen 24: Principal aplicación móvil

8.2.2.4 Artistas

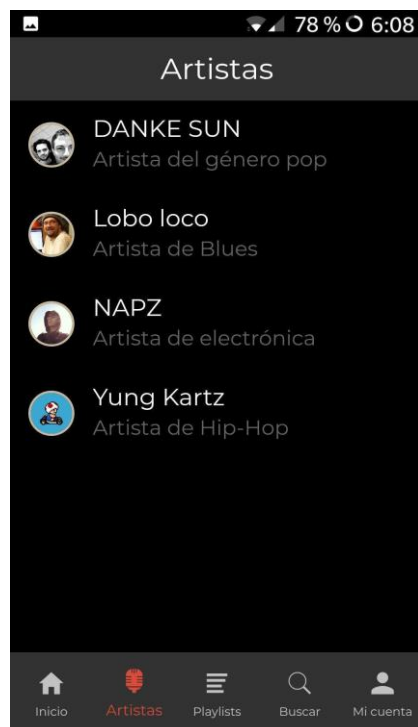


Imagen 25: Artistas aplicación móvil

8.2.2.5 Artista

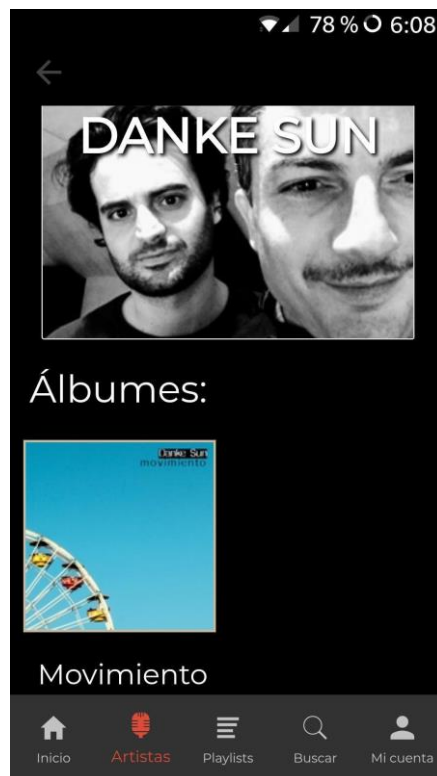


Imagen 26: Artista aplicación móvil

8.2.2.6 Álbum



Imagen 27: Álbum aplicación móvil

8.2.2.7 Álbum con canción reproduciendo



Imagen 28: Álbum con canción reproduciendo aplicación móvil

8.2.2.8 Opciones de canción



Imagen 29: Opciones canción aplicación móvil

8.2.2.9 Playlists

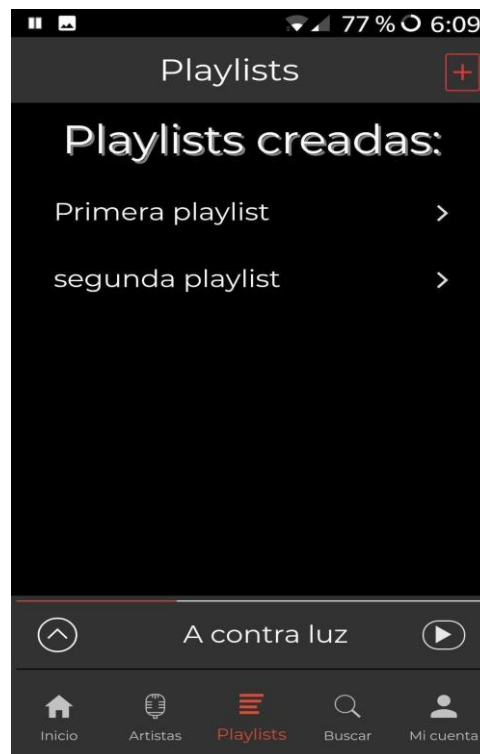


Imagen 30: Playlists aplicación móvil

8.2.2.10 Crear playlist

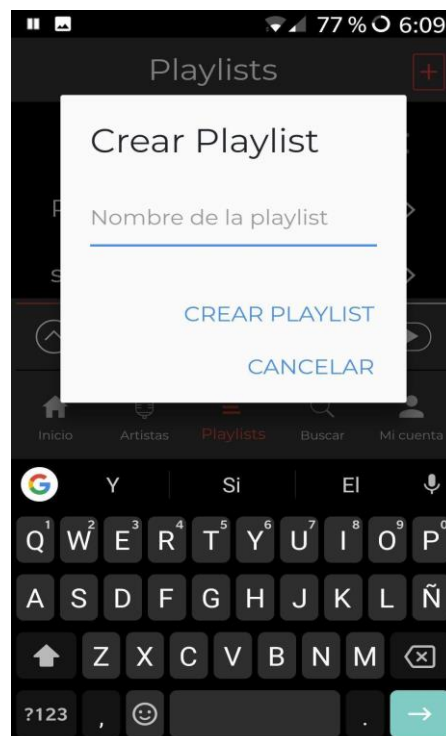


Imagen 31: Crear playlist aplicación móvil

8.2.2.11 Mi cuenta

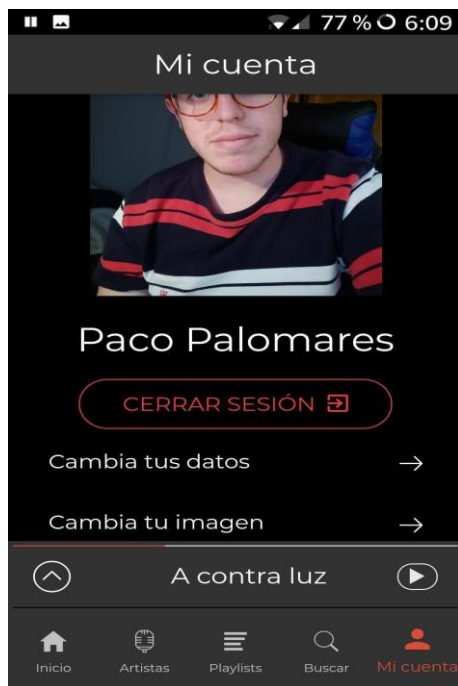


Imagen 32: Mi cuenta aplicación móvil

8.2.2.12 Cambio de imagen de usuario

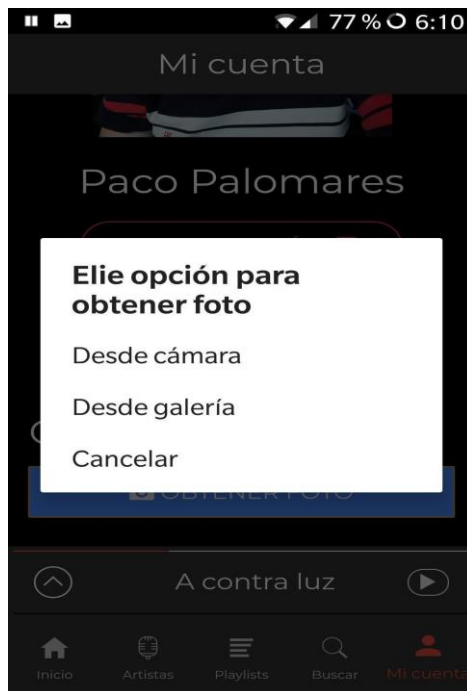


Imagen 33: Cambiar foto perfil aplicación móvil

8.2.2.13 Buscar

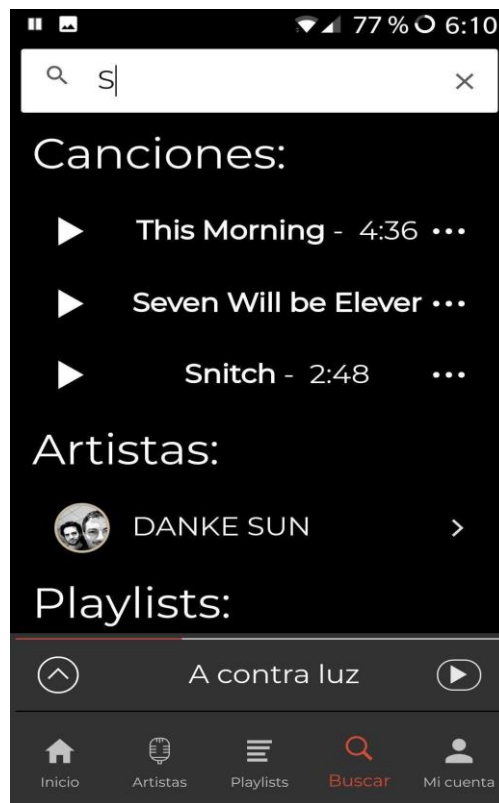


Imagen 34: Buscar aplicación móvil

8.2.2.15 Playlist

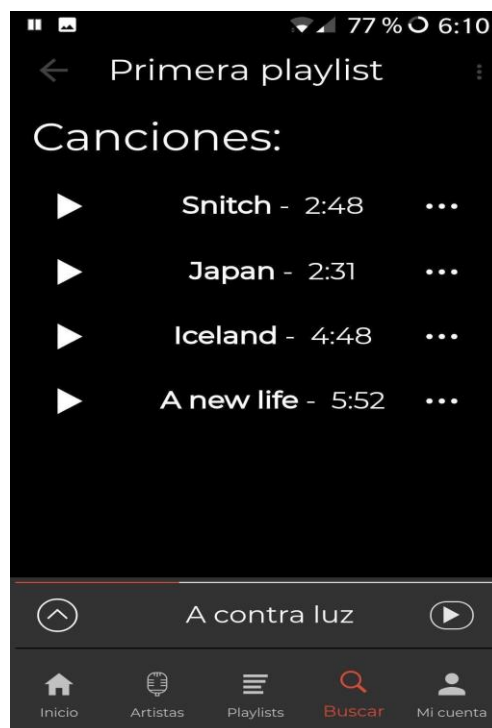


Imagen 35: Playlist aplicación móvil

8.2.2.16 Opciones de playlist

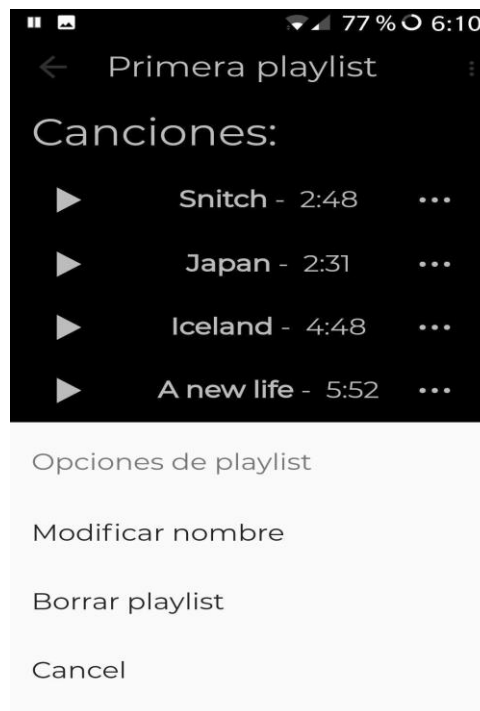


Imagen 36: Opciones playlist aplicación móvil

8.2.2.17 Playlist otro usuario



Imagen 37: Playlist otro usuario aplicación móvil

8.2.2.18 Notificación móvil para controles

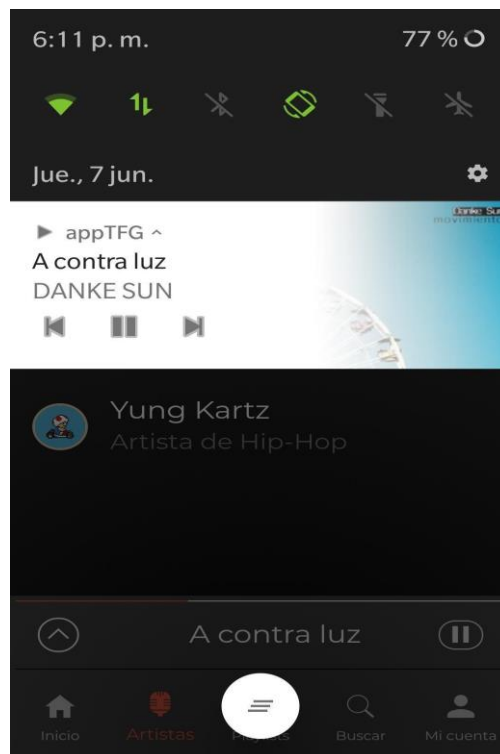


Imagen 38: Notificación con controles aplicación móvil

8.2.2.19 Reproductor

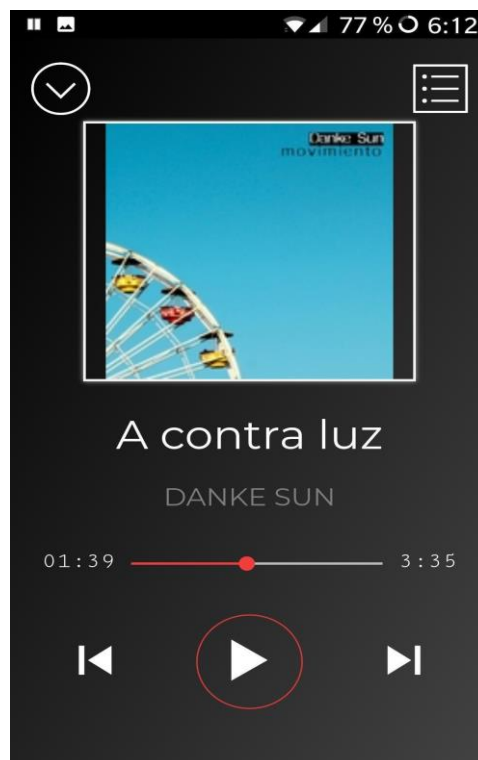


Imagen 39: Reproductor aplicación móvil

8.2.2.20 Cola de canciones

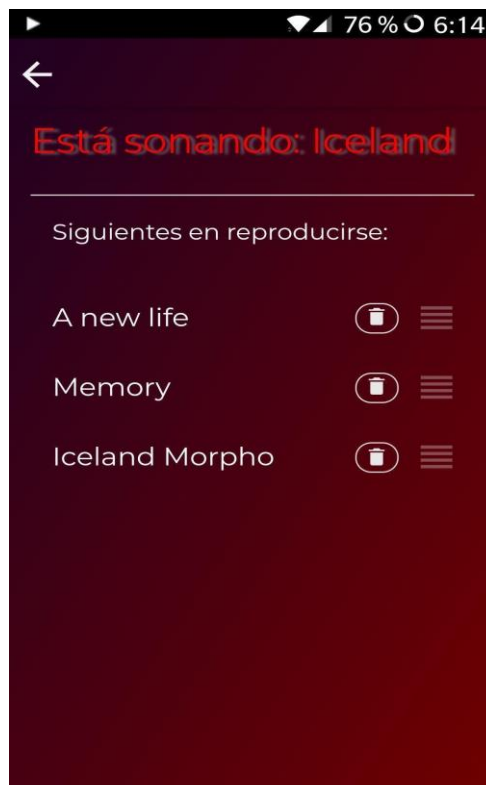


Imagen 40: Cola de canciones aplicación móvil

8.3 Glosario de términos

8.3.1 Términos

A continuación se expondrá un listado de términos específicos del dominio del problema con una breve definición de cada uno :

- **Streaming (retransmisión):** Es la distribución digital a través de una red de ordenadores , de manera que el producto es usado y descargado a la misma vez.
- **Backend:** Es la parte del servidor, el que interactúa con la base de datos y el que recibe los datos del frontend.
- **Frontend:** es la interfaz con la que interactúa el usuario y muestra los datos recibidos del Backend.
- **JavaScript:** Es un lenguaje de programación interpretado usado para la creación de aplicaciones web, tanto en frontend como en el backend.

- **Playlist:** Es una lista de canciones de diferentes artistas o de diferentes álbumes, de esta manera, el usuario escucha una lista de canciones según sus gustos sin navegar entre diversos artistas y álbumes.

8.3.2 Acrónimos

A continuación se expondrá un listado de acrónimos junto con su significado:

- **API:** Application Programming Interface (Interfaz de Programación de aplicaciones) , es el conjunto de funciones como una capa de abstracción que puede ser utilizado por otro software.
- **HTML:** Hypertext Markup Language (Lenguaje de marcas de hipertexto extendido), es el lenguaje de etiquetas principal para el desarrollo de páginas web.
- **CSS:** Cascading Style Sheets (Hojas de estilo de en cascada) , es un lenguaje usado para dar estilo a las páginas creadas con HTML.
- **IDE:** Integrated Development Environment (Entorno de desarrollo integrado) , es una aplicación de software para facilitar a un programador el desarrollo de un software.
- **HTTP:** HyperText Transfer Protocol (Protocolo de transferencia de hipertexto) , es el protocolo de comunicación que define cómo se crean y se transmiten mensajes de navegadores y servidores webs.