UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

DESARROLLO DE UNA APLICACIÓN PARA LA REPRODUCCIÓN MUSICAL VÍA STREAMING

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Marín Alós, Asier

Tutor/a: Molina Marco, Antonio

CURSO ACADÉMICO: 2021/2022

Resumen

La música es una parte fundamental en la vida de todas las personas, y los servicios de música en *streaming* han conseguido que el acceder a ella y disfrutar de tus canciones favoritas sea más sencillo y rápido que nunca. En la actualidad, solamente en el servicio de Spotify existen más de 400 millones de usuarios activos, siendo la gran pionera en este mercado. Aún así, existen alternativas que también son utilizadas por otros millones de usuarios.

En esta memoria se documentará el desarrollo de una aplicación de reproducción de música vía *streaming* en su versión de escritorio siguiendo los conceptos de una metodología ágil. Los usuarios podrán reproducir canciones y listas de reproducción y marcarlas como favoritas, crear sus propias listas de reproducción privadas o compartirlas con la comunidad, así como seguir a sus artistas favoritos, los cuales tendrán perfiles propios desde donde poder acceder a todos sus lanzamientos. Los usuarios también tendrán perfiles personalizables que podrán modificar en la sección correspondiente. Las diferentes interfaces están pensadas para facilitar el aprendizaje de nuevos usuarios, ofreciendo un diseño sencillo, intuitivo y de fácil uso, y se han diseñado y ejecutado diferentes pruebas para comprobar que los diferentes casos de uso funcionen correctamente y asegurar que los requisitos exigidos al proyecto sean satisfechos.

Para el desarrollo se han utilizado el lenguaje de programación Java, el entorno de desarrollo IntelliJ IDEA, el sistema de gestión de bases de datos MySQL y el sistema de almacenamiento de Google Drive.

Palabras clave: *streaming,* aplicación de escritorio, metodología ágil, Java, SQL, IntelliJ IDEA, Google Drive…

Resum

La música es una part fonamental en la vida de totes les persones, i els servicis de música en *streaming* han aconseguit que el accedir a ella i disfrutar de les teues cançons favorites siga més senzill i ràpid que mai. En l'actualitat, solament al servici de Spotify existeixen més de 400 milions d'usuaris actius, sent la gran pionera en aquest mercat. No obstant, existeixen alternatives que també són utilitzades per altres milions d'usuaris.

En aquesta memòria es documentarà el desenvolupament d'una aplicació de reproducció de música vía *streaming* en la seua versió d'escriptori seguint els conceptes d'una metodología àgil. Els usuaris podràn reproduir cançons i llistes de reproducció i marcarles com a favorites, crear les seues pròpies llistes de reproducció privades o compartir-les amb la comunitat, així com seguir als seus artistes favorits, els quals tindràn perfils propis des dels quals poder accedir a tots els seus llançaments. Els usuaris també tindràn perfils personalitzables que podràn modificar-se en l'apartat corresponent.

Les diferents interfícies estàn pensades per a facilitar l'aprenentatge de nous usuaris, oferint un disseny senzill, intuïtiu i de fàcil ús, i s'han dissenyat i executat diferents proves per a comprovar que els diferents casos d'ús funcionen correctament i assegurar que els requisits exigits al projecte siguen satisfets.

Per al desenvolupament s'han utilitzat el llenguatge de programació Java, l'entorn de desenvolupament IntelliJ IDEA, el sistema de gestió de bases de dades MySQL i el sistema d'emmagatzematge de Google Drive.

Paraules clau: *streaming,* aplicació d'escriptori, metodología àgil, Java, SQL, IntelliJ IDEA, Google Drive…

Abstract

Music is an essential part of all people 's life, and music streaming services made accessing it and enjoying your favorite songs easier and faster than ever. Nowadays, only in the Spotify service exist more than 400 million active users, being the greatest pioneer in this market. Despite this, there are some alternatives that are also used by millions of users.

This report will document the development of a music streaming app on its desktop version following the concepts of an agile methodology. Users will be able to play songs and playlists and save them as favorites, create their own private playlists or share them with the community, and also follow their favorite artists which will have their own profiles from which users will be able to access all artist's releases. Users will also have customizable profiles that could be modified in the corresponding section.

The different interfaces are designed to facilitate learning for new users, offering a simple and intuitive design, and different tests have been designed and executed to prove that the different use cases work as intended and to make sure that the requirements for the project are correctly fulfilled.

The app was developed using the Java programming language, the IntelliJ IDEA integrated development environment, the database management system MySQL and the Google Drive storage system.

Key words: streaming, desktop app, agile methodology, Java, SQL, IntelliJ IDEA, Google Drive…

Índice general

Índice de figuras

1.Introducción

1.1 Marco del proyecto

Este Trabajo de Fin de Grado consiste en el desarrollo de una aplicación para la reproducción de música *en streaming*, un tipo de servicio el cuál goza de gran popularidad a día de hoy y cuenta con múltiples exponentes como Spotify, Apple Music o SoundCloud [1].
La aplicación a desarrollar cuenta con todas las funcionalidades básicas que se esperan de un producto de este tipo más allá del propio reproductor, como un buscador con diversos filtros, creación de cuentas de usuario, la posibilidad de añadir canciones a listas de reproducción personalizadas, así como otras funcionalidades que serán descritas con mayor profundidad en secciones más avanzadas del documento.

Figura 1.1: aplicaciones de música en *streaming* (fuente: https://blog.syncios.com/the-best-music-streaming-apps-for-android/)

1.2 Motivación

Los servicios audiovisuales de *streaming* están altamente presentes en la vida de millones de personas en todo el mundo. Un estudio realizado en el año 2021 indica que casi el 60% de los españoles paga por algún servicio de streaming, es decir, más de la mitad de los españoles utiliza este tipo de servicios en su día a día [2]. Este dato es la prueba definitiva del enorme éxito entre la población de un estilo de servicio que llegó para quedarse, revolucionar y dominar el mercado, experimentando un gran crecimiento año tras año siendo especialmente notable el ocurrido durante los tiempos de la pandemia global ocasionada por el COVID-19.

La primera vez que el mundo escuchó la palabra *streaming* fue a mediados de la década de 1920 cuando una empresa de nombre Muzak desarrolló una plataforma de música para negocios, aunque por aquel entonces aún quedaban lejos los ordenadores tal y como los conocemos hoy. Pero no fue hasta la llegada de Internet que el *streaming* llegó a desarrollarse a niveles avanzados y permitió la compartición de contenido almacenado en la nube.

La música es una parte fundamental en la vida de muchas personas. Es muy común ver por la calle a gran cantidad de personas con dispositivos auriculares disfrutando de su música favorita, pero ésta también puede escucharse mientras se realiza cualquier tipo de actividad ya sea ejercicio, trabajo, etc. Es por eso que aplicaciones como Spotify se han convertido en algunas de las aplicaciones más descargadas y con más usuarios registrados a nivel mundial.

Ya sea en su versión móvil o en su versión para ordenadores, y tan solo requiriendo de una conexión a Internet, aplicaciones de *streaming* como la ya mencionada Spotify permite a sus usuarios escuchar todas las canciones de su catálogo (que se cuentan en decenas de millones), así como acceder a diversas formas de personalización de su experiencia, todo ello de forma rápida y con mucha mayor comodidad que utilizando el tradicional método de la descarga convencional.

Pero, ¿qué es exactamente el consumo vía *streaming?* Básicamente, la reproducción de contenido de esta forma permite al usuario disfrutar de contenido online sin tener que haberlo descargado completo previamente. Para esto se utiliza un búfer de datos, que no es más que un almacén temporal donde se va descargando el contenido al mismo tiempo que éste se va reproduciendo. Gracias a esta tecnología no solo se acaba el esperar a que una descarga finalice por completo, sino que además se ahorra espacio de almacenamiento.

El búfer es mucho más notorio en servicios de vídeo, ya que corresponden a archivos mucho más grandes que los archivos de audio. Un vídeo de un minuto en resolución HD puede llegar a los 60MB de peso, mientras que un archivo de audio en formato mp3 con una tasa de bits de 320kbps (la más alta disponible para un archivo de este formato) ocupará tan solo entre 2MB y 3MB.

1.3 Objetivos

Los principales objetivos a alcanzar durante el desarrollo de este proyecto son los siguientes:

- Realizar un análisis y diseño previos de la aplicación a desarrollar.
- Desarrollar el frontend e implementar las diversas funcionalidades que se esperarían de una aplicación perteneciente a un servicio de música en *streaming*: un reproductor, un buscador de elementos, una biblioteca…
- Desarrollar un backend que sea rápido y ofrezca una buena seguridad e integridad de los datos.
- Mostrar diferentes casos de uso que pueden realizarse en la aplicación.
- Realizar pruebas y tests que comprueben la satisfacción de los requisitos exigidos.
- Realizar una valoración final del proyecto y sacar conclusiones al respecto.

1.4 Estructura de la memoria

1. **Introducción**
   Se explican el contexto y las motivaciones del proyecto.
2. **Herramientas y tecnologías**
   Se detallan las herramientas y tecnologías utilizadas durante el desarrollo.
3. **Metodología de trabajo**
   Se introducen las metodologías de trabajo y se detalla la utilizada en el proyecto.
4. **Análisis del sistema**
   Se realiza un análisis y especificación de requisitos del sistema a diseñar.
5. **Diseño y modelado del sistema**
   Se muestra el diseño de la estructura de la aplicación y los diferentes patrones aplicados.
6. **Desarrollo del frontend**
   Se explica el diseño, desarrollo y navegabilidad de las interfaces que conforman la aplicación.

7. **Desarrollo del backend**
   Se explica la gestión de la base de datos y la comunicación con el frontend.

8. **Pruebas**
   Se detallan pruebas realizadas para comprobar la satisfacción de los requerimientos establecidos para el proyecto.

9. **Conclusiones finales**
   Se ofrecen las conclusiones finales del desarrollo, la valoración personal y una serie de posibles futuras mejoras.

10. **Bibliografía**

2.Herramientas y tecnologías

En este apartado se explicarán y detallarán las herramientas que han sido elegidas para el desarrollo del proyecto. Estas herramientas son: el lenguaje de programación Java, el IDE (Integrated Development Environment por sus siglas en inglés o Entorno de Desarrollo Integrado en español) IntelliJ IDEA de Jetbrains, el sistema de gestión de bases de datos MySQL y el sistema de almacenamiento de Google Drive.

Figura 2.1: herramientas utilizadas en el desarrollo

2.1 El lenguaje: Java

Java [3] es un lenguaje de programación introducido en 1995 y desarrollado por James Gosling y la empresa Sun Microsystems, constituida en 1983 y adquirida en 2010 por Oracle. Es uno de los tres lenguajes de programación más utilizados en la actualidad (junto a JavaScript y Python) .
Java ha sido el lenguaje de programación elegido para el desarrollo y realización de este proyecto. Estas son algunas de las razones que justifican su elección:

1. **Experiencia previa:** tanto con Java como con lenguajes similares como C++, Kotlin, etc.

2. **Orientado a objetos:** la *Programación Orientada a Objetos* (POO) es un paradigma de programación basado en el concepto de clases y objetos. Ofrece una gran serie de ventajas, como mayor facilidad en la resolución de problemas (gracias a la encapsulación), la reutilización de código mediante la herencia o la utilización del polimorfismo, entre otras.
3. **Gran soporte online y gran cantidad de librerías:** al ser un lenguaje de programación ampliamente conocido y utilizado en todo el mundo, existen numerosas guías, foros y ayudas en internet gracias a los cuales puedes encontrar solución a

numerosos problemas que puedan ocurrir durante el desarrollo. Además, existe una enorme cantidad de librerías descargables que ofrecen diferentes funcionalidades para enriquecer el proyecto y simplificar su desarrollo.

La principal ventaja que ha aportado el uso de Java durante el desarrollo del proyecto han sido las clases y librerías que han permitido la creación del sistema reproducción, especialmente las clases Media y MediaPlayer, que ofrecen distintos métodos para el tratamiento con archivos de audio que han permitido agilizar el desarrollo de esta parte del proyecto que, por obvias razones, es la más importante.

2.2 Desarrollo del frontend: IntelliJ IDEA

IntelliJ IDEA [4] es un entorno de desarrollo integrado creado por la compañía checa JetBrains, la cual es responsable también del desarrollo y creación de otros exitosos productos como PhpStorm (IDE para el desarrollo en PHP), Pycharm (IDE para el desarrollo en Python) o el lenguaje de programación Kotlin, especializado en el desarrollo para dispositivos móviles Android. De hecho, el popular IDE para el desarrollo en dispositivos Android de Google, Android Studio, está enteramente basado en el software de IntelliJ IDEA.

Entre algunas de sus características más destacables se encuentra la asistencia automática a la hora de programar (mostrando sugerencias y autocompletando código), una interfaz amigable y poco intrusiva, y un depurador muy potente y fácil de usar para una mejor detección de errores, entre otras cosas.

Debido a su facilidad de uso, sus herramientas y sus funcionalidades para la ayuda en la programación, junto a su parecido con el IDE Android Studio con el que desarrollé un proyecto no muchos meses antes de la realización de este trabajo, considero que utilizar IntelliJ IDEA ha ayudado en gran medida a la realización del proyecto y ha reducido los problemas especialmente relacionados a la creación y configuración inicial.

2.3 Gestión del backend: MySQL

MySQL [5] es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation. Se trata de la base de datos de código abierto más popular del mundo y una de las más populares a nivel general.
Para el desarrollo del proyecto se ha utilizado una base de datos de tipo SQL para la gestión del backend. Los diferentes tipos de datos se almacenan en diferentes tablas, cada una con sus respectivas columnas, y se obtienen en la aplicación mediante consultas realizadas en las clases DAO (la estructura y funcionamiento del backend se explicará en profundidad en el punto 6 de esta memoria).

Tras un período de análisis, se concluyó que era más conveniente utilizar un sistema de base de datos SQL frente a uno NoSQL como podrían ser, por ejemplo, Firebase, MongoDB o Cassandra. El principal motivo de esta decisión es que, si bien las bases de datos SQL no presentan la misma flexibilidad y gran escalabilidad que presentan las bases NoSQL, para un proyecto a pequeña escala como éste se ha decidido dar prioridad a una mayor consistencia y una mejor estructuración de los datos, ya que no se necesitarán las características de las bases NoSQL que las hacen destacar en los proyectos a enormes escalas (como el Big Data). [6]

Antes de empezar con la parte del desarrollo correspondiente a la programación, se realizó la planificación y el diseño de todas las tablas y los datos que se creían necesarios. Pese a que han existido algunos cambios durante el proceso de desarrollo y pruebas, han resultado ser mínimos.

2.4 Almacenamiento: Google Drive

Google Drive es un servicio de almacenamiento de archivos introducido por Google el 24 de abril de 2012. Es el servicio de alojamiento en la nube que se ha utilizado para guardar los archivos de audio y vídeo pertenecientes a las canciones disponibles en la aplicación. Para la comunicación con el servicio se utiliza la API de Google Drive disponible en la página para desarrolladores de Google.
La razón de su elección es que se trata de un sistema gratuito de los más conocidos y populares en todo el mundo, lo cual ha facilitado la búsqueda de información sobre su uso. Además, su implementación es poco costosa.

3.Metodología de trabajo

3.1 ¿Qué es una metodología?

La palabra metodología hace referencia al conjunto de procedimientos racionales que se llevan a cabo con el fin de alcanzar un determinado objetivo. Refiriéndonos específicamente a la metodología software, ésta puede ser definida como un conjunto de técnicas y métodos organizativos aplicados al diseño de software. Es decir, el objetivo de una metodología es el de organizar el trabajo a realizar en un proyecto para que éste pueda ser desarrollado con la mayor eficacia posible.

Existen dos grandes tipos de metodología software: las metodologías tradicionales y las metodologías ágiles. Para el desarrollo de éste proyecto se ha decidido seguir una metodología de trabajo ágil. Aún así, vamos a explicar un poco más en profundidad los dos tipos de metodología. [7]

3.2 Las metodologías tradicionales

Las metodologías tradicionales, también llamadas metodologías rígidas, son aquellas metodologías de trabajo en las cuales se establecen todos y cada uno de los requisitos del proyecto al inicio de su ciclo de producción. Este tipo de metodologías funcionan de una forma lineal (o secuencial), es decir, una etapa debe completarse antes de poder iniciar la siguiente.

Utilizando este tipo de metodología se llevará a cabo un mejor y mayor análisis de las etapas de un proyecto, así como una mejor documentación y, en definitiva, un plan de proyecto a seguir en todo momento. Se podría decir que son metodologías de desarrollo de tipo 'perfeccionista' dada la gran cantidad de análisis previos y estructuración del desarrollo.

No obstante, el mayor problema de este tipo de metodologías radica en la poca (o nula) flexibilidad a la hora de realizar cambios. Como ya se ha comentado, en este tipo de desarrollos se realiza un análisis completo de requisitos al inicio del proyecto intentando minimizar la necesidad de posibles cambios en el futuro. Es por eso que en el momento que surgen nuevas necesidades a implementar, éstas pueden tener un coste muy alto.

3.3 Las metodologías ágiles

El segundo tipo de metodologías son las llamadas 'metodologías ágiles'. Al contrario que en las metodologías tradicionales, en este tipo de metodología no se necesitan definir todos los requisitos del proyecto al inicio de éste. En este tipo de metodologías se 'fragmentan' los proyectos en diferentes partes capaces de adaptarse sobre la marcha, es decir, no se planifica o diseña todo el proyecto por adelantado (como sí ocurre en las metodologías tradicionales), sino que se va puliendo y mejorando conforme avanza al desarrollo gracias al *feedback* constante.

Como se ha dicho anteriormente, para el desarrollo de este proyecto se ha optado por trabajar con una metodología ágil siguiendo conceptos pertenecientes a las metodologías tipo Scrum [8], siendo los dos más importantes las creación de un *Product Backlog* y la división del proyecto en diferentes iteraciones o *sprints*.

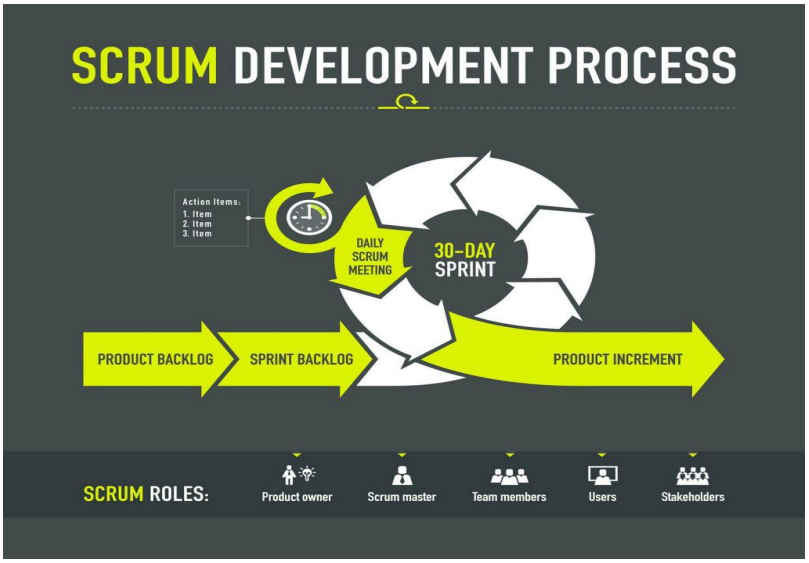

Figura 3.1: representación de la metodología *Scrum* (fuente: https://www.digite.com/agile/scrum-methodology/)

3.4 Metodología *Scrum*: división en sprints

Al inicio del proyecto se creó un *Product Backlog*, es decir, una lista de tareas pendientes ordenadas por prioridad y que, posteriormente, fueron asignadas a los distintos sprints durante sus períodos de preparación.

Anteriormente ya se ha mencionado que el proyecto fue dividido en un total de tres sprints, teniendo cada uno de ellos una duración de tres semanas (21 días). Aparte de estas tres divisiones existe también un 'Sprint 0', una fase inicial previa a todas las demás en la que se establecieron las bases del proyecto. La descripción de cada uno de los sprints realizados es la siguiente:

- Sprint inicial o Sprint 0: esta fase, previa a los tres sprints principales, se centra en establecer una serie de objetivos a cumplir y unas bases de trabajo. Las actividades principales desarrolladas son la estructuración del Backlog y el listado de características y requisitos a cumplir por el producto final.

- Sprint 1: en el primero de los sprints se tiene como objetivo crear una primera versión utilizable tanto del reproductor como del buscador sin entrar en demasiados detalles, simplemente para comprobar el correcto funcionamiento de ambas funcionalidades.
  Al final de esta etapa, en la aplicación se deberían poder realizar las siguientes acciones:
  - Buscar canciones en el buscador.
  - Reproducir una canción desde la lista de resultados del buscador.
  - Pausar y detener la reproducción de una canción.
  - Modificar el volumen del reproductor.
  - Interactuar con la barra de reproducción.

- Sprint 2: en este segundo sprint se tiene como objetivo terminar con las funcionalidades del reproductor y del buscador principal, dejándolos prácticamente en su versión definitiva. También se incluye en este sprint el desarrollo completo del sistema de registro e identificación de usuarios. Esto incluye la creación de las funcionalidades de inicio de sesión y registro de usuario, junto con las interfaces relacionadas a los perfiles de usuario para su vista y edición.
  Al final de esta etapa, en la aplicación se deberían poder realizar las siguientes acciones:
  - Iniciar sesión con una cuenta de usuario.
  - Crear una nueva cuenta de usuario.
  - Modificar elementos del perfil de usuario: contraseña, descripción, etc.
  - Buscar usuarios, artistas, álbumes y listas de reproducción desde el buscador.
  - Reproducir una canción desde una lista de reproducción.
  - Navegar listas de reproducción desde el reproductor con botones para avanzar o retroceder canciones.

- Sprint 3: el tercer y último sprint se centra principalmente en el desarrollo de la interfaz de la Biblioteca y de las funcionalidades de usuario como la creación y posterior modificación de playlists o el sistema de favoritos.

Al final de esta etapa se debe tener la aplicación con sus tres partes principales (‘Buscador’, ‘Mi perfil’ y ‘Biblioteca’) completamente terminadas y con las funcionalidades básicas completamente implementadas. Se deberían poder realizar las siguientes nuevas acciones:

- Crear una nueva playlist.
- Añadir o eliminar canciones de una playlist ya existente.
- Marcar canciones, álbumes o playlists como favoritos.
- Seguir a artistas.

Una vez descritas las etapas del proyecto, es hora de pasar al análisis del sistema.

4. Análisis del sistema

En este apartado se realizará el análisis del sistema a desarrollar. Se detallará el diagrama con los principales casos de uso que los usuarios podrán realizar en la aplicación y se realizará la especificación de requisitos funcionales y no funcionales.

4.1 Descripción del sistema a desarrollar

El sistema que se va a desarrollar es una aplicación de escritorio perteneciente a un servicio de reproducción vía *streaming*. La aplicación poseerá un buscador donde podrán encontrarse canciones, artistas, álbumes, playlists o perfiles de otros usuarios, una interfaz de personalización de perfil, y una biblioteca donde almacenar las canciones, álbumes y playlists favoritas, así como crear, editar y compartir playlists propias. Como es obvio, también dispondrá de un reproductor con todas las funcionalidades necesarias para la reproducción de canciones y listas de reproducción.
Pese a que los usuarios deberán de registrarse y crear una cuenta en la aplicación para acceder a todas las funcionalidades ofrecidas por la aplicación, la aplicación también puede utilizarse sin la necesidad de crear una cuenta.

4.2 Diagrama de casos de uso

Un diagrama de casos de uso es una forma de diagrama UML que permite visualizar las diferentes interacciones que pueden realizar los diferentes actores participantes en un sistema. En la figura 4.1 se muestra el diagrama que contiene los principales casos de uso que pueden llevarse a cabo en la aplicación.

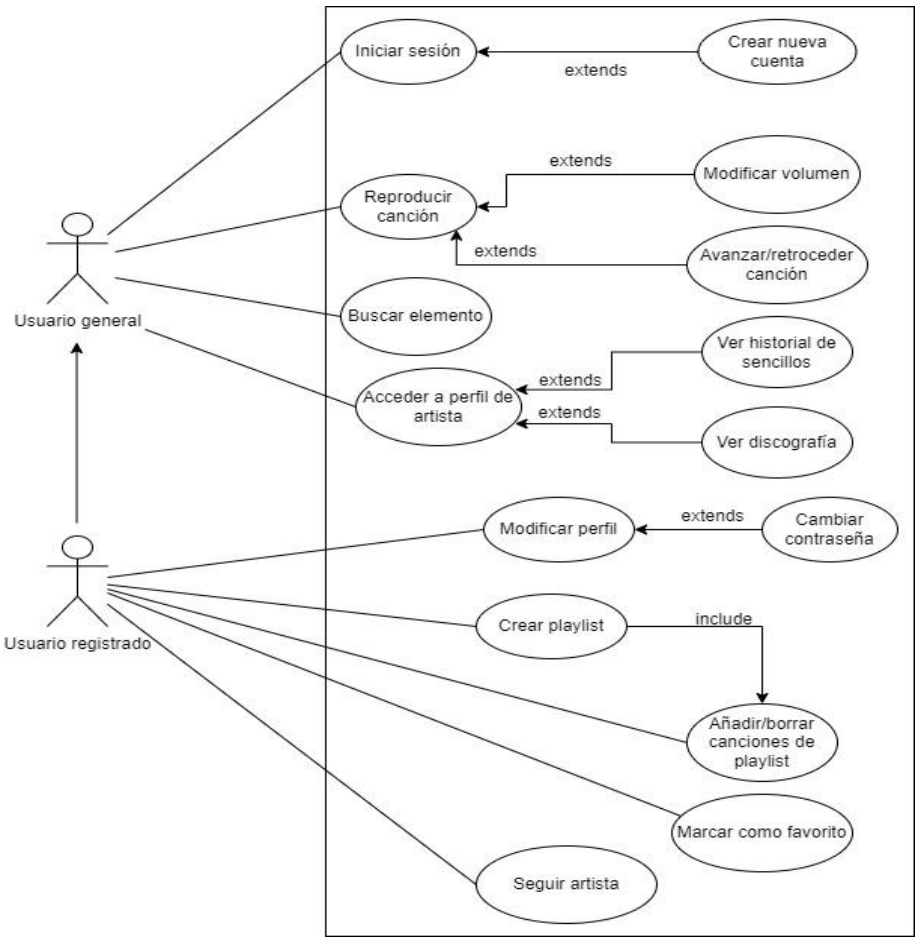

Figura 4.1: diagrama de casos de uso

Como se puede ver en el diagrama, existen dos actores en el sistema, que representan a los dos tipos de usuarios existentes en la aplicación: los usuarios generales y los usuarios registrados. Los primeros, que son aquellos que no se encuentran registrados en la aplicación, tan solo podrán acceder a las funcionalidades del buscador y del reproductor de canciones, mientras que aquellos usuarios que disponen de cuenta de usuario podrán acceder también a la modificación de perfiles, la funcionalidad de favoritos, la creación y edición de playlists y al seguimiento de artistas.

4.3 Especificación de requisitos

La especificación de requisitos (ERS) es una descripción completa del sistema que se va a desarrollar, describiendo qué es lo que hará el software y cómo lo hará. Una buena ERS debe de cumplir una serie de características, entre ellas:

- No ambigüedad
- Trazabilidad

- Fácil de verificar
- Fácil de modificar
- Consistente

Una ERS está formada por los requisitos funcionales y los requisitos no funcionales [9].

4.3.1 Requisitos funcionales

Los requisitos funcionales describen una función del sistema software o de alguno de sus componentes.
Algunos de los requisitos funcionales obtenidos después de realizar la especificación de requisitos de la aplicación a desarrollar son los siguientes:

Identificador	RQF-001
Nombre	Reproducción de una canción
Características	El usuario selecciona una canción para reproducir.
Descripción	Los usuarios deberán reproducir las canciones directamente desde el buscador o desde el contenido de un álbum o lista de reproducción.
Entradas	
Canción seleccionada	

Identificador	RQF-002
Nombre	Modificar volumen
Características	El reproductor dispone de controlador de volumen.
Descripción	El reproductor dispondrá de un controlador de volumen propio independiente del sistema sobre el que se ejecute.
Entradas	
Valor del controlador	

Identificador	RQF-003
Nombre	Avanzar/retroceder canción
Características	El usuario avanza a la siguiente canción o reproduce la canción

	anterior.
Descripción	Dentro de una lista de reproducción, al pulsar el botón de ‘Siguiente canción’ el reproductor avanzará a la canción que va a continuación, mientras que si se pulsa el botón de ‘Canción anterior’ se reproducirá la canción que precede a la canción actual.
Entradas	

Identificador	RQF-004
Nombre	Registro de usuario
Características	El usuario se registra en la aplicación.
Descripción	Los usuarios deberán registrarse en la aplicación para acceder a la funcionalidad de favoritos o a la creación de playlists, entre otras.
Entradas	
Correo electrónico de usuario Nombre de usuario Contraseña de usuario	

Identificador	RQF-005
Nombre	Inicio de sesión
Características	El usuario se identifica en la aplicación.
Descripción	Los usuarios deberán iniciar sesión utilizando el correo electrónico y una contraseña.
Entradas	
Correo electrónico de usuario Contraseña de usuario	

Identificador	RQF-006
Nombre	Modificación de contraseña
Características	El usuario cambia su contraseña actual.
Descripción	Los usuarios deberán poder modificar su contraseña en la sección ‘Configuración de perfil’, introduciendo su contraseña actual y su nueva contraseña.
Entradas	
Contraseña de usuario Nueva contraseña	

Identificador	RQF-007
Nombre	Funcionalidad ‘Favoritos’
Características	El usuario marca una canción/playlist como favorita.
Descripción	Los usuarios podrán marcar canciones y listas de reproducción como favoritas mediante un botón con forma de corazón.
Entradas	
Usuario identificado Elemento seleccionado (canción/playlist)	

Identificador	RQF-008
Nombre	Creación de listas
Características	El usuario crea una lista de reproducción propia.
Descripción	Los usuarios pueden crear listas de reproducción propias que podrán configurar como públicas (disponibles en el buscador) o privadas.
Entradas	
Nombre de nueva lista de reproducción Imagen de portada de playlist Checkbox ‘Pública’ Cancion/es a añadir	

Identificador	RQF-009
Nombre	Añadir/eliminar canciones
Características	El usuario selecciona la opción ‘Añadir canción’ o ‘Eliminar canción’ de una playlist.
Descripción	Los usuarios podrán editar sus playlists añadiendo o eliminando canciones.
Entradas	
Playlist seleccionada Cancion/es a añadir/eliminar	

Identificador	RQF-010
Nombre	Filtros de búsqueda
Características	El buscador filtra los resultados de las búsquedas.
Descripción	El buscador dispondrá de diferentes filtros para distinguir los diferentes resultados obtenidos en una búsqueda.
Entradas	
Texto introducido en el buscador Filtro seleccionado	

Identificador	RQF-011
Nombre	Mostrar discografía
Características	Se muestra una lista de todos los álbumes publicados por un artista.
Descripción	Al seleccionar la opción ‘Ver discografía’ en el perfil de un artista, se le mostrará a los usuarios una lista de los álbumes publicados por dicho artista con un orden cronológico descendente.
Entradas	
Artista	

4.3.2 Requisitos no funcionales

Los requisitos no funcionales (llamados también atributos de calidad) representan características generales, condiciones y restricciones que se imponen sobre la aplicación que se va a desarrollar. Pueden ser de muchos tipos: rendimiento, escalabilidad, eficiencia, seguridad…
Algunos de los requisitos no funcionales obtenidos después de realizar la especificación de requisitos de la aplicación a desarrollar son los siguientes:

- **Disponibilidad**:
  - La aplicación deberá estar disponible un mínimo del 99% del tiempo, siendo el 1% restante posibles errores o pruebas de mantenimiento que requieran de la suspensión temporal del servicio.
  - El tiempo de recuperación de un fallo en un servicio no debe ser superior a los cinco minutos.

- **Eficiencia**:
  - El resultado de la modificación de los datos en la base de datos deberá ser visible por los usuarios que accedan un mínimo de cinco segundos tras los cambios.
  - El tiempo que debe tardar una búsqueda en devolver resultados no debe de exceder los tres segundos.

- **Rendimiento**:
  - La aplicación deberá funcionar siempre por debajo del 15% del consumo de CPU.
  - La aplicación no deberá ocupar más de 100MB de almacenamiento en el disco duro en su versión de lanzamiento.
  - La aplicación no deberá consumir más de 300MB de memoria RAM.

- **Usabilidad**:
  - El tiempo máximo en el que un usuario promedio debe tardar en aprender a utilizar la aplicación en su totalidad es de una hora.
  - La aplicación debe proporcionar mensajes informativos para informar o prevenir posibles errores que el usuario pueda realizar.
  - El sistema debe de poseer un diseño intuitivo que facilite la navegabilidad y minimice los errores del usuario.

- **Escalabilidad**:
  - Los servidores deben alojar una gran cantidad de transacciones simultáneas de parte de múltiples usuarios.

- **Portabilidad**:
    - La aplicación debe de ser compatible para su ejecución en los sistemas operativos Windows, MacOS y Linux.

- **Mantenibilidad**:
    - El código de la aplicación debe de ser fácilmente legible y facilitar su mantenimiento a lo largo del tiempo.

5. Diseño y modelado del sistema

A continuación se describirán los patrones de diseño utilizados en el diseño de la aplicación y se explicará la estructura del sistema, los diferentes elementos que lo componen y todas las relaciones que existen entre ellos.

5.1 Patrones de diseño

Los patrones de diseño son técnicas utilizadas para la resolución de problemas durante el desarrollo de software y otros ámbitos referentes al diseño de interacción o de las interfaces. Son, en definitiva, soluciones a distintos problemas de diseño que deben poseer ciertas características para ser consideradas como patrones, siendo la principal de ellas la reutilización, es decir, que se pueda aplicar una misma solución a diferentes problemas de diseño en distintas circunstancias y sea igualmente efectiva.

Según la escala o nivel de abstracción, los patrones pueden clasificarse en:

- **Patrones de arquitectura**: patrones que expresan un esquema organizativo estructural fundamental para sistemas de software.
- **Patrones de diseño**: patrones que expresan esquemas para definir estructuras de diseño o las relaciones entre las mismas con las que construir sistemas de software.
- **Dialectos**: patrones específicos para un entorno o un lenguaje de programación concretos.

Cabe destacar también, a parte de los tipos de patrones ya mencionados, la existencia de los ‘antipatrones de diseño’, que son patrones de diseño que conducen a malas soluciones. Sirven como referencia para saber qué prácticas evitar a la hora de diseñar una solución a un problema.

A continuación se van a explicar los dos patrones principales que se han seguido para facilitar el desarrollo y la construcción del sistema. Ambos son patrones arquitectónicos utilizados de forma complementaria para lograr un mismo objetivo: la separación y diferenciación de los tipos de código en la aplicación. Estos dos patrones son: el patrón MVC (Modelo-Vista-Controlador) y el patrón DAO (*Data Access Object*).

5.1.1 El patrón MVC

El patrón MVC (Modelo - Vista - Controlador) [10] tiene el objetivo de separar el código de la aplicación según sus diferentes responsabilidades dentro de la misma, distinguiendo entre tres componentes principales:

- Modelo: es la representación de la información que se utiliza en la aplicación y gestiona todas las acciones realizadas sobre la misma, tanto consultas como actualizaciones.

- Vista: es el componente encargado de comunicarse directamente con el usuario, es decir, representa lo que el usuario visualiza y con lo que interactúa durante el uso de la aplicación (**interfaces de usuario**).

- Controlador: es el componente encargado de ser el intermediario entre el **modelo** y la **vista**. Responde a los eventos causados por el usuario al realizar alguna interacción y se encarga de obtener los datos solicitados al **modelo** para que puedan ser representados de forma visual por las **vistas**.

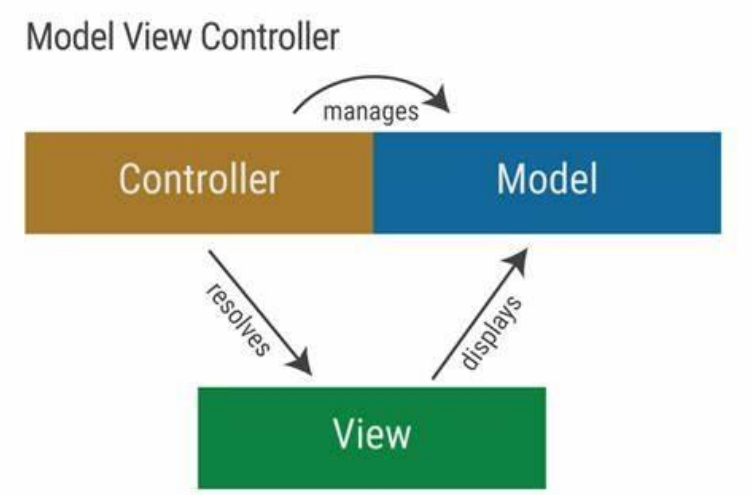

Figura 5.1: el patrón MVC (fuente: https://blog.famoser.ch/mvc-pattern/)

El funcionamiento del patrón a la hora de la ejecución de la aplicación es el siguiente:

1. El usuario interactúa con alguna de las partes de la interfaz (por ejemplo, al apretar un botón) y realiza una solicitud de datos.
2. El **controlador** se comunica tanto con la parte del **modelo** como con la **vista**. A la primera se le piden los datos solicitados por el usuario (o se realizan las

actualizaciones de datos necesarias), mientras que a la segunda se le pide la salida a la que irán destinados los datos.

3. Una vez terminada la parte de la lógica de negocio por parte del controlador, las **vistas** envían la salida al usuario, que puede ser una notificación, un cambio en la interfaz actual, mostrar una nueva interfaz, etc.

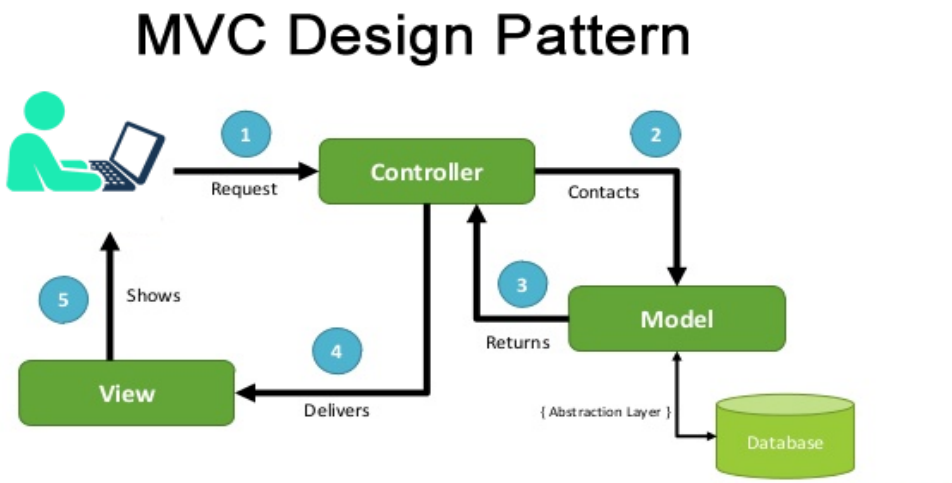

Figura 5.2: comportamiento del patrón MVC (fuente: https://www.duplicatetransaction.com/mvc-design-pattern-with-a-php-example/)

5.1.2 El patrón DAO

El patrón DAO (Data Access Object) [11] tiene como objetivo la completa separación de la lógica de negocio y la lógica de acceso a datos. La capa DAO será la encargada de proporcionar los métodos necesarios para consultar, insertar, modificar o eliminar datos de la base de datos, mientras que la capa de negocio sólo contendrá la lógica de negocio y utilizará a la DAO como medio para comunicarse con la base de datos.

Se han diseñado diferentes interfaces DAO correspondientes a las distintas fuentes disponibles en el almacén de datos. Por ejemplo, existe una interfaz correspondiente a las operaciones para los distintos tipos de búsqueda de canciones (búsqueda por nombre, por artistas involucrados, por género…). Este mismo tipo de interfaz existe también para los demás tipos de objetos.

Las interfaces creadas son implementadas en las clases DAO que serán inicializadas como objetos en la capa de negocio, la cual las utilizará para realizar las operaciones

correspondientes en base de datos. Se verán detalles más en profundidad en el apartado 7 de esta memoria.

5.2 Estructura de la aplicación (diagrama de clases)

Para analizar la estructura de la aplicación y las relaciones entre los diferentes tipos de datos que se van a tratar se ha realizado un diagrama de clases UML, mostrado a continuación en la figura 5.3.

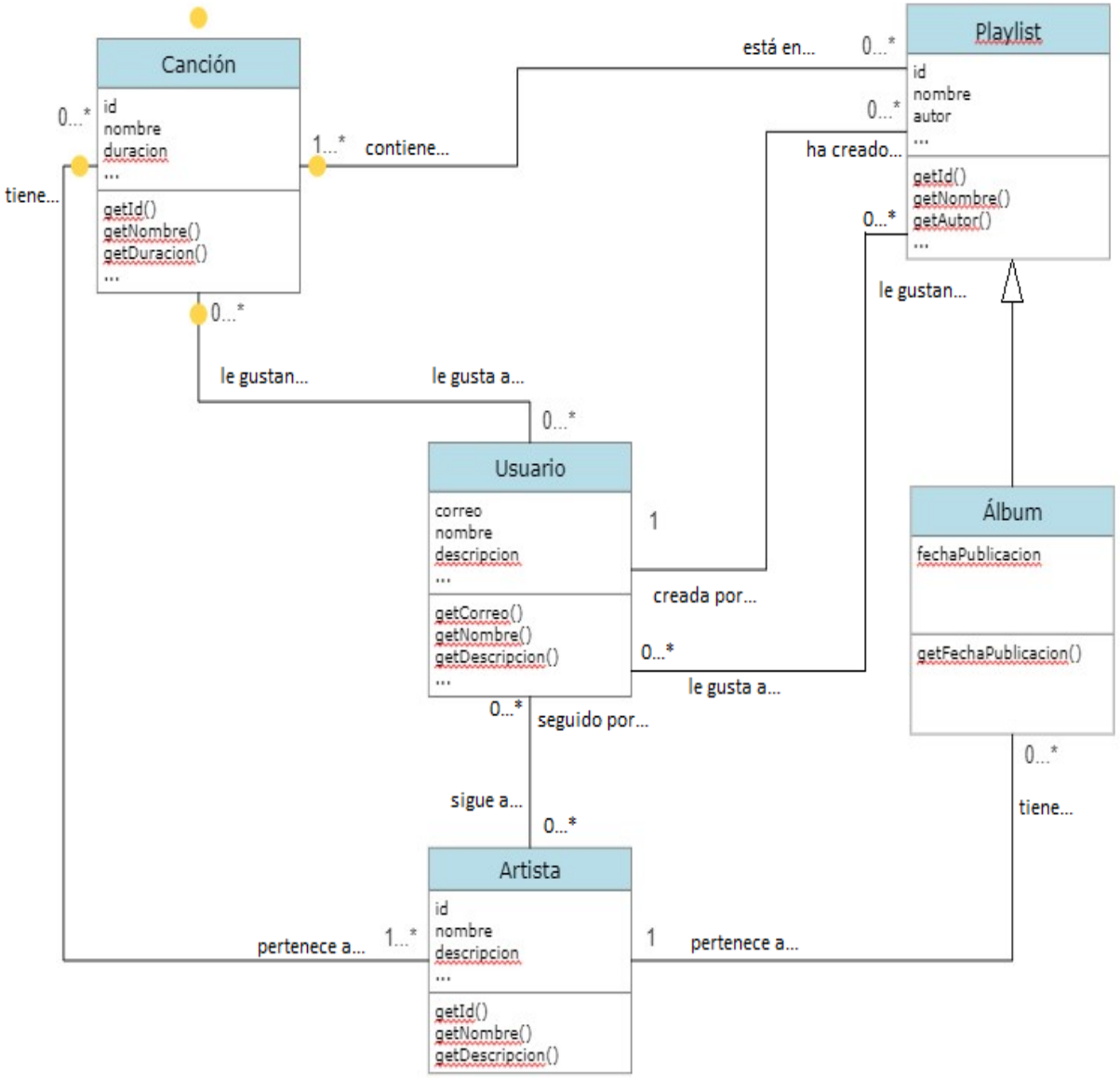

Figura 5.3: diagrama de clases del sistema

5.2.1 Definición de las clases

- Canción: esta clase contiene los datos referentes a las canciones que se encuentran disponibles para reproducir en la aplicación, entre los que destacan los más simples como el nombre y la duración de la canción, y también atributos de tipo *string* que contienen enlaces a recursos externos como la imagen de portada del sencillo y el archivo de audio correspondiente a la canción.

- Usuario: representa el usuario estándar de la aplicación. Éste podrá no sólo buscar y reproducir canciones, sino que también podrá realizar otras funcionalidades dentro de la app, como editar su propio perfil con imagen y descripción, marcar canciones, álbumes o listas de reproducción como favoritas, o crear sus propias listas de reproducción.

- Artista: representa el perfil de los artistas dentro de la aplicación. A través de esta clase se puede, por ejemplo, acceder a toda la discografía de un artista determinado o ver un top de sus canciones más populares.

- Playlist: representa listas de reproducción, es decir, conjuntos de canciones. Podrán ser privadas o ser configuradas como públicas para que todos los usuarios puedan acceder a ellas.

- Álbum: representa un trabajo discográfico de un artista. Hereda de la clase *Playlist* la gran mayoría de sus atributos dado que conceptualmente y a nivel de programación son dos clases casi idénticas.

5.2.2 Definición de las relaciones

- Canción - Artista: una canción puede pertenecer a un solo artista o a varios (si se trata de una colaboración), mientras que un artista puede haber publicado un número indefinido de canciones dentro de la app.

- Canción - Playlist: una canción puede estar en un número indefinido de playlists (o en ninguna), pero no puede existir una playlist que no contenga ninguna canción.

- Canción - Usuario: una canción puede ser marcada como favorita por un número indefinido de usuarios, mientras que un usuario puede marcar un número indefinido de canciones como favoritas.

- Usuario - Playlist: existen dos relaciones posibles entre estas dos clases.

    - Relación de favoritos: un usuario puede marcar como favoritas un número indefinido de playlists, mientras que una playlist puede ser favorita de un número indefinido de usuarios.
    - Relación de creación: un usuario puede crear indefinidas listas de reproducción, mientras que una playlist tiene un sólo creador.

- Usuario - Artista: un usuario puede seguir a un número indefinido de artistas, mientras que un artista puede tener un número indefinido de seguidores.

- Artista - Álbum: un artista puede publicar un número indefinido de álbumes, mientras que un álbum puede pertenecer a un sólo artista. Aunque cabe la posibilidad de que existan álbumes colaborativos, a nivel de código y base de datos se le asignaría la pertenencia del álbum a uno de ellos.

6. Desarrollo del frontend

En el siguiente apartado se procederá a explicar y mostrar las diferentes partes e interfaces de usuario de las cuales consta la aplicación. Inicialmente, se mencionarán las partes principales y cuál es el objetivo a realizar en cada una de ellas. A continuación, se realizará un recorrido sobre dos casos de uso reales que un usuario puede llevar a cabo en la aplicación. De esta forma se recorrerán las diferentes interfaces y se verá la relación entre ellas de una forma más clara.

6.1 Interfaces principales

- Buscador: la interfaz del buscador permite a los usuarios buscar canciones, artistas, álbumes, playlists o perfiles de otros usuarios introduciendo texto en una barra de búsqueda. Desde los resultados obtenidos en el buscador se puede acceder a la gran mayoría de subinterfaces correspondientes a los diferentes elementos de la aplicación ya mencionados.

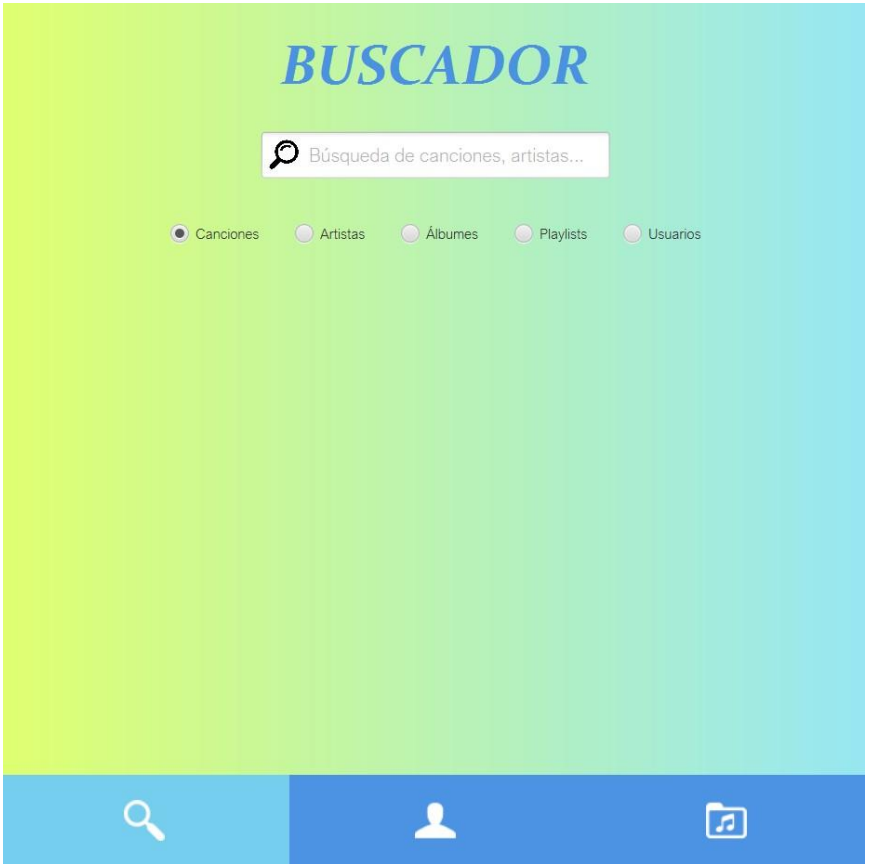

Figura 6.1: interfaz principal ‘Buscador’, sin ninguna búsqueda realizada.

- Configuración de perfil: la interfaz de configuración permite realizar modificaciones en el perfil del usuario activo como cambiar la imagen de perfil, la descripción o la contraseña. Al lado del nombre de perfil existe un botón para acceder a otras opciones como el filtrado de canciones explícitas o el cierre de la sesión actual.

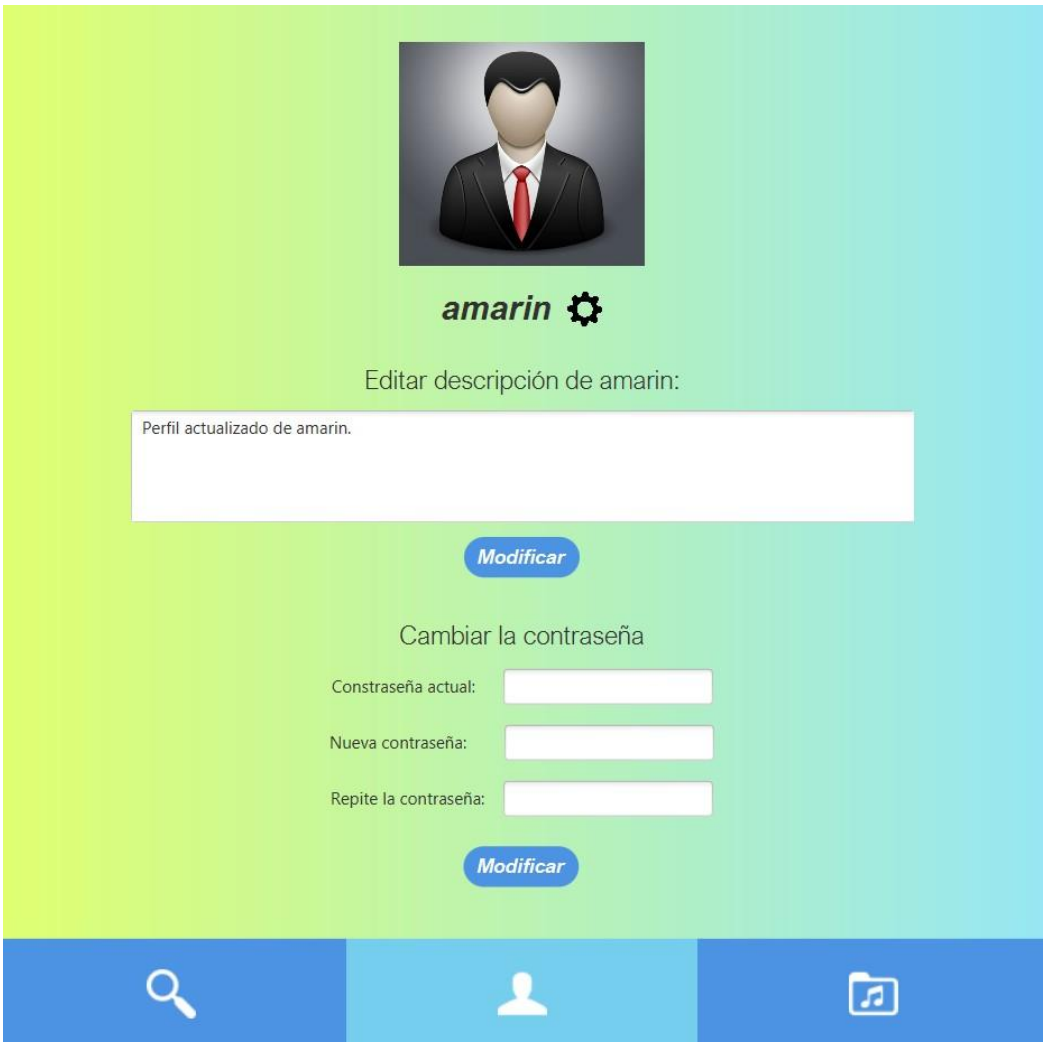

Figura 6.2: interfaz principal ‘Configuración de perfil’

- Biblioteca: la interfaz de la biblioteca contiene la playlist de las canciones favoritas del usuario, así como las diferentes playlists que ese mismo usuario ha creado (ya sean públicas o no) y las playlists que el usuario ha marcado como favoritas. Las playlists propias del usuario pueden ser editadas, por ejemplo, añadiendo nuevas canciones o eliminando las ya existentes.

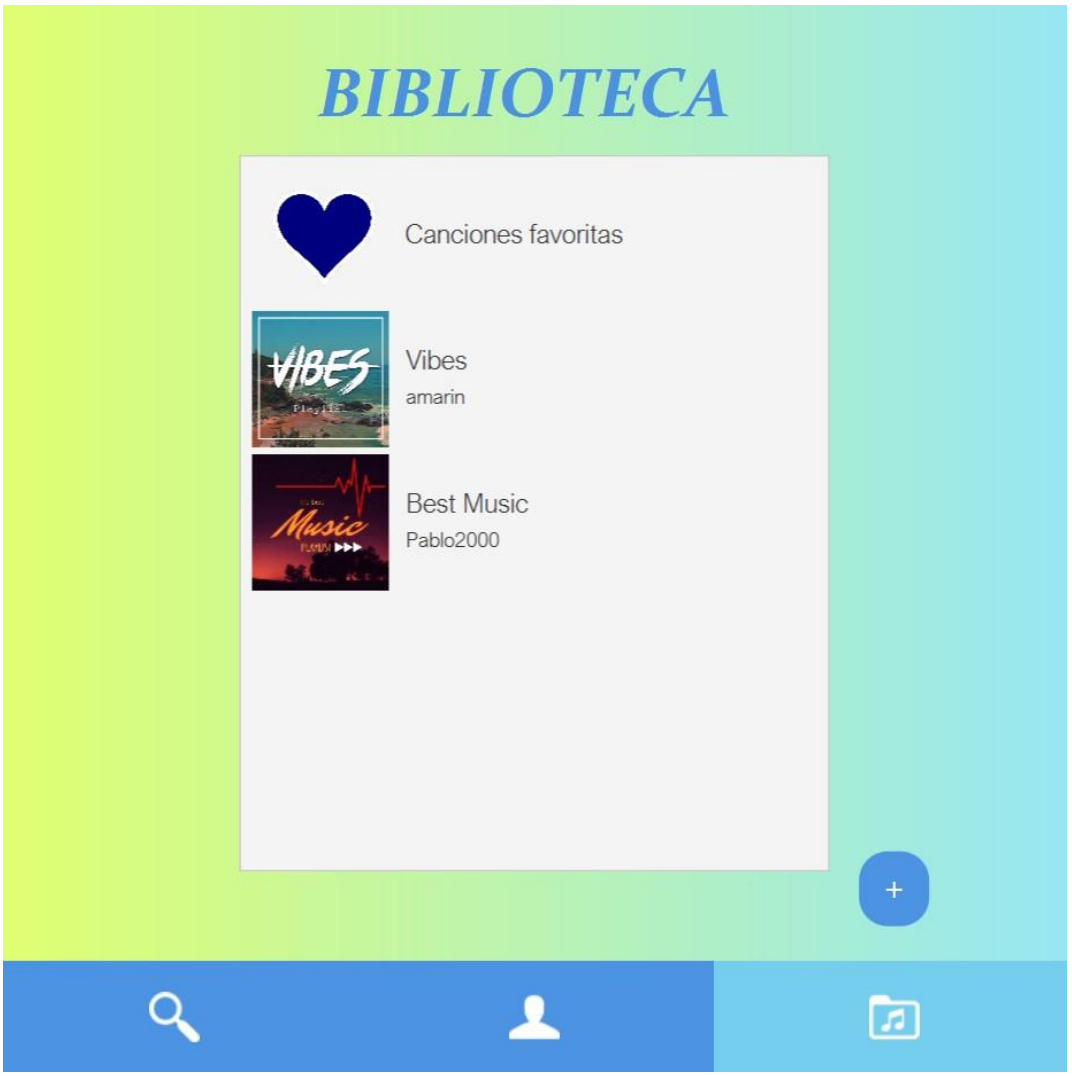

Figura 6.3: interfaz principal ‘Biblioteca’

Cabe destacar que la aplicación puede utilizarse sin necesidad de iniciar sesión. De ser este el caso, sólo se encontraría disponible la interfaz del buscador, ya que todas las acciones que se realizan en las otras interfaces requieren de que exista un usuario activo.

6.2 Mapa de navegabilidad entre interfaces

Un mapa de navegabilidad es una representación gráfica de la ordenación de la información. En este caso, al tratarse de un mapa de las interfaces, se muestra como están comunicadas las diferentes interfaces de las cuales está compuesta la aplicación, indicando entre cuáles de ellas existe una relación directa y los diferentes caminos por los que puede accederse a una interfaz determinada.

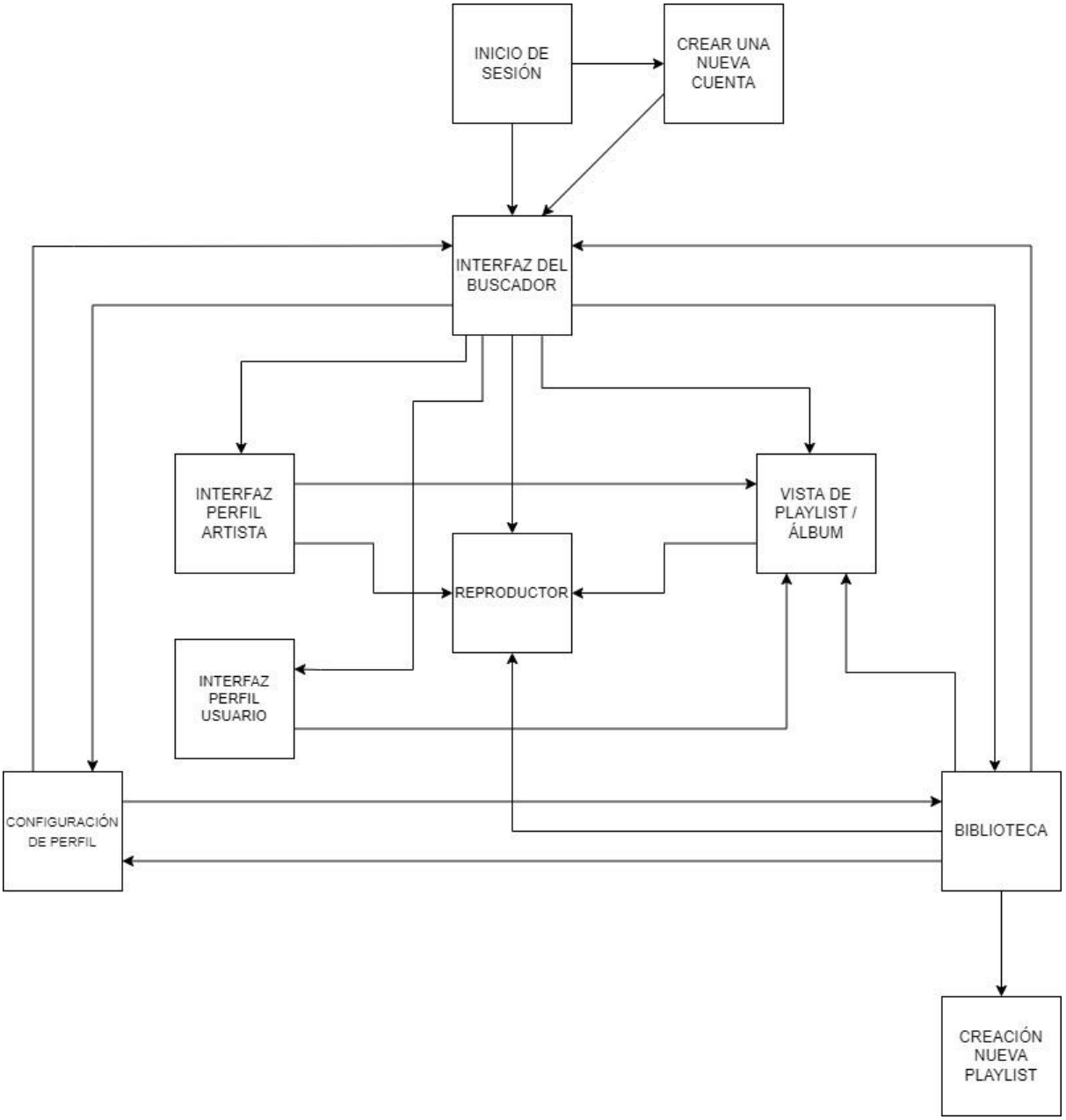

Figura 6.4: mapa de navegabilidad entre interfaces

6.3 Caso de uso: reproducción de una canción

Para mostrar con mayor claridad el funcionamiento de la aplicación, así como todas las partes e interfaces que la componen, se van a explicar detalladamente dos casos de uso que un usuario puede realizar dentro de la aplicación recorriéndolos paso a paso. El primero de estos casos será el de reproducir una canción.

Por ejemplo, vamos a suponer que el usuario está interesado en reproducir la canción ‘Photograph’, del artista Ed Sheeran. Para ello, el usuario accede a la interfaz del buscador, donde escribirá el nombre de la canción en el cuadro de texto, seleccionará el filtro de ‘Canciones’ y presionará el botón de la lupa.

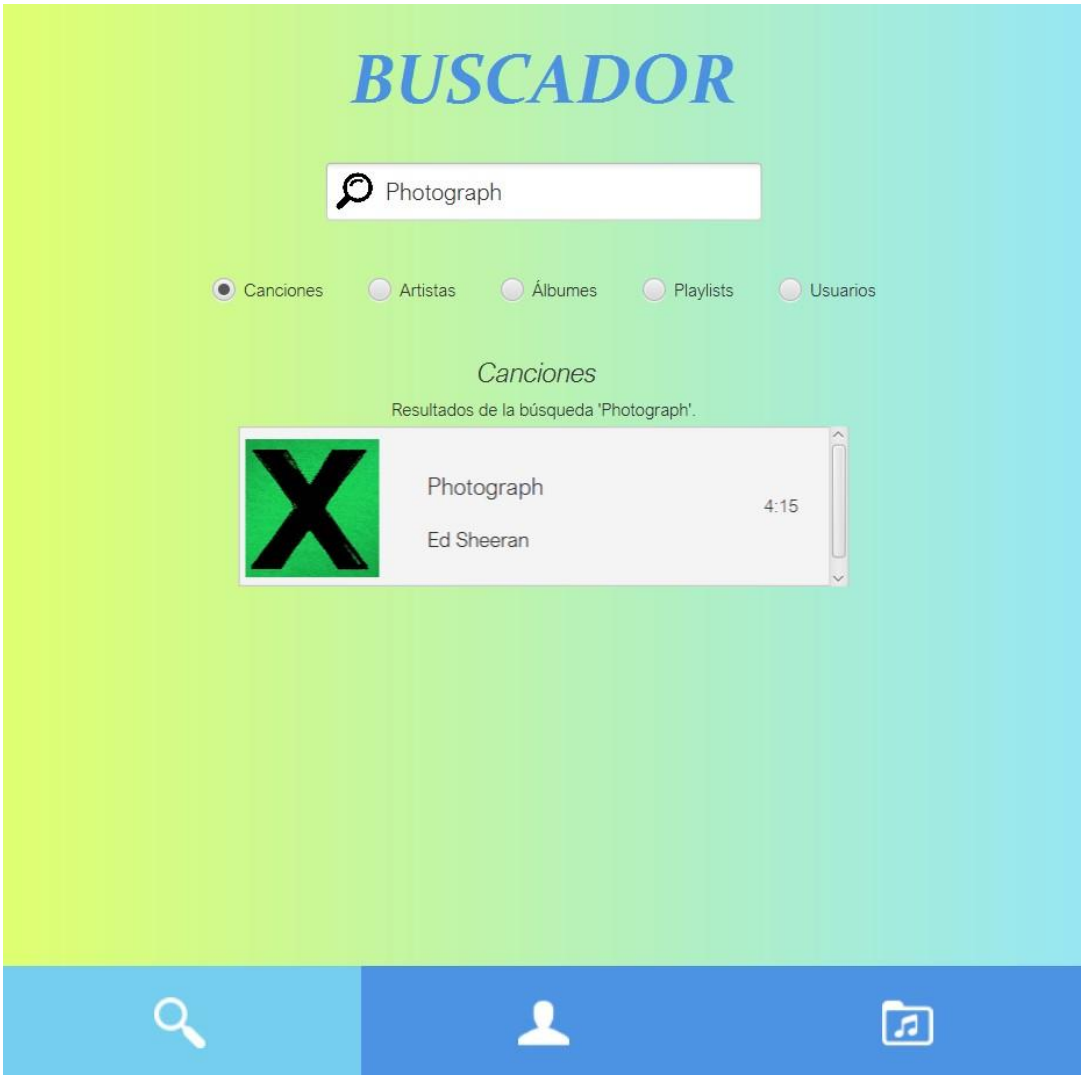

Figura 6.5: búsqueda de la canción ‘Photograph’

Una vez realizada la búsqueda, se mostrará en la misma interfaz una lista con todos los resultados que coinciden con los parámetros de búsqueda. En este caso, en la base de datos solamente existe una canción llamada ‘Photograph’, por lo tanto solo se muestra un resultado.

Al seleccionar una canción de la lista de resultados se nos mostrará directamente la interfaz del reproductor y empezará automáticamente la reproducción de la canción escogida.

Figura 6.6: interfaz del reproductor

La interfaz del reproductor contiene todas las opciones disponibles relacionadas con la reproducción de canciones, como puede verse en la figura 6.6. Dispone de los botones para la reproducción y la pausa, el botón de *stop* (que detiene la reproducción y reinicia la canción) y botones para hacer avanzar o retroceder diez segundos la canción.
También dispone de la clásica barra de reproducción, que marca el progreso actual de la canción y su duración total, y que puede manipularse directamente para hacer avanzar o retroceder la canción. Cuando esta barra de reproducción llega al final, se repetirá la canción actual (en el caso de estar reproduciendo solamente una canción) o se avanzará

automáticamente a la siguiente canción (en caso de estar reproduciendo desde una lista de reproducción/álbum). En este caso, se reiniciará la reproducción de ‘Photograph’.
En cuanto a los elementos gráficos que representan información sobre la canción, tenemos la imagen de portada y, más abajo, el nombre de la canción y del artista de la misma. Si se pulsa sobre el nombre del artista se accederá a la interfaz del perfil de dicho artista.
Cómo dos últimos elementos del reproductor, tenemos un controlador de volumen en la parte izquierda y un botón en forma de corazón al lado de la barra de reproducción. Este botón aparecerá solamente si el usuario ha iniciado sesión en un perfil ya que se corresponde con la funcionalidad de favoritos que se verá más tarde.

Aunque esta forma de búsqueda por nombre es la más rápida e intuitiva de acceder a una canción, también existen otras interfaces desde donde poder acceder a ella. Por ejemplo, podemos acceder a la misma canción desde el perfil del artista responsable, en este caso, el de Ed Sheeran. Para ello introduciremos el nombre del artista en el recuadro de texto buscador y seleccionaremos el filtro ‘Artistas’.

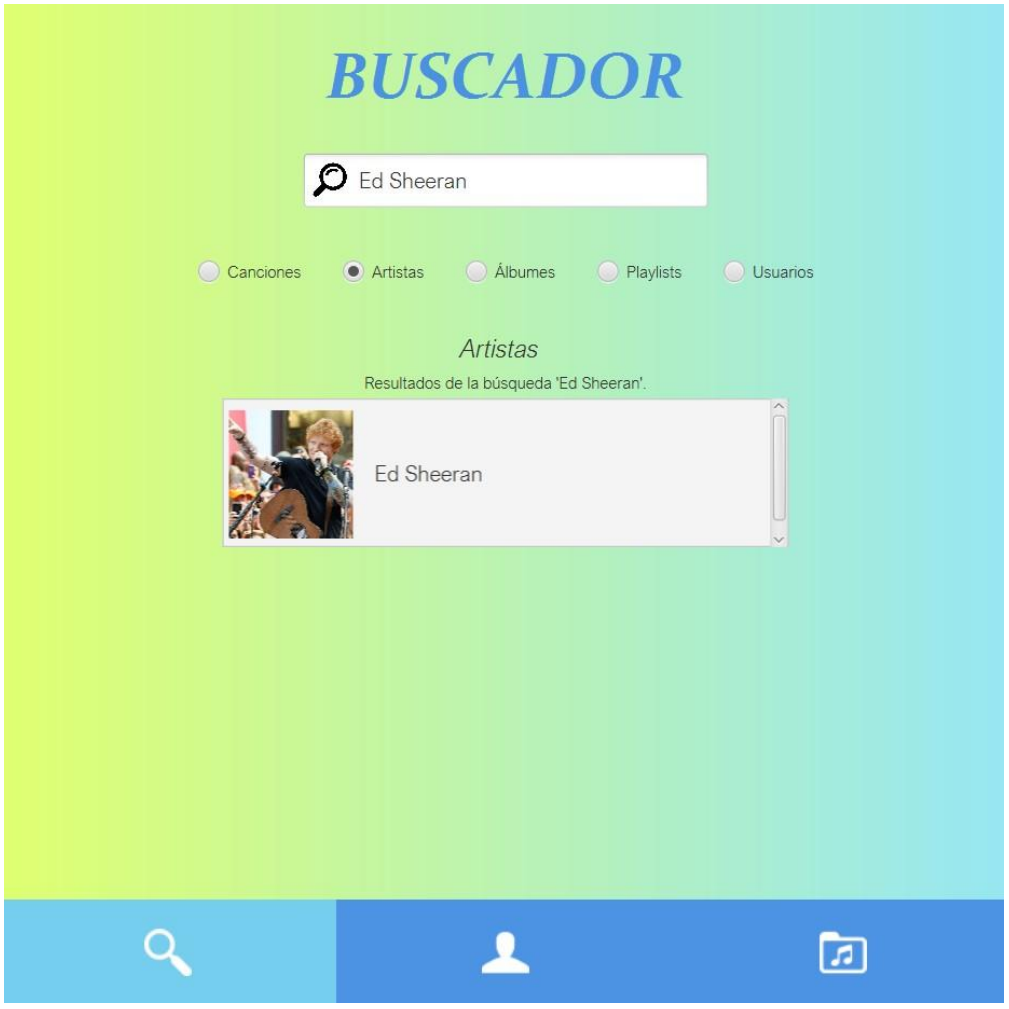

Figura 6.7: búsqueda de artista en el buscador

Al igual que en el caso anterior donde lo que se buscaba era una canción, aparecerá una lista de resultados con una serie de elementos, siendo en este caso perfiles de artistas. Una vez seleccionamos al artista en la lista se nos mostrará la interfaz del perfil de dicho artista.

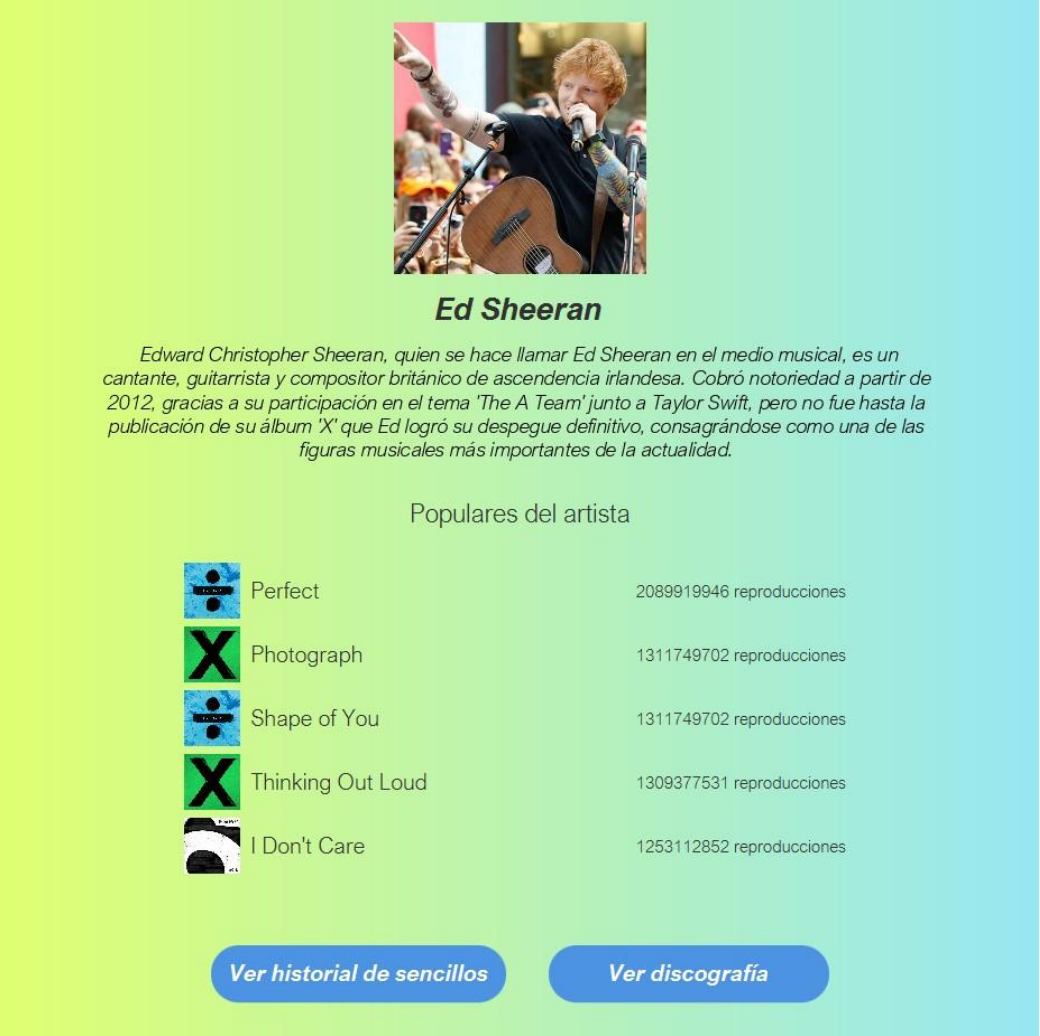

Figura 6.8: interfaz ‘Perfil de artista’

Una vez dentro de la interfaz del perfil, podemos distinguir elementos no interactuables (como la información relacionada al artista, en la parte superior) y elementos interactuables (lista de populares y botones en la parte inferior). Como podemos ver, la canción ‘Photograph’ es una de las canciones más populares de Ed Sheeran, por lo que una de las formas de acceder a la canción sería simplemente seleccionarla de la lista de canciones más populares y se nos abriría la interfaz del reproductor, donde iniciaría la reproducción automática de la canción. Cabe destacar que los números de las reproducciones han sido tomados de Spotify, por lo que son números reales.

Existe otra forma de acceder a la canción desde el perfil del artista. Suponiendo que la canción de ‘Photograph’ no fuera de las más populares, podríamos acceder a ella a través de los botones de la parte inferior de la pantalla. El botón ‘Ver historial de sencillos’ nos mostrará una lista de todos los sencillos lanzados por el artista en orden cronológico descendente, mientras que el botón ‘Ver discografía’ nos mostrará todos sus trabajos discográficos también en un orden cronológico descendente. La canción ‘Photograph’ fue lanzada inicialmente el 20 de junio de 2014 como uno de los sencillos promocionales para su segundo álbum de estudio ‘X’, y fue incluída en el mismo el día de su lanzamiento. Por lo tanto, podemos acceder a la canción desde su álbum, al cual accederemos desde la discografía.

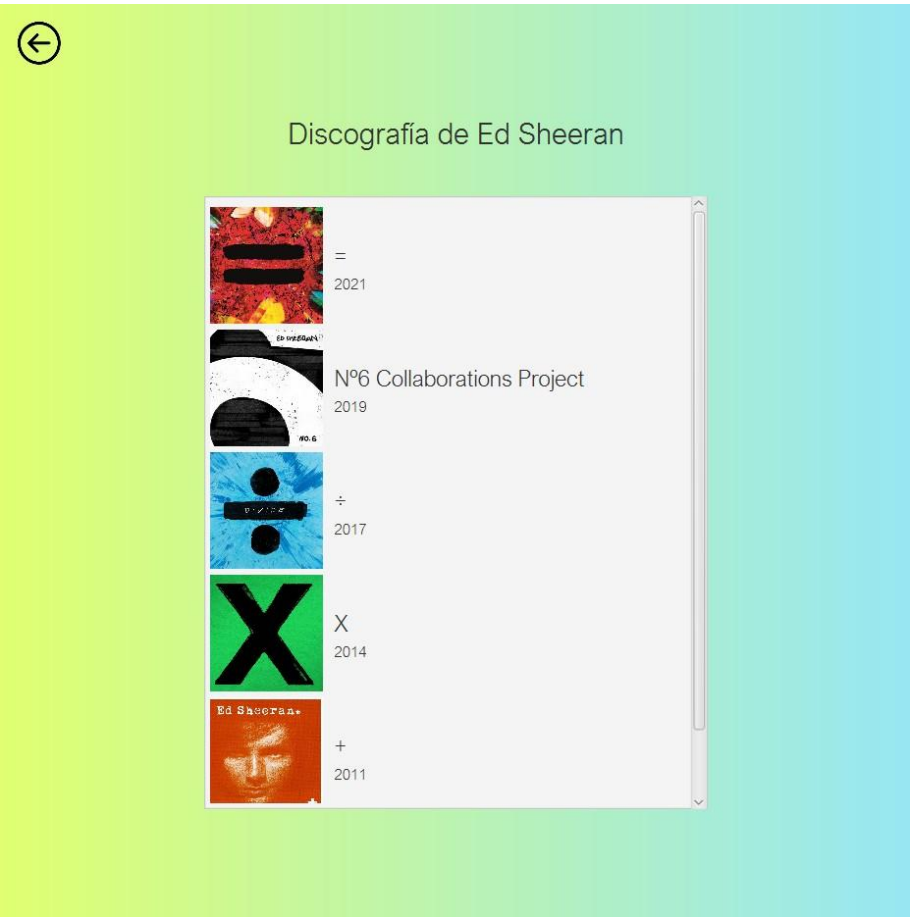

Figura 6.9: interfaz ‘Discografía de artista’

Una vez hayamos seleccionado la opción de ‘Ver discografía’ se nos mostrará la interfaz de la figura 6.9. Como se ha mencionado ya anteriormente, ‘Photograph’ pertenece al álbum ‘X’. Por tanto, seleccionaremos el cuarto elemento de la lista y accederemos a la interfaz que muestra el contenido del álbum.

Figura 6.10: interfaz ‘Contenido álbum’

En esta interfaz se nos mostrarán detalles e información del álbum seleccionado, como el nombre junto con el año de lanzamiento, el número de canciones que lo componen y el tiempo total de reproducción del álbum, junto con una lista con todas las canciones del mismo. Si deslizamos un poco hacia abajo en la lista veremos la canción ‘Photograph’, que ocupa el sexto lugar en la lista.

Por último, cabe destacar que también se puede buscar el álbum directamente desde el buscador principal seleccionando el filtro ‘Álbumes’. Al igual que las búsquedas con otros filtros saldrá una lista de elementos con los resultados obtenidos en la búsqueda. Al seleccionar un álbum accederemos directamente a la interfaz anterior.

6.4 Caso de uso: creación y acceso a una playlist

A continuación, como segundo caso de uso, vamos a explorar la funcionalidad de creación de listas de reproducción. Éstas deben crearse desde la interfaz de 'Biblioteca', que sólo se puede utilizar una vez hayamos iniciado sesión en la aplicación. Así pues, se deberá realizar un inicio de sesión si no se había hecho antes. Para acceder a la opción de iniciar sesión bastará con pulsar en el botón de 'Biblioteca' (o el de 'Configuración de perfil') de la barra inferior que aparece en la interfaz del buscador. Si no se ha iniciado sesión previamente, se redireccionará al usuario a la pantalla de inicio de sesión.

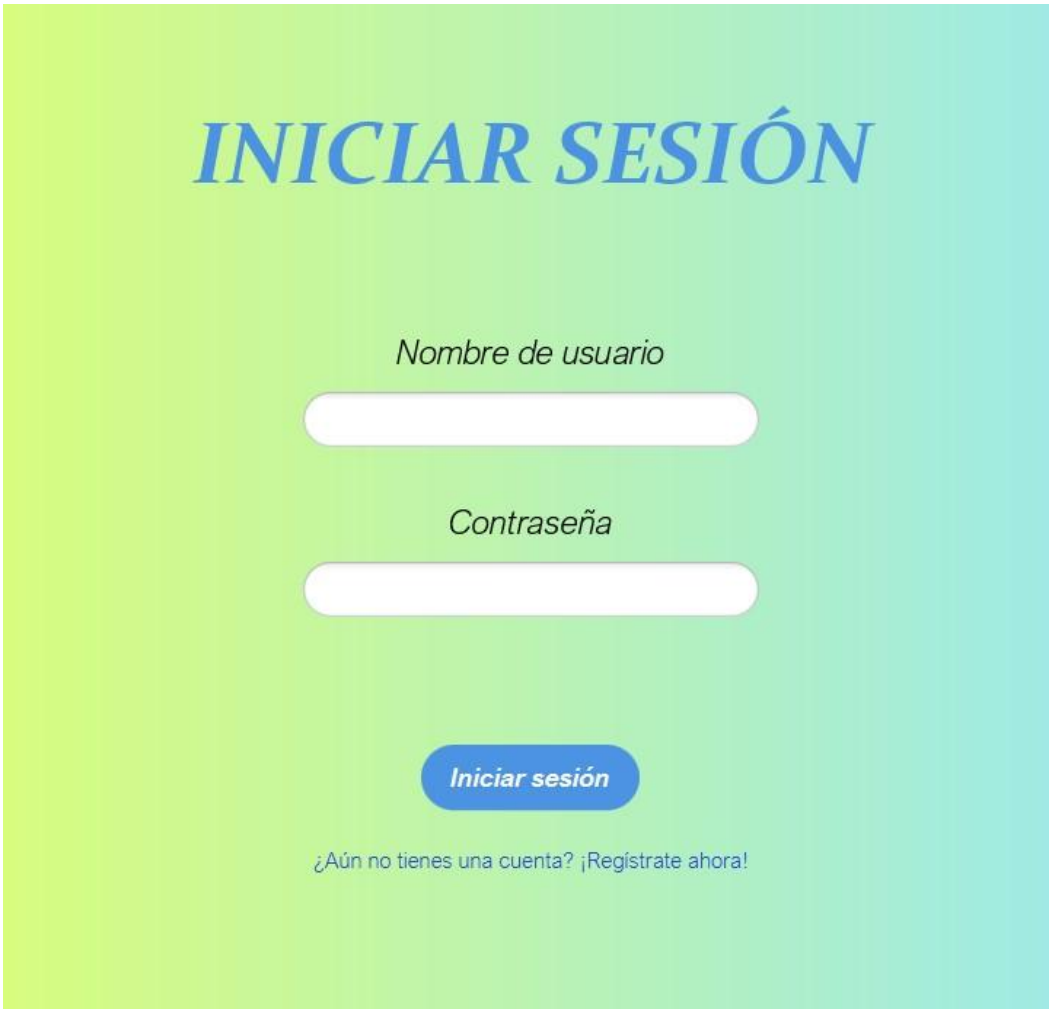

Figura 6.11: pantalla de inicio de sesión

Si el usuario no dispone de una cuenta podrá registrarse en la aplicación pulsando en el enlace situado debajo del botón de 'Iniciar sesión'.

Una vez el usuario haya iniciado sesión, para acceder a la biblioteca habrá que pulsar el botón correspondiente de la barra inferior. Al hacerlo, se mostrará la interfaz vista en la figura 5.3. En esta interfaz podemos observar principalmente una lista de elementos en pantalla, los cuales se corresponden a listas de reproducción que, o bien han sido creadas por el usuario, o bien el usuario las ha marcado como favoritas. Independientemente del número de playlists que existan en esta lista, el elemento que aparece en primera posición es el mismo para todos los usuarios. Se trata de la playlist 'Canciones favoritas', donde aparecerán todas las canciones que el usuario ha seleccionado como favoritas utilizando el botón en forma de corazón visto en la interfaz del reproductor.
En la parte inferior derecha de la pantalla, al lado de la lista de playlists, podemos ver un botón. Éste será el botón con el que accederemos a la funcionalidad de añadir una nueva playlist que veremos en este caso de uso.

La funcionalidad para añadir una nueva playlist se encuentra formada por dos subinterfaces. En la primera elegiremos el nombre de la nueva playlist, una imagen de portada, una breve descripción y seleccionaremos si queremos que la playlist sea o no pública (accesible a través del buscador).

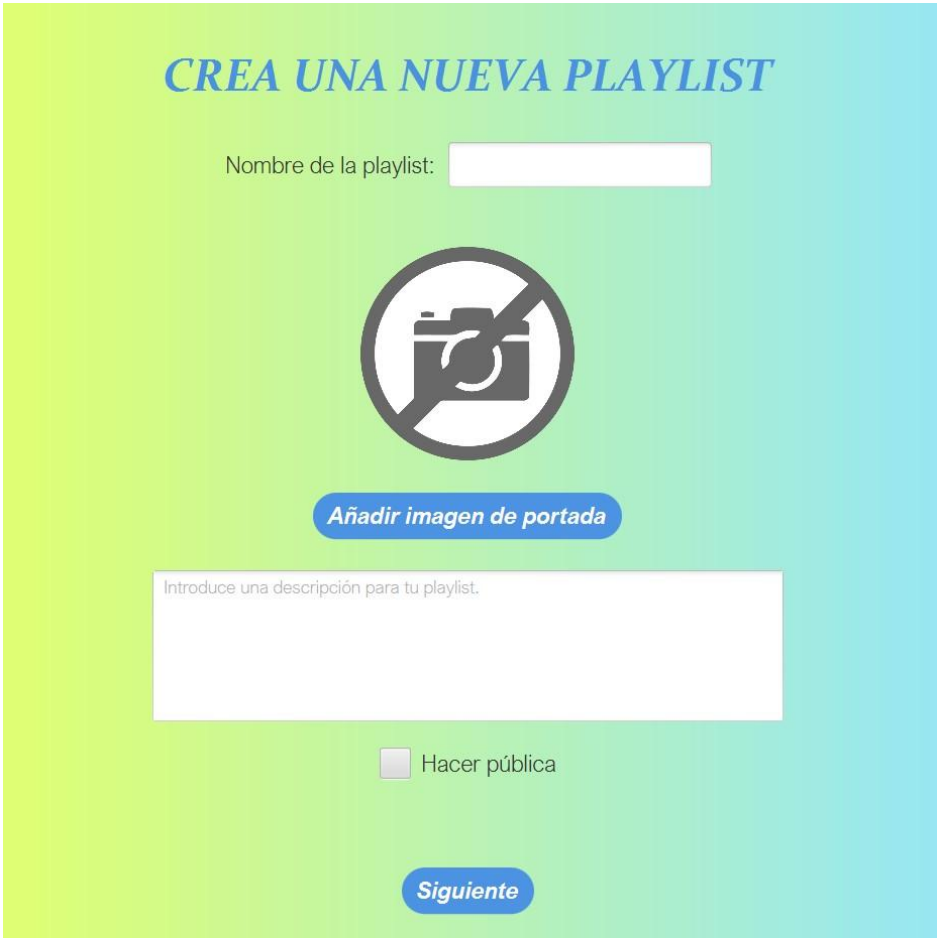

Figura 6.12: interfaz de creación de una nueva playlist

Una vez rellenados los datos y presionado el botón de ‘Siguiente’ se mostrará la segunda subinterfaz, en la que elegiremos las canciones que queremos añadir a la nueva lista de reproducción que vamos a crear.

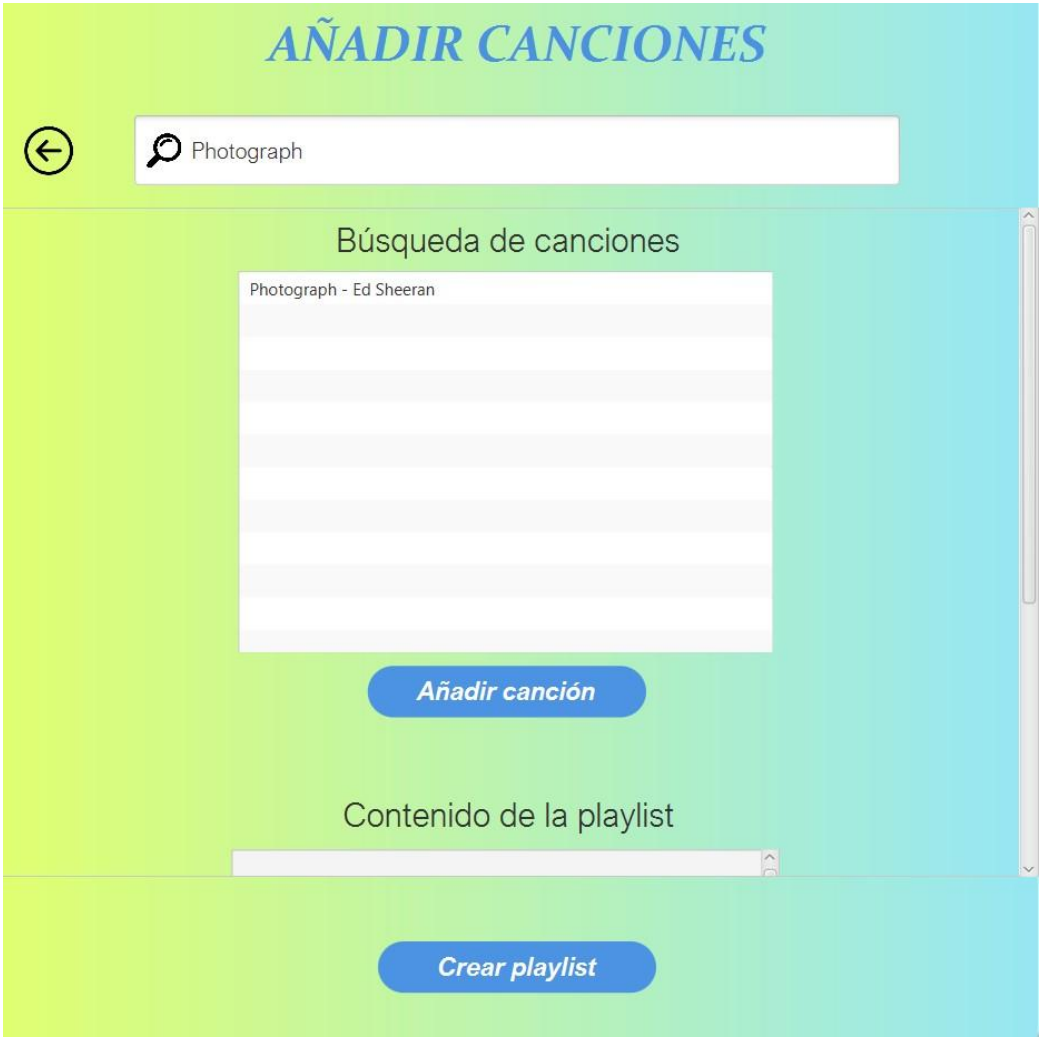

Figura 6.13: adición de canciones a una playlist

Para añadir una nueva canción empezaremos introduciendo su nombre en el recuadro de texto del buscador que se encuentra en la parte superior. Para este caso de uso vamos a utilizar como ejemplo la misma canción que en el caso anterior.
Una vez introducido el nombre y presionado el botón de buscar se cargará una lista de resultados, que contendrá el nombre de las canciones buscadas y el nombre del/los artista/s que la realizan. Para añadir la canción ‘Photograph’ a la playlist, primero la seleccionaremos de la lista de búsqueda y luego presionaremos el botón de ‘Añadir canción’. Éste botón añadirá la canción seleccionada al contenido de la playlist, que se puede visualizar si deslizamos un poco la pantalla hacia abajo. Si se desea eliminar alguna de las canciones añadidas, bastará con presionar sobre la canción y se mostrará la opción de eliminar.

Cuando la playlist esté configurada con todas las canciones deseadas, presionar el botón de ‘Crear playlist’ finalizará el proceso de creación. Podremos ver la playlist en la sección de Biblioteca o buscándola en el buscador principal en caso de haber sido configurada como pública. También podremos modificarla más tarde si accedemos a ella, pudiendo añadir y eliminar canciones o cambiar otros atributos, como hacer que sea pública o hacer que deje de serlo.

Para la segunda parte del caso (acceso a una playlist), se ha creado una playlist de ejemplo llamada ‘Best Music’. Si pulsamos sobre ella accederemos a la vista de la playlist.

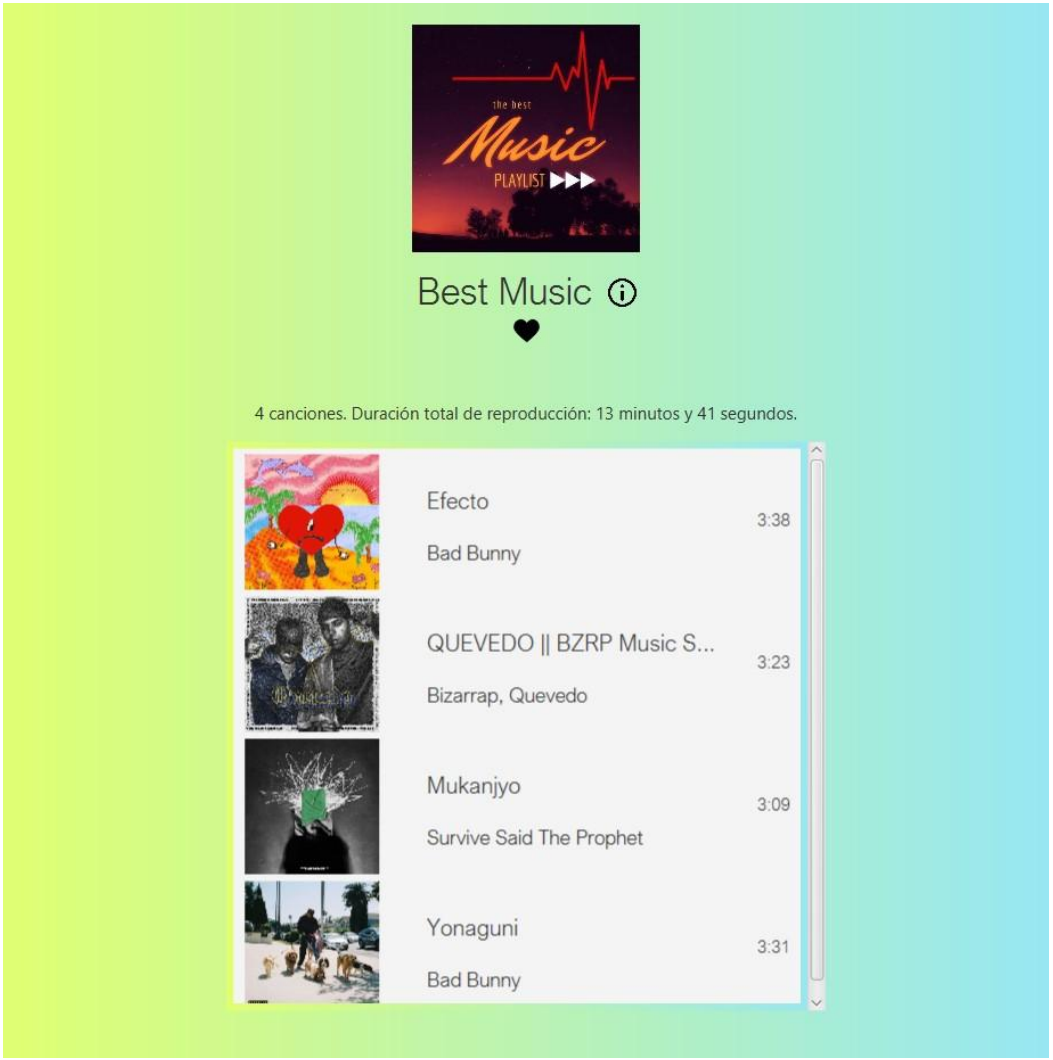

Figura 6.14: contenido de una playlist

La interfaz de contenido playlist es bastante parecida a la del contenido de un álbum (dado que un álbum no deja de ser, en esencia, una lista de reproducción dentro de la aplicación).

Así pues, solamente cabe destacar la presencia de un botón al lado del nombre de la playlist encargado de mostrar la descripción de la playlist dada por su usuario creador, y el botón de favoritos igual que el que se encontraba en el reproductor, pero esta vez marcado, indicando que el usuario activo tiene esta playlist como favorita y por eso aparece en su biblioteca.

6.4 Ejemplos de código

A continuación se mostrarán y comentarán un par de ejemplos de código Java pertenecientes a la capa de negocio y que corresponden a funcionalidades de la aplicación.

- Código para la reproducción de una canción

```
private void reproducirCancionActual() {
    slideDuracion.setValue(0);
    tiempoReproduccionActual = 0;
    tiempoReproduccion.setText("0:00");
    media = new Media(new File(cancionActual.getUrl()).toURI().toString());
    mediaPlayer = new MediaPlayer(media);
    mediaPlayer.play();
    configurarImagen(cancionActual);
    iniciarTimer();
    configurarSlider(cancionActual);
    reproductorDetenido = false;
    enPausa = false;
    if(daoCancion.esFav(usuarioActual.getNombre(),cancionActual.getId())) {
        Image img = new Image(getClass().getResource( name: "/images/fav_seleccionado.png").toString());
        botonMarcarFav.setImage(img);
    }
    System.out.println("Reproduciendo " + cancionActual.getNombre());
}
```

Figura 6.15: método java para la reproducción de una canción

El método ‘reproducirCancionActual’ pertenece al controlador de la vista del reproductor. En él se inicializa un objeto de tipo Media que contendrá el archivo correspondiente al audio de la canción a reproducir, los datos de la cual están almacenados en una variable global llamada ‘cancionActual’. También se inicializa la variable ‘media’, intancia de la clase Media en la cual se realiza la conversión del archivo de audio a un objeto manipulable por Java, y la variable ‘mediaPlayer’, instancia de la clase Media Player y la cual permite la reproducción del archivo Media.

Como puede verse, dentro del método también se encuentran llamadas a diferentes métodos que se encargan de realizar cambios en la interfaz gráfica, como por ejemplo 'configurarImagen', que cambia la imagen de la portada a la correspondiente de la canción actual, o 'configurarSlider', que ajusta la barra de reproducción a la duración de la canción a reproducir.

- Código para el cálculo de la duración de una playlist

```
private String obtenerDuracionPlaylist(ArrayList<Cancion> canciones) {
    int duracion = 0;
    for (Cancion c: canciones){
        duracion = duracion + c.getDuracion();
    }
    int minutos = duracion/60;
    int horas = minutos/60;
    int segundos = duracion%60;
    String tiempo = "";
    if(horas > 0) {
        return horas + " hora(s), " + minutos + " minutos y "+segundos+" segundos.";
    }
    else {
        return minutos + " minutos y "+segundos+" segundos.";
    }
}
```

Figura 6.16: método java para la reproducción de una canción

El fragmento de código en la figura 6.15 es el encargado de calcular la duración total de reproducción del contenido de una playlist. El método es llamado con un parámetro de tipo ArrayList<Cancion> que corresponde a las canciones que contiene la playlist.
Dentro del propio cuerpo del método, primero se inicializa una variable local 'duracion', que representará la duración total en segundos, y seguidamente se realiza un recorrido de todas las canciones del array obtenido como parámetro en la llamada del método, donde en cada iteración se obtendrá la duración de una canción con el método 'getDuracion()' de la clase 'Cancion' y se sumará a la cantidad almacenada en la variable.
Finalmente, una vez el cálculo se ha completado, se crean nuevas variables locales donde, a partir de la duración total obtenida, se obtiene el número de horas, minutos y segundos, para luego devolver una cadena de texto con todos los datos con un formato legible.

7. Desarrollo del backend

Como se ha comentado anteriormente en esta memoria, para la gestión del backend se ha utilizado el sistema de gestión de bases de datos MySQL, el servicio de almacenamiento en la nube ofrecido por Google Drive y el patrón DAO para la conexión de los elementos del frontend con la lógica de acceso a datos. A continuación se detallarán un par de ejemplos de tablas SQL utilizadas para el almacenamiento de los datos tratados en la aplicación y algunos de los diferentes métodos y solicitudes de datos a las interfaces DAO desde los métodos del frontend.

7.1 Tablas SQL

Las tablas SQL son las estructuras donde se encuentra almacenada la información que se solicitará a la capa lógica de la aplicación. Para su gestión se ha utilizado el Workbench de MySQL, una herramienta visual de diseño y administración de bases de datos. Estas tablas contienen los datos correspondientes a los elementos que componen la aplicación (canciones, artistas, álbumes, playlists y usuarios), así como las relaciones que existen entre ellos.

7.1.1 La tabla ‘Cancion’

Column	Type
ID	int(11)
nombre	varchar(100)
duracion	int(11)
fecha	date
reproducciones	int(11)
album	int(11)
esExplicita	varchar(45)
img	varchar(200)
url	varchar(200)

Figura 7.1: columnas de la tabla ‘Cancion’

La tabla ‘Cancion’ es una de las tablas que representa una de las clases de datos principales utilizadas en la aplicación, junto con las tablas ‘Artista’, ‘Álbum’, ‘Playlist’ y ‘Usuario’. En la figura 7.1 pueden verse todas las columnas de las cuales está compuesta.
Un ejemplo de consulta a esta tabla sería el siguiente:

SELECT nombre FROM tfg_bd.cancion WHERE duracion < 240;

Esta consulta obtiene los nombres de todas las canciones con una duración menor de cuatro minutos.
Lo que más cabe destacar de esta tabla son las dos últimas columnas: ‘img’ y ‘url’. Estas dos columnas tienen un ‘path’ con la dirección de los almacenes que contienen tanto las imágenes asociadas a las canciones en la aplicación como los archivos de audio correspondientes a cada canción. Estos almacenes se encuentran en la nube y pertenecen al servicio de Google Drive, y los datos son obtenidos mediante la realización de peticiones con una API proporcionada por Google.

7.1.2 La tabla ‘canciones_artistas’

La tabla ‘canciones_artistas’ es una tabla que, a diferencia de la anterior, no representa una clase o un tipo de objeto dentro de la aplicación, sino que se trata de una tabla que relaciona las canciones con los artistas que participan en ellas utilizando los identificadores (ID) de ambos como claves ajenas.

artista	cancion
5	6
5	7
5	8
5	9
5	10
5	11
5	12
5	13
5	14
5	15
5	16
5	17
5	18
5	19
5	20

Figura 7.2: resultado de una consulta a la tabla ‘canciones_artistas’

La figura 7.2 es el resultado de realizar la siguiente consulta:

SELECT * FROM tfg_bd.canciones_artistas where artista=5;

Esta consulta obtiene todos los elementos cuyo valor en la columna ‘artista’ es 5. Esto significa que se muestran todas las relaciones entre el artista con id=5 y las canciones en las que éste está involucrado.

Otras tablas similares a esta son ‘canciones_favoritas’, que relaciona el ID de un usuario con el ID de las canciones que marca como favoritas, o ‘seguidores_artistas’, que relaciona el ID de un usuario con el ID de los artistas a los que sigue.

7.2 Comunicación con la base de datos: las clases DAO

Dentro del código de la aplicación, existen un total de seis interfaces/clases DAO (pertenecientes al patrón descrito en el apartado 5.1.2 de esta memoria) encargadas de comunicarse con la base de datos:

- RegLoginDAO: esta interfaz es la encargada de realizar las consultas y solicitudes relacionadas con la gestión e identificación de los usuarios de la aplicación. Es utilizada principalmente en las pantallas de inicio de sesión y registro.

- CancionDAO: esta interfaz es la encargada de realizar las consultas y búsquedas relacionadas con las canciones como, por ejemplo, obtener las canciones de un artista o las canciones favoritas de un usuario.

- ArtistaDAO: esta interfaz es la encargada de realizar las consultas y búsquedas relacionadas con los artistas como, por ejemplo, obtener los artistas involucrados en una canción concreta.

- AlbumDAO: esta interfaz es la encargada de realizar las consultas y búsquedas relacionadas con los álbumes. Uno de los métodos más utilizados de esta interfaz es el de obtener todos los álbumes de un artista.

- PlaylistDAO: esta interfaz es la encargada de realizar las consultas y búsquedas relacionadas con las playlists. Contiene métodos de búsqueda, modificación y creación de listas de reproducción.

- UsuarioDAO: esta interfaz es la encargada de realizar las consultas y búsquedas relacionadas con los usuarios registrados en la aplicación.

Estas seis interfaces DAO son implementadas en los controladores de las vistas de la aplicación e inicializadas como objetos globales dentro de las clases. Por ejemplo, la instancia que se corresponde con la implementación de las interfaces CancionDAO y ArtistaDAO es la siguiente:

```
private CancionDAOImpl cancionDAO = new CancionDAOImpl();
private ArtistaDAOImpl artistaDAO = new ArtistaDAOImpl();
```

De este modo, cada vez que se quiera realizar consultas relacionadas a canciones o artistas se deberán referenciar estos objetos, que serán los encargados de comunicarse con la base de datos y devolver los resultados obtenidos (en caso de ser una solicitud SELECT).

```
private ArrayList<Cancion> buscarCanciones(String cancion) {
    ArrayList<Cancion> res = daoCancion.obtenerPorNombre(cancion);
    return res;
}
```

Figura 7.3: llamada a CancionDAO desde un controlador

En la figura 7.3 se muestra el ejemplo de una consulta al DAO de canciones. En este caso se quieren obtener todas las canciones cuyos nombres contengan el texto introducido en el buscador.

```
@Override
public ArrayList<Cancion> obtenerPorNombre(String nombre) {
    ArrayList<Cancion> resultado = new ArrayList<~>();
    con = conDB.getConnection();
    String consulta = "SELECT * FROM tfg_bd.cancion WHERE nombre like '"+nombre+"%'";
    try {
        ps = con.prepareStatement(consulta);
        rs = ps.executeQuery();
        if (rs != null){
            while(rs.next()){
                Cancion can = new Cancion();
                can.setId(rs.getInt( columnIndex: 1));
                can.setNombre(rs.getString( columnIndex: 2));
                can.setDuracion(rs.getInt( columnIndex: 3));
                can.setFechaPublicacion(rs.getDate( columnIndex: 4));
                can.setReproducciones(rs.getInt( columnIndex: 5));
                can.setAlbum(rs.getInt( columnIndex: 6));
                can.setEsExplicita(rs.getBoolean( columnIndex: 7));
                can.setImagen(rs.getString( columnIndex: 8));
                can.setUrl(rs.getString( columnIndex: 9));

                resultado.add(can);
            }
        }
        con.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
    return resultado;
}
```

Figura 7.4: método de CancionDAO

El método mostrado en la figura 7.4 es el método de la clase ‘CancionDAOImpl’ el cual se encarga de obtener la información solicitada en la llamada vista en la figura 6.3. Obtiene como parámetro un *string* correspondiente al texto introducido en el buscador.

Tras establecer conexión con la base de datos se realiza la consulta de tipo SELECT detallada en la variable ‘consulta’, que obtiene todas las canciones cuyos nombres contienen el texto obtenido como argumento del método. Una vez se han obtenido los resultados almacenados en la variable rs, de tipo ResultSet, se realiza un recorrido por ellos transformando los datos obtenidos en cada iteración en objetos de tipo ‘Cancion’ para que éstos puedan ser utilizados por la aplicación.

Al finalizar el recorrido, el método devuelve todos los objetos de tipo ‘Canción’ obtenidos, y éstos son finalmente recibidos en el método que inicialmente realizó la llamada a la clase DAO.

A lo largo de este proceso se ha conseguido una comunicación del frontend con el backend desacoplando la capa de negocio con la lógica de la base de datos, todo ello gracias a la implementación del patrón DAO que, además, facilitará la modificación del código en un futuro.

8. Pruebas

Para comprobar que los requisitos establecidos previamente al desarrollo del proyecto se han diseñado y ejecutado una serie de pruebas cuyos resultados se van a detallar en este apartado. Los requisitos probados han sido de eficiencia, rendimiento y usabilidad.

8.1 Pruebas de eficiencia

En primer lugar, en lo referente a las pruebas de eficiencia, se ha comprobado el siguiente requisito (establecido en el punto 4.3.2):

- El tiempo que debe tardar una búsqueda en devolver resultados no debe de exceder los tres segundos.

Los resultados de las pruebas han sido los siguientes:

Número de elementos buscados	Tiempos hasta mostrar los resultados
1	+ 0.41 segundos - 0.49 segundos
10	+ 0.53 segundos - 0.68 segundos
50	+ 0.97 segundos - 1.37 segundos

Se han realizado cinco pruebas para cada una de las cantidades de elementos y se ha anotado el mejor y el peor resultado en cada caso. Como podemos ver, en el peor de los casos se ha tardado 1.37 segundos en realizar una búsqueda que ha devuelto cincuenta resultados, por lo que podemos concluir que el requisito se satisface.

8.2 Pruebas de rendimiento

Para las pruebas de rendimiento se han probado todos los requisitos especificados en el punto 4.3.2. El primero de ellos es el siguiente:

- La aplicación deberá funcionar siempre por debajo del 15% del consumo de CPU.

Se ha anotado el consumo de CPU de la aplicación realizando diferentes funciones y los resultados han sido los siguientes:

Acción realizada	**Consumo de CPU (%)**
Iniciar sesión en una cuenta	6.8
Buscar elementos en el buscador	10.1
Reproducción de una canción	2.5

Los resultados anotados pertenecen a los mayores picos de uso de la CPU al realizar las acciones descritas en cada caso. Como en todos los casos es menor al 15%, podemos concluir que el requisito se satisface correctamente.

El segundo requisito a probar es el siguiente:

- La aplicación no deberá ocupar más de 100MB de almacenamiento en el disco duro en su versión de lanzamiento.

Después de exportar la aplicación a un archivo ejecutable, el tamaño del archivo es de 45.4 MB, por lo que este requisito también puede darse como satisfecho.

Por último, el tercer requisito de rendimiento a probar es el siguiente:

- La aplicación no deberá consumir más de 300MB de memoria RAM.

Se han realizado las pruebas que para el primer requisito, y los resultados han sido los siguientes:

Acción realizada	**Consumo de RAM (MB)**
Iniciar sesión en una cuenta	124.2
Buscar elementos en el buscador	210.5
Reproducción de una canción	232.7

Una vez más, podemos ver que los resultados obtenidos en las pruebas no superan el máximo especificado por el requisito (300MB), por lo que este tercer requisito también se satisface.

8.3 Pruebas de usabilidad

Los requisitos de usabilidad acordados durante la especificación de requisitos del proyecto son los siguientes:

- El tiempo máximo en el que un usuario promedio debe tardar en aprender a utilizar la aplicación en su totalidad es de una hora.
- La aplicación debe proporcionar mensajes informativos para informar o prevenir posibles errores que el usuario pueda realizar.
- El sistema debe de poseer un diseño intuitivo que facilite la navegabilidad y minimice los errores del usuario.

Después de la descripción de las interfaces principales que componen la aplicación, el mapa de navegabilidad entre las mismas y la realización de los dos casos de uso, todo ello explicado y detallado en el punto 6 de esta memoria, se ha llegado a la conclusión de que los requisitos especificados se satisfacen correctamente ya que, como se ha visto, las interfaces son bastante simples e intuitivas, sin grandes cantidades de opciones simultáneas que pueden abrumar al usuario, y se puede acceder a las diferentes interfaces desde diferentes caminos, todos ellos fácilmente identificables y sin posibilidad de pérdida. La aplicación muestra alertas informativas cuando una acción no puede realizarse o, en el caso contrario, cuando se necesita realizar una acción obligatoria o corregir un error para continuar (por ejemplo, cuando la contraseña durante el inicio de sesión es incorrecta o cuando un correo electrónico utilizado para crear una nueva cuenta no se encuentra disponible), por lo que el segundo de los requisitos también se satisface.

9. Conclusiones finales

Tras haber finalizado el desarrollo del proyecto, se puede concluir que se han cumplido los objetivos propuestos al inicio del mismo, ya que el resultado final es una aplicación de música en streaming que realiza con éxito todas las funcionalidades principales que se plantearon en la fase de diseño de la misma.
En cuanto a las tres partes principales se puede destacar lo siguiente del resultado final:

- El buscador es totalmente funcional y sencillo tanto para la búsqueda como para el filtrado de resultados. Se podrían añadir más filtros específicos para cada tipo de búsqueda, como por ejemplo el año de publicación de una canción o un álbum, pero finalmente se ha decidido basar las búsquedas solamente en el nombre, ya que junto a la imagen de portada y al/los artista/s se puede identificar fácilmente cuál de los elementos de la búsqueda es el deseado.

- La configuración del perfil, si bien no contiene una enorme variedad de opciones, es perfectamente funcional y aplica correctamente todos los cambios que se realizan a los perfiles, tanto estéticos (imagen y descripción) como los cambios a funcionalidades (filtrado de canciones explícitas) y la gestión de contraseñas.

- La biblioteca recopila correctamente todas las listas de reproducción relacionadas a un usuario, tanto las propias como las marcadas como favoritas. La creación de las mismas y su posterior modificación se realizan de una forma bastante fácil y rápida.

No obstante, tras finalizar el desarrollo de la aplicación se han anotado nuevas ideas sobre nuevas funcionalidades y cambios que podrían implementarse en la misma de cara a posibles futuras versiones. Algunas de ellas son las siguientes:

- Expandir el buscador, por ejemplo, con una pestaña de ‘Tendencias’ donde se muestren cuales son las canciones más populares del momento.
- Una funcionalidad para descargar canciones en el almacenamiento local, ya que actualmente se necesita estar conectado de forma permanente para reproducir cualquier canción.
- Un sistema de recomendación de música dependiendo de la música escuchada recientemente o de los gustos musicales del usuario, que podrían definirse en la configuración del perfil.
- Nueva característica en el perfil de los usuarios: un top de sus canciones más escuchadas, por ejemplo, durante el último mes, o una lista que indique sus artistas o géneros más escuchados.
- Seguimiento de usuarios. Actualmente solo pueden seguirse artistas, pero el seguimiento de usuarios al estilo red social también estaba incluido en las ideas iniciales.

- Rediseño estético, ya que actualmente si bien la aplicación tiene un diseño intuitivo y fácil de dominar, no resulta demasiado llamativa en cuanto a lo visual.

Por último, realizando una valoración a nivel personal, considero que en este trabajo he volcado todos los conocimientos que he adquirido a lo largo de todos los años cursados en la universidad. Si bien no se ha profundizado de forma muy extensa en algunas partes concretas, considero que he demostrado mis capacidades en diferentes áreas y he aplicado los conceptos más importantes de la ingeniería del software como el análisis, diseño y modelado de sistemas, la utilización de una metodología de trabajo y la realización de tests y pruebas de software para la correcta satisfacción de requisitos. A nivel de programación, los conocimientos que poseía de Java me han permitido un desarrollo sin demasiadas complicaciones, y la experiencia en el uso del lenguaje SQL también me ha facilitado la realización de consultas y la gestión de la base de datos del proyecto.

10. Bibliografía

- [1] *"Las mejores plataformas de música en streaming hoy en día"*. Digitaltrends ES. https://es.digitaltrends.com/entretenimiento/plataformas-de-musica-en-streaming/

- [2] *"¿Cómo es el consumo de streaming en España y el resto del mundo?"*. Digitalis. https://diariodigitalis.com/comercio-electronico/2021/08/02/como-es-el-consumo-de-streaming-en-espana-y-el-resto-del-mundo/

- [3] *"Java"*. Oracle. https://www.java.com/es/

- [4] *"IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains"*. JetBrains. https://www.jetbrains.com/idea/

- [5] *"MySQL"*. MySQL. https://www.mysql.com/

- [6] *"NoSQL vs SQL: diferencias y ventajas de cada sistema"*. Unir. https://www.unir.net/ingenieria/revista/nosql-vs-sql/

- [7] *"Diferencias entre metodologías ágiles y tradicionales en proyectos"*. Ealde Business School. https://www.ealde.es/metodologias-agiles-tradicionales-proyectos/

- [8] *"¿Qué es la metodología Scrum?"*. Digite. https://www.digite.com/es/agile/que-es-scrum/

- [9] *"Diferenciar entre requisitos funcionales y no funcionales"*. KryptonSolid. https://kryptonsolid.com/diferenciar-entre-requisitos-funcionales-y-no-funcionales/

- [10] *"Qué es MVC?"*. desarrolloweb.com . https://desarrolloweb.com/articulos/que-es-mvc.html

- [11] *"Data Access Object (DAO)"*. Reactive Programming. https://reactiveprogramming.io/blog/es/patrones-arquitectonicos/dao

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Los Objetivos de Desarrollo Sostenible o Objetivos Globales son 17 objetivos globales interconectados diseñados para ser un «plan para lograr un futuro mejor y más sostenible para todos». Fueron establecidos en el año 2015 por la Asamblea General de las Naciones Unidas y se espera que logren alcanzarse para el año 2030.

En la siguiente tabla se mencionan todos los ODS y se indica el grado de relación que mantiene el proyecto desarrollado con cada uno de ellos.

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**		**X**		
ODS 4. **Educación de calidad.**			**X**	
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**		**X**		**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Reflexión sobre la relación del TFG con los ODS y con el/los ODS más relacionados.

De los objetivos de calidad mencionados anteriormente, he concluido que el proyecto puede relacionarse en cierto grado con tres de ellos. Estos ODS son:

- **Salud y bienestar:** es conocido que escuchar música puede ser beneficioso para la salud de las personas. Ayuda a mejorar el bienestar emocional, mejora el estado de ánimo, disminuye la ansiedad y ayuda al control de las emociones. Es por eso que,

aunque no se trate de una aplicación con propósitos relacionados a la salud, al tratarse de un servicio musical se pueden obtener todos los beneficios mencionados anteriormente.

- **Educación de calidad:** aunque en el actual estado de la aplicación solamente puede reproducirse música, la aplicación podría expandirse para, por ejemplo, permitir la reproducción de podcasts que tuvieran fines educativos o divulgativos sobre diferentes temas y que podrían estar destinados a todo tipo de público. Es por esto que creo que la aplicación podría ver aumentada su relación con este ODS si se incluyera esta posibilidad.

- **Trabajo decente y crecimiento económico:** si la aplicación siguiera su desarrollo con un equipo de trabajo mayor, inversiones económicas y un mayor tiempo de desarrollo, podría finalmente crearse una nueva alternativa competitiva dentro del mercado de los servicios de *streaming.* Esto supondría la creación de una empresa detrás de la app con todo lo que esto conlleva, como la creación de puestos de trabajo.