



AMUSIC

Desarrollo de aplicaciones multiplataforma

Proyecto Fin de Grado

CIPP Virgen de Gracia

Alberto Muñoz Bautista

Índice

1. Presentación del proyecto	3
1.1 Explicación resumida.....	3
1.2 Estudio de mercado	4
Conocer a nuestro target	4
Conocer a la competencia	5
Análisis DAFO	5
1.3 Valor del producto	7
2. Planificación de tareas y estimación de costes.....	7
2.1 Planificación y organización de tareas	7
2.2 Estimación de costes y recursos	13
2.3 Herramientas usadas	15
2.4 Gestión de riesgos	16
Carga postural	16
Condiciones ambientales.....	17
Aspectos psicosociales.....	17
3. Análisis de la solución	18
3.1 Análisis de la solución	18
Requisitos funcionales	18
Requisitos no funcionales	20
Requisitos de información.....	20
3.2 Análisis de escenarios (casos de uso)	21
4. Diseño de la solución	25
4.1 Diseño de la interfaz de usuario y prototipos	25
Prototipado de Usuarios	27
Prototipado de Administradores.....	35
4.2 Diagrama de clases	37
Diagrama de clases de AMusic.....	40
4.3 Diseño de la persistencia de la información.....	41

Diagrama de la persistencia de la información de AMusic	43
4.4 Arquitectura del sistema	44
5. Implementación de la solución	46
5.1 Análisis tecnológico.....	46
Tecnología del cliente:	47
Tecnología del servidor:.....	49
5.1 Elementos a implementar.....	52
6. Testeo y pruebas de la solución	65
6.1 Plan de pruebas.....	65
6.2 Solución a problemas encontrados.....	99
7. Lanzamiento y puesta en marcha.....	100
7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema.....	100
7.2 Manual de uso	101
8. Valoración y conclusiones.....	101
9. Bibliografía	102

1. Presentación del proyecto

1.1 Explicación resumida

Esta aplicación, llamada AMusic, será un programa para poder escuchar, organizar y compartir tus play list de música favoritas.

Será desarrollada en WPF siguiendo el patrón de diseño modelo - vista - modelo de vista (MVVM) y cuya base de datos estará alojada en Firebase.

El objetivo de esta aplicación, más allá de que el usuario pueda disponer de toda la música del momento, será proporcionar al usuario una experiencia única, a través de un correcto funcionamiento del programa y de un diseño atractivo y armonioso basado sobre todo en su sencillez.

Este será el símbolo e ícono de la aplicación:



Nada más entrar en la aplicación, nos encontraremos con un Menú de acceso, un login, en este, mostraremos el ícono de la aplicación y los distintos campos que tendremos que llenar para poder acceder. El login es obligatorio, no existe la posibilidad de acceso anónimo. Si no tenemos credenciales con los que poder acceder, habrá un botón que nos llevará a otra pestaña donde estarán todos los campos que tendremos que llenar para poder tener una cuenta. Quiero destacar que va a haber dos tipos de usuarios, los usuarios normales y los usuarios administradores.

El objetivo de los usuarios administradores será gestionar toda la aplicación, estos van a poder subir música a la base de datos y administrar los usuarios.

Los usuarios normales van a poder disfrutar del resto de funcionalidades de la aplicación, una vez superen el login, su primera visión va a ser la música más popular del momento, la que los administradores consideren que es la música

más escuchada, los usuarios van a poder crear listas de reproducción con su música favorita. En la parte superior de la pantalla, en un menú superior, van a tener un acceso a su cuenta de usuario, para cambiar los datos que habían añadido en el registro, a su lado va a haber una barra de búsqueda donde podrán buscar las canciones que quieran escuchar, las listas de reproducción y otros usuarios registrados. Cuando busquemos otro usuario tendremos acceso a las listas de reproducción que este tenga públicas, además podremos solicitar a seguir a otros usuarios.

Habrá una opción donde podrán guardar las listas de reproducción de otros usuarios en favoritos, para así estar en todo momento actualizados.

1.2 Estudio de mercado

Puesto que AMusic es un proyecto completamente nuevo, nos interesa hacer un estudio de mercado.

Un estudio de mercado es un método por el cual vamos a poder conocer cuántos individuos o empresas hacen la actividad que nosotros tenemos pensado hacer, es decir, analizar la competencia. Además vamos a poder ver si hay demanda insatisfecha en el lugar donde pensamos iniciar la actividad y vislumbrar si nuestro proyecto va a tener aceptación entre el público.

Hoy en día es esencial antes de lanzar nuevos productos o servicios, porque a menudo existe una brecha entre lo que las empresas quieren ofrecer y lo que los consumidores desean.

En nuestro estudio de mercado nos vamos a centrar en tres cosas, en conocer a nuestro público o target, en informarnos sobre la competencia y por último vamos a fijar nuestros objetivos haciendo un análisis DAFO.

Conocer a nuestro target

Podemos decir que nuestro target es nuestro público objetivo y nuestro público potencial. De esta manera, cuando hablamos de qué es el target, nos referimos a aquel grupo de personas que debido a sus cualidades y características tiene un alto potencial, o existe una alta probabilidad de que pueda llegar a ser en el futuro un consumidor de nuestro producto o servicio.

El perfil sociocultural de nuestro target no es importante, ya que el nivel educativo, las tradiciones o las costumbres no van a influir en nuestra aplicación. Lo que vamos a considerar importante será el perfil demográfico ya que la edad y el lugar serán dos factores a tener en cuenta.

Nuestra aplicación estará destinada para un público joven, con un **rango de edad entre 14 y 50 años**. Como ya hemos dicho anteriormente nuestra

aplicación se va a caracterizar por su simpleza, esto, es una característica pensada para ampliar este rango de edad ya que las personas más mayores no están tan familiarizadas con las nuevas tecnologías y simplificar todo al máximo nos puede ayudar a ampliar nuestro target. La localización también será importante porque estará pensada para un **público hispanohablante**.

Conocer a la competencia

Las empresas competidoras son aquellas que operan en el mismo mercado o sector donde piensas implementar tu idea de negocio.

Cuando hablamos de competencia, podemos establecer dos grados:

- **Competencia directa:** son aquellas empresas que operan en el mismo mercado.
- **Competencia indirecta:** son empresas que operan en tu mismo mercado, se dirigen a los mismos clientes, pero ofrecen un servicio o producto sustituto o alternativa.

Este será el punto más flojo de este proyecto, ya que como competencia tanto directa como indirecta tenemos a grandes empresas muy asentadas en este terreno y con mucho apoyo detrás.

Como competencia directa, el gran rival será **Spotify**.

Esta empresa sueca, fue fundada en 2006 y es nuestro rival más fuerte y más directo ya que ofrece exactamente nuestros mismos servicios. Pero intentaremos que el factor diferencial sea tanto la gran sencillez de nuestra aplicación, así como su usabilidad.

Como competencia indirecta tenemos muchas empresas, pero vamos a darle más importancia a **YouTube**, ya que aunque es una empresa dedicada más enfocada al vídeo, muchos artistas la usan para anunciar y sacar sus nuevas canciones. Y tiene detrás el apoyo de una gigante de la informática como es Google.



Análisis DAFO

El análisis DAFO, también conocido como FODA o DOFA, es un marco básico, analítico, que evalúa los puntos fuertes y débiles de una organización, así como sus posibles oportunidades y amenazas. Toma la información de un

análisis ambiental y lo separa en las fortalezas y debilidades internas, así como sus oportunidades y amenazas externas.

El análisis DAFO debe ser corto y simple, y debe evitar la complejidad y el exceso de análisis, ya que mucha de la información es subjetiva. Por lo tanto, se debe utilizar como una guía y no como una receta.

Deben distinguirse entre factores internos y externos, es decir, los dos primeros dependen totalmente del proyecto y los dos externos son variables.

- **Debilidades:** impiden que una organización se desempeñe en su nivel óptimo. Tienen el potencial de reducir el progreso o aportan una ventaja a la competencia. Una organización necesita minimizar las debilidades y analizar la forma en que se pueden mejorar.
- **Fortalezas:** describen en lo que una organización sobresale, permitiendo que las decisiones se tomen de manera que se obtenga una ventaja competitiva
- **Amenazas:** son todos aquellos factores que tienen el potencial de afectar negativamente a una organización.
- **Oportunidades:** se refieren a factores externos favorables que una organización puede utilizar para sacar ventaja.



1.3 Valor del producto

En general, creemos que tenemos un producto muy completo y que va a ser muy útil y muy aceptado, como ya hemos comentado antes, nuestra mayor preocupación va a ser, no precisamente la aceptación de nuestros targets, sino la aceptación de la competencia, ya que son empresas muy grandes, muy asentadas y con mucho poder y apoyo económico detrás, y va a ser muy difícil plantarles cara. Pero la solución a esto y el factor que nosotros hemos considerado que puede ser diferencial, es el poseer un producto de calidad y nuestras ganas de hacerlo bien.

2. Planificación de tareas y estimación de costes

2.1 Planificación y organización de tareas

La planificación y organización de tareas, básicamente, se trata de un proceso que consiste en identificar, organizar y establecer las actividades que se deben llevar a cabo dentro de un proyecto.



Esto, es fundamental para gestionar el tiempo de forma efectiva, ya que hay que tener en cuenta que, aunque todas las actividades están compuestas de diferentes tareas, generalmente, estas no tienen el mismo nivel de relevancia dentro de los procesos ya que unas son más urgentes que otras, algunas son más complejas, otras requieren más esfuerzo, ... Por ello, es indispensable organizar planificar y organizar nuestras tareas, estas son algunas de las ventajas de las que vamos a disfrutar:

- **Aumento de la productividad:** Dedicarle tiempo a ver y a organizar las tareas por importancia, duración y otros criterios nos va a hacer aumentar la productividad sin incrementar el tiempo trabajado.
- **Más tiempo libre:** El incremento de la productividad tendrá como resultado acabar las tareas antes, lo que conlleva mucho más tiempo libre.
- **Conocer las prioridades:** El haber estudiado y organizado nuestras tareas nos va a llevar a conocer la lista de tareas prioritarias y por tanto no nos vamos a concentrar en algo que no sea de urgencia.
- **Más motivación:** Las listas de tareas tienen el poder de motivarte, al tachar lo que está pendiente de tu lista, experimentaremos una sensación de logro que nos estimulará para completar el resto de tareas pendientes.
- **Menos estrés:** La mayor parte del estrés generado por el proyecto está relacionado con la falta de tiempo y gracias a una gran planificación y organización de tareas podremos reducir los niveles de estrés y ansiedad.

En AMusic vamos a organizar la planificación y organización de tareas ayudándonos de varias herramientas. Primero hemos hecho la planificación de tareas, organizando todas las tareas en tareas principales y subtareas. A continuación, voy a explicar tarea por tarea.

The screenshot shows a project management interface. At the top, there is a dark header with the title "1. Presentación del proyecto #1". Below the header, there are three status indicators: a green button labeled "Open", a blue button labeled "3 tasks done", and a text "AlbertoMunozBautista opened this issue 16 hours ago · 0 comments". The main content area has a dark background. On the left, there is a profile picture of AlbertoMunozBautista. Next to it, the user's name is listed along with their comment history ("commented 16 hours ago · edited") and a "Propina ..." button. In the center, there is a list of checked items: "Explicación resumida", "Estudio de mercado", and "Valor del producto". At the bottom of this list, there is a note about estimated and actual completion dates: "Fecha estimada para finalizar: 15/04/2022" and "Fecha finalizada: 12/04/2022".

Tarea donde realizaremos una explicación resumida del proyecto, un estudio de mercado y hablaremos sobre el valor de nuestro producto. La fecha estimada para finalizar esta tarea era el 15/04/2022 y la hemos finalizado el 12/04/2022.

2. Planificación de tareas y estimación de costes #2

 Open

 1 of 4 tasks

AlbertoMunozBautista opened this issue 16 hours ago · 0 comments



AlbertoMunozBautista commented 16 hours ago · edited

 Propina ...

- Planificación y organización de tareas
- Estimación de costes y recursos
- Herramientas usadas
- Gestión de riesgos

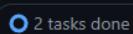
Fecha estimada para finalizar: 06/05/2022

Fecha finalizada: 06/05/2022

Tarea donde vamos realizar la planificación y organización de tareas, la estimación de costes y recursos y hablaremos sobre las herramientas usadas y la gestión de riesgos. La fecha estimada para finalizar esta tarea es el 06/05/2022.

3. Análisis de la solución #3

 Open

 2 tasks done

AlbertoMunozBautista opened this issue 16 hours ago · 0 comments



AlbertoMunozBautista commented 16 hours ago · edited

 Propina ...

- Análisis de la solución
- Análisis de escenarios

Fecha estimada para finalizar: 22/05/2022

Fecha finalizada: 22/05/2022

Tarea donde vamos a realizar un análisis de la solución y de escenarios. La fecha estimada para finalizar esta tarea es el 22/05/2022 y la hemos finalizado el 22/05/2022.

4. Diseño de la solución #4

 Open

 3 tasks

AlbertoMunozBautista opened this issue 16 hours ago · 0 comments



AlbertoMunozBautista commented 16 hours ago

 Propina ...

- Diseño de la interfaz de usuario y prototipos
- Diseño de la persistencia de la información
- Diseño de la arquitectura del sistema

Fecha estimada para finalizar: 23/05/2022

Fecha finalizada:

Tarea donde vamos a realizar tres tipos de diseños, el de la interfaz de usuario, el de la persistencia y el de la arquitectura. La fecha estimada para finalizar esta tarea es el 23/05/2022.

5. Implementación de la solución #5

 Open

 3 tasks

AlbertoMunozBautista opened this issue 16 hours ago · 0 comments



AlbertoMunozBautista commented 16 hours ago

  Propina ...

- Justificación tecnológica
- Aspectos esenciales en la implementación
- Desarrollo de la funcionalidad indicada por la tutora

Fecha estimada para finalizar: 12/06/2022

Fecha finalizada:

Tarea donde vamos a realizar la justificación tecnológica, y hablaremos sobre los aspectos esenciales de la implementación y el desarrollo de la funcionalidad indicada por la tutora. La fecha estimada para finalizar esta tarea es el 12/06/2022.

6. Testeo y pruebas de la solución #6

 Open

 2 tasks

AlbertoMunozBautista opened this issue 12 hours ago · 0 comments



AlbertoMunozBautista commented 12 hours ago

  Propina ...

- Plan de pruebas
- Solución a problemas encontrados

Fecha estimada para finalizar: 13/06/2022

Fecha finalizada:

Tarea donde vamos a realizar un plan de pruebas y vamos a poner solución a los problemas encontrados. La fecha estimada para finalizar esta tarea es el 13/06/2022.

7. Lanzamiento y puesta en marcha #7

 Open

 2 tasks

AlbertoMunozBautista opened this issue 12 hours ago · 0 comments



AlbertoMunozBautista commented 12 hours ago · edited

  Propina ...

- Aspectos relevantes del despliegue y puesta en marcha del sistema
- Manual de uso

Fecha estimada para finalizar: 14/06/2022

Fecha finalizada:

Tarea donde vamos a ver los aspectos más relevantes de despliegue y a hacer el manual de uso. La fecha estimada para finalizar esta tarea es el 14/06/2022.

8. Valoración y conclusiones #8

[Open](#) AlbertoMunozBautista opened this issue 12 hours ago · 0 comments



AlbertoMunozBautista commented 12 hours ago

Propina ...

Fecha estimada para finalizar: 15/06/2022

Fecha finalizada:

Tarea donde vamos a realizar una breve valoración y a exponer nuestra conclusión sobre el proyecto. La fecha estimada para finalizar esta tarea es el 15/06/2022.

9. Bibliografía y recursos utilizados #9

[Open](#) AlbertoMunozBautista opened this issue 12 hours ago · 0 comments



AlbertoMunozBautista commented 12 hours ago

Propina ...

Fecha estimada para finalizar: 15/06/2022

Fecha finalizada:

Tarea donde vamos a exponer toda nuestra bibliografía y los recursos utilizados. La fecha estimada para finalizar esta tarea es el 15/06/2022.

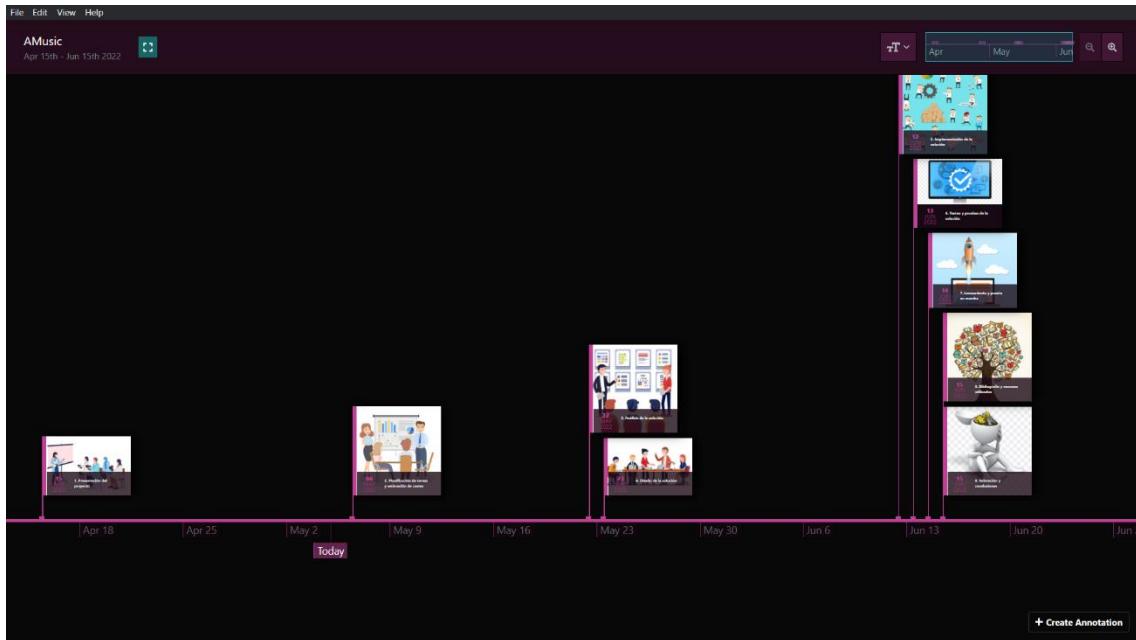
Una vez hemos definido las tareas vamos a organizarlas, para ello hemos usado dos herramientas. La primera herramienta con la que vamos a trabajar será **GitHub Boards**, aquí hemos creado un tablero kanban básico donde hemos metido y organizado todas las tareas y subtareas.

The screenshot shows a GitHub Boards kanban board titled "AMusic". It has three columns: "To do", "In progress", and "Done".

- To do:** Contains cards for tasks #9, #8, #4, #5, and #7.
- In progress:** Contains cards for tasks #2 and #1.
- Done:** Contains cards for tasks #3 and #1.

Each card includes the task title, ID, and the fact that it was opened by AlbertoMunozBautista. The "In progress" column also shows the progress of task #2 as 1 of 4 tasks completed.

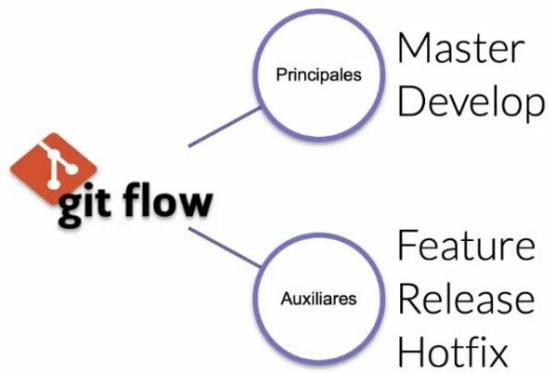
La segunda herramienta con la que vamos a trabajar será **GitKraken Timeline**, que será una herramienta mucho más gráfica, donde podremos ver todas las tareas y sus correspondientes fechas en la que nosotros hemos estimado que vamos a acabarlas.



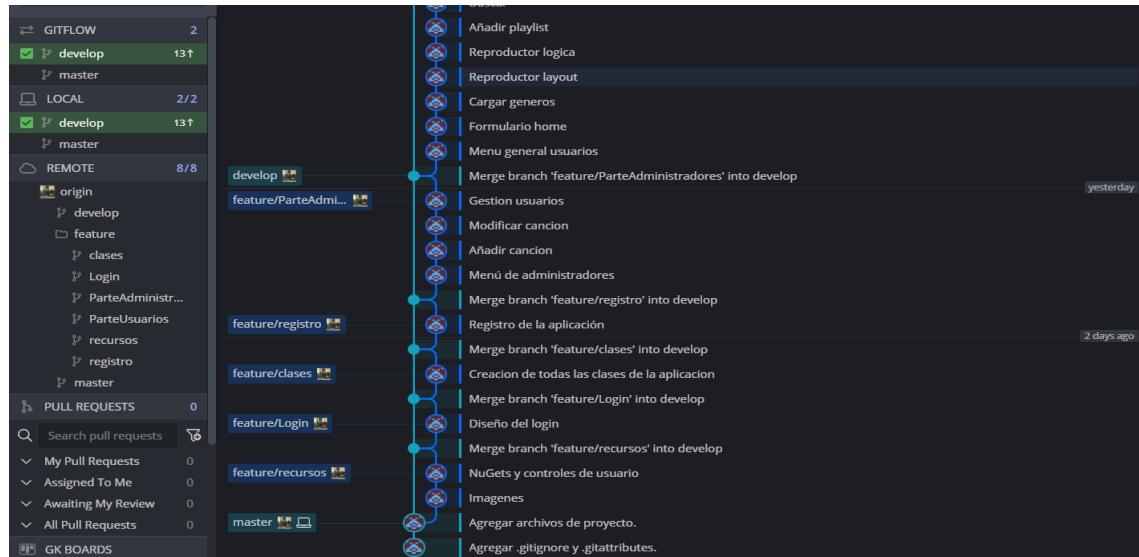
Como flujo de trabajo vamos a usar **GitFlow**. Este flujo de trabajo es una estrategia creada para mejorar la organización de ramas dentro del repositorio y, de esta forma, dar más fluidez.

Gitflow trabaja con diferentes ramas en su flujo de trabajo.

- **Las ramas principales:** son la rama Master y Develop.
- **Las ramas de soporte:** van a ser las ramas de Feature, Release y Hotfix.



Este es una captura del estado actual de las ramas de mi proyecto, vamos a ir viendo como poco a poco va cogiendo forma y evoluciona.



Para finalizar la planificación, voy a hablar sobre la documentación, ya que un buen proyecto tiene que tener detrás un buen respaldo, y ese respaldo se consigue teniendo una sólida documentación.

Para la documentación de mi proyecto, voy a usar la [Wiki de Github](#) complementándola con un archivo **readme** en el propio repositorio.

2.2 Estimación de costes y recursos

El éxito de la gestión de un proyecto se apoya en el proceso de estimar **correctamente los recursos de las actividades** que lo componen.



La estimación de recursos consiste en la planificación y garantía de la disponibilidad de los mismos para asegurar el buen desarrollo y éxito de un proyecto. La gestión de los recursos disponibles es esencial para cualquier proyecto y debe tenerse en cuenta incluso antes de que el mismo comience.

La gestión de las cantidades de recursos necesarios, así como su optimización, son dos de los factores clave para cumplir a cabalidad con la entrega de un proyecto.

En este proceso siempre hay conjeturas (de ahí que sea una estimación y no un pronóstico exacto), y si nos equivocamos con la estimación, nos podemos ver envueltos en problemas.

Nosotros vamos a realizar la estimación del coste teniendo en cuenta que vamos a cobrar 20€ por hora trabajada.

La duración de una tarea va a variar en función a los riesgos asociados (retrasos o adelantos que puedan ocurrir), y la variabilidad intrínseca a la tarea (aspectos como la motivación, el cansancio... pueden afectar al tiempo necesario para hacer determinadas tareas).

Nosotros vamos a realizar una estimación por analogía, que consiste en aproximar la duración de una tarea en base a los datos históricos disponibles referentes a la duración real de la misma tarea en proyectos anteriores, personalmente tengo poca experiencia en este tipo de proyectos, pero usará la que me ha otorgado "[SGEntregasAlbertoSheila](#)" (proyecto en Visual Studio que realicé hace unos meses) para realizar dicha tarea.

Los costes totales de desarrollo de la aplicación los vamos a calcular **estimando los costos de Firebase** ya que es una herramienta difícil de cuantificar y te cobran en función de datos almacenados en la nube, autorizaciones, ... Y teniendo en cuenta que el resto de herramientas que vamos a usar son totalmente gratuitas, el único coste que va a tener la aplicación será el coste de producción.

Se cobrarán unos extras de gastos de mantenimiento que hacen referencia a luz y gastos derivados del trabajo.

Tarea	Cantidad (h)	Precio (€)	Importe (€)
Desarrollo de la aplicación	140h	20 €	2800.00 €
Gasto de Mantenimiento	140h	2 €	280 €
Firebase	-	250 €	250 €
		Subtotal	3330.00 €
		IVA 21%	699.30 €
		TOTAL	4029.30 €

2.3 Herramientas usadas

Las herramientas de programación o herramientas de desarrollo de software, son programas informáticos que los desarrolladores de software utilizan para crear, depurar, mantener, encontrar solución de errores o apoyar programas y aplicaciones.

A continuación, en este apartado no solo vamos a exponer las herramientas de programación usadas, sino que vamos a exponer todas las herramientas que nos van a ayudar a realizar este proyecto.



Visual Studio es un entorno de desarrollo integrado para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C# y Visual Basic. La aplicación se va a desarrollar íntegramente en este entorno.



GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.



GitKraken es una herramienta multiplataforma, que nos ayuda a manejar Git de manera sencilla, lo cual incide en nuestra productividad



Figma es un editor de gráficos vectorial y una herramienta de generación de prototipos, principalmente basada en la web.



Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles.



Microsoft Word es un software de procesamiento de textos.



Draw.io es un software de dibujo de gráficos multiplataforma y de código abierto.

2.4 Gestión de riesgos

Cualquier proyecto, tenga la característica que tenga, presenta problemas. Cuando hablamos de proyectos de implantación de software es casi imposible no pensar en ellos y en cómo ejecutar una correcta gestión de riesgos.

Dichos riesgos están asociados al excesivo sedentarismo, las posturas incorrectas y la carga mental por el estrés y las funciones muy repetitivas. Básicamente, los riesgos de las tareas de oficina se pueden dividir en tres categorías: carga postural, condiciones ambientales y aspectos psicosociales.

Carga postural

Los daños de salud más habituales son las molestias y lesiones musculares y los trastornos circulatorios provocados por:

- Un espacio o entorno inadecuados.
- Un mal diseño de la silla y/o mesa del trabajo.
- La incorrecta ubicación del ordenador u otros elementos informáticos.

Condiciones ambientales

Los problemas de iluminación, como reflejos, deslumbramientos o zonas mal iluminadas, la temperatura inadecuada o la existencia de fuentes de ruido excesivo son los principales motivos de diversas molestias y problemas de salud por condiciones ambientales. Los más frecuentes son:

- Alteraciones y fatiga visual.
- Trastornos respiratorios.
- Problema de concentración, irritabilidad y nerviosismo.

Aspectos psicosociales

Una organización inadecuada de las tareas y trabajos o conflictos en las relaciones entre compañeros o con los superiores, a menudo provocan problemas de carácter psíquico o psicosomático como:

- Nerviosismo.
- Depresión.
- Ansiedad.
- Fatiga crónica.

Algunas medidas preventivas y correctivas que siempre deben incluirse en un plan de gestión de seguridad en el trabajo en las oficinas son las siguientes:

- El entorno de trabajo debe ser lo suficientemente espacioso para que no se tengan que adoptar posturas forzadas o estáticas.
- La silla debe permitir la movilidad de la espalda y de las piernas, así como adaptarse a los movimientos del trabajador. El asiento ha de ser estable y garantizar la libertad de movimientos y una postura cómoda. El asiento ha de llegar, como mínimo, hasta la parte media de la espalda.
- La mesa debe tenerla altura (entre 72 y 77 cm) y medidas adecuadas (como mínimo 160 cm de ancho por 90cm de profundidad).
- Otro aspecto importante es la pantalla del ordenador: debe estar situada frente a nosotros a una distancia mínima de 55cm. Las radiaciones deben reducirse al mínimo, la imagen ha de ser estable y sin destellos y los caracteres bien definidos.
- La temperatura en el interior de los lugares donde se realizan trabajos sedentarios, como las oficinas, debe ser de entre 17 y 26º centígrados y la humedad entre el 30 y el 70%. Si el ruido procedente del exterior es excesivo, se deberá atenuar con la instalación de los componentes estructurales necesarios en paredes, techos y ventanas, con el fin de lograr un suficiente aislamiento acústico.
- Los riesgos asociados con los aspectos psicosociales se pueden prevenir en gran medida con: una organización equilibrada de las tareas, sin que se someta a los empleados a cargas excesivas de trabajo, mejorar la comunicación y el manejo de conflictos y fomentar medidas para conciliar la vida laboral con la familiar y personal.

3. Análisis de la solución

3.1 Análisis de la solución

Requisitos funcionales

Los requisitos funcionales son una descripción explícita del comportamiento que debe tener una solución de software y qué información debe manejar.

Por lo tanto, los requisitos funcionales:

- Expresan las capacidades o cualidades que debe tener la solución para satisfacer los requerimientos de los interesados.
- Se expresan en términos de cuál debe ser el comportamiento de la solución y qué información debe manejar.
- Deben proporcionar una descripción lo suficientemente detallada para permitir el desarrollo e implementación de la solución.
- Son los que más influyen en si la solución será aceptada o no por los usuarios.

En el caso de AMusic, los requisitos funcionales van a ser los siguientes:

RF1 → El sistema debe permitir al usuario ver una pantalla de inicio con el logo de la aplicación.

RF2 → El sistema debe permitir al usuario registrarse en la base de datos.

RF3 → El sistema debe avisar si el usuario ya está registrado.

RF4 → El usuario debe poder iniciar sesión en el sistema con sus credenciales.

RF5 → El sistema no debe permitir acceder a ningún usuario sin credenciales.

RF6 → El sistema debe proporcionar solo al usuario administrador los permisos para gestionar los usuarios.

RF7 → El sistema debe proporcionar solo al usuario administrador los permisos para añadir música.

RF8 → El usuario va a poder reproducir música del sistema.

RF9 → El usuario va a poder organizar la música por play list.

RF10 → El sistema debe ofrecer al usuario la música con las canciones más de moda en ese momento.

RF11 → El sistema ofrecerá una serie de filtros para que los usuarios puedan buscar música.

RF12 → El sistema tendrá un menú superior con distintas opciones.

RF13 → El sistema tendrá una barra de búsqueda donde el usuario podrá el nombre de canciones, de play list o de otros usuarios de la aplicación.

RF14 → El sistema tendrá una opción donde el usuario podrá modificar sus datos.

RF15 → El usuario podrá seguir a otros usuarios.

RF16 → El usuario podrá añadir a favoritos las listas de reproducción de otros usuarios.

RF17 → El usuario podrá poner sus play list públicas o privadas.

RF18 → El usuario administrador podrá registrar los artistas.

RF19 → El sistema no permitirá añadir una canción sin que se haya registrado previamente un artista.

RF20 → El sistema permitirá al usuario administrador añadir, borrar y modificar usuarios.

Requisitos no funcionales

Los requisitos no funcionales representan las características generales y restricciones de la aplicación o sistema que se esté desarrollando

En el caso de AMusic, los requisitos no funcionales van a ser los siguientes:

RNF1 → Todos los datos se almacenarán en la base de datos de Firebase.

RNF2 → La autorización del usuario se realizará mediante Firebase.

RNF3 → La aplicación será desarrollada en WPF siguiendo el patrón de diseño modelo - vista - modelo de vista (MVVM)

Requisitos de información

Los requisitos de información contienen información relevante del sistema con el que vamos a trabajar.

En el caso de AMusic, los requisitos de información van a ser los siguientes:

RI1 → Usuario: Representan todos los usuarios del sistema. Vamos a trabajar con estos datos:

- idUsuario
- nombreUsuario
- contraseña
- tipo
- correo
- teléfono
- foto

RI2 → Favorito: Representa las play list favoritas de los usuarios. Vamos a trabajar con estos datos:

- idFavorito
- idUsuario
- idPlayList

RI3 → PlayList: Representan la forma en que los usuarios pueden agrupar las canciones, Vamos a trabajar con estos datos:

- idPlaylist
- idUsuario
- nombre
- visibilidad

RI4 → CancionesPlaylist: Representan las canciones que van a tener las playlist:

- idCancionPlaylist
- idCancion
- idPlaylist

RI5 → Canción: Representan las canciones del sistema. Vamos a trabajar con estos datos:

- idCancion
- nombre
- idGenero
- artista
- fechaAgregacion

RI6 → Género: Representan los géneros musicales de las canciones. Vamos a trabajar con estos datos:

- idGenero
- nombre

RI7 → Seguido: Representan los amigos que tiene cada usuario en su lista.:

- idSeguido
- idUsuario1
- idUsuario2

3.2 Análisis de escenarios (casos de uso)

En este apartado vamos a realizar un análisis de escenario, y para ello vamos a hacer y un diagrama de casos de uso.

Un diagrama de caso de uso es un tipo de diagrama UML y se usa frecuentemente para analizar varios sistemas. Permiten visualizar los diferentes tipos de roles en un sistema y cómo esos roles interactúan con el sistema.

Como ya he mencionado, los diagramas de casos de uso se utilizan para reunir los requisitos de uso de un sistema. Dependiendo de sus necesidades, puede utilizar esos datos de diferentes maneras. A continuación, se presentan algunas formas de usarlas.

- **Identificar las funciones y la forma en que los roles interactúan con ellas:** El propósito principal de los diagramas de casos de uso.
- **Para una visión de alto nivel del sistema:** Especialmente útil cuando se presenta a los administradores o a las partes interesadas. Se pueden destacar los papeles que interactúan con el sistema y la funcionalidad proporcionada por el sistema sin profundizar en el funcionamiento interno del sistema.
- **Identificar los factores internos y externos:** Esto puede parecer simple pero en grandes proyectos complejos un sistema puede ser identificado como una función externa en otro caso de uso.

Los diagramas de caso de uso consisten en 4 objetos.

- Actor
- Caso de uso
- Sistema
- Paquete

Los objetos se explican con más detalle a continuación.

Actor

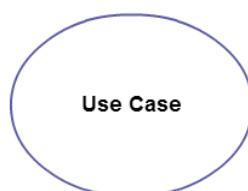
El actor en un diagrama de casos de uso, es cualquier entidad que desempeñe un papel en un sistema determinado. Puede ser una persona, una organización o un sistema externo y normalmente se dibuja como el esqueleto que se muestra a continuación. En el caso de esta aplicación el actor será el propio usuario que esté usando la aplicación, dependiendo de los permisos que tenga, podrá ser el usuario administrador o el usuario común.



Actor

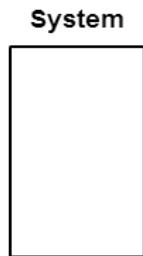
Caso de uso

Un caso de uso representa una función o una acción dentro del sistema. Está dibujado como un óvalo y nombrado con la función.



Sistema

El sistema se utiliza para definir el alcance del caso de uso y se dibuja como un rectángulo. Este es un elemento opcional pero útil cuando se visualizan sistemas



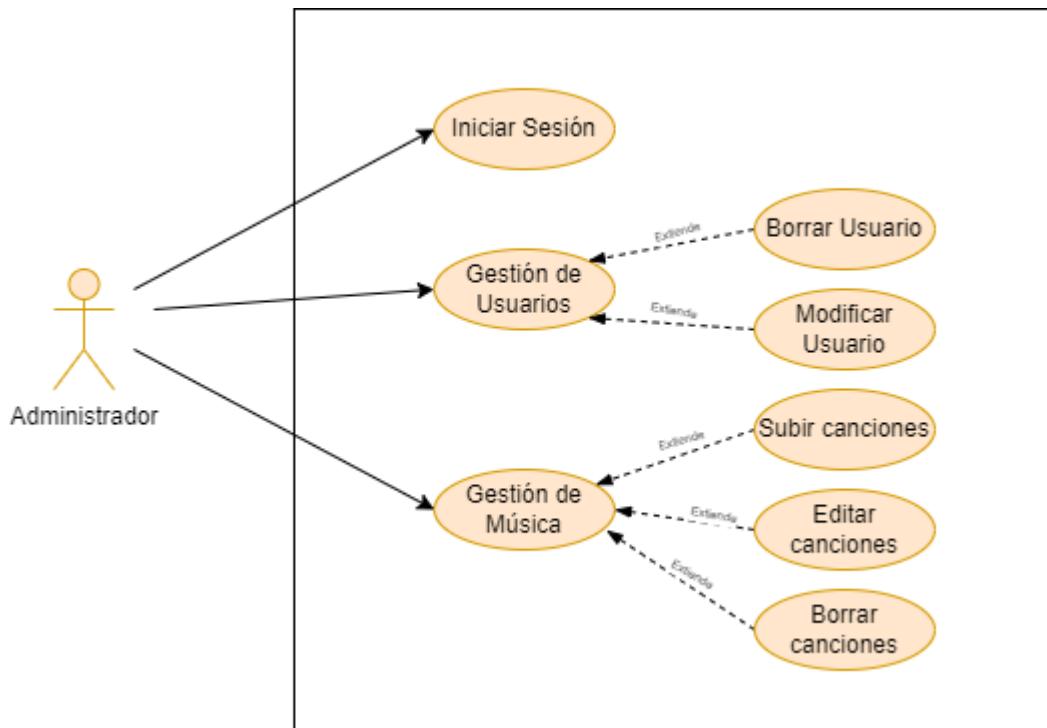
grandes. Por ejemplo, puede crear todos los casos de uso y luego utilizar el objeto del sistema para definir el alcance que abarca su proyecto. O incluso puedes usarlo para mostrar las diferentes áreas cubiertas en los diferentes lanzamientos.

Paquete

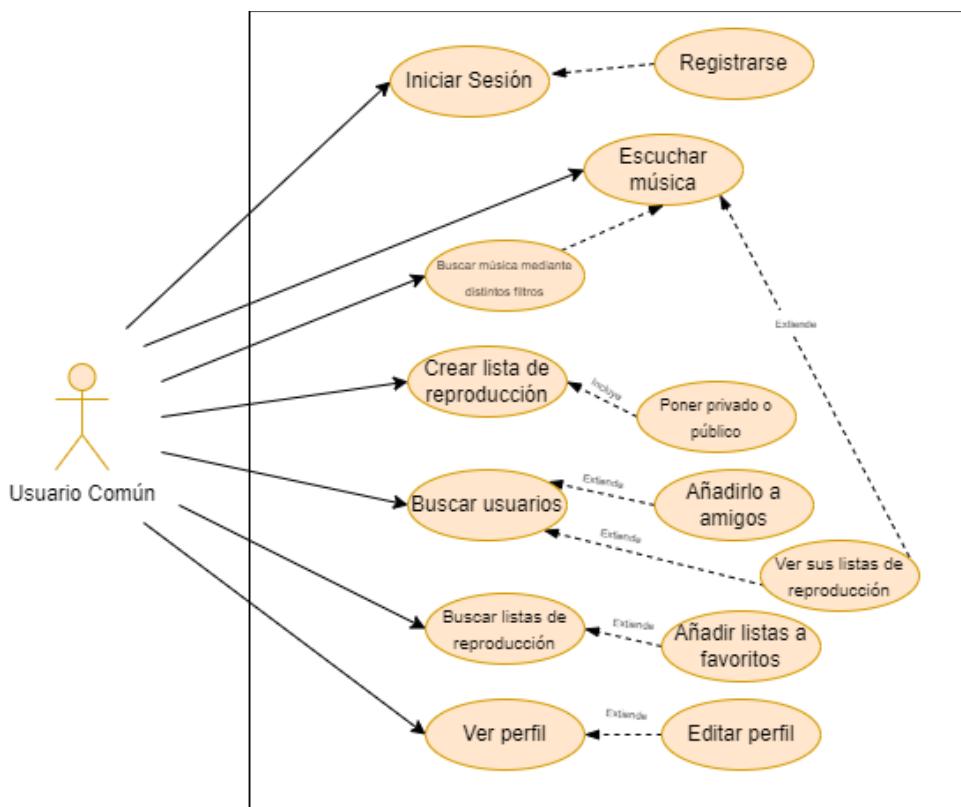
El paquete es otro elemento opcional que es extremadamente útil en diagramas complejos. De manera similar a los diagramas de clase, los paquetes se utilizan para agrupar los casos de uso. Se dibujan como la imagen que se muestra a continuación.



El diagrama de casos de uso de esta aplicación desde el punto de vista de los administradores, será el siguiente.



El diagrama de casos de uso de esta aplicación desde el punto de vista de los usuarios comunes, será el siguiente.



4. Diseño de la solución

4.1 Diseño de la interfaz de usuario y prototipos

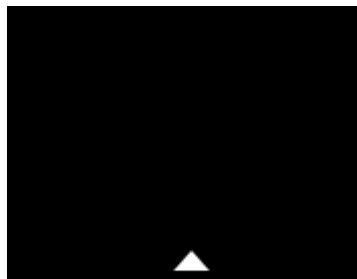
Muchos suelen confundir prototipo con diseño, e incluso no le dan la importancia necesaria, sin embargo, en nuestro caso si se la vamos a dar para ofrecer el mejor producto.

Un prototipo es un boceto, borrador o esquema desecharable donde podemos probar distintas opciones de estructura para una interfaz digital sin tener que gastar mucho tiempo ni dinero.



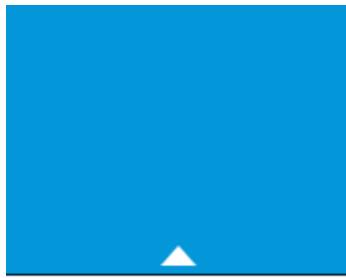
Además, un prototipo nos ayuda a realizar varias pruebas y corregir de manera rápida los posibles errores que vayamos encontrando, evitando problemas graves en un futuro, además de ir viendo si vamos o no por un buen camino.

Vamos a empezar diciendo los colores de la aplicación, nuestro color principal será el negro.



#000000

Y nuestro color secundario será un azul.



#0396D9

Ahora vamos a explicar el prototipado de AMusic.

Además de en la documentación tengo todo el diseño y prototipado explicado en mi canal de Youtube, podéis acceder copiando el siguiente enlace en el navegador:

<https://youtu.be/7NGpF4S0cfA>

Prototipado de Usuarios

Este es el diseño y prototipado de la aplicación cuando el usuario **no** es un administrador.



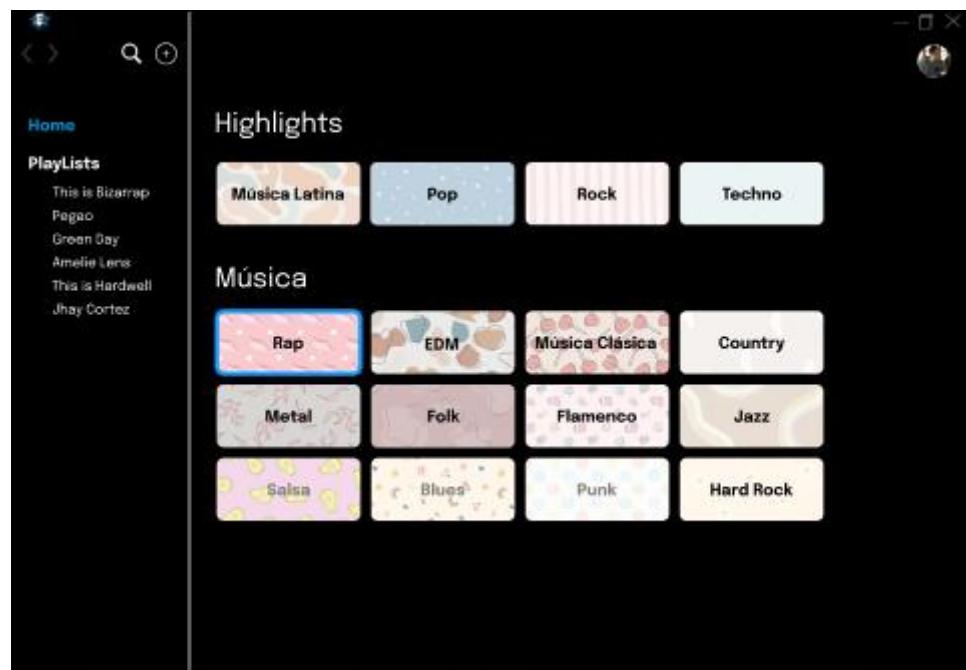
Lo primero que vamos a encontrar nada más iniciar la aplicación será un splash screen con el logo y el nombre de nuestra aplicación.



Cuando pasen 2 segundo, nos llevará directos al login, donde tener cuenta es obligatorio, por lo que si es la primera vez que entramos, tendremos que pasar obligatoriamente por el registro.

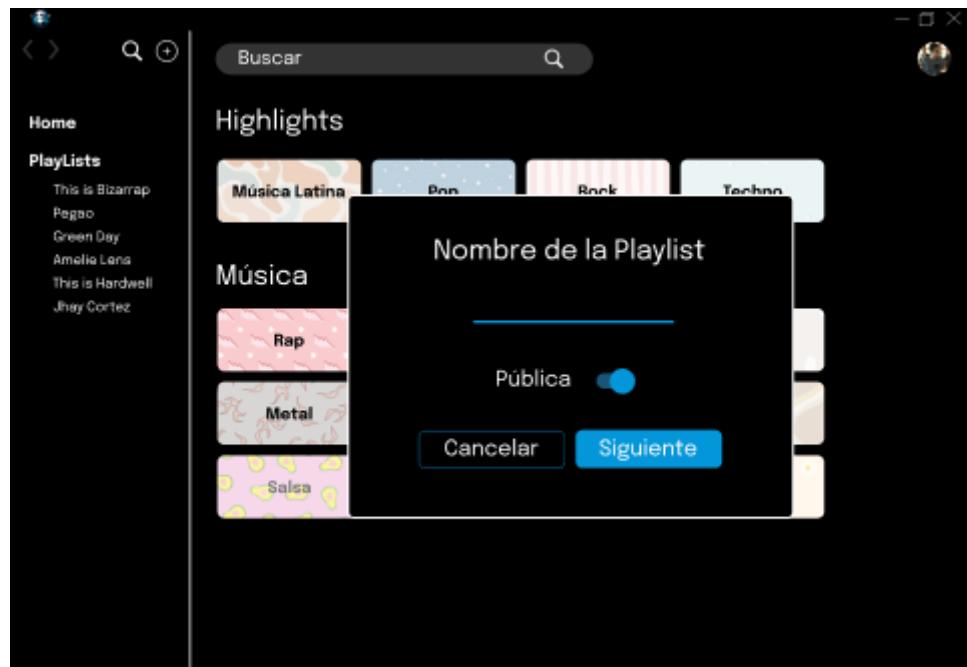


En el registro podremos crearnos una cuenta, y si todos los datos son correctos, podremos acceder al menú principal.

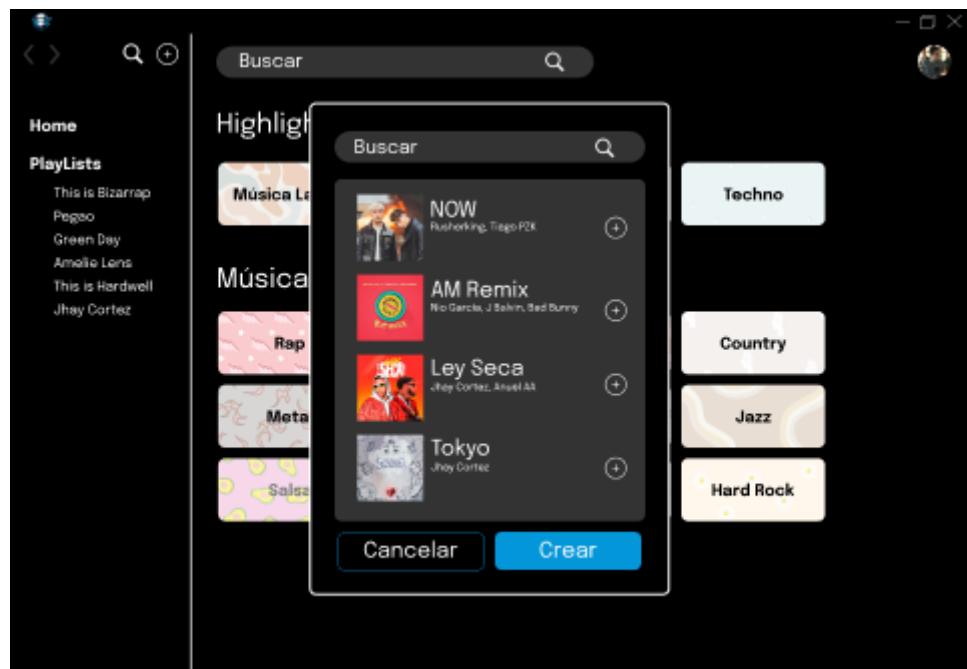


Este será nuestro menú principal, donde tendremos toda la música agrupada por géneros musicales. A la izquierda tendremos todas las playlist que nosotros hemos creado y las que tenemos guardadas como favoritos, a la izquierda, arriba, tendremos dos iconos, el de la derecha es para crear una nueva playlist y el de la izquierda, es para buscar en la aplicación,

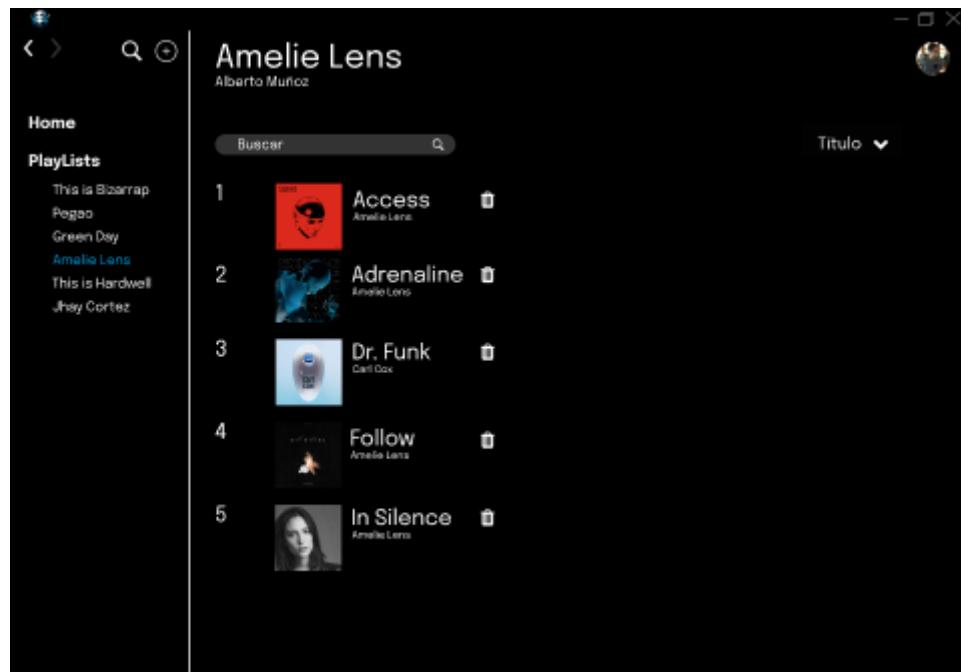
podremos buscar por usuarios, música y playlists. A la derecha arriba tendremos nuestra foto, pulsando en ella podremos acceder y cambiar nuestros datos de perfil y ver la gente a la que seguimos.



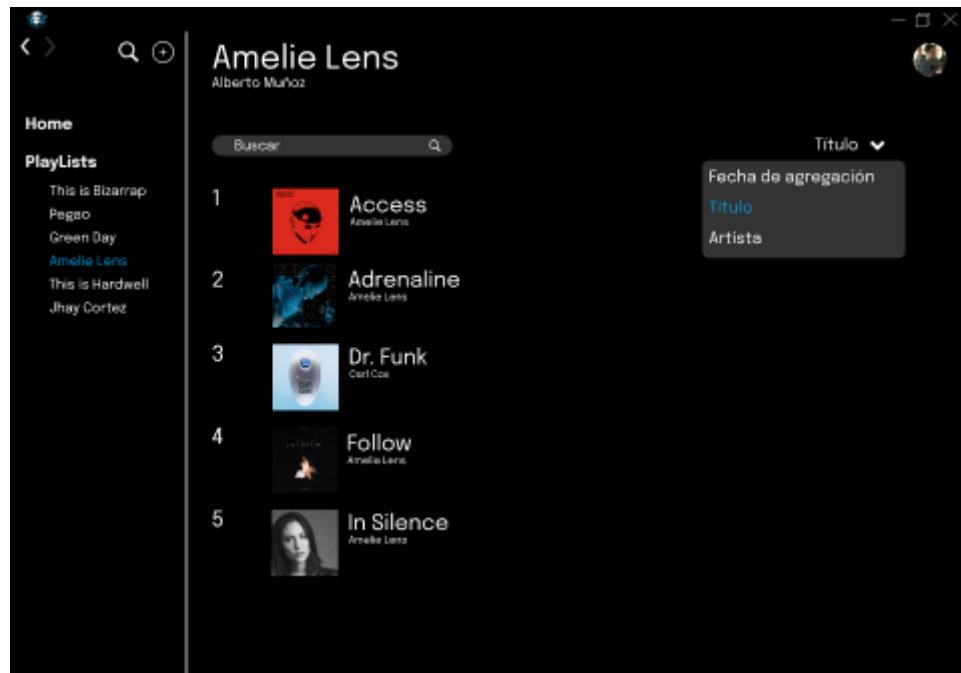
Al darle al botón de agregar una nueva playlist nos saldrá una ventana emergente, donde tendremos que elegir un nombre para esta y elegir si será pública o privada.



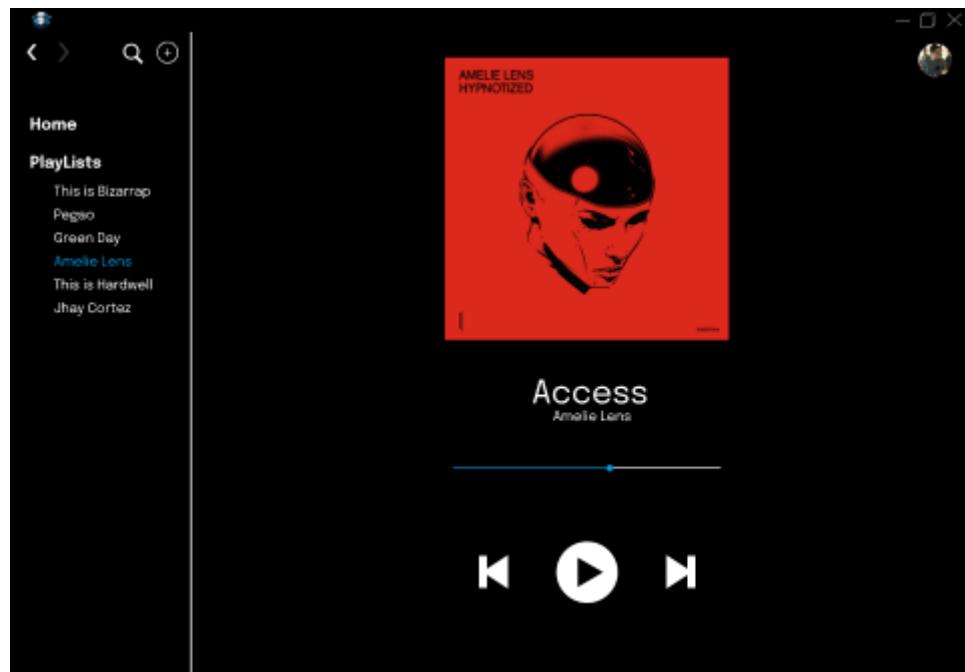
Dándole a siguiente, podremos meter todas las canciones que van a componer nuestra playlist.



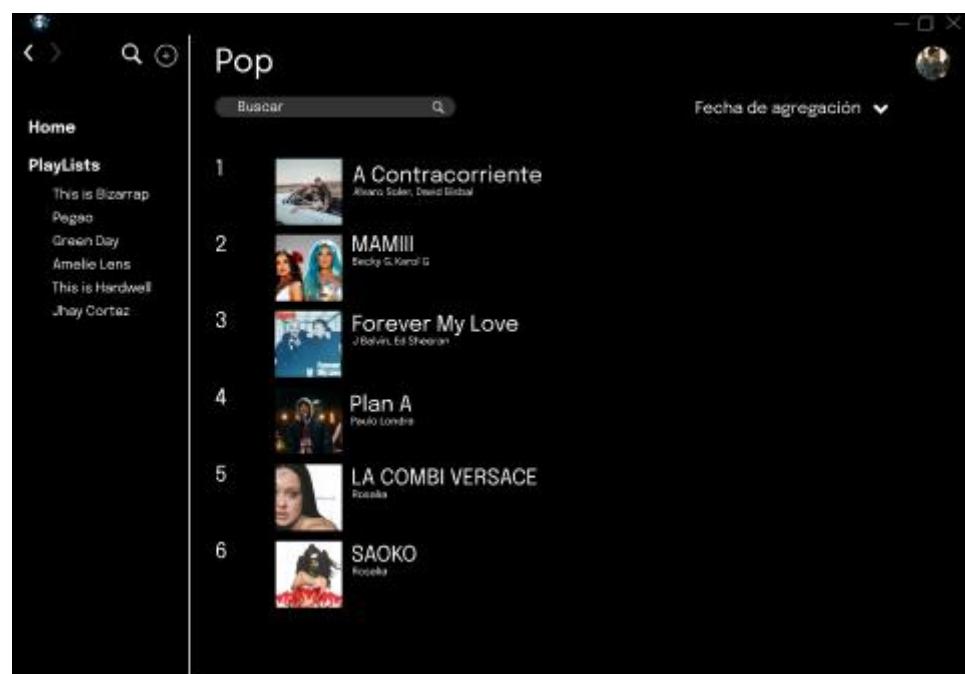
Desde el menú principal, al seleccionar una playlist tendremos acceso a las canciones que la componen.



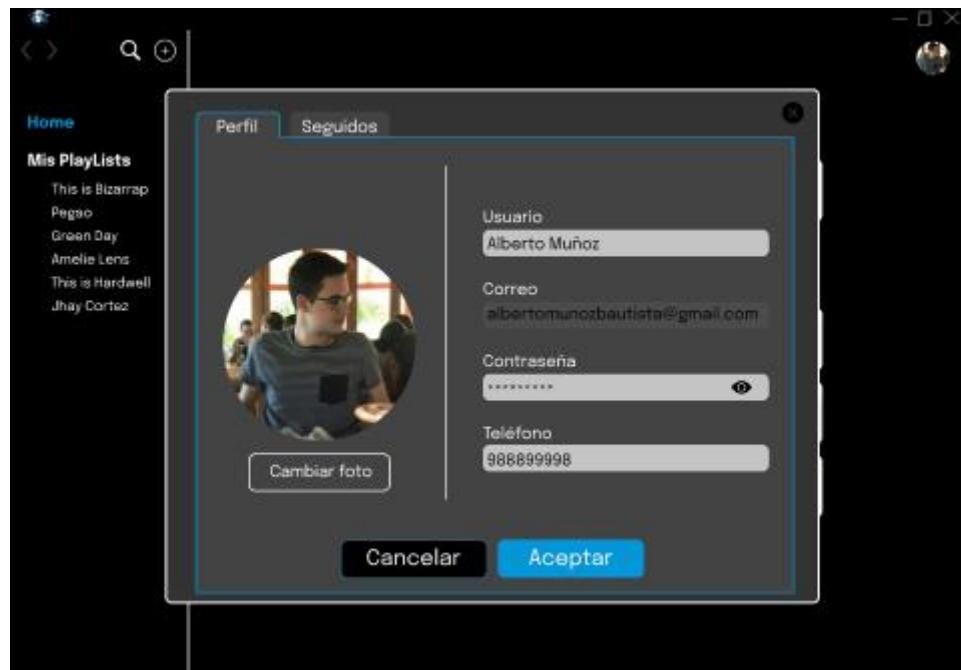
Arriba a la derecha de esta ventana, podremos ordenar las canciones por fecha de agregación, título y artista.



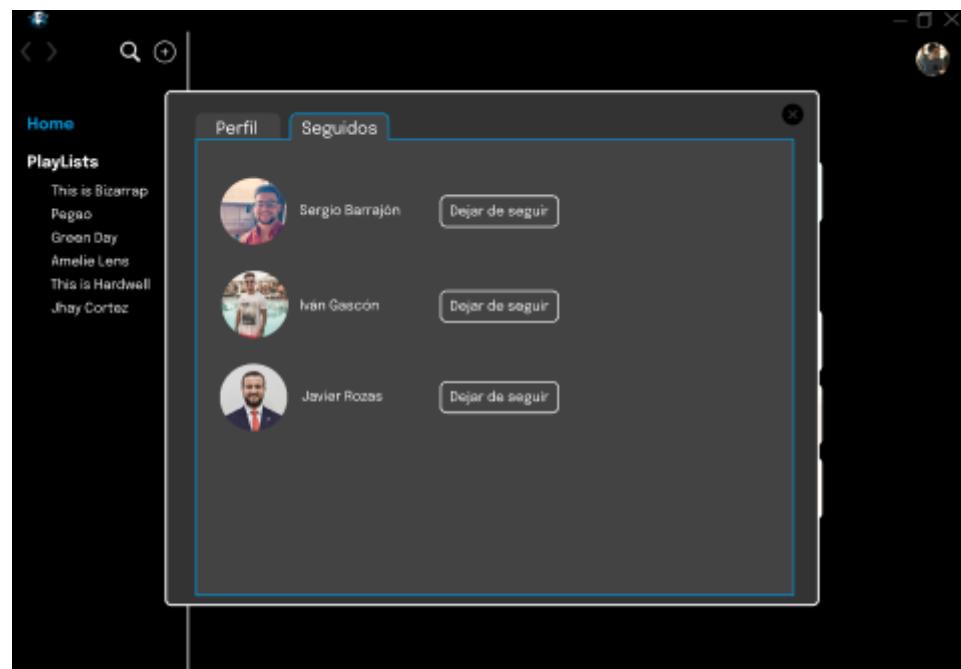
Y al pulsar sobre una canción, podremos reproducirla.



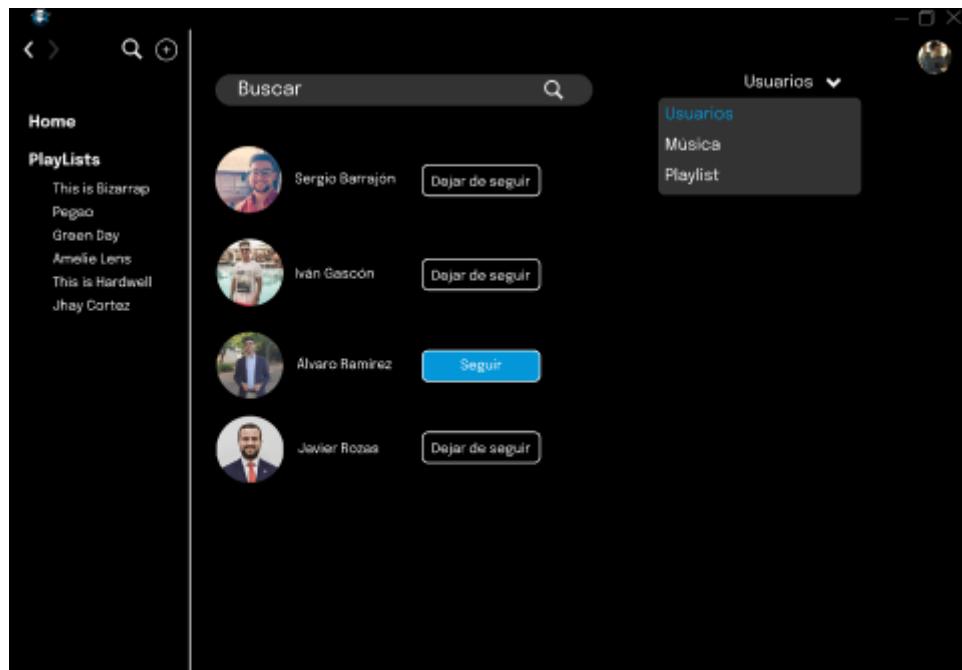
Desde el menú principal, al seleccionar un género musical, tendremos acceso a todas las canciones de ese género, también las podremos ordenar de diferentes maneras.



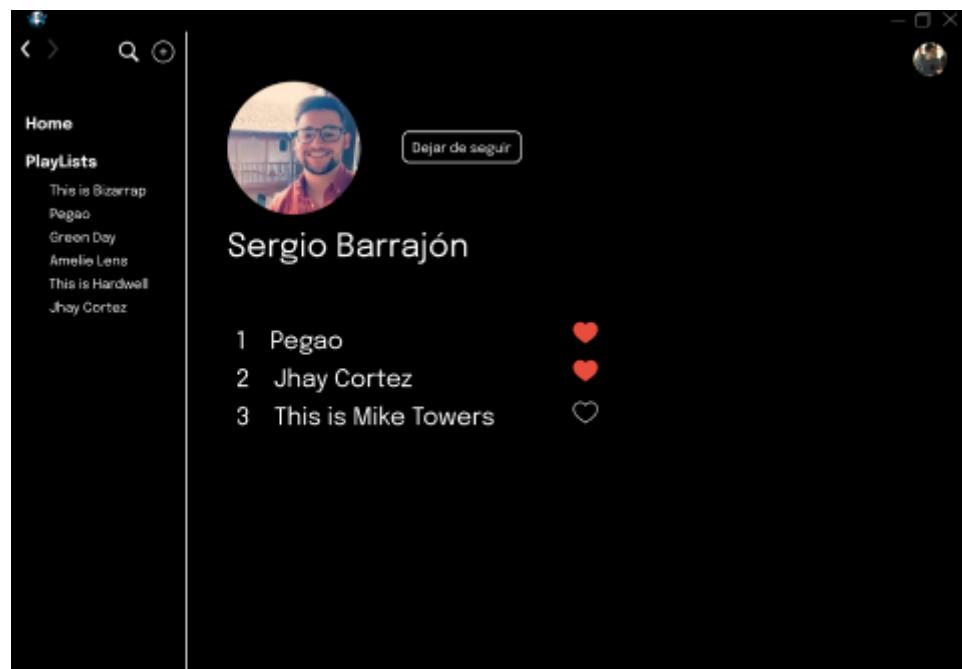
Desde el menú principal, al darle a nuestra foto de perfil, nos aparecerá una ventana emergente, donde tendremos dos pestañas, al seleccionar la pestaña de perfil, tendremos acceso a los datos de este, podremos añadir una foto personalizada y cambiar todo menos el correo.



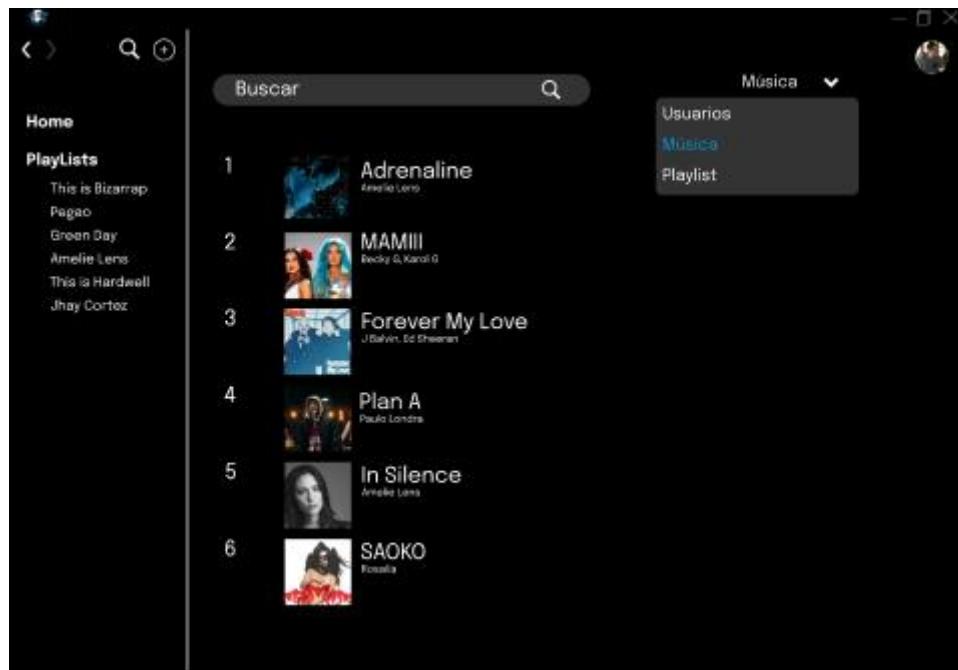
Al seleccionar la pestaña de seguidos, podremos ver toda la gente que seguimos.



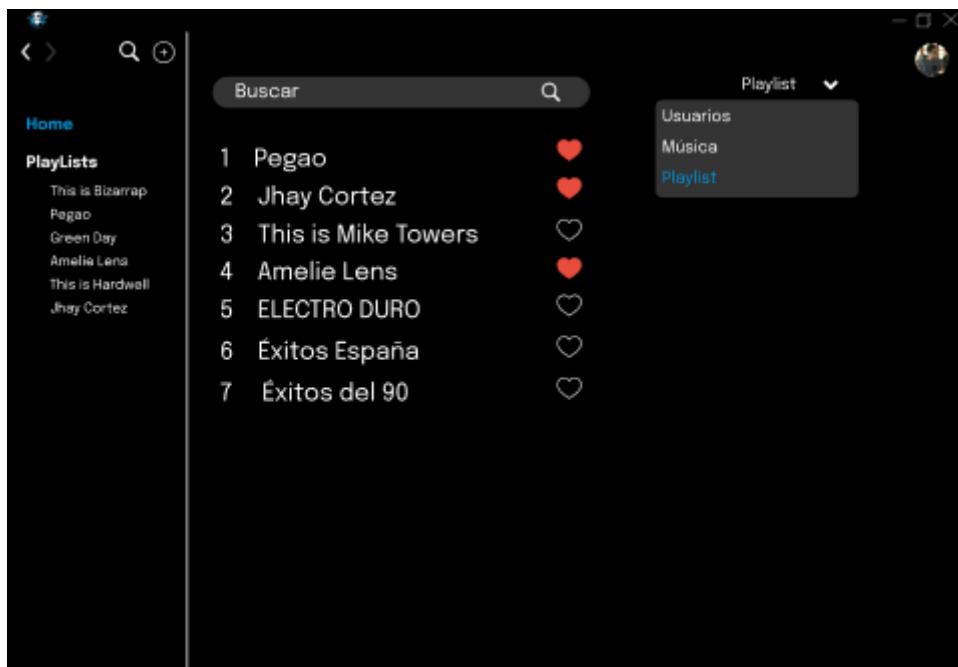
Desde el menú principal, al seleccionar el ícono de la lupa, tendremos acceso a un buscador, en este podremos buscar por usuarios, música y playlists.



Al buscar por usuarios, podremos seguirlos, y al hacer clic sobre uno, podremos ver todas sus playlists públicas y añadirlas a favoritos.



Al buscar por música tendremos acceso a todas las canciones de la aplicación.



Al buscar por playlists tendremos acceso a todas las playlists de la aplicación.

Prototipado de Administradores

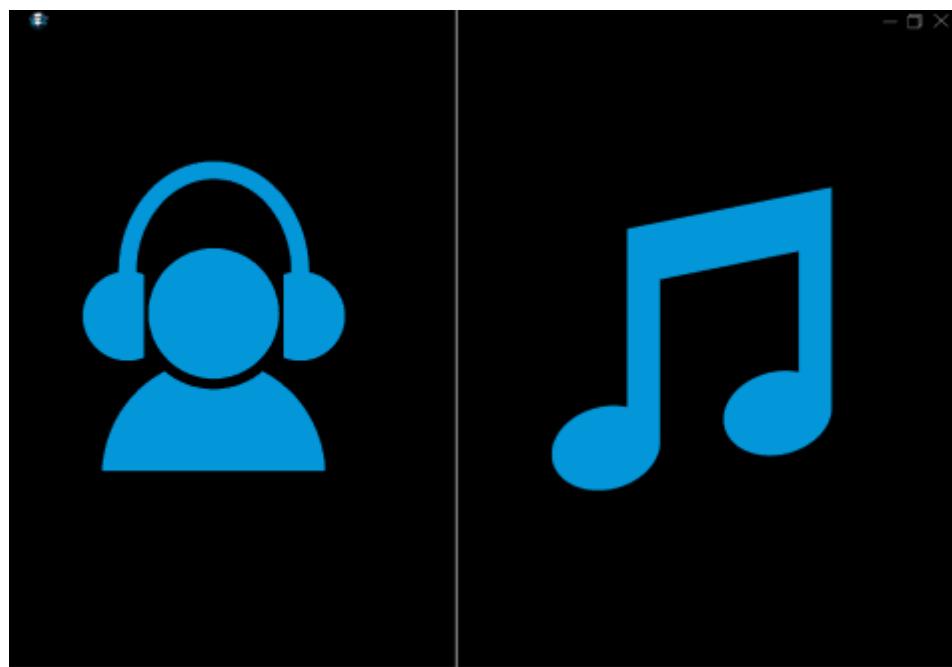
Este es el diseño y prototipado de la aplicación cuando el usuario es un administrador.



El splash screen será el mismo que el de los usuarios comunes, saldrá el logo de la aplicación y tras 3 segundos, nos llevará al login.



El login también será exactamente el mismo, ya que es la misma aplicación, lo único que se diferencia es el tipo de cuenta, si es una cuenta de tipo administrador, entraremos en otro menú.



Este será el menú de los administradores, donde tendremos acceso a la gestión de usuarios y de canciones.

A screenshot of a Windows-style application window showing a user management interface. At the top is a header row with columns labeled: IDUsuario, Usuario, Correo, Contraseña, Teléfono, and Foto. Below the header is a table with 10 rows of data. At the bottom center is a button labeled "Eliminar".

IDUsuario	Usuario	Correo	Contraseña	Teléfono	Foto

Eliminar

Al pulsar sobre los usuarios, nos llevará a una tabla donde estarán todos los usuarios de la aplicación, y pulsando el botón eliminar y teniendo seleccionado un usuario, lo podremos eliminar.

IDCanción	Género	Nombre	Artista	Foto	Fecha de agregación

Al pulsar sobre las canciones, nos llevará a una tabla donde estarán todas las canciones de la aplicación, y desde aquí las podremos añadir, eliminar y modificar.

4.2 Diagrama de clases

El diagrama de clases recoge las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del sistema y sus relaciones con los demás objetos, pero no muestra información temporal.

Con el fin de facilitar la comprensión del diagrama, se pueden incluir paquetes como elementos del mismo, donde cada uno de ellos agrupa un conjunto de clases.

Este diagrama no refleja los comportamientos temporales de las clases, aunque para mostrarlos se puede utilizar un diagrama de transición de estados.

Los elementos básicos del diagrama son:

Clases:

Una clase describe un conjunto de objetos con propiedades (atributos) similares y un comportamiento común. Los objetos son instancias de las clases.

No existe un procedimiento inmediato que permita localizar las clases del diagrama de clases. Estas suelen corresponderse con sustantivos que hacen referencia al ámbito del sistema de información y que se encuentran en los documentos de las especificaciones de requisitos y los casos de uso.

Dentro de la estructura de una clase se definen los atributos y las operaciones o métodos:

- Los atributos de una clase representan los datos asociados a los objetos instanciados por esa clase.
- Las operaciones o métodos representan las funciones o procesos propios de los objetos de una clase, caracterizando a dichos objetos.

El diagrama de clases permite representar clases abstractas. Una Clase abstracta es una clase que no puede existir en la realidad, pero que es útil conceptualmente para el diseño del modelo orientado a objetos. Las clases abstractas no son instanciables directamente sino en sus descendientes. Una clase abstracta suele ser situada en la jerarquía de clases en una posición que le permita ser un depósito de métodos y atributos para ser compartidos o heredados por las subclases de nivel inferior.

Las clases y en general todos los elementos de los diagramas, pueden estar clasificados de acuerdo a varios criterios, como por ejemplo su objetivo dentro de un programa. Esta clasificación adicional se expresa mediante un Estereotipo. Los tipos son:

- Objetos Entidad.
- Objetos límite o interfaz.
- Objetos de control.

Estos son estereotipos de clases. Un estereotipo representa una la meta-clasificación de un elemento.

Dependiendo de la herramienta utilizada, también se puede añadir información adicional a las clases para mostrar otras propiedades de las mismas, como son las reglas de negocio, responsabilidades, manejo de eventos, excepciones, etc.

Formulario de Reservas	
+ título : Titulo	
+ prestatario: Informacion_prestatario	
+ botonBuscarTítulo_Pulsado ()	
+ botonBuscarPrestatario_Pulsado()	
+ botonOk_Pulsado ()	
+ botonCancelar_Pulsado ()	
+ títuloResultado ()	
+ prestatarioResultado ()	
- comprobarEstado ()	
+ FormularioDeReservas ()	
# botonEliminarTítulo ()	

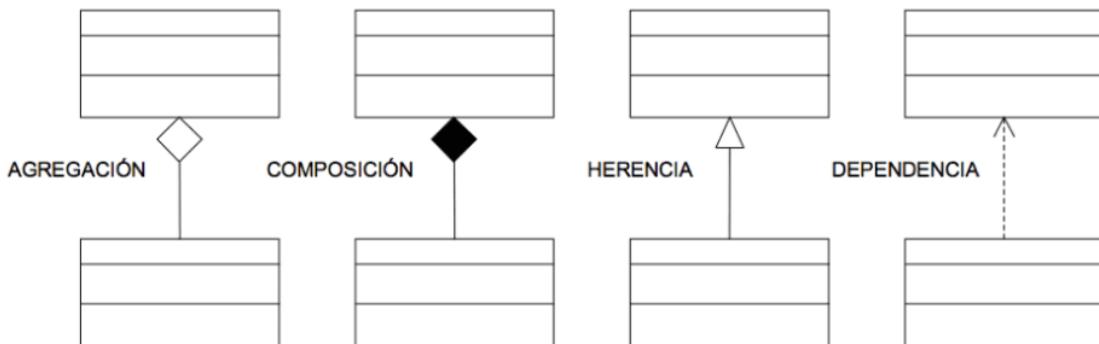
Relaciones:

Los tipos más importantes de relaciones estáticas entre clases son los siguientes:

- **Asociación:** Las relaciones de asociación representan un conjunto de enlaces entre objetos o instancias de clases. Es el tipo de relación más general, y denota básicamente una dependencia semántica. Por ejemplo, una Persona trabaja para una Empresa. Cada asociación puede presentar elementos adicionales que doten de mayor detalle al tipo de relación:

- Rol, o nombre de la asociación, que describe la semántica de la relación en el sentido indicado. Por ejemplo, la asociación entre Persona y Empresa recibe el nombre de trabaja para, como rol en ese sentido.
- Multiplicidad, que describe la cardinalidad de la relación, es decir, especifica cuántas instancias de una clase están asociadas a una instancia de la otra clase. Los tipos de multiplicidad son: Uno a uno, uno a muchos y muchos a muchos.
- **Herencia:** Las jerarquías de generalización/especialización se conocen como herencia. Herencia es el mecanismo que permite a una clase de objetos incorporar atributos y métodos de otra clase, añadiéndolos a los que ya posee. Con la herencia se refleja una relación “es_un” entre clases. La clase de la cual se hereda se denomina superclase, y la que hereda subclase. La generalización define una superclase a partir de otras. Por ejemplo, de las clases profesor y estudiante se obtiene la superclase persona. La especialización o especificación es la operación inversa, y en ella una clase se descompone en una o varias subclases. Por ejemplo, de la clase empleado se pueden obtener las subclases secretaria, técnico e ingeniero.
- **Agregación:** La agregación es un tipo de relación jerárquica entre un objeto que representa la totalidad de ese objeto y las partes que lo componen. Permite el agrupamiento físico de estructuras relacionadas lógicamente. Los objetos “son-parte-de” otro objeto completo. Por ejemplo, motor, ruedas, carrocería son parte de automóvil.

- **Composición:** La composición es una forma de agregación donde la relación de propiedad es más fuerte, e incluso coinciden los tiempos de vida del objeto completo y las partes que lo componen. Por ejemplo, en un sistema de Maquina de café, las relaciones entre la clase máquina y producto, o entre máquina y depósito de monedas, son de composición.
- **Dependencia:** Una relación de dependencia se utiliza entre dos clases o entre una clase y una interfaz, e indica que una clase requiere de otra para proporcionar alguno de sus servicios.



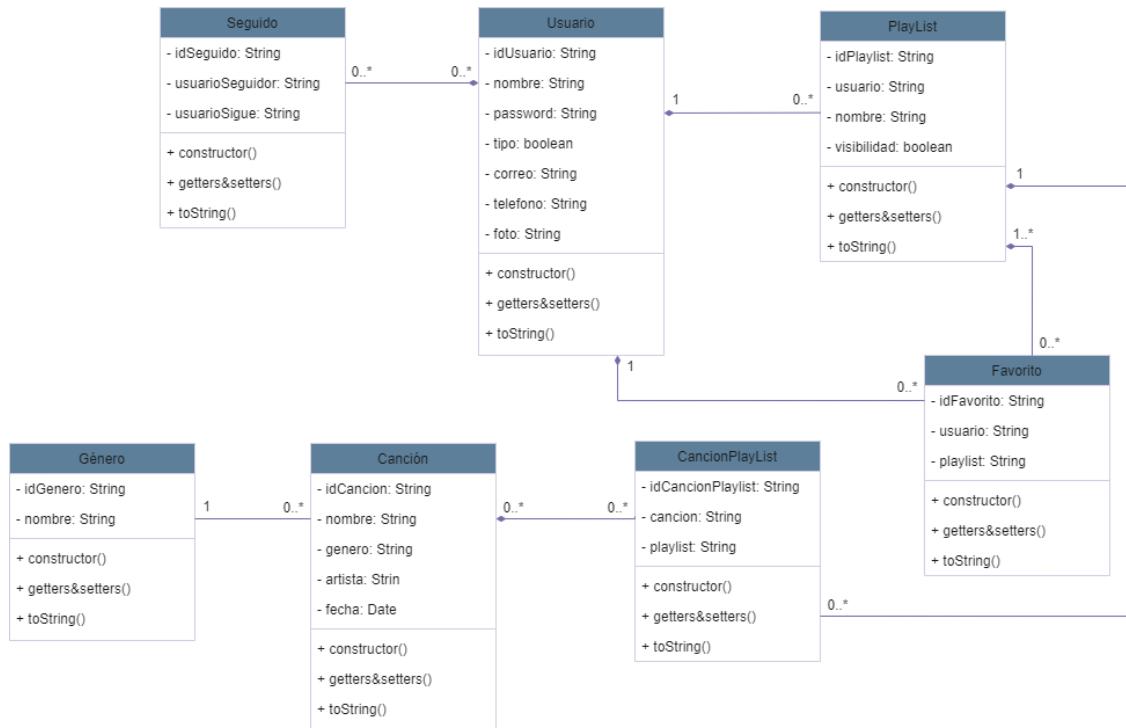
Interfaces:

Una interfaz es una especificación de la semántica de un conjunto de operaciones de una clase o paquete que son visibles desde otras clases o paquetes. Normalmente, se corresponde con una parte del comportamiento del elemento que la proporciona.

Paquetes:

Los paquetes se usan para dividir el modelo de clases del sistema de información, agrupando clases u otros paquetes según los criterios que sean oportunos.

Diagrama de clases de AMusic



Este será el diagrama de clases de mi aplicación, ahora voy a explicar las relaciones.

Seguido-Usuario

Existe una relación de composición entre un usuario y un seguido. Un usuario podrá seguir a 0 o a muchos usuarios y un usuario podrá ser seguido por 0 o muchos usuarios.

Usuario-PlayList

Existe una relación de composición entre usuario y playlist. Un usuario podrá tener 0 o muchas playlist, pero una playlist siempre será de un único usuario.

PlayList-CancionPlayList

Existe una relación de composición entre playlist y cancionPlaylist. Una playlist podrá tener 0 o muchas CancionesPlaylist, pero una CancionPlaylist siempre será de una única playlist.

Favorito-Usuario

Existe una relación de composición entre favorito y usuario. Un usuario podrá tener 0 o muchos favoritos, pero un favorito siempre será de un único usuario.

Favorito-Playlist

Existe una relación de composición entre favorito y playlist. Una playlist podrá estar en 0 o muchos favoritos, y en los favoritos podremos meter 1 o muchas playlists.

CanciónPlayList-Canción

Existe una relación de composición entre canción y CancionPlayList. Una canción puede estar en 0 o en muchas CancionesPlayList y una CancionPlayList podrá contener 0 muchas canciones.

Canción-Género

Existe una relación de agregación entre género y canción. Una canción solo tendrá un género, pero un género puede estar en 0 o en muchas canciones.

4.3 Diseño de la persistencia de la información

La persistencia es la capacidad de un lenguaje de programación o entorno de desarrollo de programación para, almacenar y recuperar el estado de los objetos de forma que sobrevivan a los procesos que los manipulan.

En este proyecto he decidido utilizar un modelo no relacional (NoSQL), en concreto la base de datos de Google Firebase, la razón por la que he elegido esta opción son las distintas ventajas que nos brinda este modelo que voy a exponer a continuación.

Las bases de datos NoSQL están diseñadas para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Estas bases de datos son ampliamente reconocidas porque son fáciles de desarrollar, gracias a su funcionalidad y el rendimiento a escala.

Estas bases de datos utilizan una variedad de modelos de datos para acceder y administrar datos, están optimizadas específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles.



Hay distintos tipos de bases de datos NoSQL:

- **Clave-valor:** Las bases de datos clave-valor son altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos y tecnología publicitaria se prestan particularmente bien con el modelo de datos clave-valor.
- **Documentos:** En el código de aplicación, los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el uso del mismo formato de modelo de documento que emplean en el código de aplicación. La naturaleza flexible, semiestructurada y jerárquica de los documentos y las bases de datos de documentos permite que evolucionen según las necesidades de las aplicaciones. Este modelo funciona bien con catálogos, perfiles de usuario y sistemas de administración de contenido en los que cada documento es único y evoluciona con el tiempo.
- **Gráficos:** El propósito de una base de datos de gráficos es facilitar la creación y ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraudes...
- **En memoria:** Las aplicaciones de juegos y tecnología publicitaria tienen casos de uso como tablas de clasificación, tiendas de sesión y análisis en tiempo real que requieren tiempos de respuesta de microsegundos y pueden tener grandes picos de tráfico en cualquier momento.

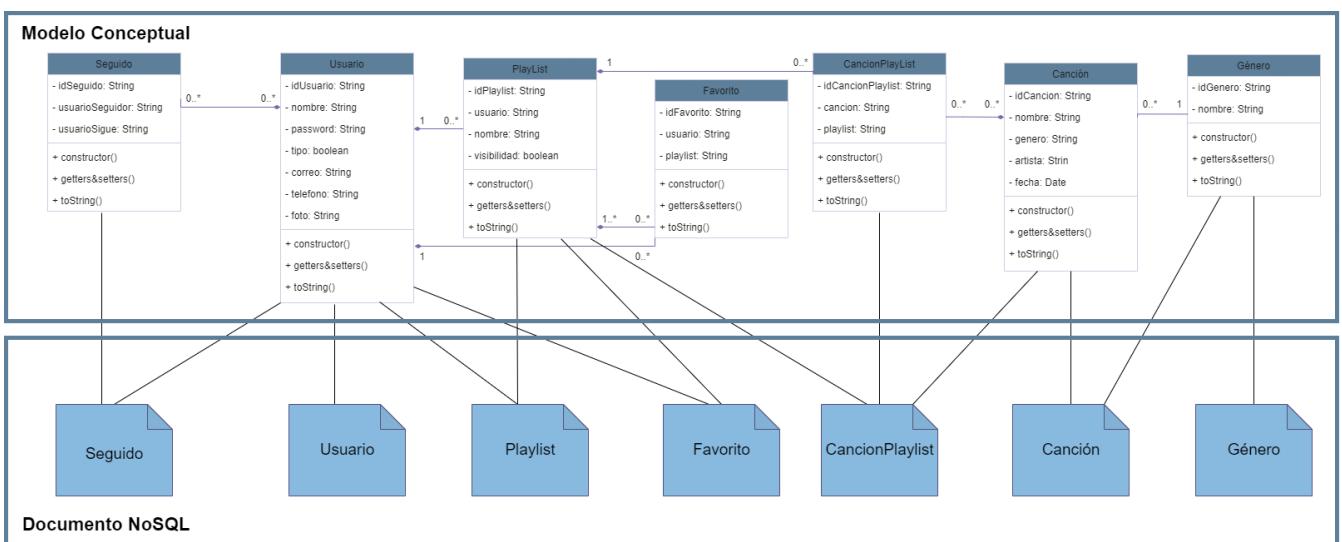
Las bases de datos NoSQL se adaptan perfectamente a muchas aplicaciones modernas, como dispositivos móviles, web y juegos, que requieren bases de datos con unas grandes cualidades para proporcionar excelentes experiencias de usuario, algunas de estas cualidades son:

- **Flexibilidad:** Las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.
- **Escalabilidad:** Las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de hardware en lugar de escalar añadiendo servidores caros y sólidos.
- **Alto rendimiento:** Las bases de datos NoSQL están optimizadas para modelos de datos específicos y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.
- **Altamente funcional:** Las bases de datos NoSQL proporcionan APIs altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

Una vez expuestas y analizadas todas las ventajas de estas bases de datos, y teniendo en cuenta los requisitos que vamos a tener en AMusic. Esta es la opción por la que me voy decantar, ya que va a ser una base de datos sujeta a muchos cambios, a un gran crecimiento y queremos que nuestra base de datos cuente con la mayor optimización y rapidez posible.

Diagrama de la persistencia de la información de AMusic

Ahora vamos a mostrar el diagrama de la base de datos NoSQL de AMusic. En la parte superior tendremos las clases y las relaciones entre ellas, y en la parte inferior tendremos los documentos donde se guardan los datos y estarán relacionados con las clases.

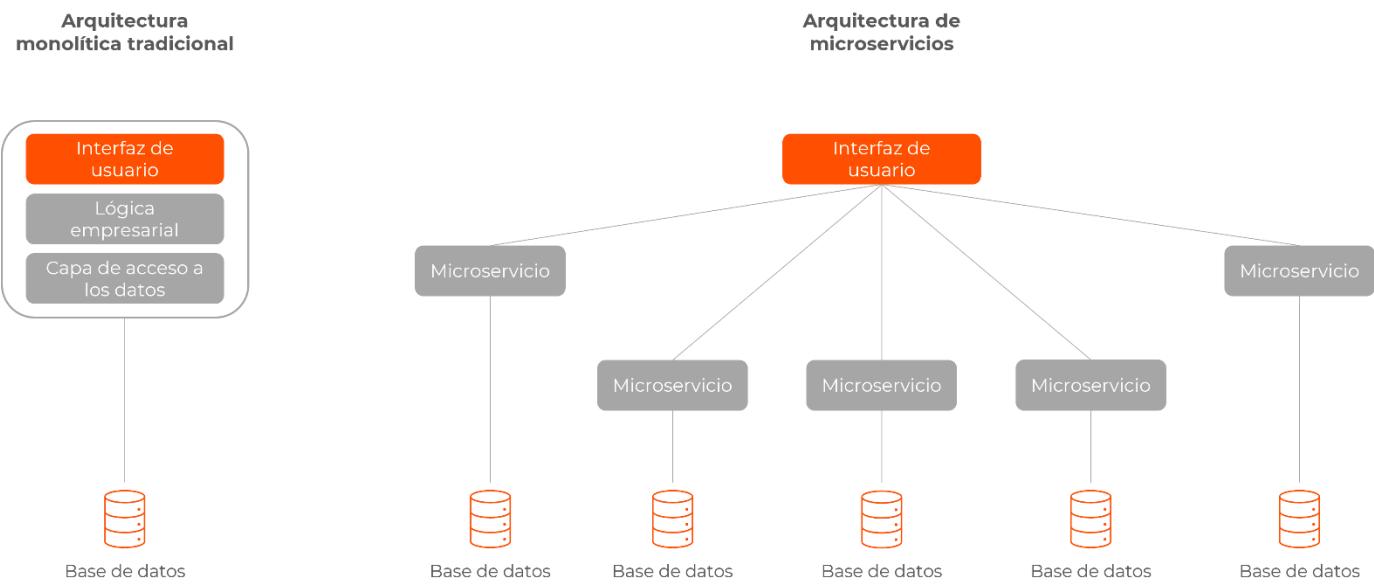


- **Seguido:** Tendrá la información de la clase Seguido y el idUsuarioSeguidor e idUsuarioSeguido serán las referencias de la clase Usuario.
- **Usuario:** Tendrá la información de la clase usuario.
- **Playlist:** Tendrá información de la clase playlist y el idUsuario será la referencia de la clase usuario.
- **Favorito:** Tendrá la información de la clase favorito y el idFavorito será la referencia de la clase playlist y usuario.
- **CancionPlaylist:** Tendrá la información de la clase CanciónPlayLists y el idCanciónPlaylist será la referencia de la clase playlist y canción.
- **Canción:** Tendrá la información de la clase canción y el idCanción será la referencia de la clase género.
- **Género:** Tendrá a información de la clase género.

4.4 Arquitectura del sistema

Tradicionalmente el diseño de software se ha realizado con arquitectura monolítica, en la que el software se estructura de forma que todos los aspectos funcionales quedan acoplados y sujetos en un mismo programa. En este tipo de sistema, toda la información está alojada en un servidor, por lo que no hay separación entre módulos y las diferentes partes de un programa están muy acopladas. Esto genera un problema a largo plazo, ya que se trata de un sistema no escalable de manera sencilla, por eso aparece la arquitectura de microservicios.

La idea de los microservicios es dividir los sistemas en partes individuales, permitiendo que se puedan tratar y abordar los problemas de manera independiente, así, mientras que en una arquitectura monolítica el software se desarrolla como una única unidad, una arquitectura de microservicios funciona con un conjunto de pequeños servicios que se ejecutan de manera autónoma e independiente.

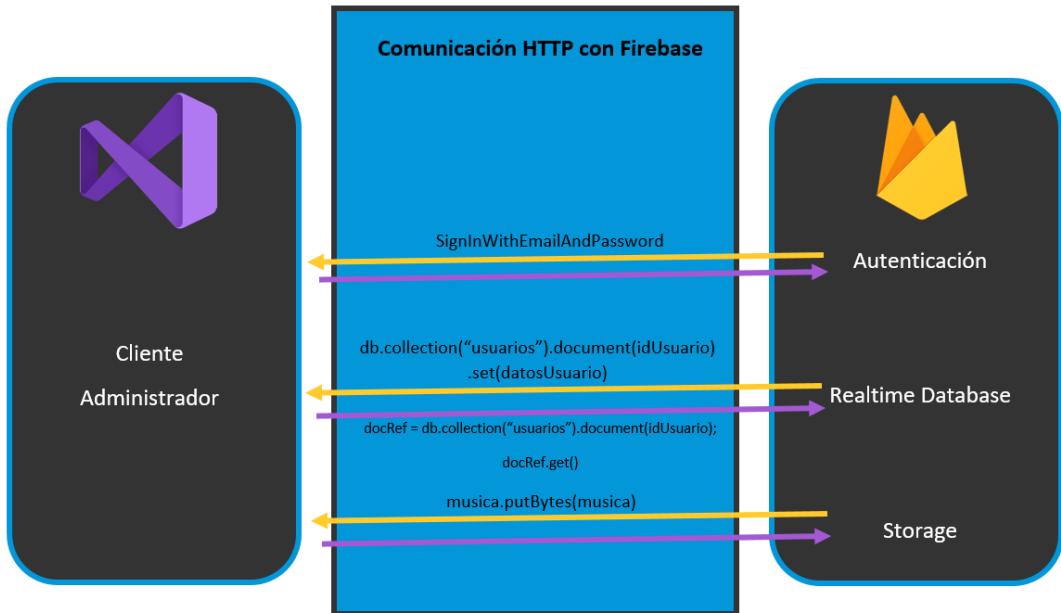


En esta arquitectura, cada microservicio es un código que puede estar en un lenguaje de programación diferente, y que desempeña una función específica. Los microservicios se comunican entre sí a través de APIs y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

Los microservicios han creado infraestructuras IT más adaptables y flexibles, porque si se quiere modificar solamente un servicio, no es necesario alterar el resto de la infraestructura. Cada uno de los servicios se puede desplegar y modificar sin que ello afecte a otros servicios o aspectos funcionales de la aplicación.

Además de lo nombrado anteriormente, estas son las ventajas de los microservicios:

- **Modularidad:** al tratarse de servicios autónomos, se pueden desarrollar y desplegar de forma independiente. Además, un error en un servicio no debería afectar la capacidad de otros servicios para seguir trabajando según lo previsto.
- **Escalabilidad:** como es una aplicación modular, se puede escalar horizontalmente cada parte según sea necesario, aumentando el escalado de los módulos que tengan un procesamiento más intensivo.
- **Versatilidad:** se pueden usar diferentes tecnologías y lenguajes de programación. Lo que permite adaptar cada funcionalidad a la tecnología más adecuada y rentable.
- **Rapidez de actuación:** el reducido tamaño de los microservicios permite un desarrollo menos costoso, así como el uso de "contenedores de software" permite que el despliegue de la aplicación se pueda llevar a cabo rápidamente.
- **Mantenimiento simple y barato:** al poder hacerse mejoras de un solo módulo y no tener que intervenir en toda la estructura, el mantenimiento es más sencillo y barato que en otras arquitecturas.
- **Agilidad:** se pueden utilizar funcionalidades típicas (autenticación, trazabilidad, etc) que ya han sido desarrolladas por terceros, no hace falta que el desarrollador las cree de nuevo.



Este es el diagrama de arquitectura de AMusic, el cual nos va a ayudar a visualizar la estructura general de alto nivel de la aplicación. En este diagrama podremos distinguir el cliente, el servidor y las comunicaciones que se realizan. La aplicación estará formada por los microservicios de autenticación, realtime database y storage.

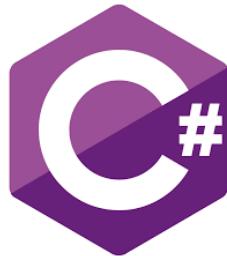
5. Implementación de la solución

5.1 Análisis tecnológico

A la hora de empezar con un proyecto es muy importante elegir bien las tecnologías implicadas, las tecnologías que se van a usar y esta es una decisión difícil ya que hay tecnologías que se ajustan más a nosotros en lo personal (las tenemos más o menos dominadas y nos gustan más o menos) y tecnologías que se ajustan más o menos a nuestro proyecto, y a veces hay que elegir y dependiendo de muchos factores usaremos unas tecnologías u otras.

En este apartado voy a exponer algunas de las tecnologías que he estado tanteando, las voy a definir, voy a decir sus principales características, y al final voy a decir la conclusión y a la decisión que he llegado.

Tecnología del cliente:



C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de JAVA aunque incluye mejoras derivadas de otros lenguajes.

El nombre de C Sharp fue inspirado por el sino #, el cual se lee como Sharp en inglés para la notación musical. Aunque C# forma parte de la plataforma .NET, esta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.



Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.

Se caracteriza por ser un lenguaje simple, ya que ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

Es un lenguaje orientado a objetos, distribuido y con una arquitectura neutral.



Dart es un lenguaje de código abierto desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos con análisis estático de tipo. Desde la primera versión estable en 2011, este lenguaje ha cambiado mucho, tanto el lenguaje en sí como en sus objetivos principales. Con la versión 2.0, el sistema de tipo de Dart pasó de opcional a estático y desde su llegada Flutter se ha convertido en el principal objetivo del lenguaje.

Flutter es un framework de Dart para crear aplicaciones multiplataforma con un único código. A diferencia de otros frameworks multiplataforma, el código de una aplicación de Flutter se compila a código nativo, por lo que el rendimiento alcanzado es superior a aplicaciones basadas en web-views, además Flutter no utiliza componentes nativos, sino que viene con sus propios componentes, llamados widgets, por lo que la misma aplicación se verá igual en cualquier dispositivo, independientemente de su sistema operativo o la versión. Gracias a ello, el desarrollador no tiene que preocuparse por que el diseño de su aplicación se vea mal en dispositivos antiguos.

Además de aplicaciones móviles, con Flutter también se pueden hacer páginas web y aplicaciones de escritorio, aunque el soporte para páginas web está en beta, y por aplicaciones de escritorio está en technical preview, por lo que quien lo quiera usar tendrá que esperar un tiempo más hasta que sea estable.



Kotlin es un lenguaje de programación de código abierto creado por JetBrains que se ha popularizado gracias a que se puede utilizar para programar aplicaciones Android.

Kotlin destaca por las ventajas que tiene respecto a Java a la hora de desarrollar aplicaciones móviles, además de por presentar características como simplificar la lectura del código y del propio desarrollo de este, algunas de las ventajas de Kotlin son:

- **Interoperabilidad con código Java:** una de las características principales de Kotlin es que está diseñado para iteroperar completamente con la sintaxis del lenguaje de Java. Es decir, con una base de código existente escrita en Java, puede interactuar con Kotlin y viceversa.

- **Curva de aprendizaje sencilla:** La sencillez de la sintaxis permite una curva de aprendizaje fluida, intuitiva y fácil de usar, perfecta para los que quieran aprender su primer lenguaje de programación. Además, como es de código abierto, hay un gran apoyo de la comunidad de Kotlin, lo que supone una gran ventaja.
- **Menor tiempo de programación:** Uno de los puntos fuertes de Kotlin es que elimina el código redundante, además de ser compacto y conciso, lo que optimiza mucho el proceso de escritura de código y evita la repetición.
- **Orientado a objetos y programación funcional:** Aunque lo habitual en el desarrollo de apps móviles es un paradigma a objetivos, Kotlin demuestra que también se puede trabajar de la mano de la programación funcional. La posibilidad de trabajar con lambdas en este entorno simplifica las tareas más comunes y tediosas en el desarrollo.
- **Corritunas:** Otra de las grandes ventajas de Kotlin es que las corrutinas optimizan la programación asíncrona. Simplifican así el aburrido trabajo de las llamadas de red y acceso a las bases de datos, y dejan atrás los callbacks.
- **Desarrollo multiplataforma:** Kotlin se puede utilizar para cualquier tipo de desarrollo, desde la web del lado del servidor y del lado del cliente, hasta Android e iOS. Como el lenguaje se ejecuta en JVM, permite compartir el código entre diferentes plataformas.
- **Flexibilidad:** Kotlin da a los desarrolladores libertad de trabajar con el estilo que elijan. Por tanto, es un lenguaje altamente flexible que tiene construcciones funcionales y orientadas a objetos. Todo ello se traduce en una mejor experiencia a la hora de programar.

Tecnología del servidor:



Firebase es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible en distintas plataformas (iOS, Android y web), por lo que es más rápido trabajar en el desarrollo.

Fue creada en 2011 pero pasó a ser parte de Google en 2014 comenzando como una base de datos en tiempo real. Sin embargo, se añadieron más funciones y características:

- **Realtime database:** Es una de las herramientas más destacadas y esenciales de Firebase, son bases de datos en tiempo real. Estas se alojan en la nube, son NoSQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real.
- **Autenticación de usuarios:** Ofrece un sistema de autenticación de usuarios que permite tanto el registro propiamente dicho (mediante email y contraseña) como el acceso utilizando perfiles de otras plataformas externas (Google, Facebook...).

- **Almacenamiento en la nube:** Cuenta con un sistema de almacenamiento donde los desarrolladores puedan guardar los ficheros de sus aplicaciones. Este almacenamiento es de gran ayuda para tratar los archivos de los usuarios.
- **CrashReporting:** Esta funcionalidad detecta y ayuda a solucionar los problemas de la app, consiguiendo un informe de errores muy detallado y organizado.
- **TestLab:** Con esta opción podremos testear la app en dispositivos Android virtuales basados en los parámetros que configuremos.
- **Remote Config:** La configuración remota nos va a servir para modificar ciertas funciones, aspectos o incluso la apariencia de la aplicación sin que sea necesario publicar una actualización.
- **Cloud Messaging:** Esta funcionalidad nos permitirá el envío de notificaciones y mensajes a diversos usuarios en tiempo real y a través de varias plataformas.

La función principal de Firebase es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.



Amazon Web Services (AWS) es una plataforma de servicios de nube que proporciona una variedad de servicios de infraestructura tales como almacenamiento, redes, bases de datos, servicios de aplicaciones, potencia de cómputo, entre otros, los cuales permiten el crecimiento de las empresas.

- **Funcionalidad:** Cuenta con una cantidad de servicios y características incluidas.
- **Gran comunidad:** Tiene una de las comunidades más grandes y dinámicas del mercado, con millones de clientes activos y decenas de miles de socios en todo el mundo.
- **Seguro:** Está diseñado para ser uno de los entornos de informática de nube más flexible y seguro.
- **Innovación:** Utiliza las últimas tecnologías para experimentar e innovar de forma más rápida.



SpringBoot es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el framework Spring Core. Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar.

Estas son las principales características que tiene Spring Boot:

- **Configuración:** Cuenta con un complejo módulo que auto configura todos los aspectos de nuestra aplicación para poder simplemente ejecutar la aplicación sin tener que definir absolutamente nada.
- **Resolución de dependencias:** Solo hay que determinar qué tipo de proyecto estamos utilizando y este se encargará de resolver todas las librerías y dependencias para que este funcione.
- **Despliegue:** Puede desplegar las aplicaciones mediante un servidor web integrado, como es el caso de Tomcat.
- **Métricas:** Por defecto, Spring Boot cuenta con servicios que permiten consultar el estado de salud de la aplicación, permitiendo así saber si la aplicación está encendida o apagada y la memoria utilizada y disponible entre otras muchas cosas.
- **Extensible:** Permite la creación de complementos, los cuales ayudan a que la comunidad de Software Libre cree nuevos módulos que faciliten aún más el desarrollo.

Para la parte del cliente, he decidido decantarme con C# ya que tenía decidido hacer una aplicación de escritorio, y además de que es un lenguaje que conozco y que he trabajado, me siento bastante cómodo usándolo y creo que posee las características con las que quiero enfocar mi aplicación. La otra opción era Dart, ya que con Flutter también podría desarrollar una aplicación de escritorio, pero como esa parte está en beta y no está estable, me he decantado por C#.

Para la parte del servidor, voy a usar Firebase, ya que es una plataforma bastante completa, y ya la he usado con anterioridad, aunque no con C#.

5.1 Elementos a implementar

En este apartado voy a realizar una explicación del desarrollo de los elementos a implementar.

Voy a empezar explicando el login. Esta es la segunda pantalla que veremos (por detrás del splash screen) en esta pantalla tendremos dos cuadros de texto para escribir nuestro usuario y nuestra contraseña.

```
1 referencia
private void btnLogin1_Click(object sender, EventArgs e)
{
    String user = txtUsuario.Text;
    String password = txtPassword.Text;

    if (string.IsNullOrWhiteSpace(user) || string.IsNullOrWhiteSpace(password))
    {
        MessageBox.Show("Rellene todos los campos!");
    }
    else
    {
        FirebaseResponse res = cliente.Get(@"Usuarios/" + user);
        Usuario resUsuario = res.ResultAs<Usuario>();
        Usuario curUsuario = new Usuario()
        {
            nombre = user,
            password = password
        };

        if (Usuario.loginCorreoPassword(resUsuario, curUsuario))
        {
    
```

Una vez le escribamos nuestro usuario y contraseña, primero nos comprobará que no nos hemos dejado ninguno de los dos campos de texto vacíos y una vez hayamos pasado este filtro, comprobaremos que nuestro usuario tiene esas credenciales, esto lo haremos con el método loginCorreoPassword.

```
1 referencia
public static bool loginCorreoPassword(Usuario u1, Usuario u2)
{
    if (u1 == null || u2 == null)
    {
        return false;
    }

    if (u1.nombre != u2.nombre)
    {
        error = "El nombre de usuario no existe!";
        return false;
    }
    else if (u1.password != u2.password)
    {
        error = "La contraseña es incorrecta!";
        return false;
    }

    return true;
}
```

Si superamos este otro filtro, dependiendo de nuestro tipo de usuario (normal o administrador) nos iremos a un menú u otro.

```
//dependiendo del tipo de usuariion que esa se nos abrirá un menú u otro.
if (resUsuario.tipo)
{
    FrmMenuAdmin frmMenuAdmin = new FrmMenuAdmin();
    frmMenuAdmin.Show();
    this.Hide();
}
else
{

    FrmMenuUser frmMenuUser = new FrmMenuUser(resUsuario);
    frmMenuUser.Show();
    this.Hide();
}
```

Desde el menú administrador tendremos la posibilidad de gestionar tanto los usuarios como las canciones.

Al hacer clic sobre la opción de gestionar usuarios, rellenaremos una tabla con usuarios.

```
IFirebaseConfig ifc = new FirebaseConfig()
{
    AuthSecret = "eqBhoT5oymn26ukYh7HFYGFx568xn9MIK8jrzo7a",
    BasePath = "https://amusicprueba-default-rtdb.firebaseio.com/"
};

IFirebaseClient cliente;
```

Nos conectamos a firebase, a través del link de nuestra base de datos y su secreto.

```
2 referencias
private void cargarDatagrid()
{
    FirebaseResponse res = cliente.Get(@"Usuarios");
    Dictionary<string, Usuario> data = JsonConvert.DeserializeObject<Dictionary<string, Usuario>>(res.Body.ToString());
    llenarDataGrid(data);
}

1 referencia
private void llenarDataGrid(Dictionary<string, Usuario> record)
{
    dgvUsuarios.Rows.Clear();
    dgvUsuarios.Columns.Clear();

    dgvUsuarios.Columns.Add("usuario", "Usuario");
    dgvUsuarios.Columns.Add("correo", "Correo");
    dgvUsuarios.Columns.Add("contraseña", "Contraseña");
    dgvUsuarios.Columns.Add("telefono", "Teléfono");
    dgvUsuarios.Columns.Add("tipo", "Admin");
    dgvUsuarios.Columns.Add("foto", "Foto");

    foreach (var item in record)
    {
        dgvUsuarios.Rows.Add(item.Value.nombre, item.Value.correo, item.Value.password, item.Value.telefono, item.Value.tipo);
    }
}
```

Accedemos a la tabla usuarios, la convertimos en un diccionario y rellenamos una tabla con sus valores.

Usaremos la misma técnica para las canciones, voy a mostrar una imagen de como quedarán en la tabla.

Nombre	Artista	Fecha	Género	Canción	Carátula
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Caminaré	Natos, Waor y Maka	08/06/2022	Flameco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Disuestos a morir	Natos, Waor, CRO, Homer ...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

En la gestión de canciones podremos añadir, eliminar y modificar las mismas.

```
2 referencias
private Cancion getSelectedSong()
{
    int rowindex = dgvCanciones.CurrentCell.RowIndex;
    //int columnindex = dgvUsuarios.CurrentCell.ColumnIndex;

    String idCancion = dgvCanciones.Rows[rowindex].Cells[4].Value.ToString();

    //MessageBox.Show(nombreUsuario);

    FirebaseResponse result = cliente.Get(@"Cancion/" + idCancion);
    Cancion cancionSeleccionada = result.ResultAs<Cancion>();

    //MessageBox.Show(usuarioSeleccionado.nombre.ToString());

    return cancionSeleccionada;
}
```

```

1 referencia
private void btnEliminar1_Click(object sender, EventArgs e)
{
    if (dgvCanciones.SelectedRows.Count > 0)
    {

        DialogResult dialogResult = MessageBox.Show("¿Desea borrar la canción?", "ATENCIÓN", MessageBoxButtons.YesNo);
        if (dialogResult == DialogResult.Yes)
        {
            Cancion cancionSeleccionada = getSelectedSong();
            cliente.Delete("Cancion/" + cancionSeleccionada.idCancion);

            MessageBox.Show("Cancion eliminada con éxito!");
            cargarDatagrid();
        }
    }
    else
    {
        MessageBox.Show("Ninguna fila seleccionada!");
    }
}

```

Para la opción de eliminar, al pulsar el botón, si hemos seleccionado alguna fila, llamamos al método getSelectedSong que nos devolverá la canción que hemos marcado en la tabla, una vez tenemos la canción, la eliminamos de firebase.

Para la opción de modificar, al pulsar el botón se nos abrirá un nuevo formulario al que le pasaremos la canción seleccionada y donde cargaremos sus componentes.

```

1 referencia
private void cargarComponentes()
{
    FirebaseResponse result = cliente.Get(@"Genero/" + cancion.genero);
    Genero genero = result.ResultAs<Genero>();

    this.cmbGenero.Text = genero.nombre;
    this.txtArtista.Text = cancion.artista;
    this.txtNombre.Text = cancion.nombre;
}

```

También cargaremos el combobox de géneros

```

1 referencia
private void cargarGeneros()
{
    FirebaseResponse res = cliente.Get(@"Genero");
    Dictionary<string, Genero> data = JsonConvert.DeserializeObject<Dictionary<string, Genero>>(res.Body.ToString());

    foreach (var item in data)
    {
        listaIdGenero.Add(item.Value.idGenero);
        listaNombreGenero.Add(item.Value.nombre);
    }

    for (int i = 0; i < listaNombreGenero.Count; i++)
    {
        this.cmbGenero.Items.Add(listaNombreGenero[i].ToString());
    }

    //cmbGenero.SelectedIndex = 0;
}

```

Hacemos una consulta a la tabla género de la base de datos, guardamos sus valores en diferentes arraylists y lo cargamos en el combobox.

```
1 referencia
private void btnAceptar1_Click(object sender, EventArgs e)
{
    String artista = txtArtista.Text;
    String nombre = txtNombre.Text;
    Cancion c;

    try
    {
        String genero = this.listaIdGenero[cmbGenero.SelectedIndex].ToString();
        c = new Cancion(cancion.idCancion, nombre, cancion.cancion, cancion.caratula, genero, artista, cancion.fecha);
    }
    catch (Exception)
    {
        c = new Cancion(cancion.idCancion, nombre, cancion.cancion, cancion.caratula, cancion.genero, artista, cancion.fecha);
    }

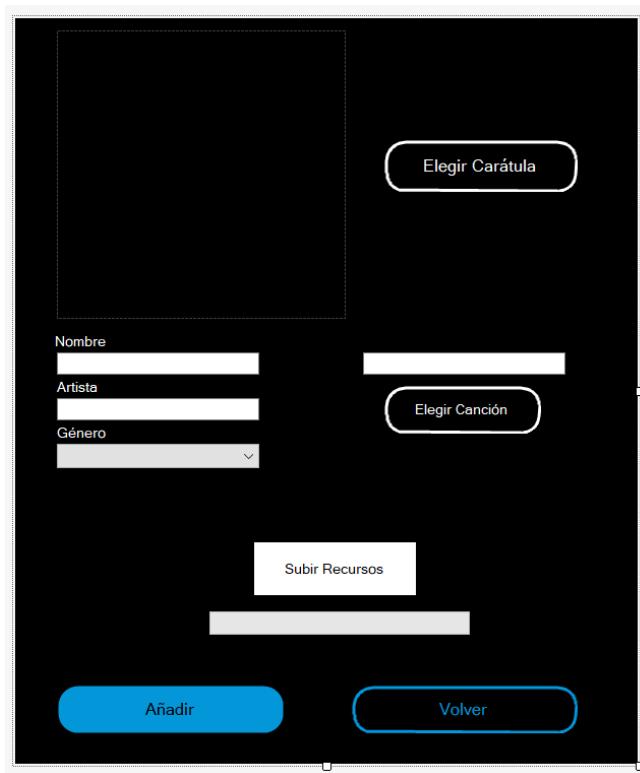
    cliente.Update("Cancion/" + cancion.idCancion, c);

    MessageBox.Show("Canción Modificada");

    this.Close();
    frmGestionCanciones.cargarDatagrid();

    //FrmGestionCanciones frmGestionCancionesNuevo = new FrmGestionCanciones();
    //frmGestionCancionesNuevo.Show();
}
```

Si hacemos clic a aceptar, recogemos los datos de los distintos componentes que queremos cambiar, creamos una nueva canción y la añadimos a la tabla canción de firebase.



Primero a través de unos open file dialog, seleccionamos tanto la carátula como la canción.

```

1 referencia
private void btnCaratula1_Click(object sender, EventArgs e)
{
    //filtramos el openfile dialog
    ofdCaratula.Filter = "Imágenes(*.jpg; *.jpeg; *.gif; *.bmp; *.png)|*.jpg; *.jpeg; *.gif; *.bmp; *.png";
    ofdCaratula.FileName = "";

    //si elegimos la imagen, se cambia
    if (ofdCaratula.ShowDialog() == DialogResult.OK)
    {
        picCaratula.Image = new Bitmap(ofdCaratula.FileName);
        caratulaURL = ofdCaratula.FileName;

    }
}

1 referencia
private void btnCancion1_Click(object sender, EventArgs e)
{
    //filtramos el openfile dialog
    ofdCancion.Filter = "Canción(*.mp3);*.mp3";
    ofdCancion.FileName = "";

    //si elegimos la imagen, se cambia
    if (ofdCancion.ShowDialog() == DialogResult.OK)
    {
        //picCaratula.Image = new Bitmap(ofdCaratula.FileName);
        //caratulaURL = ofdCaratula.FileName;

        cancionURL = ofdCancion.FileName;
        txtCancion.Text = Path.GetFileName(cancionURL);

    }
}

```

Una vez tengamos seleccionados ambos recursos, los subimos al storage de firebase.

```

1 referencia
private async void insertarCaratula()
{
    //limpiamos la imagen del picture box
    picCaratula.Image.Dispose();

    //cogemos la url de la imagen
    var stream = File.Open(caratulaURL, FileMode.Open);

    //cogemos el nombre de la imagen
    String nombreImagen = Path.GetFileName(caratulaURL);

    //la insertamos
    var task = new FirebaseStorage("amusicprueba.appspot.com")
        //Caratulas
        //Usuarios
        .Child("Images")
        .Child("Caratulas")
        .Child(nombreImagen)
        .PutAsync(stream);

    // Await the task to wait until upload is completed and get the download url
    String downloadUrl = await task;

    caratulaStorage = downloadUrl;

}

```

```

1 referencia
private async void insertarCancion()
{
    //cogemos la url de la canción
    var stream = File.Open(cancionURL, FileMode.Open);

    //cogemos el nombre de la canción
    String nombreCancion = Path.GetFileName(cancionURL);

    //la insertamos
    var task = new FirebaseStorage("amusicprueba.appspot.com")
        //Carátulas
        //Usuarios
        .Child("Musica")
        .Child(nombreCancion)
        .PutAsync(stream);

    // Await the task to wait until upload is completed and get the download url
    task.Progress.ProgressChanged += (s, e) => prbProgreso.Value = e.Percentage; //Console.WriteLine($"Progress: {e.Percentage} %");

    String downloadUrl = await task;
    cancionStorage = downloadUrl;

}

```

Ambos son métodos asíncronos se van a ir ejecutando en segundo plano, en el caso de la canción, rescatamos la canción escogida y la subimos al storage a la carpeta de música, después vamos rellenando la barra de progreso conforme se vaya subiendo, y una vez subida, obtenemos el enlace de descarga. Para las carátulas es prácticamente lo mismo, cabe destacar que hacemos un `.Dispose()` al picture box de la carátula ya que si no le quitamos el recurso, no nos deja subirlo ya que está siendo utilizado por el picture box.

```

1 referencia
private void btnAceptar1_Click(object sender, EventArgs e)
{
    Guid UUID = Guid.NewGuid();
    String idCancion = UUID.ToString();
    String nombre = txtNombre.Text.ToString();
    String genero = this.listaIdGenero[cmbGenero.SelectedIndex].ToString();
    String artista = txtArtista.Text.ToString();
    String fecha = DateTime.Now.ToString("d");

    Cancion cancion = new Cancion(idCancion, nombre, cancionStorage, caratulaStorage, genero, artista, fecha);
    SetResponse res = cliente.Set(@"Cancion/" + idCancion, cancion);

    MessageBox.Show("Canción añadida");

    this.Close();
    frmGestionCanciones.cargarDatagrid();
}

```

Y finalmente para subir la canción, recogemos todos los valores de los campos y los subimos a realtime database de firebase.



En el menú de los usuarios comunes, quiero destacar tres cosas, arriba a la derecha se nos va a cargar la foto del usuario que actualmente esté conectado, a la izquierda se nos van a cargar las playlist del usuario y dentro del formulario se nos van a ir cargando los distintos formularios hijos.

Cargar la foto ha sido bastante sencillo, ya que desde el login le pasamos el usuario que se ha logueado.

```
2 referencias
public FrmMenuUser(Usuario usuarioLogueado)
{
    InitializeComponent();

    this.usuarioLogueado = usuarioLogueado;

    cargar();
    cargarLista();
    // MessageBox.Show(usuarioLogueado.foto);
    //picFoto.ImageLocation = "";
    var request = WebRequest.Create(usuarioLogueado.foto);
    using (var response = request.GetResponse())
    using (var stream = response.GetResponseStream())
    {
        picFoto.Image = Bitmap.FromStream(stream);
    }

    cargarFormulario();
}
```

Y con este método nos bajamos la imagen de firebase y la metemos en el picturebox.

Cargar las playlist sí que es algo más complejo ya que tenemos que hacer consultas a firebase.

```
2 referencias
public void cargarLista()
{
    FirebaseResponse res = cliente.Get(@"Playlist");
    Dictionary<string, Playlist> data = JsonConvert.DeserializeObject<Dictionary<string, Playlist>>(res.Body.ToString());
    llenarLista(data);
}

1 referencia
private void llenarLista(Dictionary<string, Playlist> data)
{
    lstPlaylist.Clear();
    lstPlaylist.Columns.Clear();

    lstPlaylist.Columns.Add("Nombre", 75);
    lstPlaylist.Columns.Add("Id", 0);

    foreach (var item in data)
    {
        if (item.Value.usuario == usuarioLogueado.nombre)
        {
            String[] row = { item.Value.nombre, item.Value.idPlaylist };
            var itemListView = new ListViewItem(row);
            lstPlaylist.Items.Add(itemListView);
            listaIdPlaylist.Add(item.Value.idPlaylist);
        }
    }
}
```

Hacemos una consulta a la tabla playlist y nos guardamos en una variable toda la tabla, y le decimos que, si el usuario propietario de la playlist es el mismo que el usuario logueado, se nos guarde en la lista.

Al tanto al cargar una playlist, como al seleccionar un género musical, nos llevará a una lista de canciones. En el caso de las playlist, hacemos dos consultas a firebase.

```
1 referencia
private void cargarLista()
{
    FirebaseResponse res = cliente.Get(@"CancionPlaylist");
    Dictionary<string, CancionPlaylist> data = JsonConvert.DeserializeObject<Dictionary<string, CancionPlaylist>>(res.Body.ToString());
    llenarLista(data);
}
```

En la primera consulta, cogemos todas las canciones que pertenecen a una playlist.

```
1 referencia
private void llenarLista(Dictionary<string, CancionPlaylist> data)
{
    foreach (var item in data)
    {
        if (item.Value.playlist == playlist.idPlaylist)
        {
            listaCanciones.Add(item.Value.cancion);
        }
    }

    lstCancion.Clear();
    lstCancion.Columns.Clear();

    lstCancion.Columns.Add("Nombre", 150);
    lstCancion.Columns.Add("Artista", 150);
    lstCancion.Columns.Add("Id", 0);

    FirebaseResponse res = cliente.Get(@"Cancion");
    Dictionary<string, Cancion> data2 = JsonConvert.DeserializeObject<Dictionary<string, Cancion>>(res.Body.ToString());

    foreach (var item in data2)
    {
        for (int i = 0; i < listaCanciones.Count; i++)
        {
            if (item.Value.idCancion == listaCanciones[i].ToString())
            {
                String[] row = { item.Value.nombre, item.Value.artista, item.Value.idCancion };
                var itemListView = new ListViewItem(row);
                lstCancion.Items.Add(itemListView);
            }
        }
    }
}
```

Y nos quedamos con las que pertenecen a la playlist que hemos seleccionado. Despues, hacemos una consulta a la tabla canciones y nos quedamos con las que su id sea el mismo que las anteriores.

En el caso de la lista de canciones por género.

```
foreach (var item in data)
{
    if (item.Value.genero == idGenero)
    {
        listaIdCanciones.Add(item.Value.idCancion);

        Cancion c = new Cancion(item.Value.idCancion, item.Value.nombre, item.Value.cancion, item.Value.caratula, item.Value.genero, item.Value.url);
        listaCanciones.Add(c);

        WebClient wc = new WebClient();
        byte[] bytes = wc.DownloadData(item.Value.caratula);
        MemoryStream ms = new MemoryStream(bytes);
        System.Drawing.Image img = System.Drawing.Image.FromStream(ms);
        ms.Dispose();

        this.imageList.Images.Add(img);
    }
}

this.imageList.ImageSize = new Size(100, 100);
this.lstCancion.View = View.Details;
lstCancion.Columns.Add("", 300);

lstCancion.SmallImageList = imageList;
int cont = 0;
foreach (var item in data)
{
    if (item.Value.genero == idGenero)
    {
        lstCancion.Items.Add(item.Value.nombre, cont);
        cont++;
    }
}
```

Solo hacemos una consulta a la tabla canciones y nos quedamos con la que el género sea el mismo que el elegido. Despues añadimos el id de la canción a un array list y guardamos las carátulas en un image list.

Por ultimo, volvemos a recorrer la lista de canciones que su género es el elegido y metemos su nombre junto con su foto en la lista.

Quiero destacar que en esta ventana hay una text box para buscar y su funcionamiento es el mismo que el de cargar, pero ademas de filtrar canciones por el género seleccionado, se fija tambien en si su nombre contiene las letras que se hayan escrito en el buscar.

```
int cont = 0;
foreach (var item in data)
{
    if (item.Value.genero == idGenero && item.Value.nombre.Contains(texto))
    {
        lstCancion.Items.Add(item.Value.nombre, cont);
        cont++;
    }
}
```

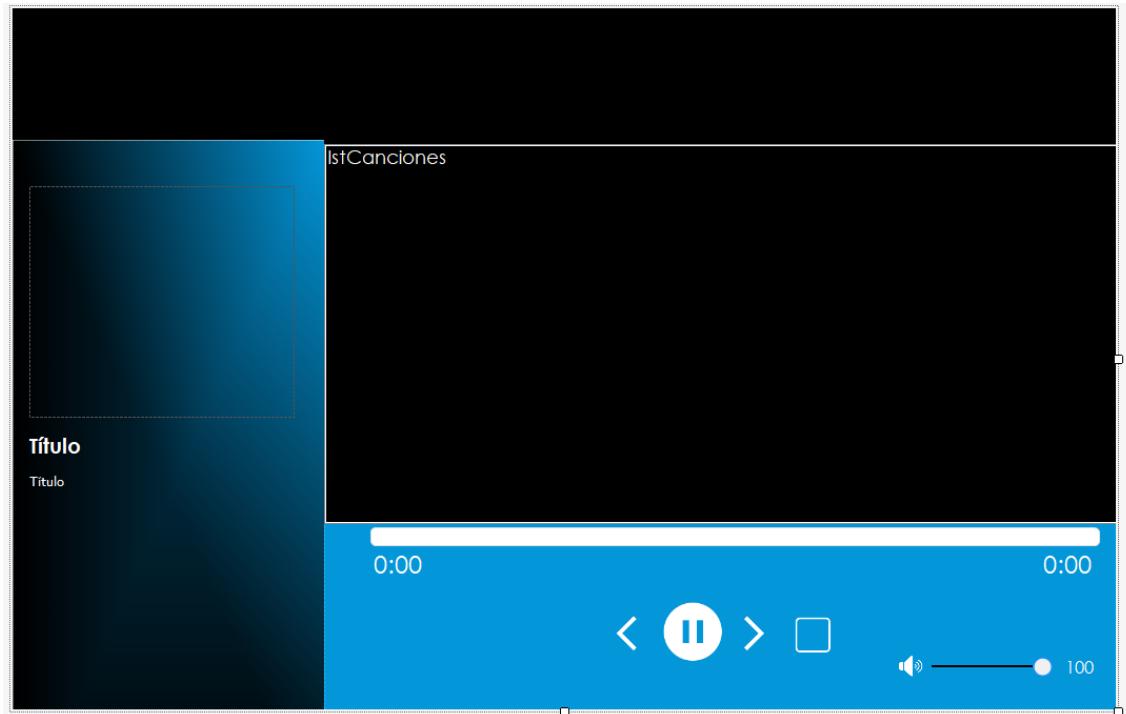
Si hacemos doble clic alguna canción de la lista, recuperamos el índice en el que está esa canción y se lo mandamos al reproductor junto con la lista de canciones entera.

```
1 referencia
private void lstCancion_DoubleClick(object sender, EventArgs e)
{
    recuperarCancion();
    FrmReproductorMusica frmReproducirMusica = new FrmReproductorMusica(listaCanciones, index);
    frmReproducirMusica.MdiParent = frmMenuUser;
    frmReproducirMusica.Dock = DockStyle.Fill;
    frmReproducirMusica.Show();
}

1 referencia
private void ...recuperarCancion()
{
    int fila = lstCancion.FocusedItem.Index;
    index = fila;
    /*
    String idCancion = listaIdCanciones[fila].ToString();

    FirebaseResponse res = cliente.Get(@"Cancion/" + idCancion);
    Cancion cancion = res.ResultAs<Cancion>();*/
}
```

El reproductor tiene este diseño.



Cuando se carga el reproductor, se carga la lista de canciones que le llega en la lista.

```

1 referencia
private void anadirLista(ArrayList listaCanciones)
{
    for (int i = 0; i < listaCanciones.Count; i++)
    {
        Cancion c = (Cancion)listaCanciones[i];
        lstCanciones.Items.Add(c.nombre);
    }
}

```

Recorremos el array y lo vamos metiendo a la lista.

```

4 referencias
private void startPlayer(int i)
{
    Cancion c = (Cancion)listaCanciones[i];
    player.URL = c.cancion;
    player.Ctlcontrols.next();
    player.Ctlcontrols.play();

    var request = WebRequest.Create(c.caratula);
    using (var response = request.GetResponse())
    using (var stream = response.GetResponseStream())
    {
        picCaratula.Image = Bitmap.FromStream(stream);
    }

    lblArtista.Text = c.artista;
    lblTitulo.Text = c.nombre;
}

```

Y también llamamos al método startPlayer pasándole el índice de la canción que previamente hemos seleccionado, para que así, aunque se carguen todas las canciones en la lista se reproduzca la seleccionada. También se cargan el artista y el título en sus respectivos labels y la foto de la carátula en el picturebox.

```

1 referencia
private void barSonido_ValueChanged(object sender, Utilities.BunifuSlider.BunifuScrollBar.ValueChangedEventArgs e)
{
    player.settings.volume = barSonido.Value;
    lblSonido.Text = barSonido.Value.ToString();
}

```

Cuando movemos la barra del sonido cambiamos también el sonido del reproductor.

```
2 referencias
private void stopPlayer()
{
    player.Ctlcontrols.stop();
}
```

Cuando queremos parar la canción llamamos al método stop.

```
1 referencia
private void siguiente()
{
    //proCancion.Value = 0;
    if (startIndex == lstCanciones.Items.Count - 1)
    {
        //startIndex = lstCanciones.Items.Count - 1;

        startIndex = 0;
        lstCanciones.SelectedIndex = startIndex;
    }
    else if (startIndex < lstCanciones.Items.Count)
    {
        startIndex = startIndex + 1;
        lstCanciones.SelectedIndex = startIndex;
    }

    startPlayer(startIndex);
}
```

Cuando hacemos clic en siguiente, hacemos varias comprobaciones, si es la última canción de la playlist, pasa a la primera de la lista, y si no pasa a la siguiente.

El método de atrás sería parecido.

```

1 referencia
private void timCancion_Tick(object sender, EventArgs e)
{
    lblTiempo1.Text = player.Ctlcontrols.currentPositionString;
    lblTiempo2.Text = player.Ctlcontrols.currentItem.durationString.ToString();
    if (player.playState == WMPLib.WMPPlayState.wmppsPlaying)
    {
        proCancion.Value = (int)player.Ctlcontrols.currentPosition;
    }
}

1 referencia
private void player_PlayStateChange(object sender, AxWMPLib._WMPocxEvents_PlayStateChangeEvent e)
{
    if (player.playState == WMPLib.WMPPlayState.wmppsPlaying)
    {
        proCancion.MaximumValue = (int)player.Ctlcontrols.currentItem.duration;
        timCancion.Start();
    } else if (player.playState == WMPLib.WMPPlayState.wmppsPaused)
    {
        timCancion.Stop();
    } else if (player.playState == WMPLib.WMPPlayState.wmppsStopped)
    {
        timCancion.Stop();
        proCancion.Value = 0;
    }
}

```

Para la barra de progreso de la canción y para los labels del tiempo tendremos un timer. El timer se activa cuando el reproductor cambia de estado y hay una canción reproduciéndose. Al activarse el timer, en el label1 pone el tiempo actual de la canción y en el label2 pone el tiempo total de la canción, y si la canción se está reproduciendo la barra de progreso va avanzado. El valor máximo de la barra de progreso lo coge cuando empieza una nueva canción.

6. Testeo y pruebas de la solución

6.1 Plan de pruebas

Existen distintas técnicas de software testing que permiten verificar que el desarrollo cumple con los requerimientos explícitos definidos por el cliente. Es importante destacar que existen distintos tipos de pruebas que pueden ser aplicadas para detectar las desviaciones que afectan la calidad del entregable.

Una de las más esenciales son las pruebas funcionales, que se recomienda aplicar en las distintas etapas del desarrollo de software, con el fin de garantizar de forma temprana que el sistema cumple con la calidad y funcionalidad esperada por el usuario.

El testing funcional tiene el objetivo principal de revisar que el software funciona para lo que fue creado, y es esencial debido a que permite comprobar que los requisitos y definiciones en ventajas de las pruebas funcionales de software son:

- Permiten detectar tempranamente defectos y comportamientos inesperados en el software para lograr un producto libre de fallos.
- Aumentan la confianza de los usuarios, asegurando que el sistema se ejecute de la forma esperada.

- Ahorran costos y tiempo a la empresa, ya que mientras antes se encuentren los defectos, menor será el esfuerzo para corregirlos.
- El proceso de testing funcional considera todo tipo de flujos definidos tanto en el proceso de negocios, como en casos de uso y requisitos declarados por el usuario final.

¿Qué tipos de pruebas funcionales para software existen?

- **Pruebas unitarias:** Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeños test que validan el comportamiento de un objeto y la lógica.

Estos test suelen realizarse durante la fase de desarrollo de aplicaciones de software o móviles. Normalmente se llevan a cabo por los propios desarrolladores de la aplicación.

Con estas pruebas se detectan errores que no se podrían detectar hasta fases más avanzadas de pruebas. Por lo que realizar estas pruebas supone al final, un ahorro de tiempo.

Hay dos tipos de enfoques principales:

- Enfoque estructural o de caja blanca: Se verifica la estructura interna del componente con independencia de la funcionalidad establecida para el mismo.
- Enfoque funcional o de caja negra: Se comprueba el correcto funcionamiento de los componentes del sistema de información, analizando las entradas y salidas y verificando si el resultado es el esperado.

- **Pruebas de integración:** En estas pruebas se examinan las interfaces entre grupos de componentes o subsistemas para asegurar que son llamados cuando es necesario y que los datos que se transmiten son los correctos.

A menudo se combinan los tipos de prueba unitarias y de integración, hay dos tipos de pruebas de integración.

- Integración incremental: Se combina el siguiente componente que se debe probar con el conjunto de componentes que ya están probados y se va incrementando progresivamente el número de componentes a probar.

Hay tres estrategias de integración, de arriba abajo, de abajo a arriba y estrategias combinadas.

- Se prueba cada componente por separado y posteriormente se integran todos de una vez realizando las pruebas pertinentes.

- **Pruebas del sistema:** Las pruebas del sistema tienen como objetivo ejercitarse profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen.

En esta etapa se pueden distinguir los siguientes tipos de pruebas, cada una con un objetivo diferente:

- **Pruebas funcionales:** Dirigidas a asegurar que el sistema de información realiza correctamente todas las funciones que se han detallado.
 - **Pruebas de comunicaciones:** Determinan que las interfaces entre los componentes del sistema funcionan adecuadamente.
 - **Pruebas de rendimiento:** Consisten en determinar que los tiempos de respuesta están dentro de los intervalos establecidos.
 - **Pruebas de volumen:** Consisten en examinar el funcionamiento del sistema cuando está trabajando con grandes volúmenes de datos.
 - **Pruebas de sobrecarga:** Consisten en comprobar el funcionamiento del sistema en el umbral límite de los recursos, sometiéndolo a cargas masivas.
 - **Pruebas de disponibilidad de datos:** Consisten en demostrar que el sistema puede recuperarse ante fallos.
 - **Pruebas de facilidad de uso:** Consisten en comprobar la adaptabilidad del sistema a las necesidades de los usuarios.
 - **Pruebas de operación:** Consisten en comprobar la correcta implementación de los procedimientos de operación.
 - **Pruebas de entorno:** Consisten en verificar las interacciones del sistema con otros sistemas dentro del mismo entorno.
 - **Pruebas de seguridad:** Consisten en verificar los mecanismos de control de acceso al sistema para evitar alteraciones indebidas en los datos.
- **Pruebas de usuario:** Estas pruebas con un método de investigación cualitativa que se basan en la observación y el análisis de cómo un grupo determinado de usuarios utiliza nuestro producto.

Durante la prueba vamos a observar el grado de eficacia, eficiencia y satisfacción con el que los usuarios de nuestro producto logran concretar objetivos específicos.

Estas pruebas nos brindan beneficios significativos para nuestros proyectos, ya que nos permiten detectar problemas de usabilidad por muy bajo costo, ya que este tipo de estudios es mucho menos costoso que otros.

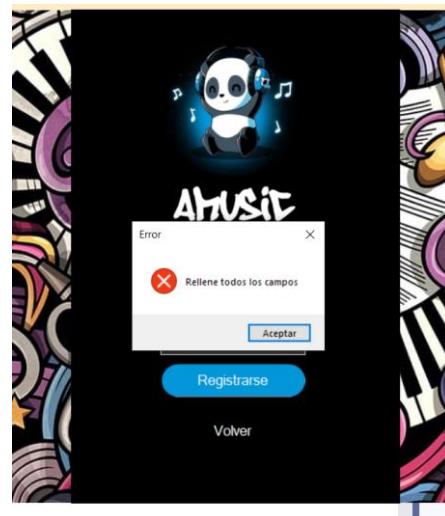
Y lo más importante nos ayudan a recordar que **nosotros no somos el usuario**.

A continuación, voy a mostrar las pruebas realizadas para el proyecto de AMusic.

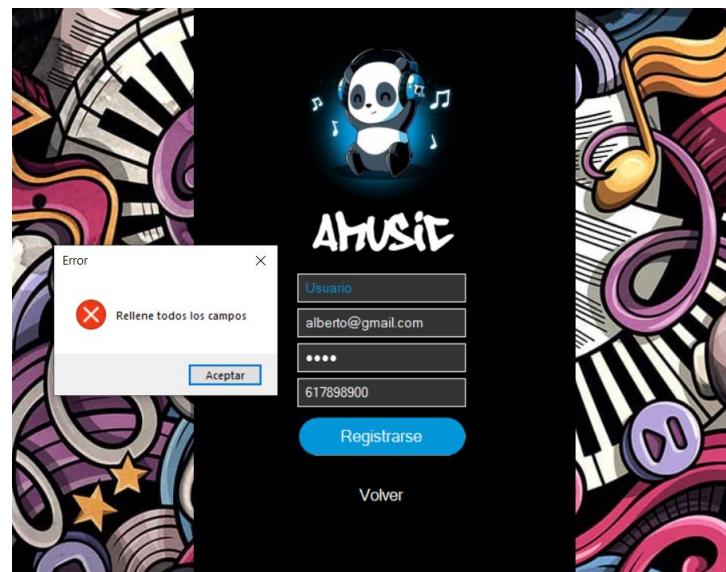
- Caso de prueba 1: Registro

En esta prueba comprobaremos que el usuario es capaz de registrarse y acceder a la aplicación desde el registro sin problemas, además también comprobaremos que no puede introducir datos incorrectos o nulos.

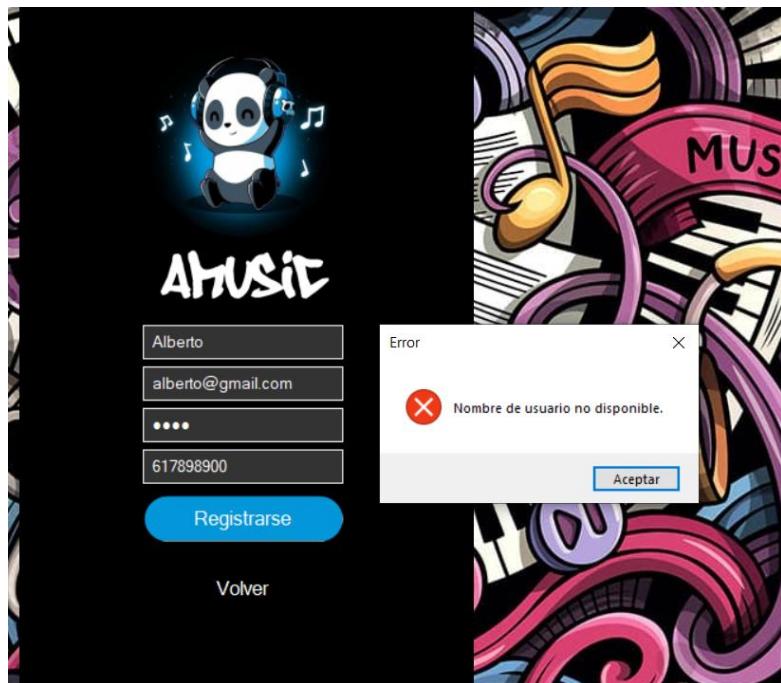
Hemos intentado registrarnos con todos los campos sin llenar y nos muestra un mensaje de error, **el resultado es el esperado**.



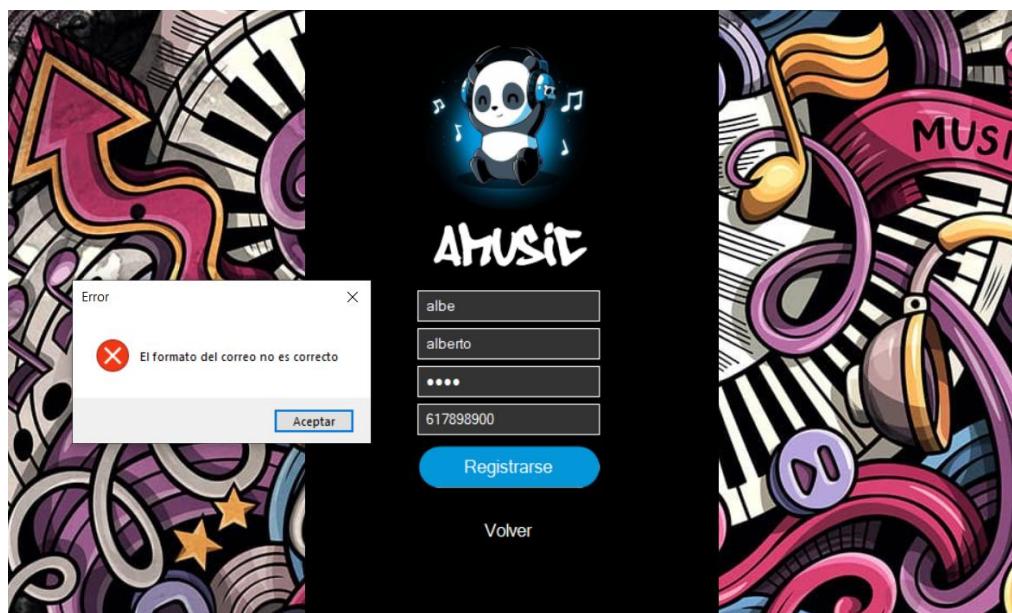
Hemos intentado registrarnos con algún campo sin llenar y nos muestra un mensaje de error, **el resultado es el esperado**.



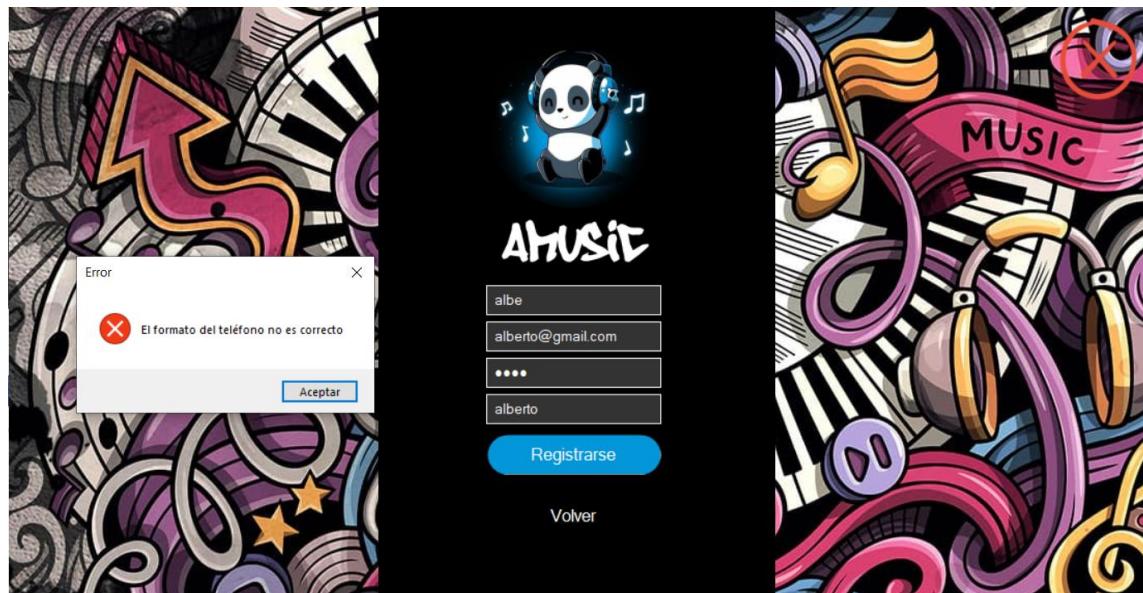
Hemos intentado registrarnos usando un nombre de usuario ya existente, no nos deja registrarnos y nos muestra un mensaje de error, **el resultado es el esperado**.



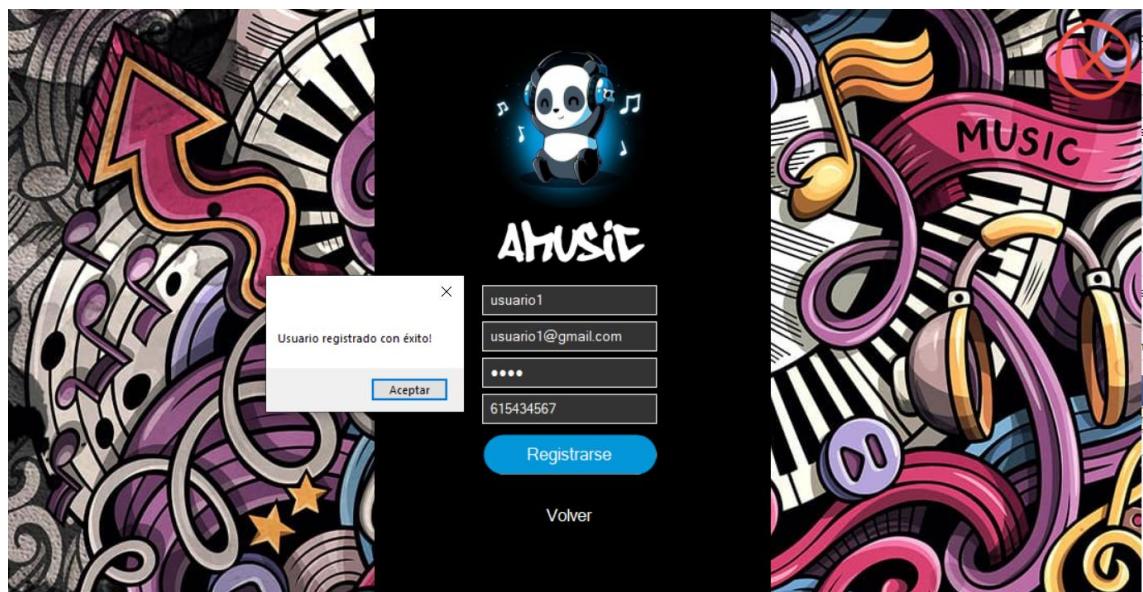
Hemos intentado registrarnos usando un correo electrónico no válido, no nos deja registrarnos y nos muestra un mensaje de error, **el resultado es el esperado**.



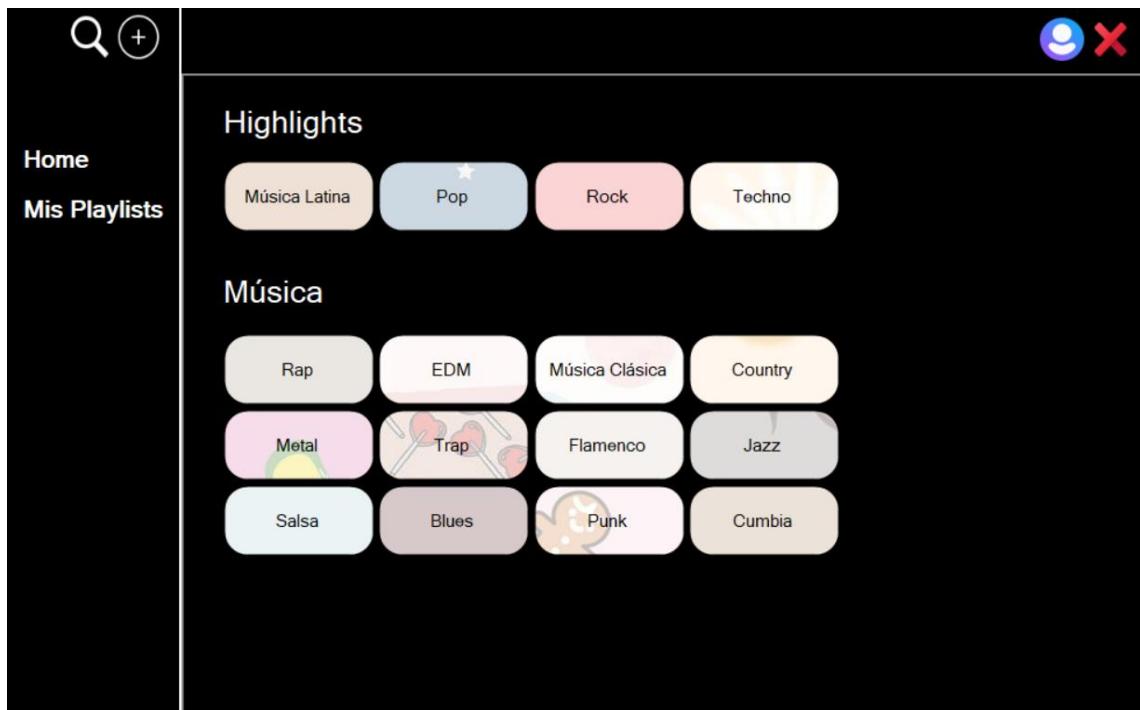
Hemos intentado registrarnos usando un teléfono no válido, no nos deja registrarnos y nos muestra un mensaje de error, **el resultado es el esperado**.



Hemos intentado registrarnos rellenando todos los campos y poniendo todos los datos correctos, nos registra en la base de datos y podemos acceder al menú principal de usuarios, **el resultado es el esperado**.



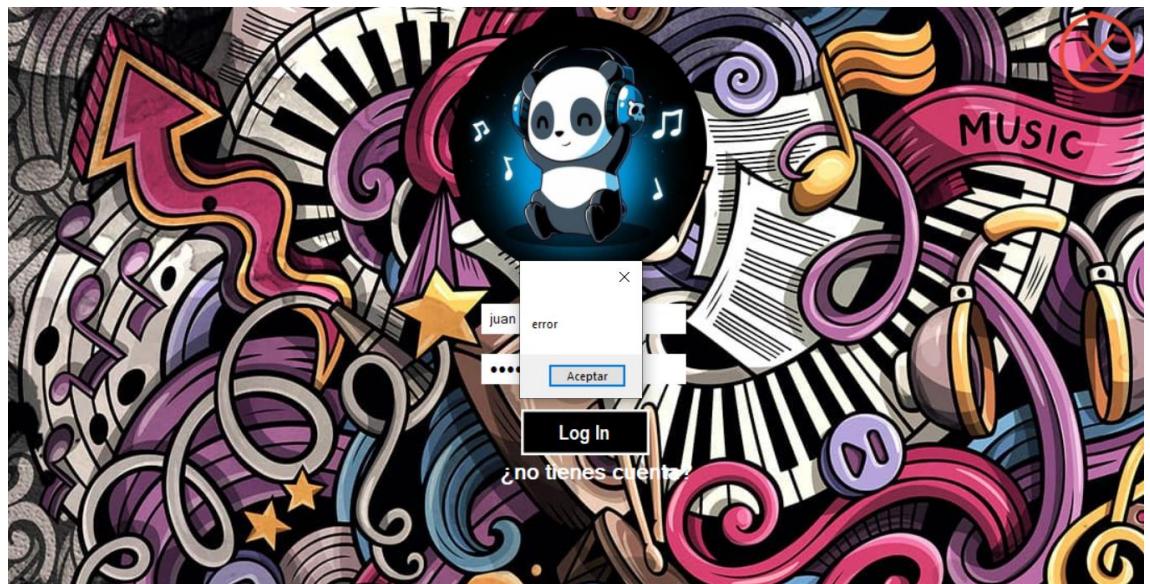
```
↳ — usuario1
    — correo: "usuario1@gmail.com"
    — foto: "https://firebasestorage.googleapis.com/v0/b/amusicprueba.appspot.com/o/Images%2FUsuarios%2Fperfil.png?alt=media&token=f73a063d-dc44"
    — nombre: "usuario1"
    — password: "holi"
    — telefono: "615434567"
    — tipo: false
```



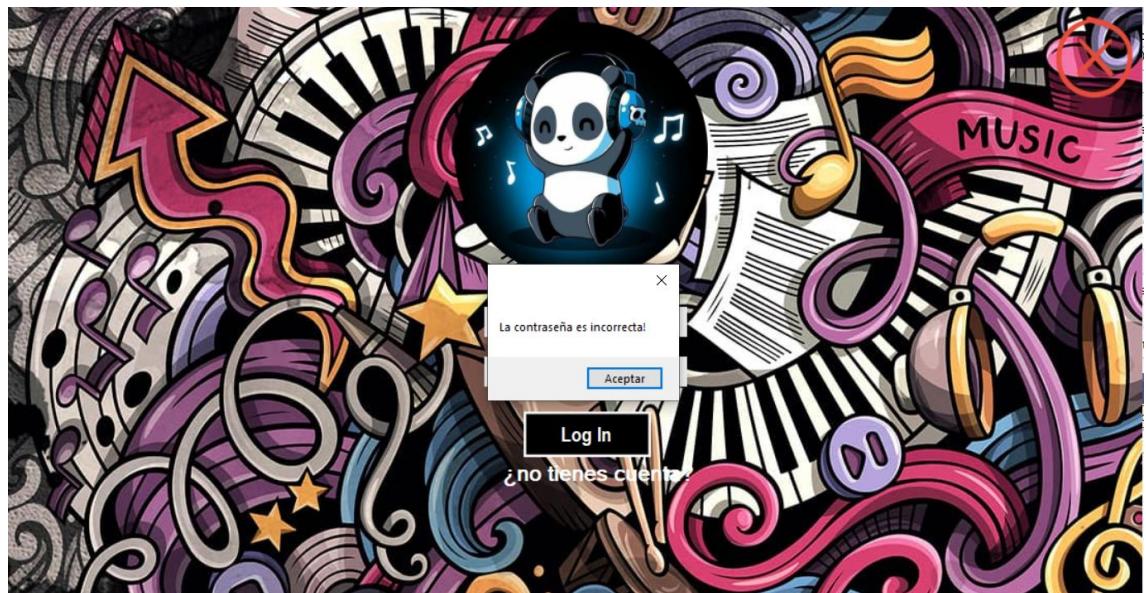
- *Caso de prueba 2: Login*

En esta prueba comprobaremos que el usuario que ya dispone de una cuenta es capaz de iniciar sesión sin problemas, además testearemos que pasa si se intenta iniciar sesión sin credenciales.

Hemos intentado iniciar sesión con un usuario que no existe y nos muestra un mensaje de error, **el resultado es el esperado**.



Hemos intentado iniciar sesión introduciendo mal la contraseña de un usuario que existe, nos salta un mensaje de error, **el resultado es el esperado**.



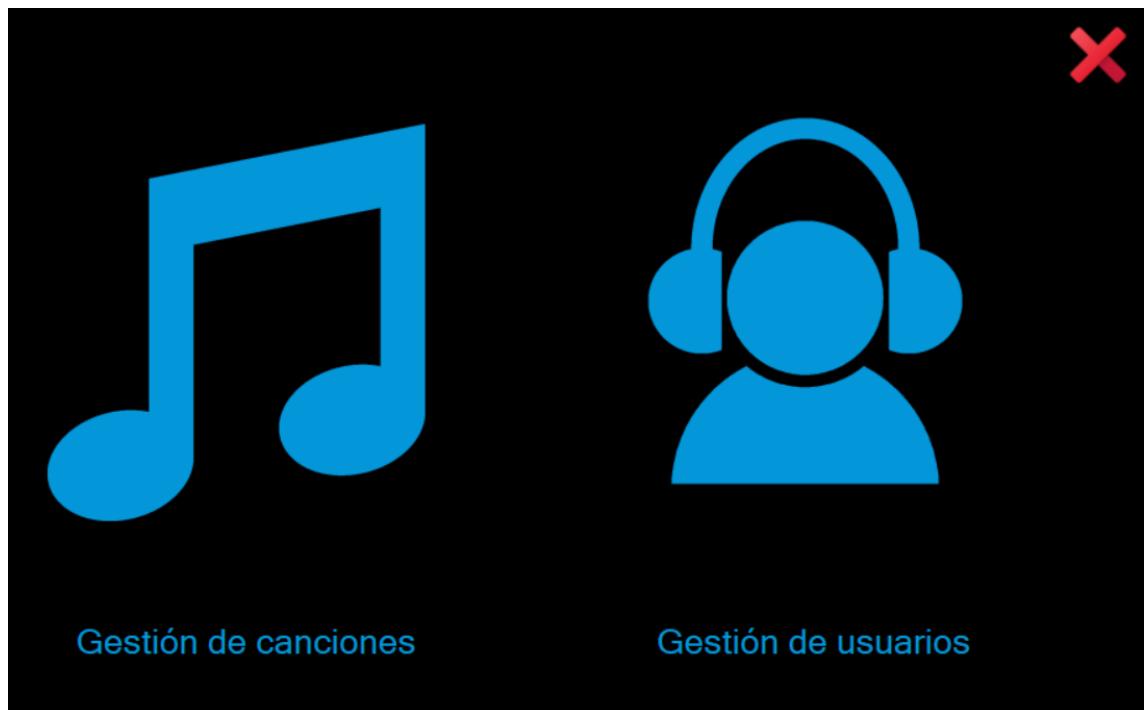
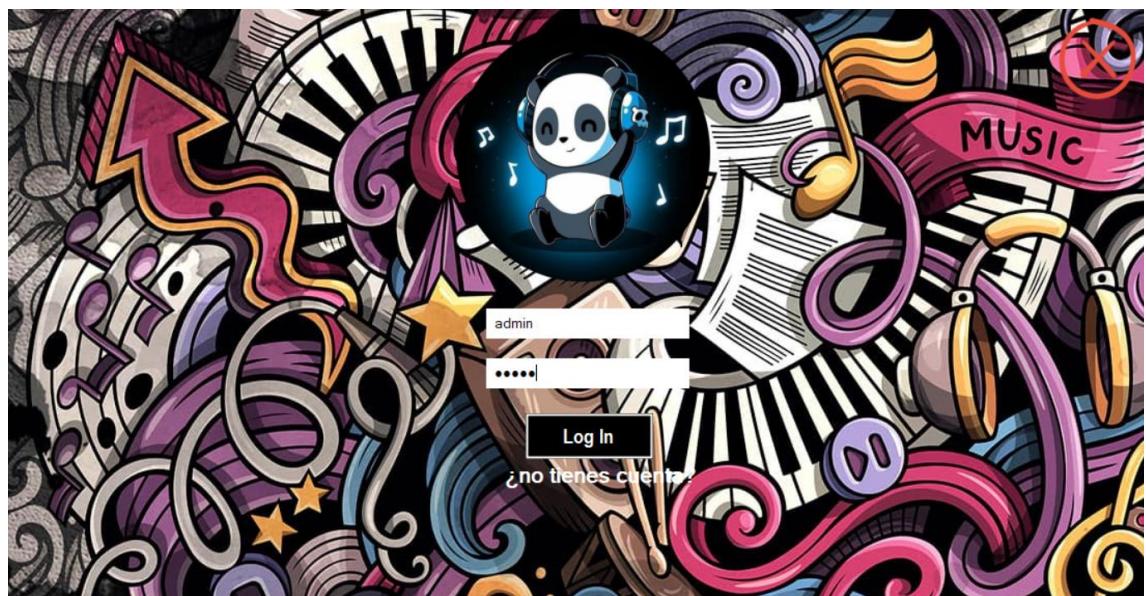
Hemos intentado iniciar sesión introduciendo bien todas las credenciales del usuario y nos inicia sesión, **el resultado es el esperado**.



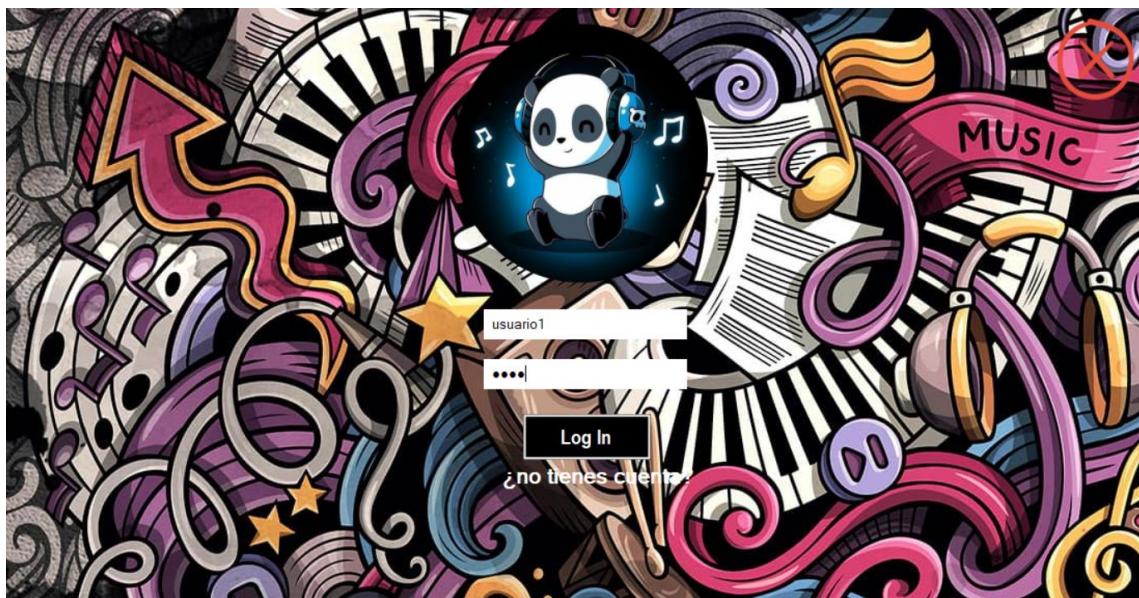
- Caso de prueba 3: Tipos de usuario

En esta prueba comprobaremos que se distinguen dos tipos de usuarios, los administradores y los comunes, y por tanto cuando iniciamos sesión nos llevará a sus respectivos menús.

Hemos intentado iniciar sesión con un usuario administrador y nos lleva al menú de los administradores, **el resultado es el esperado**.



Hemos intentado iniciar sesión con un usuario común y nos lleva al menú de los usuarios, el resultado es el esperado.



A mobile application interface. On the left, a sidebar shows 'Home' and 'Mis Playlists' with a search bar at the top. The main content area features a 'Highlights' section with four categories: 'Música Latina' (selected), 'Pop', 'Rock', and 'Techno'. Below this is a 'Música' section with a grid of nine genre cards: Rap, EDM, Música Clásica, Country, Metal, Trap, Flamenco, Jazz, Salsa, Blues, Punk, and Cumbia. The 'Metal' card is highlighted with a yellow sun icon. The 'Trap' card has a red 'X' icon. The 'Punk' card has a small orange circle icon.

- Caso de prueba 4: Cargar listas administradores

En esta prueba comprobaremos que desde el menú de los administradores se cargan tanto la lista de todas las canciones y de todos los usuarios.

Hemos intentado cargar la lista de los usuarios y se carga con éxito, **el resultado es el esperado**.

Usuario	Correo	Contraseña	Teléfono	Admin	Foto
Alberto	alberto@gmail.com	holi	678982939	False	https://firebasestorage.goo...
Alejandra	alejandra@gmail.com	holi	654333445	False	https://firebasestorage.goo...
Cristian	cristian@gmail.com	holi	654322245	False	https://firebasestorage.goo...
admin	admin@admin.com	admin	665554433	True	https://firebasestorage.goo...
holi	holi@mail.com	holi	4343434343	False	https://firebasestorage.goo...
prueba	prueba@gmail.com	holi	654443322	False	https://firebasestorage.goo...
usuario1	usuario1@gmail.com	holi	615434567	False	https://firebasestorage.goo...

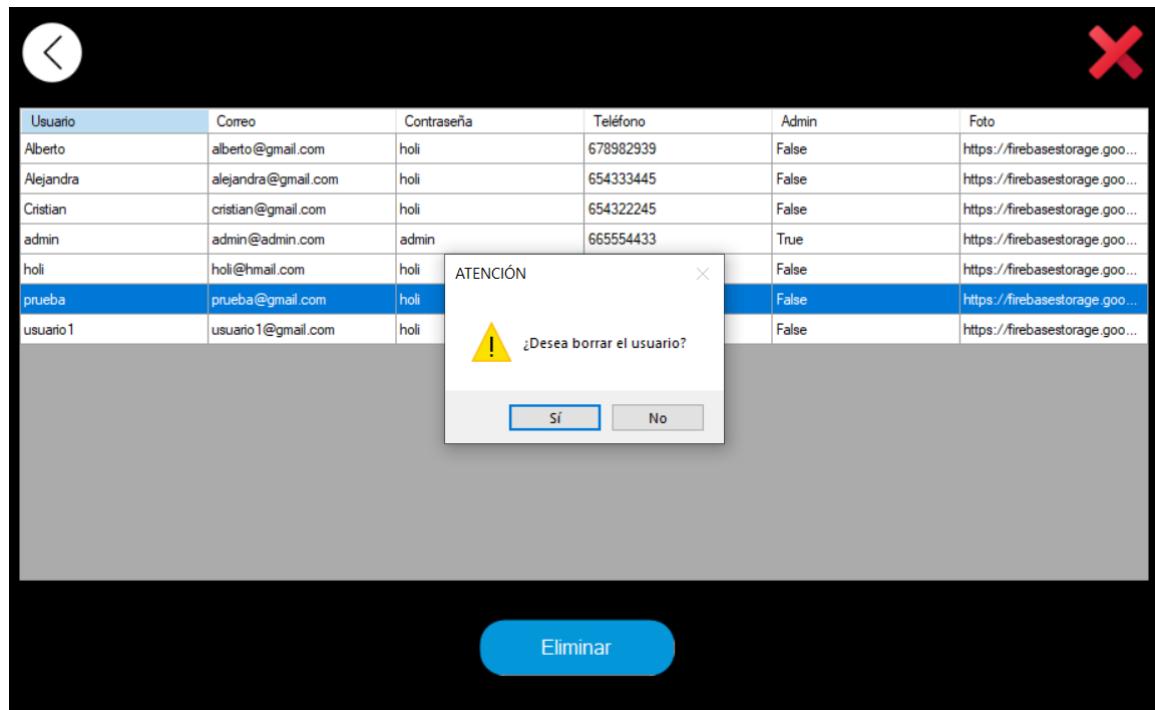
Hemos intentado cargar la lista de las canciones y se carga con éxito, **el resultado es el esperado**.

Nombre	Artista	Fecha	Género	Canción	Carátula
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Caminaré	Natos, Waor y Maka	08/06/2022	Flamenco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

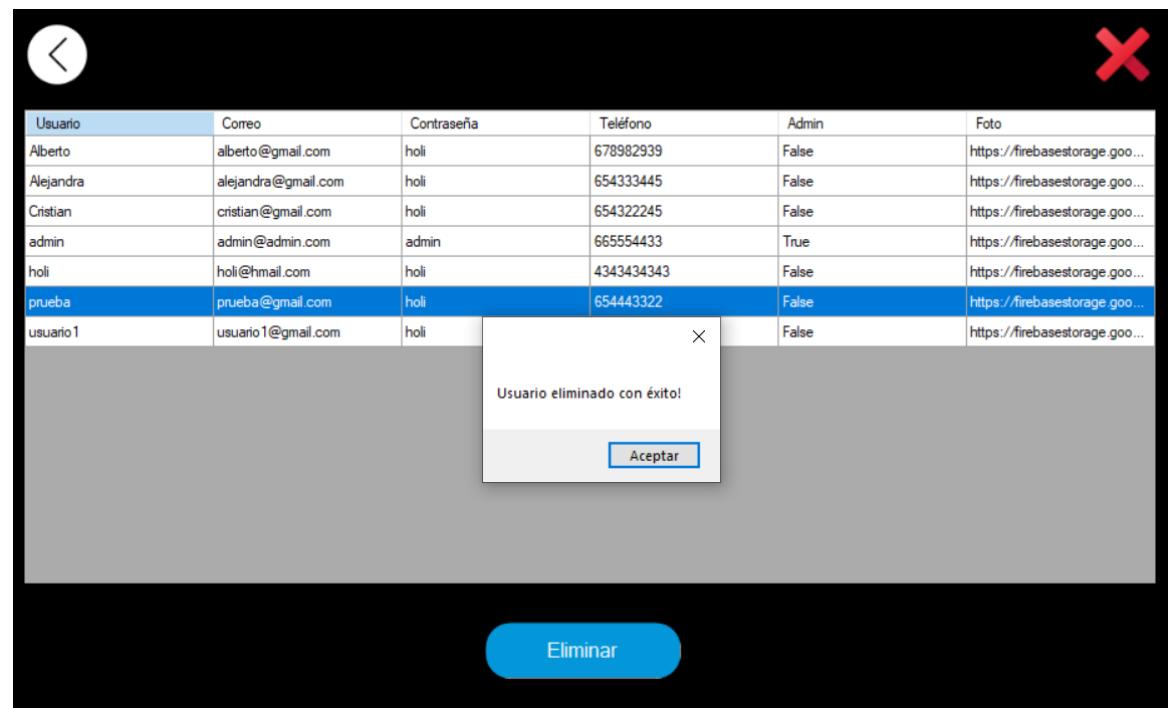
- Caso de prueba 5: Gestión de usuarios

En esta prueba comprobaremos que desde el menú de los administradores podremos gestionar todos los usuarios.

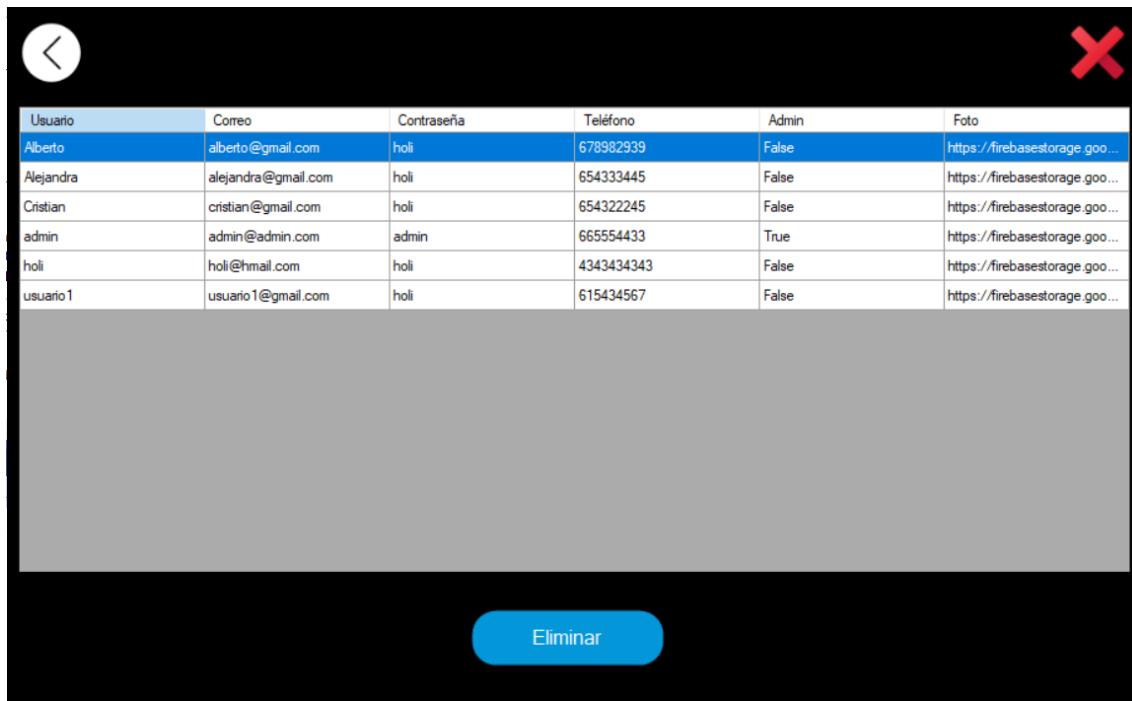
Hemos intentado eliminar un usuario y comprobamos que pregunta antes de eliminar, el resultado es el esperado.



Hemos seleccionado la opción 'sí' y comprobamos que el usuario se elimina con éxito, el resultado es el esperado.



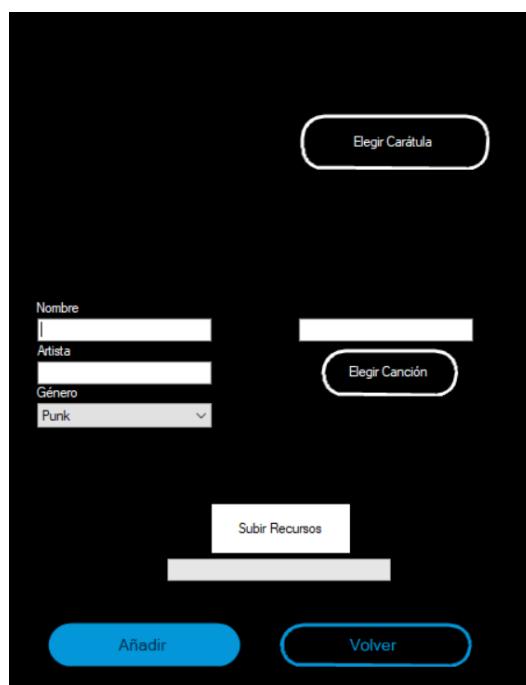
Queremos comprobar que al eliminar un usuario la lista de usuarios se recarga con éxito, el resultado es el esperado.



- *Caso de prueba 5: Gestión de canciones*

En esta prueba comprobaremos que desde el menú de los administradores podremos gestionar todas las canciones.

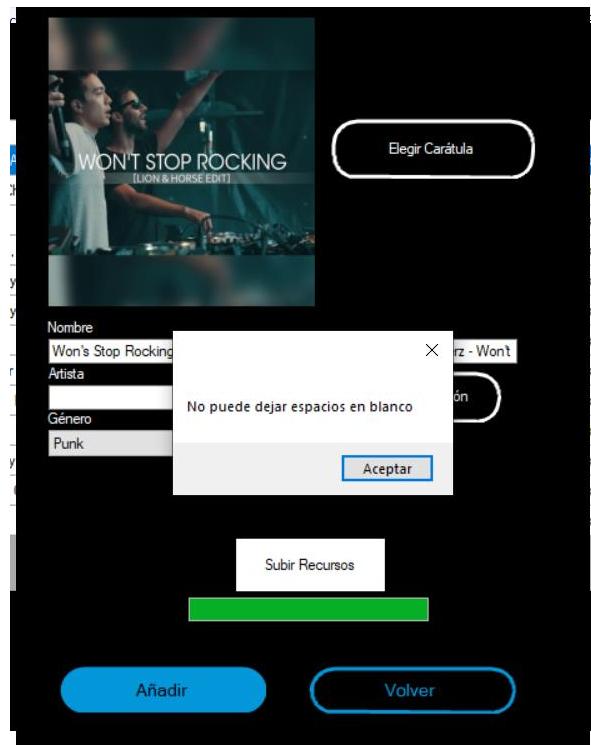
Hemos intentado añadir una canción y se nos abre una ventana para añadirla, el resultado es el esperado.



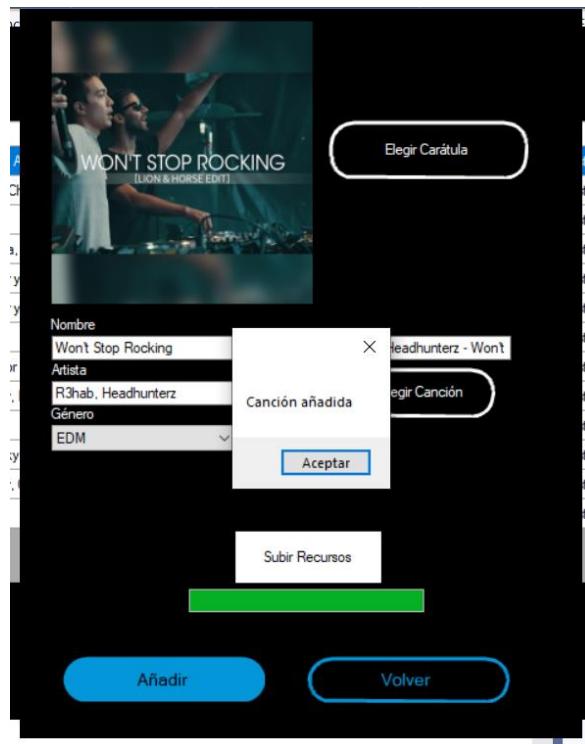
Hemos intentado añadir una canción haber subido los recursos antes, no nos deja, el resultado es el esperado.



Hemos intentado añadir una canción habiendo subido todos los recursos, pero sin llenar todos los campos, no nos deja, el resultado es el esperado.



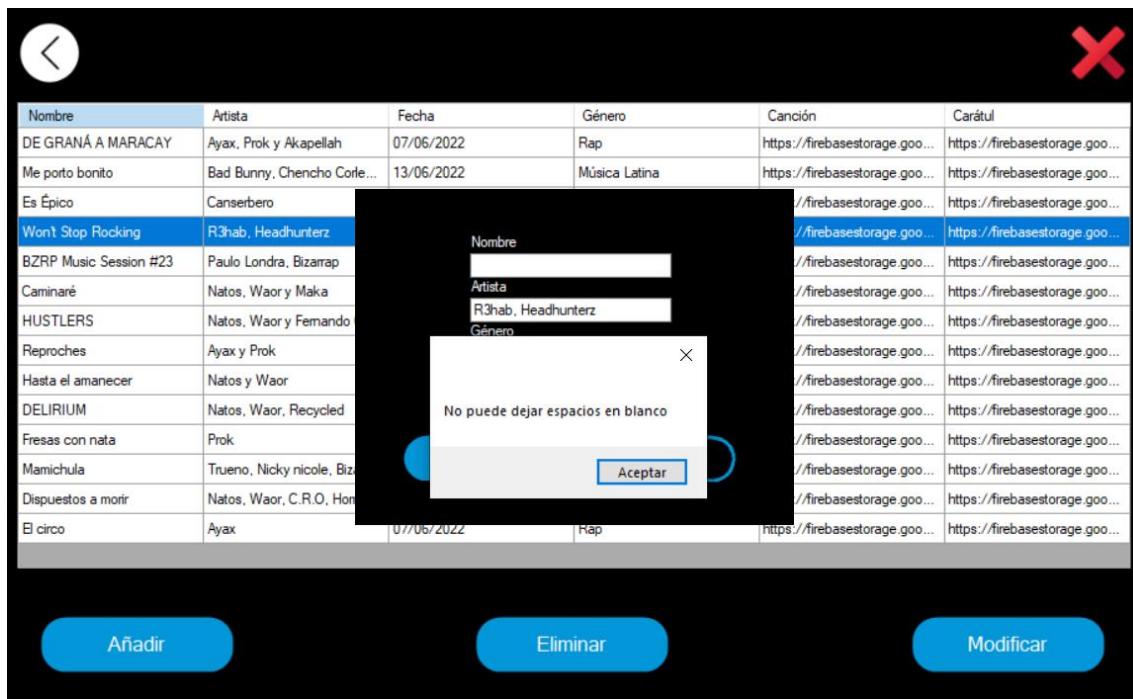
Hemos intentado añadir una canción rellenando todos los campos, la canción se añade, el resultado es el esperado.



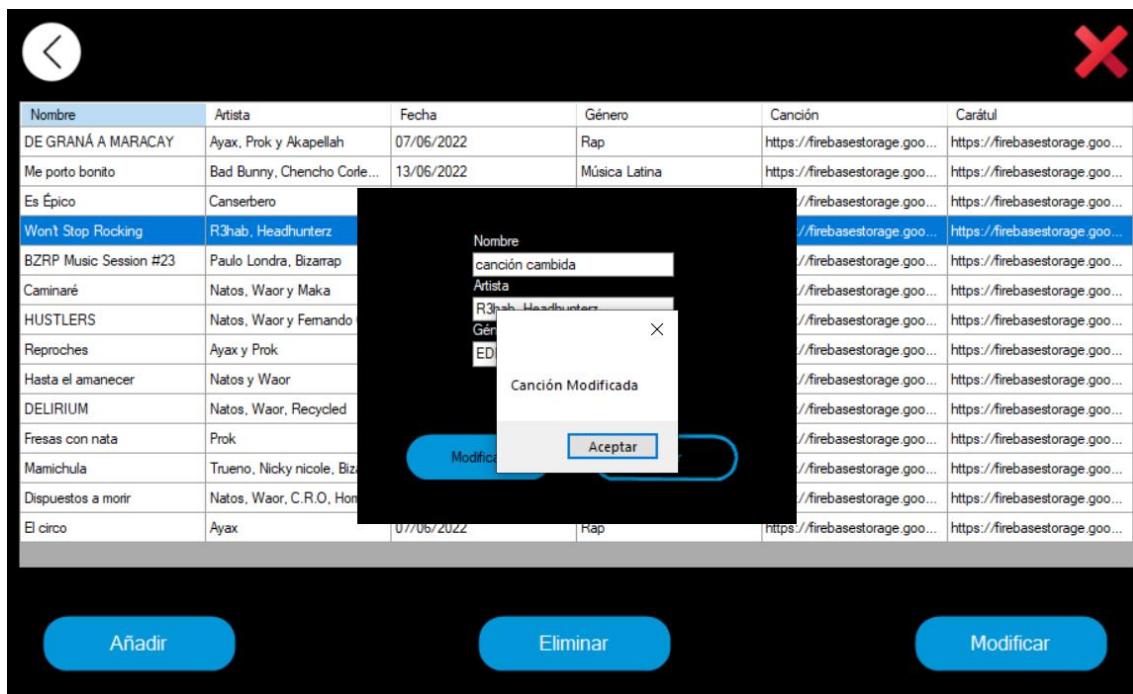
Queremos comprobar que cuando se añade una canción la lista se recarga, el resultado es el esperado.

Nombre	Artista	Fecha	Género	Canción	Carátula
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Won't Stop Rocking	R3hab, Headhunterz	19/06/2022	EDM	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Caminaré	Natos, Waory Maka	08/06/2022	Flamenco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waory Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizar...	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

Hemos intentado modificar una canción dejando espacios vacíos, no nos deja y no salta un mensaje de error, el resultado es el esperado.



Hemos intentado modificar una canción sin dejar ningún campo sin rellenar, el resultado es el esperado.



Queremos comprobar que cuando se modifica una canción la lista se recarga, el resultado es el esperado.

Nombre	Artista	Fecha	Género	Canción	Carátula
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
canción cambiada	R3hab, Headhunterz	19/06/2022	EDM	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Caminaré	Natos, Waor y Maka	08/06/2022	Flameco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

[Añadir](#) [Eliminar](#) [Modificar](#)

Hemos intentado eliminar una canción para que salte un mensaje de aviso, el resultado es el esperado.

Nombre	Artista	Fecha	Género	Canción	Carátula
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
canción cambiada	R3hab, Headhunterz	19/06/2022	EDM	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Caminaré	Natos, Waor y Maka	08/06/2022	Flameco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

[Añadir](#) Eliminar [Modificar](#)

Queremos que al darle a confirmar nos elimine la canción, el resultado es el esperado.

Nombre	Artista	Fecha	Género	Canción	Carátul
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
canción cambia	R3hab, Headhunterz	19/06/2022	EDM	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Camariné	Natos, Waor y Maka	08/06/2022	Flamenco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022		https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

Añadir **Eliminar** **Modificar**

Queremos comprobar que cuando se elimine una canción la lista se recargue, el resultado es el esperado.

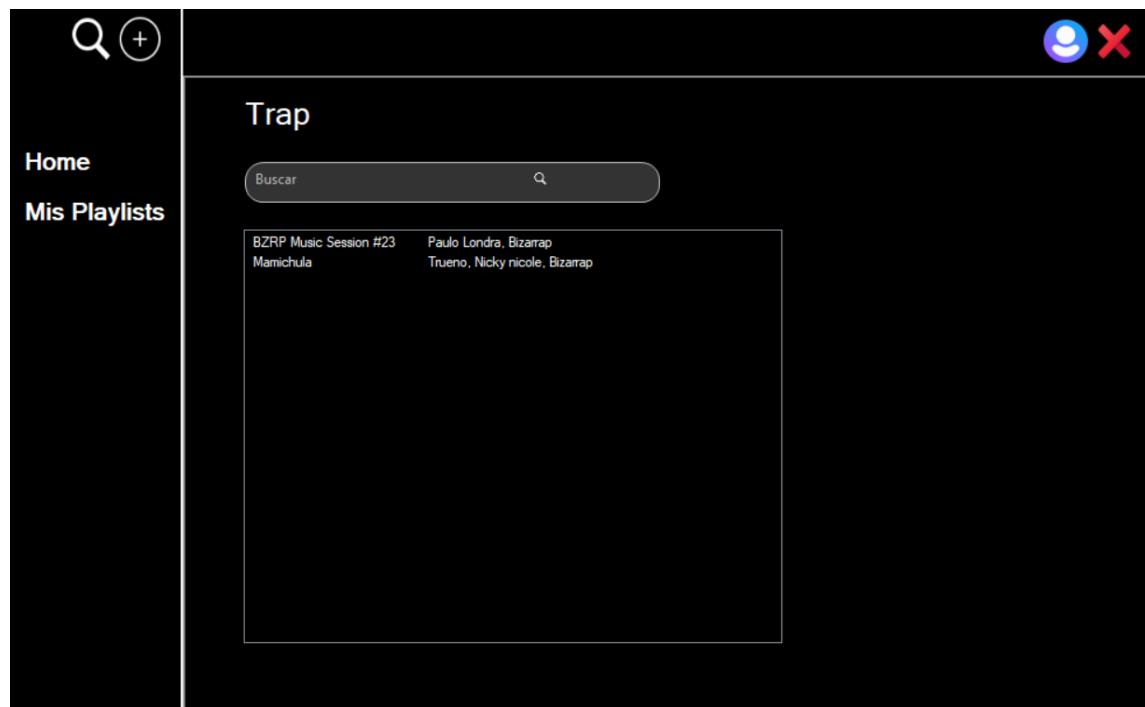
Nombre	Artista	Fecha	Género	Canción	Carátul
DE GRANÁ A MARACAY	Ayax, Prok y Akapellah	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Me porto bonito	Bad Bunny, Chencho Corle...	13/06/2022	Música Latina	https://firebasestorage.goo...	https://firebasestorage.goo...
Es Épico	Canserbero	02/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
BZRP Music Session #23	Paulo Londra, Bizarrap	07/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Camariné	Natos, Waor y Maka	08/06/2022	Flamenco	https://firebasestorage.goo...	https://firebasestorage.goo...
HUSTLERS	Natos, Waor y Fernando Co...	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Reproches	Ayax y Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Hasta el amanecer	Natos y Waor	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
DELIRIUM	Natos, Waor, Recycled	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Fresas con nata	Prok	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
Mamichula	Trueno, Nicky nicole, Bizarrap	02/06/2022	Trap	https://firebasestorage.goo...	https://firebasestorage.goo...
Dispuestos a morir	Natos, Waor, C.R.O, Homer...	18/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...
El circo	Ayax	07/06/2022	Rap	https://firebasestorage.goo...	https://firebasestorage.goo...

Añadir **Eliminar** **Modificar**

- Caso de prueba 6: Cargar canciones según su género

En esta prueba comprobaremos que, al hacer clic en los diferentes géneros, se nos cargarán todas las canciones que tenga ese género.

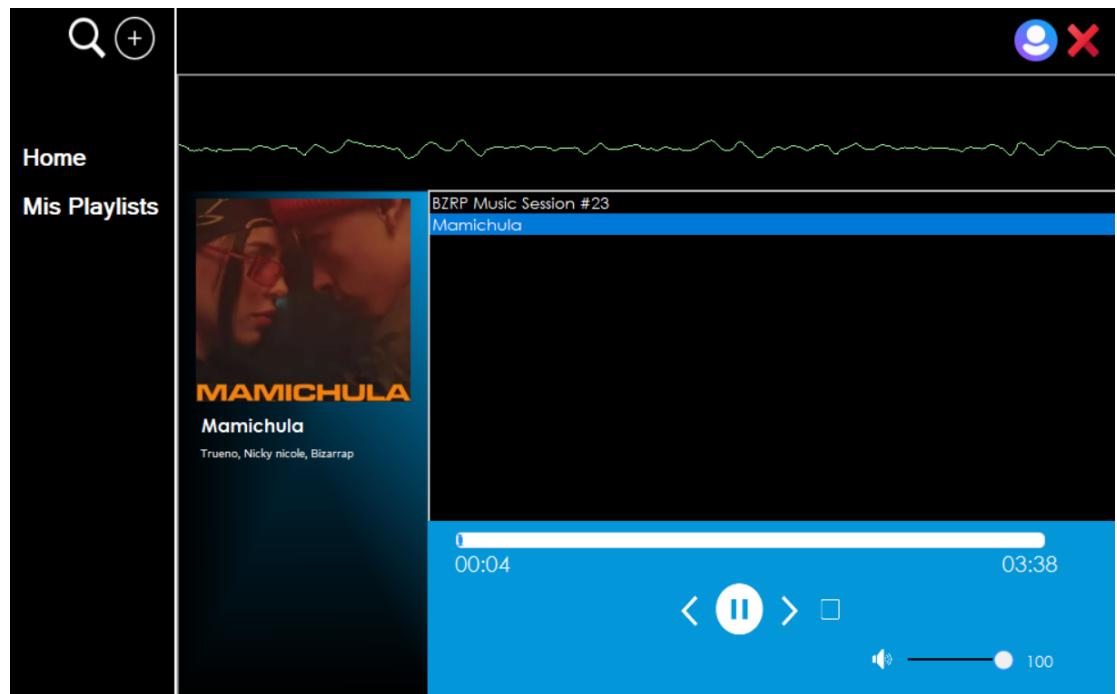
Hemos hecho clic en los diferentes géneros y se nos cargan sus canciones, **el resultado es el esperado**.



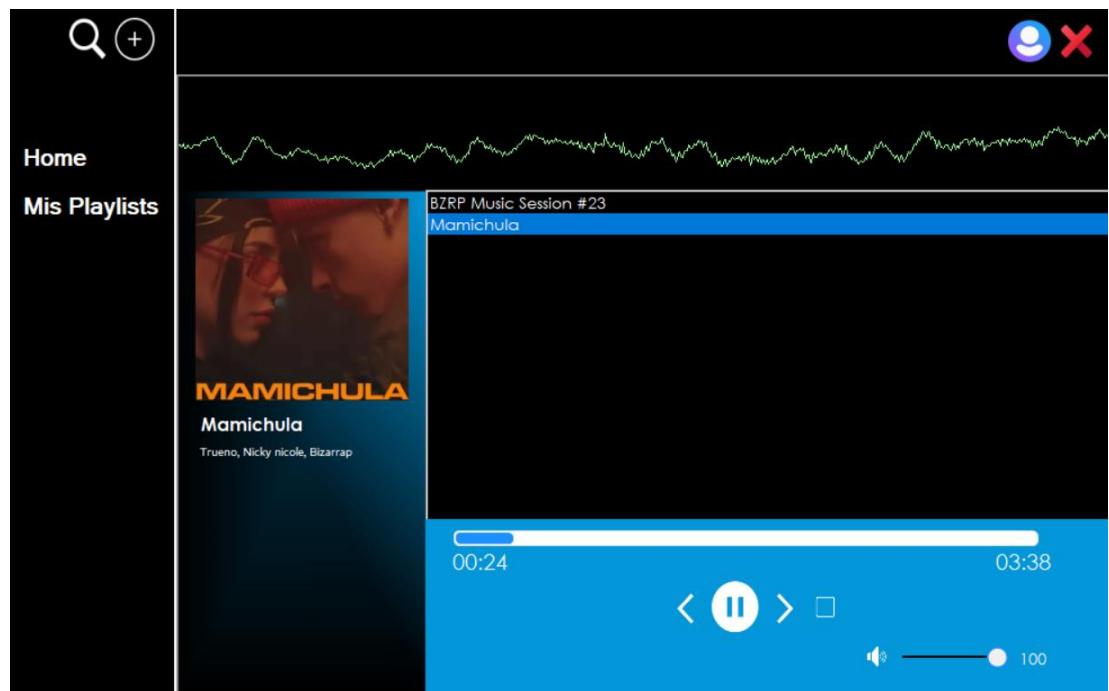
- Caso de prueba 7: Reproducir canciones por su género

En esta prueba comprobaremos que, al hacer clic una canción se nos reproduce esa canción y se nos cargan las demás en el reproductor.

Hemos hecho clic en la canción y se nos reproduce, **el resultado es el esperado**.



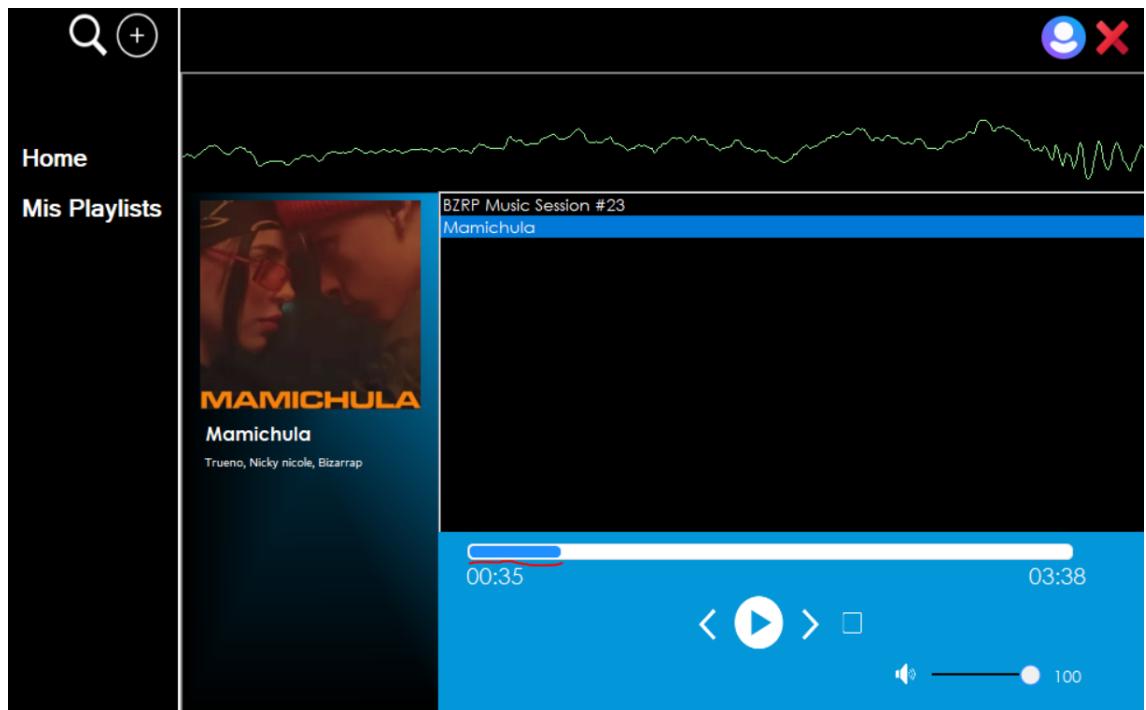
Hemos comprobado que al hacer clic en una canción además de reproducirse la canción seleccionada se cargan todas las canciones de ese género en el reproductor, **el resultado es el esperado**.



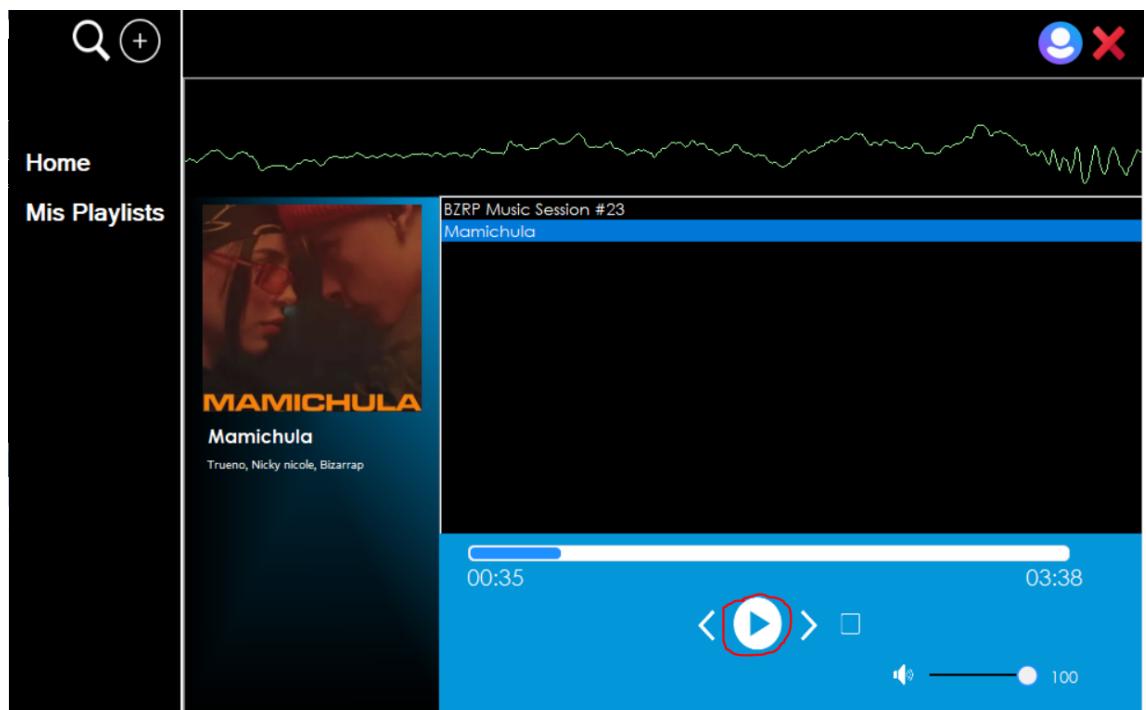
- Caso de prueba 8: Testear reproductor de música

En esta prueba comprobaremos que el reproductor de música funciona correctamente.

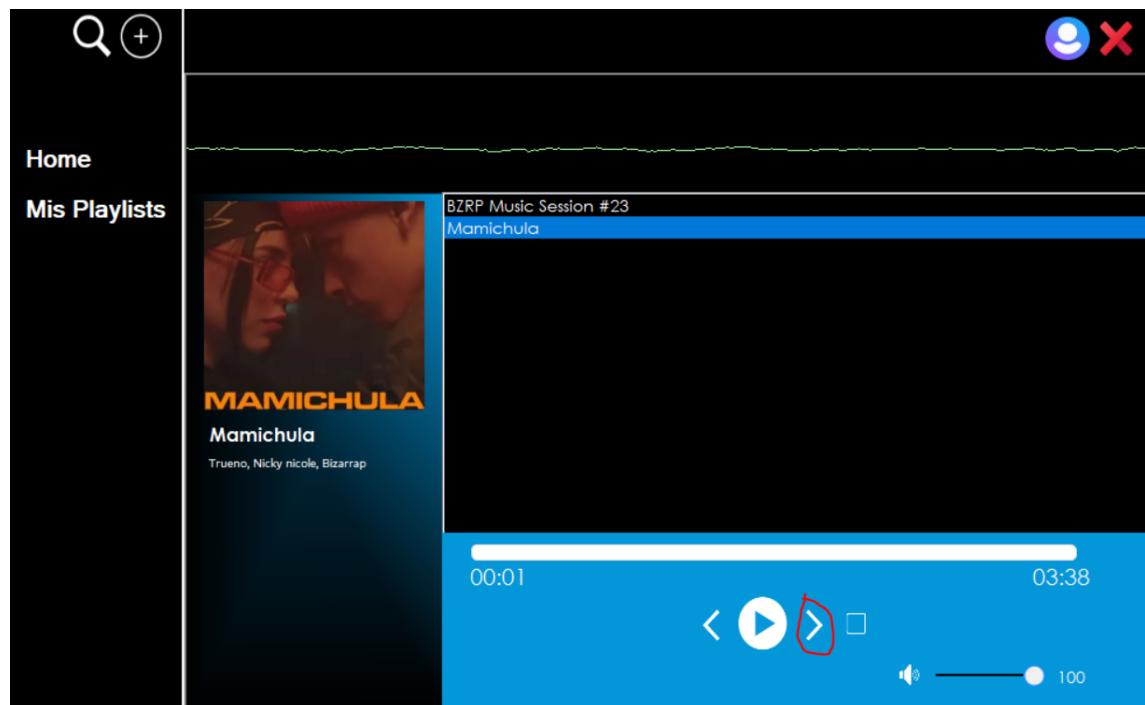
Hemos comprobado que la barra del tiempo va avanzando conforme se reproduce la canción, **el resultado es el esperado**.



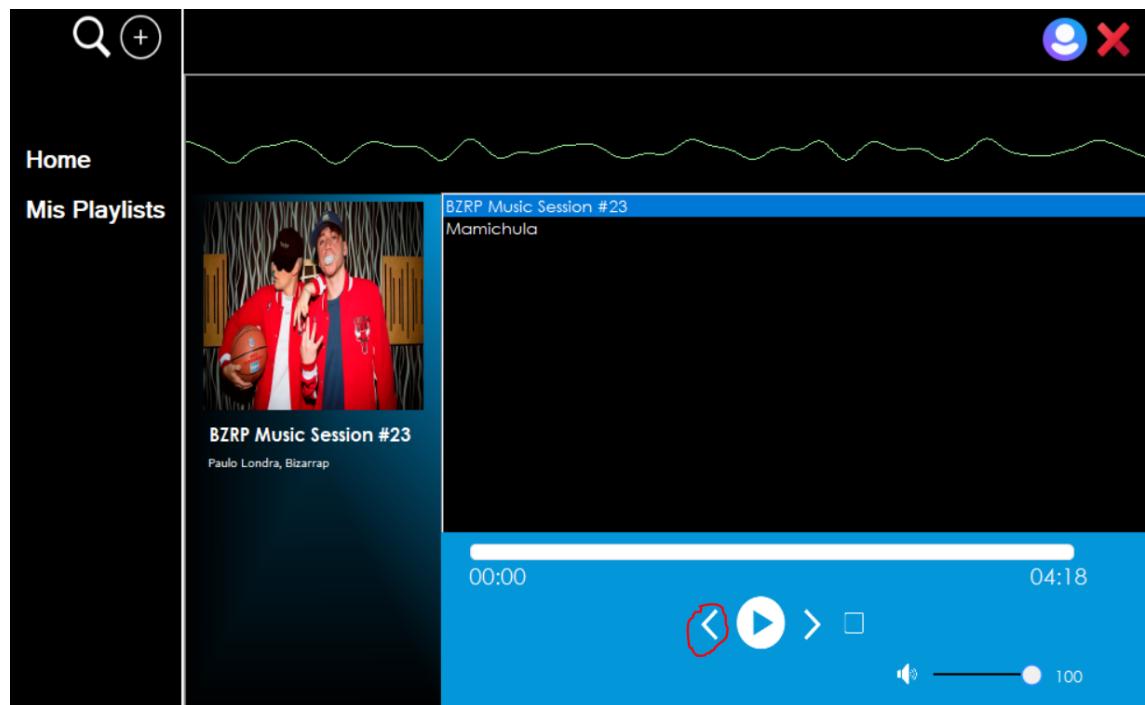
Hemos hecho clic el botón de pause y la canción se pone en pausa, **el resultado es el esperado**.



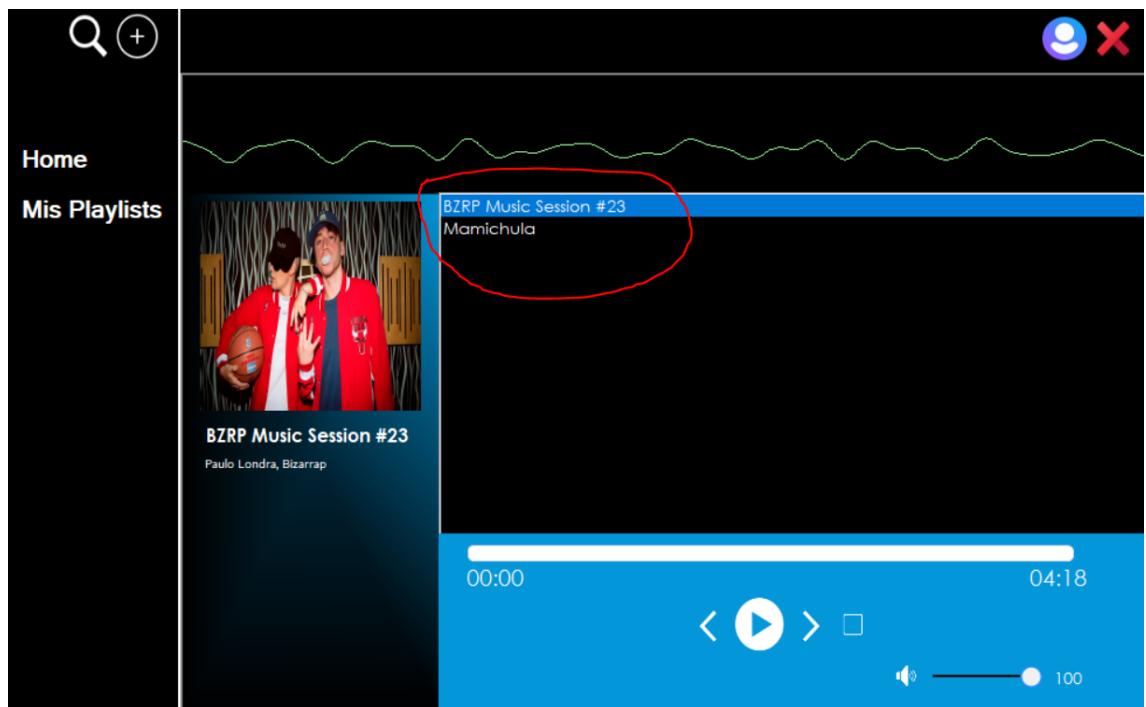
Hemos hecho clic el botón de siguiente canción y se reproduce la siguiente canción, el resultado es el esperado.



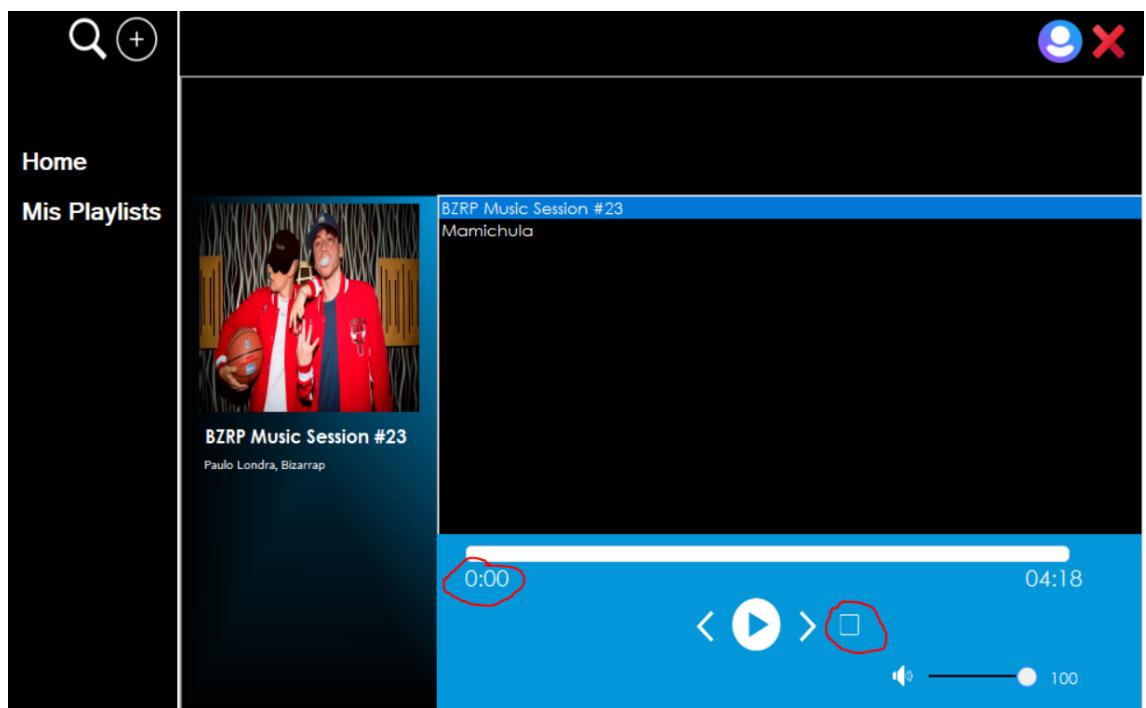
Hemos hecho clic el botón de anterior canción y se reproduce la canción anterior, el resultado es el esperado.



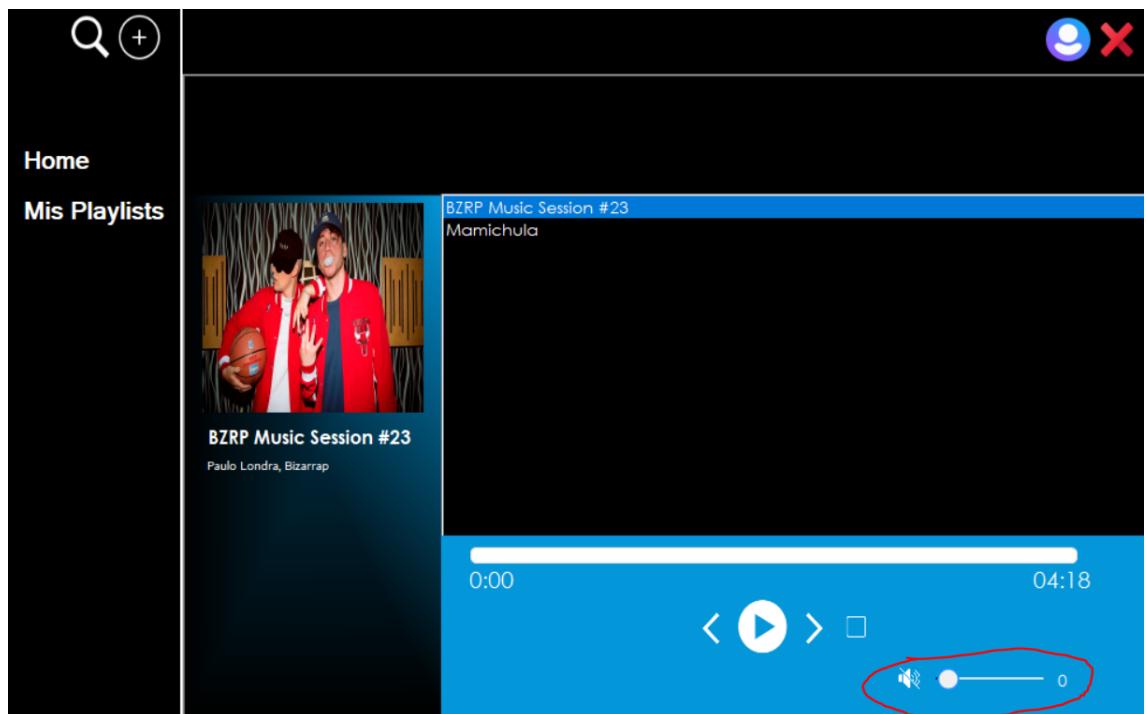
Hemos intentado cambiar de canción haciendo clic en otra canción de la lista de canciones, la canción se cambia, el resultado es el esperado.



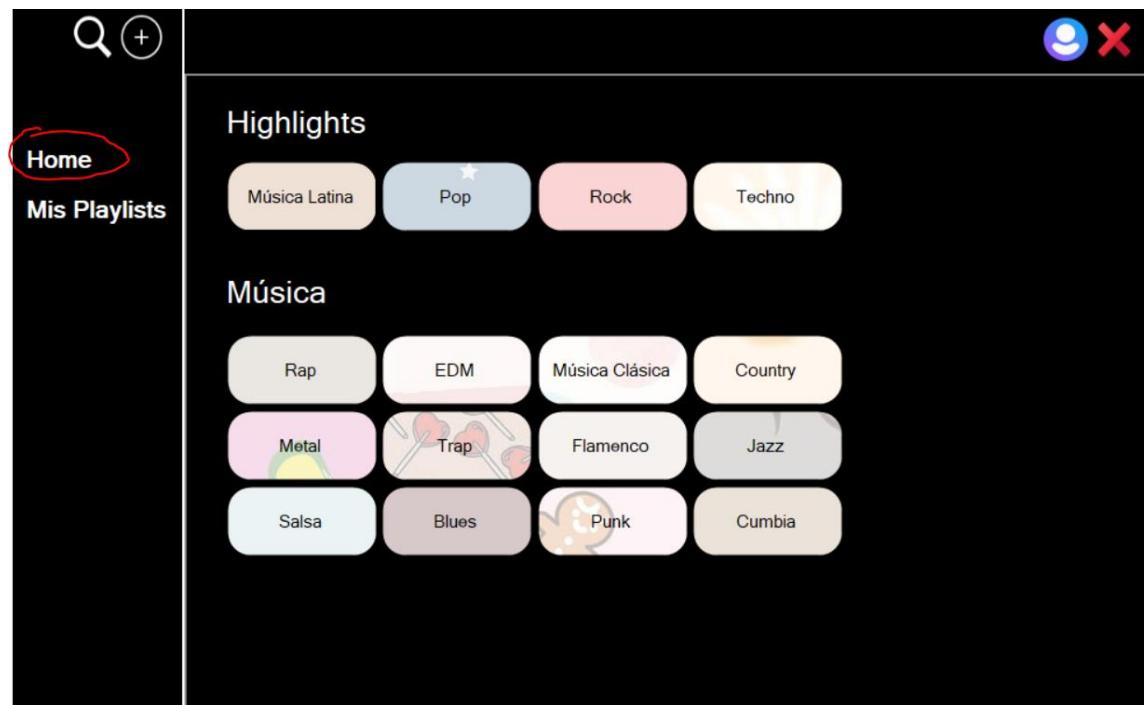
Hemos intentado parar la canción haciendo clic en el botón de stop, la canción se para y el tiempo se pone a 0:00, el resultado es el esperado.



Hemos intentado cambiar el volumen del sonido de la canción, el volumen se cambia y la imagen también, el resultado es el esperado.



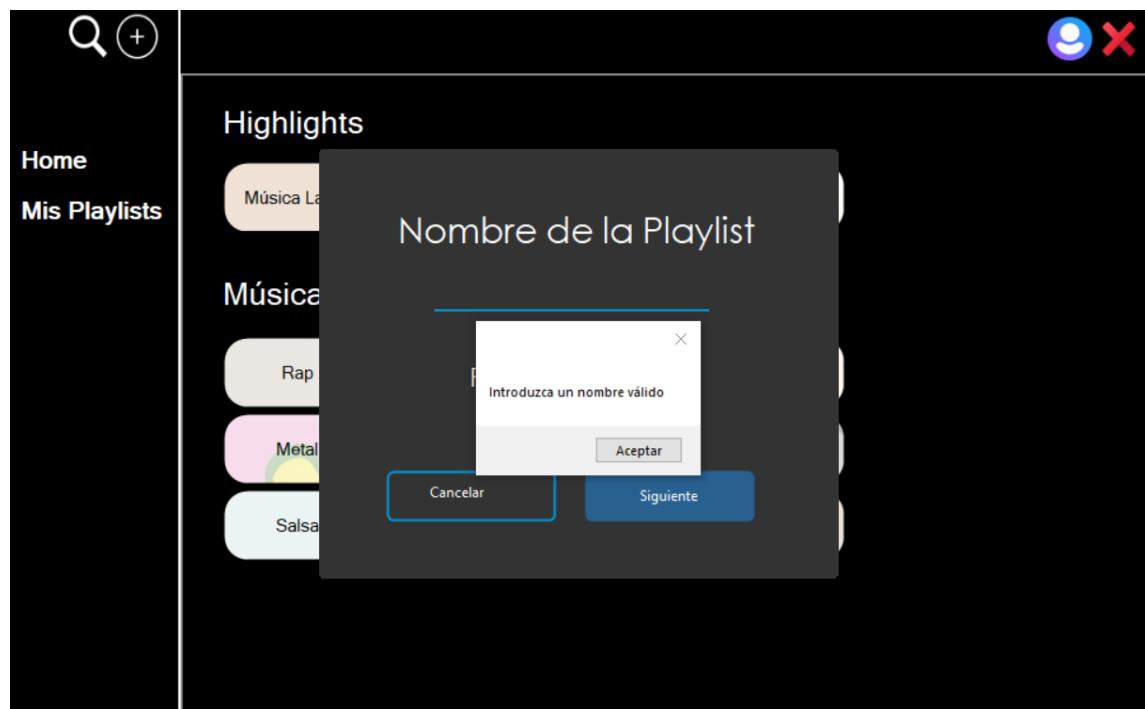
Hemos intentado irnos a la página principal y la canción se para, el resultado es el esperado.



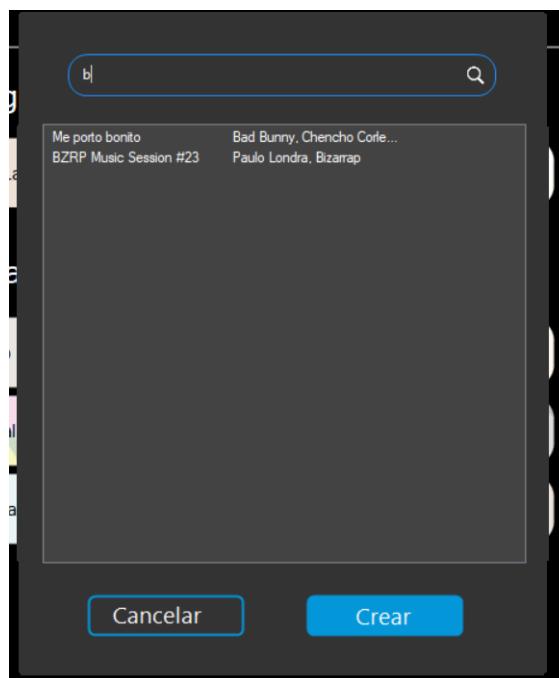
- Caso de prueba 9: Añadir playlist

En esta prueba comprobaremos que podemos añadir correctamente una playlist.

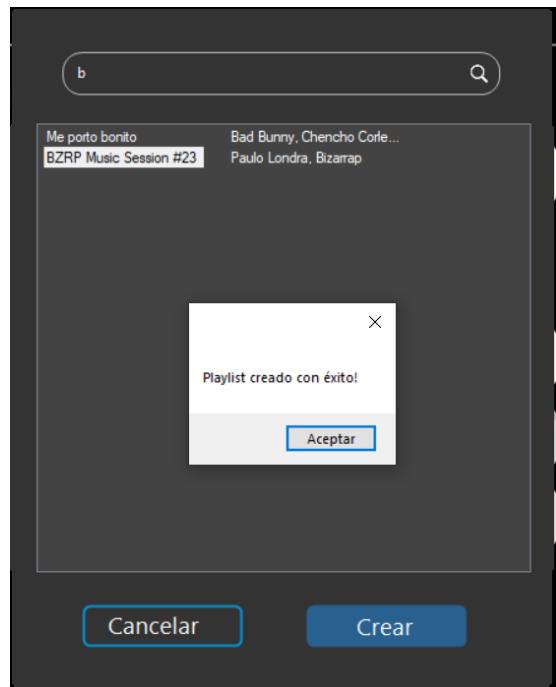
Hemos comprobado que al intentar añadir una playlist sin llenar el nombre no nos deja, **el resultado es el esperado**.



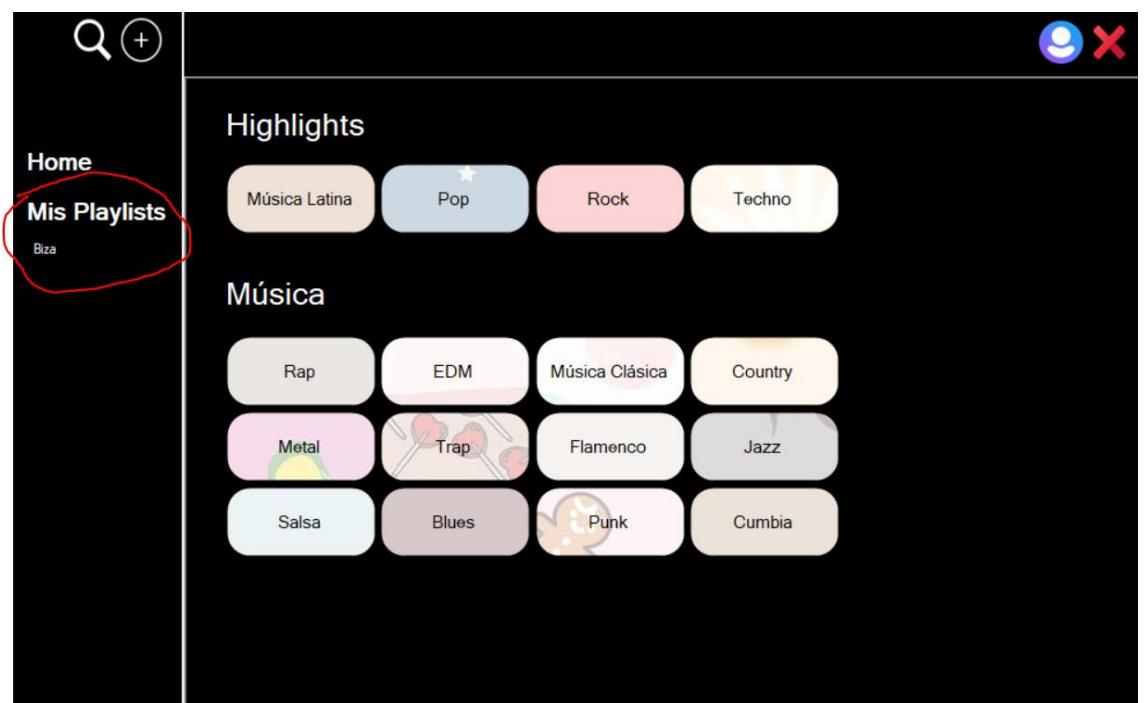
Hemos comprobado que, al intentar filtrar por nombre de canciones, nos las filtra bien, **el resultado es el esperado**.



Hemos comprobado que, al seleccionar canciones nos crea nuestra playlist, **el resultado es el esperado**.



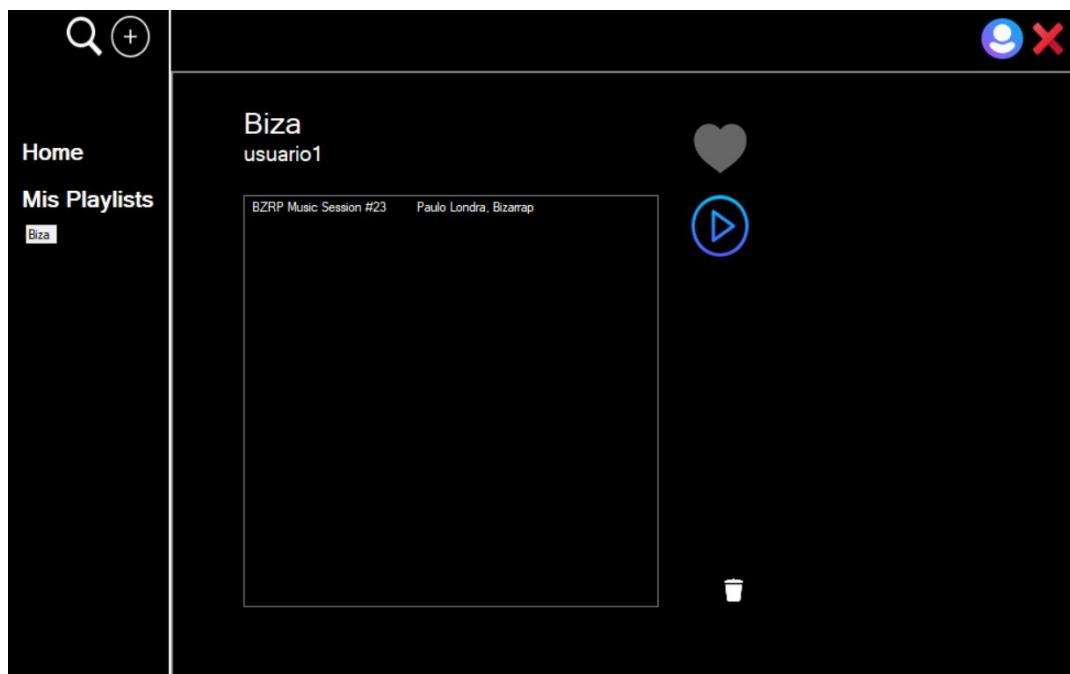
Hemos comprobado que, cuando creamos una playlist nos aparece en el apartado de 'Mis playlists', **el resultado es el esperado**.



- Caso de prueba 10: Ver mis playlist

En esta prueba comprobaremos que podemos acceder a nuestras playlist.

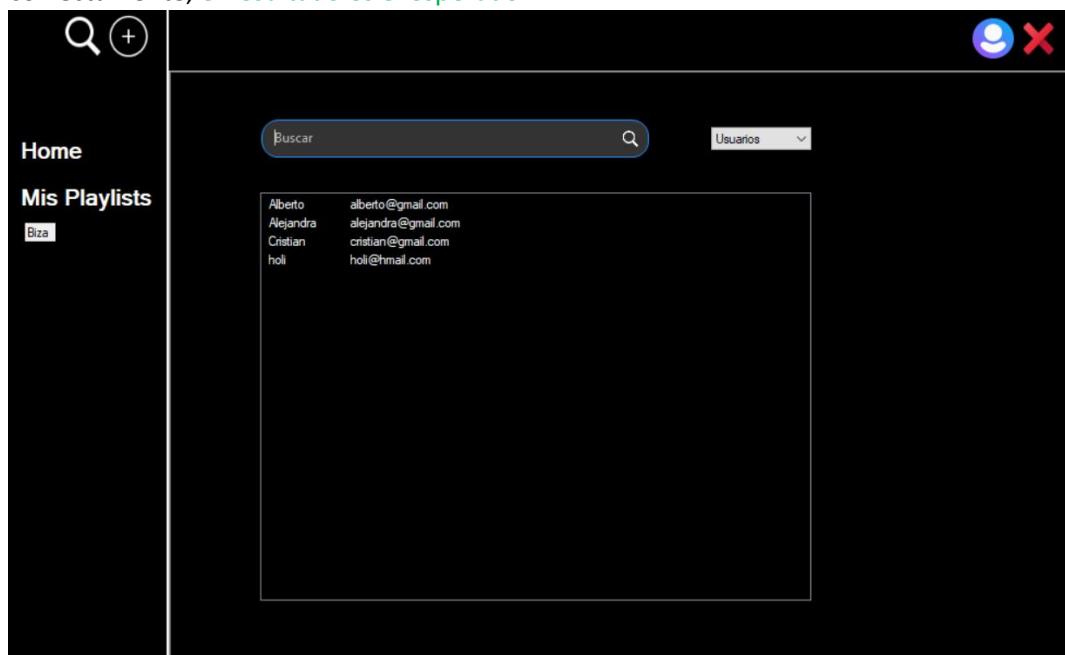
Hemos comprobado que, al intentar acceder a nuestras playlist, podemos acceder, el resultado es el esperado.



- Caso de prueba 11: Funcionalidad del buscar

En esta prueba comprobaremos que el buscar funciona perfectamente.

Hemos comprobado que, al cambiar el combo box del formulario, la lista se rellena correctamente, el resultado es el esperado.



The screenshot shows a dark-themed mobile application interface. On the left, there's a vertical sidebar with a search icon at the top, followed by 'Home' and 'Mis Playlists'. Under 'Mis Playlists', there's a small thumbnail labeled 'Biza'. The main content area has a search bar at the top with the placeholder 'Buscar' and a magnifying glass icon. To the right of the search bar is a dropdown menu set to 'Música'. Below the search bar is a list of song titles grouped into two columns:

DE GRANÁ A MARACAY	Ayax, Prok y Akapellah
Me puto bonito	Bad Bunny, Chencho Corone
Es Épico	Canserbero
BZRP Music Session #23	Paulo Londra, Bizarrap
Caminaré	Natos, Waor y Maka
HUSTLERS	Natos, Waor y Fernando Costa
Reproches	Ayax y Prok
Hasta el amanecer	Natos y Waor
DELIRIUM	Natos, Waor, Recycled
Fresas con nata	Prok
Manichula	Trueno, Nicky nicole, Bizarrap
Dispuestos a morir	Natos, Waor, C.R.O, Homer el Mero Mero
El circo	Ayax

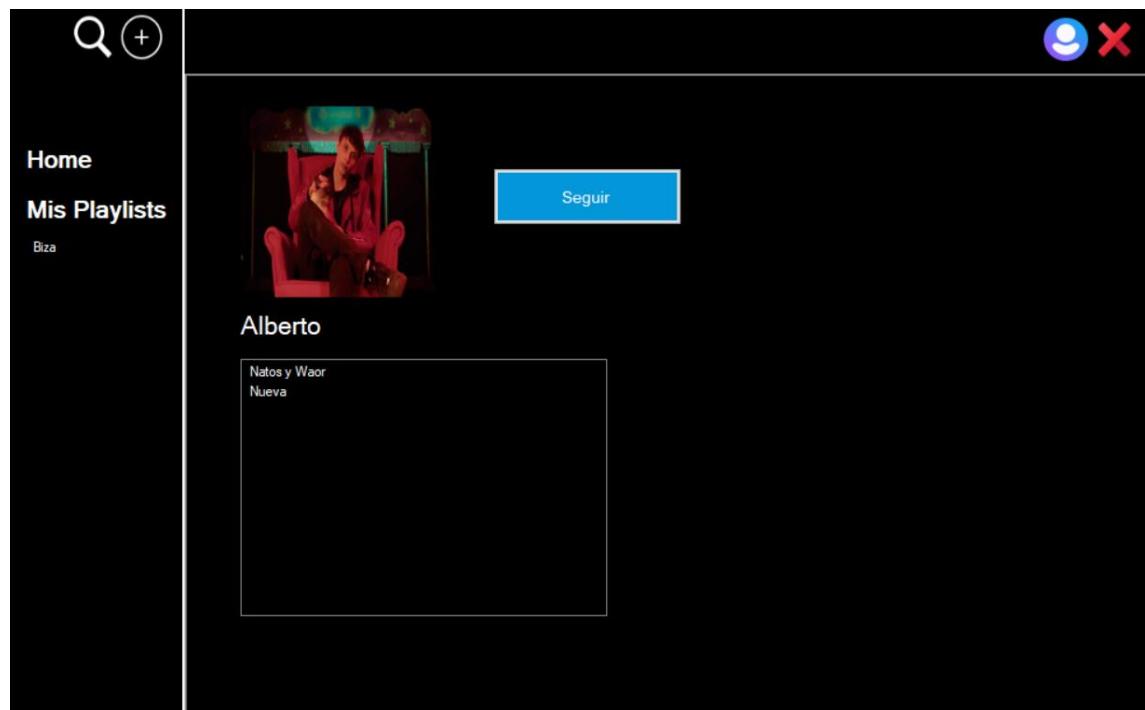
This screenshot shows the same dark-themed mobile application interface as the first one. The sidebar on the left includes 'Home', 'Mis Playlists' (with a 'Biza' thumbnail), and a search icon. The main content area features a search bar with 'Buscar' and a magnifying glass icon, and a dropdown menu set to 'Playlist'. Below these are the names of several playlists:

- Alee
- Natos y Waor
- Top Éxitos
- Nueva
- Truenito
- Biza
- Bad Bunny

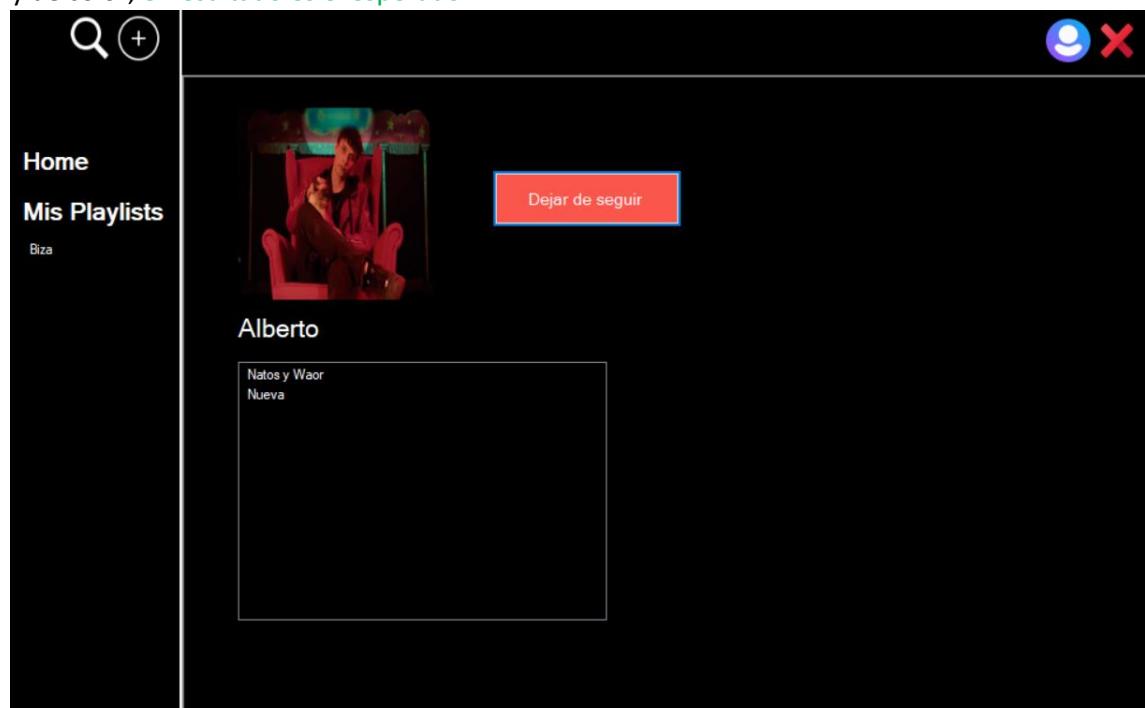
- Caso de prueba 12: Otros usuarios

En esta prueba comprobaremos que la interacción con otros usuarios funciona correctamente.

Hemos comprobado que, desde el buscar, al hacer clica otros usuarios, **el resultado es el esperado**.



Hemos comprobado que, al darle al botón seguir, lo seguimos y el botón cambia de texto y de color, **el resultado es el esperado**.

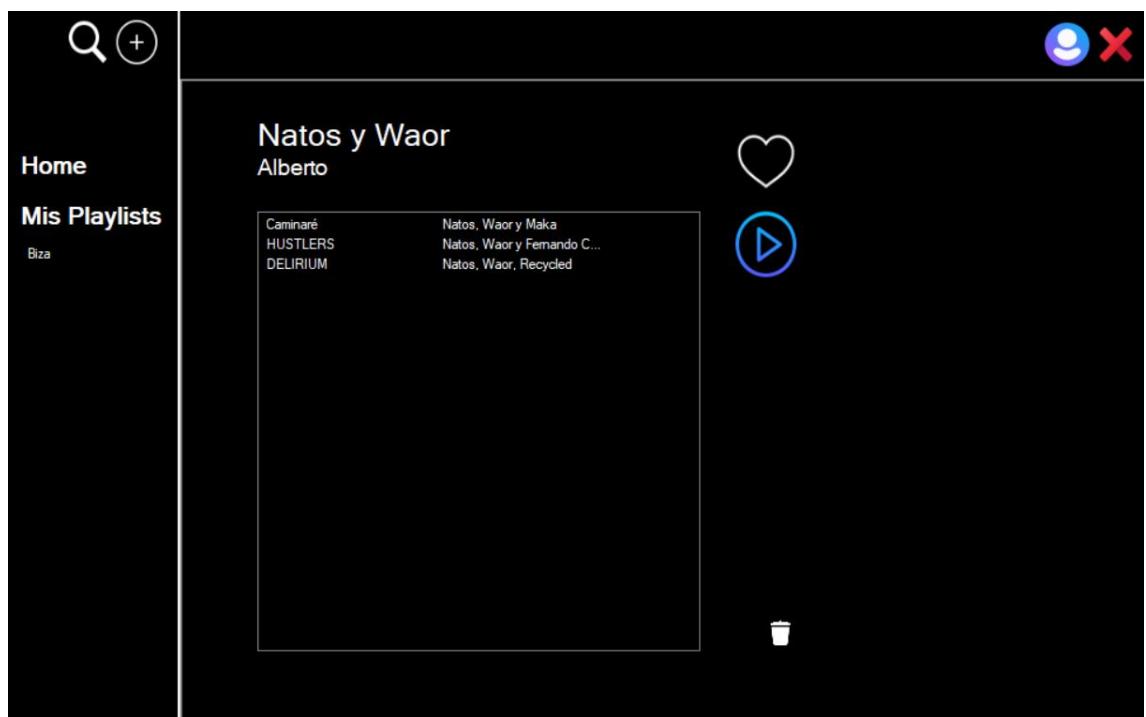


```
  ↴ — usuario1Alberto
      └── idSeguido: "usuario1Alberto"
      └── usuarioSeguido: "Alberto"
      └── usuarioSeguidor: "usuario1"
```

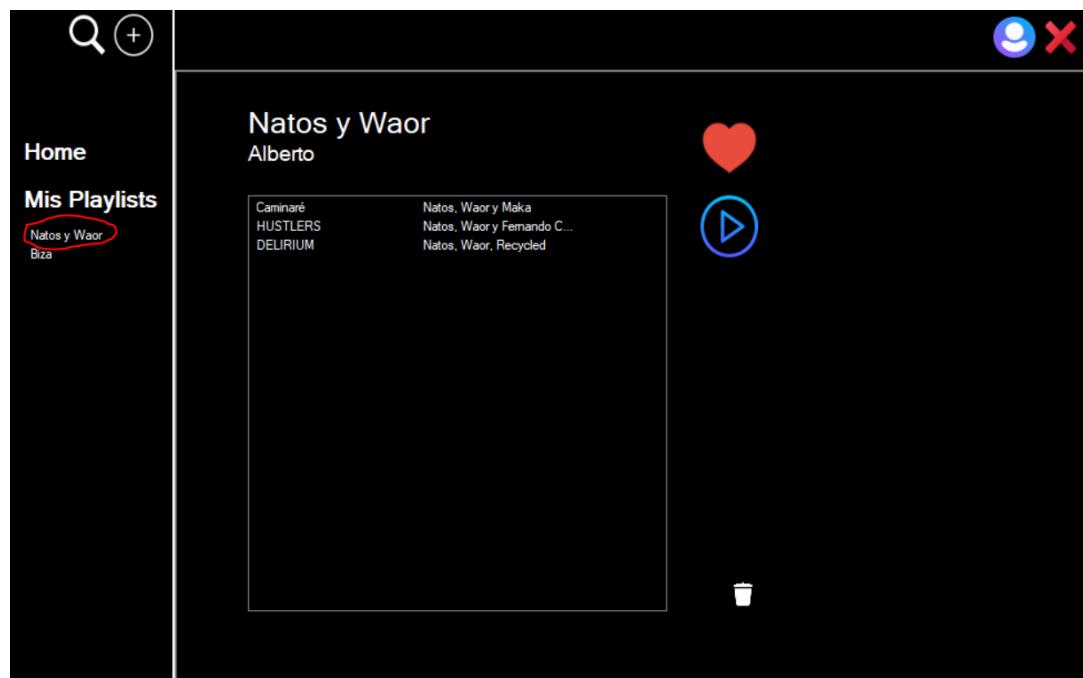
- *Caso de prueba 13: Playlist*

En esta prueba comprobaremos que tenemos acceso a las distintas playlists públicas de otros usuarios, además de intentar borrar, y reproducir playlists nuestras y de otros usuarios.

Hemos comprobado que, podemos ver otras playlist de otros usuarios, **el resultado es el esperado**.

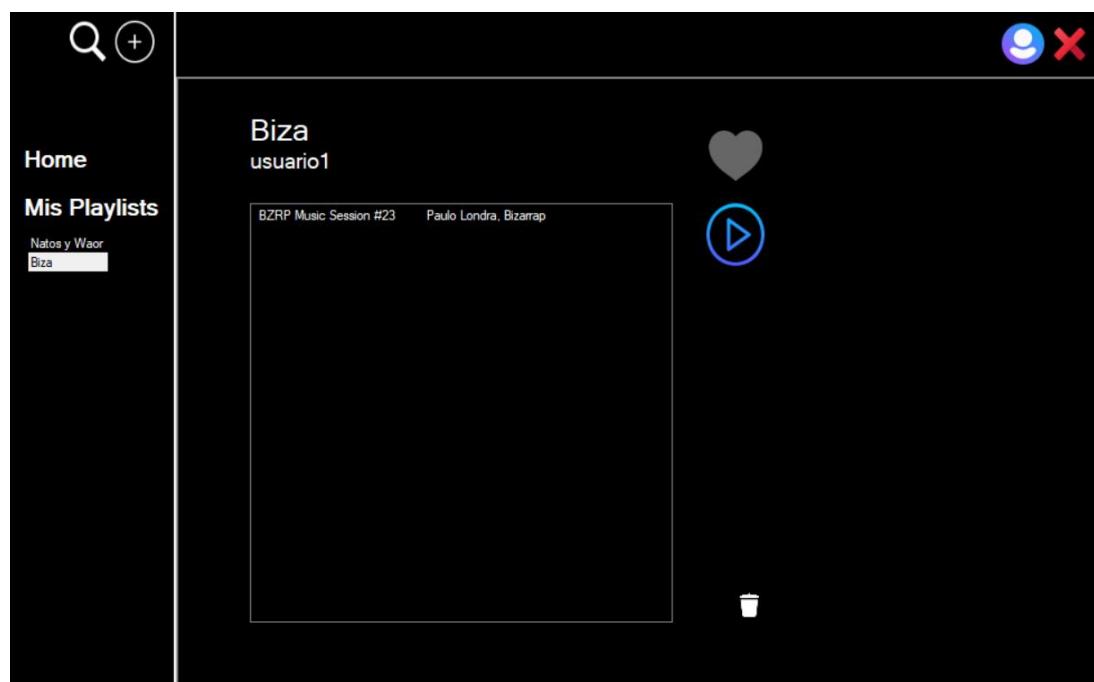


Hemos comprobado que podemos darle a favoritos a las playlist de otros usuarios y, además de que el botón del corazón cambie a rojo, se nos añadirá la playlist a nuestras playlist, **el resultado es el esperado**.

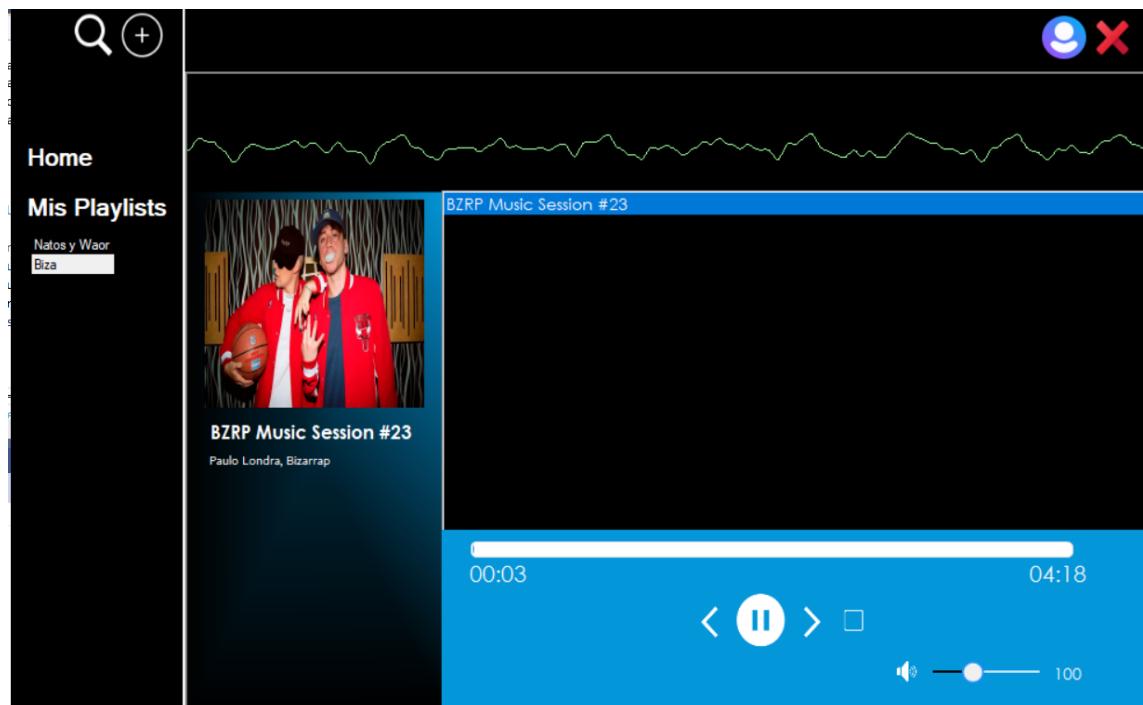


```
▼ — usuario1Natos y Waor
  └── idFavorito: "usuario1Natos y Waor"
  └── playlist: "52fc4b1f-d930-47cc-ac81-b2ceae0f3aff"
  └── usuario: "usuario1"
```

Hemos comprobado que no podemos poner en favoritos a una playlist que hemos creado nosotros, el resultado es el esperado.



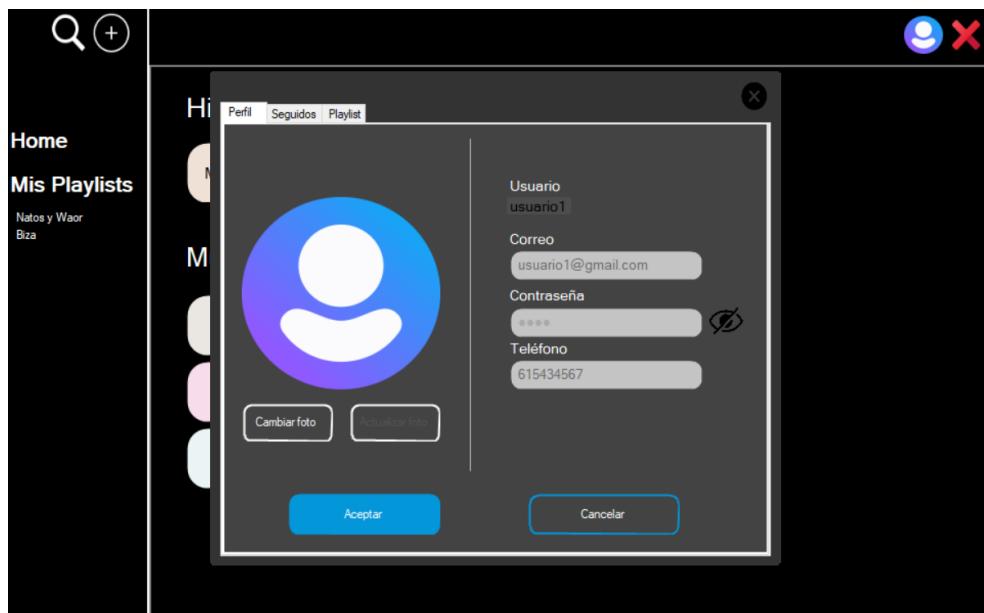
Hemos comprobado que solo podemos eliminar playlist que sean nuestras, **el resultado es el esperado**.



- Caso de prueba 14: Editar usuario

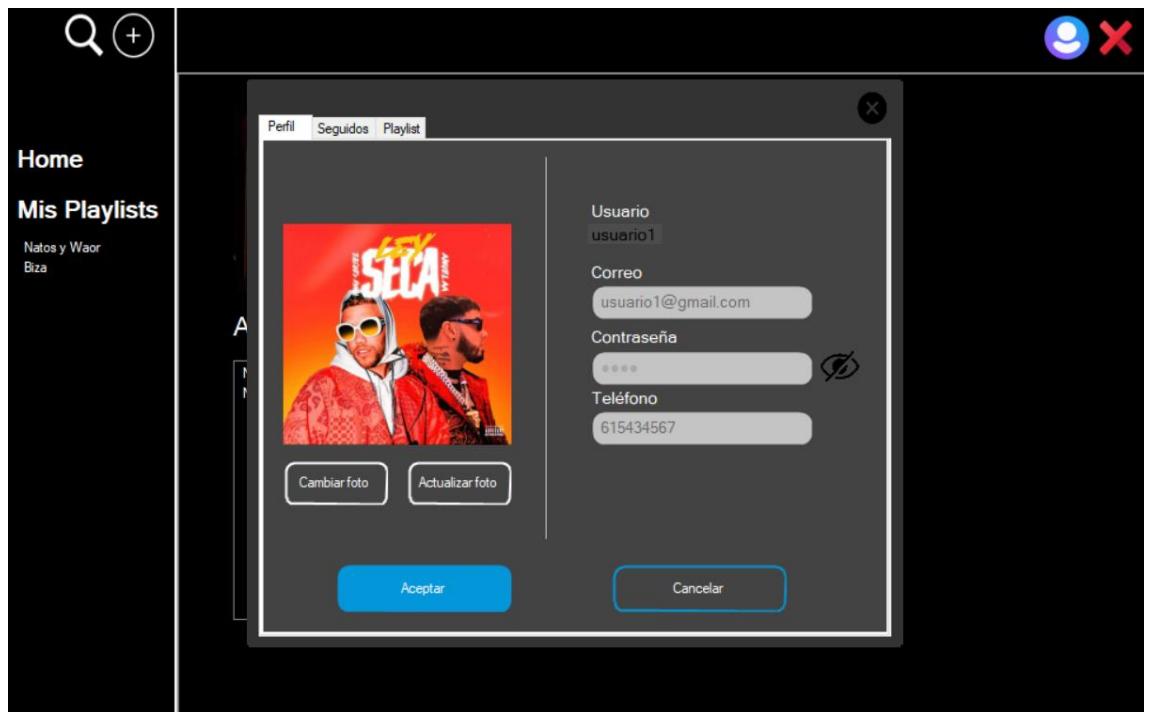
En esta prueba comprobaremos que podemos editar nuestro propio usuario, además de tener acceso a una lista con nuestras playlist y los usuarios que seguimos.

Hemos comprobado que, al hacer clic en nuestro usuario, nos carga todos nuestros datos, una lista con los usuarios que seguimos y una lista con nuestras playlists, **el resultado es el esperado**.

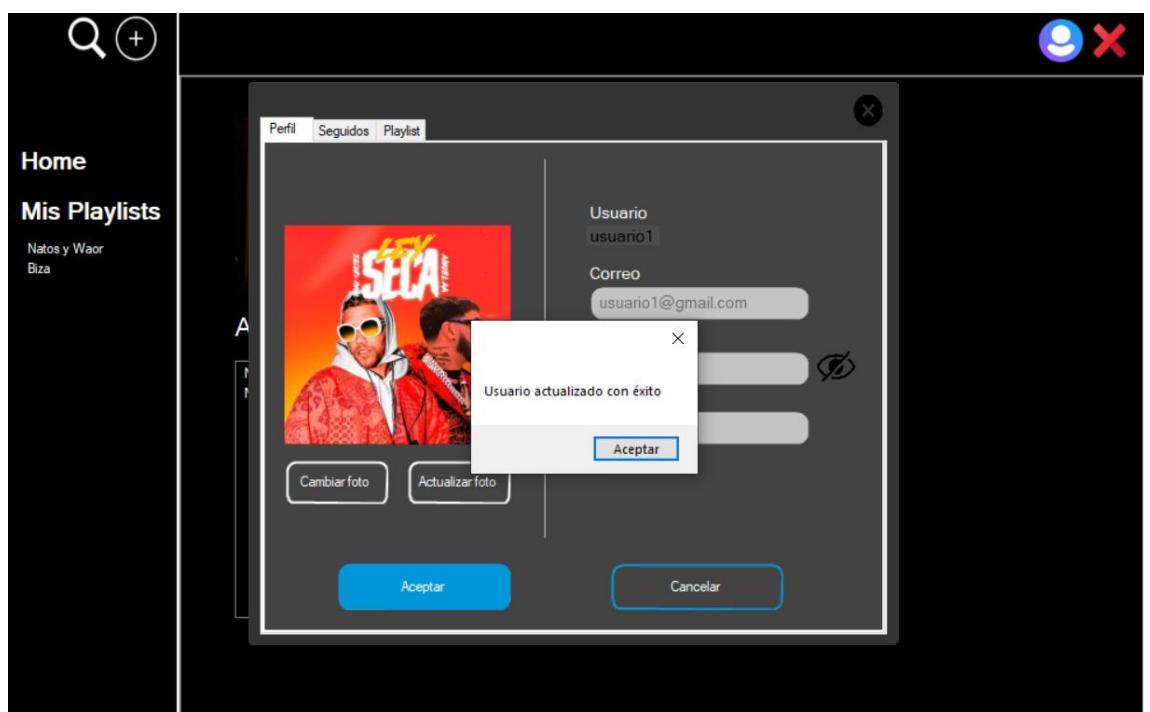




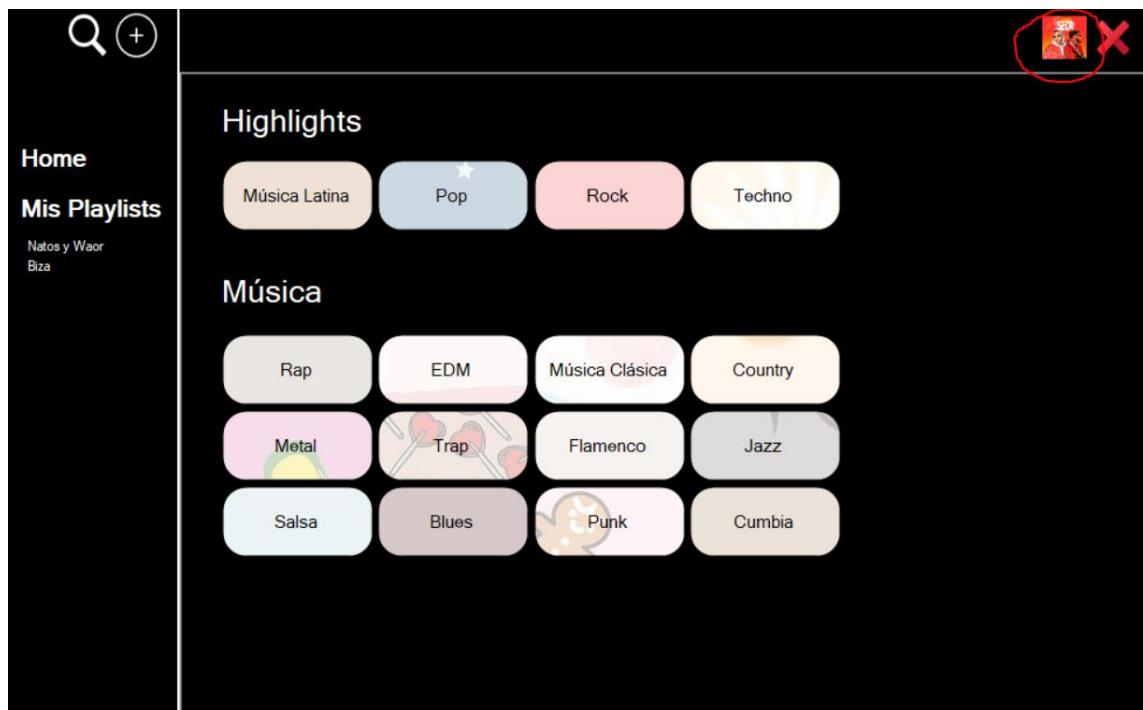
Hemos comprobado que, al hacer clic en cambiar foto, la foto podemos modificar nuestra foto, **el resultado es el esperado**.



Hemos comprobado que, podemos cambiar el correo, la contraseña y el teléfono de nuestro usuario, **el resultado es el esperado**.



Hemos comprobado que, una vez cambiada la foto de nuestro usuario, la foto de arriba a la derecha se nos cambia, el resultado es el esperado.



6.2 Solución a problemas encontrados

He ido solucionando los problemas encontrados a medida que desarrollaba la app, la mayoría de los problemas que he encontrado han sido respecto a firebase y a las listas, ya que la velocidad de descargar de firebase es muy baja y me ha dado muchos problemas.

7. Lanzamiento y puesta en marcha

7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema

En este apartado hablaremos sobre los aspectos más relevantes del despliegue, tanto del servidor como de nuestra aplicación.

Cuando hablamos de despliegue nos referimos a todas las actividades que hacen que un sistema de software esté disponible para su uso.

Estas actividades son:

- **Lanzamiento:** La actividad de publicación se deriva del proceso de desarrollo completado y, a veces, se clasifica como parte del proceso de desarrollo en lugar del proceso de implementación. Incluye todas las operaciones para preparar un sistema para el compilado y la transferencia a los sistemas informáticos en los que se ejecutará en producción.
- **Instalación y activación:** La instalación implica establecer algún tipo de comando, acceso directo, script o servicio para ejecutar el software. Para sistemas complejos puede implicar la configuración del sistema, posiblemente haciendo preguntas al usuario final sobre su uso previsto, o preguntándoles directamente cómo les gustaría que se configuren. Y la activación es la actividad de iniciar el componente ejecutable de software por primera vez.
- **Desactivación:** La desactivación es la actividad inversa a la activación y se refiere al cierre de cualquier componente que ya se esté ejecutando de un sistema.
- **Desinstalación:** La desinstalación es el inverso a la instalación, es la eliminación de un sistema que ya no es necesario.
- **Actualización:** Este proceso reemplaza una versión anterior de todo o parte de un sistema de software con una versión más reciente.
- **Actualización incorporada:** La automatización de los procesos para instalar actualizaciones va totalmente automática hasta iniciada y controlada por el usuario.
- **Seguimiento de las versiones:** Los sistemas de seguimiento de la versión ayudan al usuario a encontrar e instalar actualizaciones a los sistemas de software.
- **Adaptación:** La actividad de adaptación también es un proceso para modificar un sistema de software que se ha instalado anteriormente, se diferencia de la actualización en que las adaptaciones son iniciadas por eventos locales, mientras que la actualización es una consecuencia de la disponibilidad de la nueva versión.

En el caso del lanzamiento de AMusic:

- **La parte del servidor (Firebase).**

En lanzamiento en la parte del servidor (ya que usamos Firebase) es bastante sencilla, no porque su lanzamiento sea fácil, sino porque lo hace Firebase por nosotros, **desde el momento de su creación, Firebase ya se encuentra desplegado**, nosotros lo único que tenemos que hacer es configurarlo y meternos en la consola ya que el único requisito que tenemos que cumplir es tener una cuenta de Google.

- **La parte del servidor (Visual Studio).**

El lanzamiento en visual studio lo haremos creando un archivo ejecutable de la aplicación.

7.2 Manual de uso

El manual de uso de AMusic es el siguiente:

<https://github.com/AlbertoMunozBautista/AMusic/blob/develop/AMusic/Documentacion/Manual%20de%20usuario%20AMusic.pdf>

8. Valoración y conclusiones

Llegado a este punto del proyecto, donde veo todo con más retrospectiva, en general, no ha sido un camino fácil, pero a pesar de todas las dificultades que se han ido presentando y todas las trabas que ha tenido el camino, puedo decir que ha merecido la pena el esfuerzo.

Al principio del proyecto lo veía todo con mucha ilusión y ganas de empezar, y, gracias a que iba viendo que el proyecto seguía adelante y no se atascaba, he seguido manteniendo esas ganas hasta el final cosa que también me ha ayudado a acabar.

El realizar el proyecto durante las prácticas, además de tener un punto extra de dificultad y compromiso, también me ha servido para coger una dinámica de trabajo correcta, que he podido aplicar a este proyecto y gracias a ello, he cumplido todas mis expectativas temporales pudiendo realizar el plazo de todas las entregas.

Otro punto extra de dificultad, ha sido el realizar una documentación de este calibre ya que, en mi opinión, es una documentación muy extensa que no estaba acostumbrado a hacer, pero he de decir, que me ha servido para tener las cosas mucho más claras ya que también me ha ayudado en la gestión del tiempo, además me ha ayudado tener plasmado todo lo que iba a usar en mi proyecto, ya que al tratar varios puntos como las tecnologías a usar, el diagrama de clases,... me ha servido para tener unas ideas mucho más claras y fijas.

Durante el transcurso de estos años, he aprendido mucho de las diferentes asignaturas que hemos tenido, todos estos conocimientos han sido vitales para el desarrollo de este último proyecto y he intentado plasmarlos todos en la medida de lo posible, todas las asignaturas han sido importantes y fundamentales.

A modo de concluir, quería destacar que para mí todo esto ha sido una experiencia positiva ya que me ha ayudado a aprender y a ser un poquito más profesional.

9. Bibliografía

<http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html>

<http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html>

<https://www.qualtrics.com/es/gestion-de-la-experiencia/investigacion/estudio-de-mercado/>

<https://marketing4ecommerce.net/que-es-el-target/>

<https://debitoor.es/quia-pequenas-empresas/marketing/como-analizar-la-competencia#:~:text=El%20an%C3%A1lisis%20de%20la%20competencia,econom%C3%ADa%20de%20mercado%3A%20la%20competencia.>

<https://www.arimetrics.com/glosario-digital/dafo>

<https://creately.com/blog/es/diagramas/tutorial-diagrama-caso-de-uso/#importance>

<https://www.eaprogramas.es/bloq/negocio/empresa/cinco-beneficios-de-la-planificacion-de-tareas>

<https://www.isotools.org/2015/12/08/la-gestion-de-los-riesgos-laborales-en-oficinas/>

<https://edit.org/es/blog/anlisis-foda-editable-online-imprimir>

<https://attachmedia.com/blog/prototipos-importancia/>

<https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-clases/>

<https://aws.amazon.com/es/nosql/>

<https://decidesoluciones.es/arquitectura-de-microservicios/>

https://es.wikipedia.org/wiki/C_Sharp

<https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart>

https://www.plainconcepts.com/es/kotlinandroid/#Caracteristicas_y_ventajas_de_Kotlin

<https://www.verity.cl/blog/5-pruebas-funcionales-que-todo-software-necesita>