Especificación de requerimientos Musicloud

Irving Iván Gumesindo Trujillo

Contenido

[Introducción 3](#_Toc48867775)

[Proceso 4](#_Toc48867776)

[Artefactos de análisis y diseño 5](#_Toc48867777)

[Clases y caracteristicas de usuarios 5](#_Toc48867778)

[Requerimientos funcionales. 6](#_Toc48867779)

[Backend 6](#_Toc48867780)

[Frontend 6](#_Toc48867781)

[Requerimientos no funcionales. 7](#_Toc48867782)

[Modelo de dominio preliminar. 8](#_Toc48867783)

[Diagrama de clases 9](#_Toc48867784)

[Diagrama de casos de uso 10](#_Toc48867785)

[Diagrama de paquetes 10](#_Toc48867786)

[Diagrama de casos de uso de Cuenta 11](#_Toc48867787)

[Diagrama de casos de uso de Reproductor 12](#_Toc48867788)

[Diagramas de casos de uso de Creador de contenido 12](#_Toc48867789)

[Diagrama de casos de uso de búsqueda 13](#_Toc48867790)

[CU01.- Iniciar sesión 14](#_Toc48867791)

[Prototipo de CU01 14](#_Toc48867792)

[CU02.- Crear Cuenta 14](#_Toc48867793)

[Prototipo de CU02 14](#_Toc48867794)

[CU04 Modificar cuenta 15](#_Toc48867795)

[Prototipo CU04 15](#_Toc48867796)

[CU05 Convertirse en creador de contenido 17](#_Toc48867797)

[Prototipo CU05 17](#_Toc48867798)

[CU06 Mostrar página de inicio 18](#_Toc48867799)

[Página principal 18](#_Toc48867800)

[Prototipo CU06 19](#_Toc48867801)

[Cu07 Mostrar álbumes 19](#_Toc48867802)

[Prototipo CU07 19](#_Toc48867803)

[CU08.- Consultar estudio 21](#_Toc48867804)

[Prototipo de CU08 21](#_Toc48867805)

[CU09.- Subir álbum 21](#_Toc48867806)

[Prototipo CU09 21](#_Toc48867807)

[CU12.- Consultar artistas 23](#_Toc48867808)

[Prototipo CU12 23](#_Toc48867809)

[CU13.- Consultar playlist 24](#_Toc48867810)

[Prototipo CU13 24](#_Toc48867811)

[CU14.- Consultar Biblioteca 26](#_Toc48867812)

[Prototipo CU14 26](#_Toc48867813)

[CU15 Crear Playlist 27](#_Toc48867814)

[Prototipo CU15 27](#_Toc48867815)

[CU16 Consultar Biblioteca Propia 28](#_Toc48867816)

[Prototipo CU16 28](#_Toc48867817)

[CU17 Agregar canción a biblioteca 28](#_Toc48867818)

[Prototipo CU17 28](#_Toc48867819)

[CU18 Mostrar canciones playlist 30](#_Toc48867820)

[Prototipo CU18 30](#_Toc48867821)

[CU19.- Agregar canción a playlist 31](#_Toc48867822)

[Prototipo CU19 31](#_Toc48867823)

[CU20 Reproducir canción 32](#_Toc48867824)

[Prototipo CU20 32](#_Toc48867825)

[CU21.- Mostrar canciones álbum 32](#_Toc48867826)

[Prototipo CU21 32](#_Toc48867827)

[Diagrama de despliegue 34](#_Toc48867828)

[Diagrama de componentes 35](#_Toc48867829)

[Diseño de API con Canvas 36](#_Toc48867830)

[Anexos 38](#_Toc48867831)

[Análisis estático de código con SonarQube 38](#_Toc48867832)

[Confiabilidad 38](#_Toc48867833)

[Seguridad 38](#_Toc48867834)

[Mantenibilidad 38](#_Toc48867835)

[Duplicación 39](#_Toc48867836)

[Pruebas del servidor 39](#_Toc48867837)

[Referencias 42](#_Toc48867838)

# Introducción

En el presente documento, se presentarán de manera general los requerimientos del sistema Musicloud. Servicio de música bajo demanda (streaming) en su versión para escritorio. El propósito de Musicloud es proporcionar al usuario la capacidad de reproducir música a través de internet o localmente, esta última gracias a la característica de descargar música.

Otra de las características proporcionadas es el poder subir su propria música, también se podrán crear lista de reproducción y gestionarlas, se permitirá la búsqueda de contenido, abarcando álbumes, artistas y playlist.

Musicloud surge con la finalidad de aplicar los conocimientos adquiridos en la experiencia educativa de Desarrollo de sistemas en red, donde se busca construir los servicios antes mencionados, para ello se utilizarán los lenguajes de programación JavaScript y Ruby , estos servicios serán construidos en la versión de Linux Ubuntu 20.04 , para el desarrollo de la API REST se utilizará el entorno de Nodejs el cual está basado en JavaScript, apoyado del framework Express que nos proporcionará las funcionalidades como el enrutamiento. Para el servicio de reproducción se utilizará Ruby, para construir estos servicios se utilizará Apache Thrift para las interfaces de servicio

Para la construcción del Fronted se hará uso del lenguaje C# ejecutado en de .NET Framework, se llegó esta elección dado que ya se tiene experiencia en el desarrollo y ayudará a optimizar el tiempo de elaboración.

Para el despliegue de los servidores se utilizará Docker, creando una imagen de cada contenedor para posterior correrlas en contenedores y permitir que el cliente haga peticiones a dichos contenedores con puertos específicos.

Este documento también describirá el diseño, la arquitectura y las tecnologías a utilizar para el desarrollo del sistema de streaming.

# Proceso

Para el proceso de desarrollo se pensó en ICONIX, está metodología comprende cuatro principales disciplinas: Análisis de requerimientos, análisis y diseño preliminar, diseño detallado e implementación. Por cuestiones del tipo de desarrollo solo se tomarán ciertas características, sobre todo para las primeras fases, es decir, se aplicará el análisis de requerimientos para identificar los requerimientos funcionales y el modelado de dominio, no se contarán con los apartados de diseño preliminar y detallado tal como lo describe esta metodología, es decir, no se realizarán diagramas de robustez ni diagramas de secuencia, solo modelo de dominio y diagrama de clases, sumando los respectivos diagramas de despliegue y componentes.

Se llevará a cabo en iteraciones con una duración de 2 semanas, a continuación, se muestran las iteraciones con las listas de tareas correspondientes en las fechas planeadas.

Primera iteración

Incremento 1

|  |  |  |
| --- | --- | --- |
| Requerimientos | Fecha | Descripción |
| Mayo | Análisis de sistemas |
| Mayo | Modelo de casos de uso |
| Mayo | Modelo de dominio |
| Mayo | Identificación de requerimientos funcionales y no funcionales |

Incremento 2

|  |  |  |
| --- | --- | --- |
| Requerimientos | Fecha | Descripción |
| 18 – 24 de mayo | Análisis de sistemas |
| 18 – 24 de mayo | Análisis de cliente de escritorio Spotify |
| 18- 24 de mayo | Modificar requerimientos funcionales con respecto al Backend |
| Diseño | 25- 31 de mayo | Diagrama de clases |
| 25- 31 de mayo | Diseño de recursos (rutas) |
| 25- 31 de mayo | Alinear recursos con peticiones HTTP |

Segunda iteración

Incremento 1

|  |  |  |
| --- | --- | --- |
| Requerimientos | Fecha | Descripción |
| 1 – 15 de junio | Modificación con base a la revisión |
| Diseño | 1-15 de junio |  |
| Construcción | 1-15 de junio | Exploración y elección de tecnologías |
| 1-15 | Diseño de API |
| Despliegue | 1-15 | Exploración y elección de tecnologías |

Como se puede ver se tenía planeado llevar una calendarización del trabajo que se iba a realizar, funcionó para el segundo incremento, pero después de ello se perdió el seguimiento y se optó por trabajar de manera libre.

# Artefactos de análisis y diseño

En este apartado del documento se muestran los artefactos de análisis y diseño, se enuncian los requerimientos funcionales y no funcionales, también se muestran los diagramas de casos de uso organizados por funcionalidades del sistema.

El respectivo modelo de clases de Musicloud, seguido se muestran los prototipos y descripciones de cada caso de uso.

## Clases y caracteristicas de usuarios

* Creador de contenido
  + Usuario que se registra como creador, el cual tiene la capacidad de subir música personal, modificarla, o eliminarla, también puede promocionar sus canciones.
* Consumidor de música
  + Usuario que puede escuchar la música en línea o de ser necesario descargarla y escucharla sin conexión, puede crear sus propias listas de producción y su biblioteca personalizada. Mantiene la experiencia de manejar la calidad de audio, y escuchar listas de radio personalizadas.

## Requerimientos funcionales.

### Backend

**RF01.-** El sistema debe permitir la suscripción de usuarios.

**RF02.-** El sistema debe permitir subir canciones al usuario para su biblioteca propia con un máximo de 250 canciones.

**RF03.-** El sistema debe guardar un historial de reproducción del usuario.

**RF04.-** El sistema debe generar a partir de una canción, una estación de radio con canciones del mismo género.

**RF05.-** El sistema debe permitir al usuario descargar las canciones para escucharlas sin conexión.

### Frontend

**RF06.-** El sistema debe permitir al usuario registrarse como Artista (Creador de contenido).

**RF07.-** El sistema debe permitir al usuario agregar su propia biblioteca de música.

**RF08.-** El sistema debe permitir agregar una canción al final de la lista de reproducción actual.

**RF09.-** El sistema debe permitir agregar una canción a continuación de la reproducción actual.

**RF10.-** El sistema debe permitir gestionar sus propias playlist.

**RF11**.- El sistema debe permitir agregar canción a me gusta.

**RF12.-** El sistema debe contar con una biblioteca de canciones compuesta por: Álbumes, Artistas, Canciones y Géneros.

**RF13.-** El sistema debe permitir registrar álbumes con sus respectivas canciones para el creador de contenido.

**RF14.-** El sistema debe permitir al usuario realizar búsquedas personalizadas de álbumes, artistas y playlist.

## Requerimientos no funcionales.

**Seguridad**.

El sistema debe permitir la autenticación de usuarios.

El sistema debe mantener integra la información de usuarios cifrando las contraseñas.

**Disponibilidad**.

El sistema debe estar disponible el 99% del tiempo en una semana.

El sistema debe ser capaz de estar disponible, aunque una función no esencial esté deshabilitada.

**Escalabilidad.**

El sistema debe ser capaz de mantener el mismo rendimiento, aunque la demanda de usuarios crezca en al menos un 50%.

**Heterogeneidad**

El sistema debe estar construido con Javascript y ruby del lado del servidor en un sistema Linux.

El sistema debe poder ejecutarse en cliente de escritorio

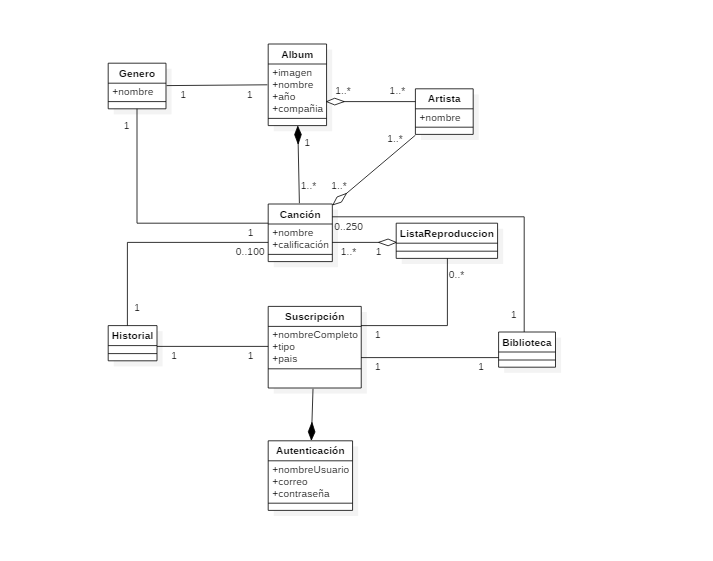
**Confiabilidad**

El sistema manejará los errores o fallos que sucedan durante la ejecución del cliente y del servidor.

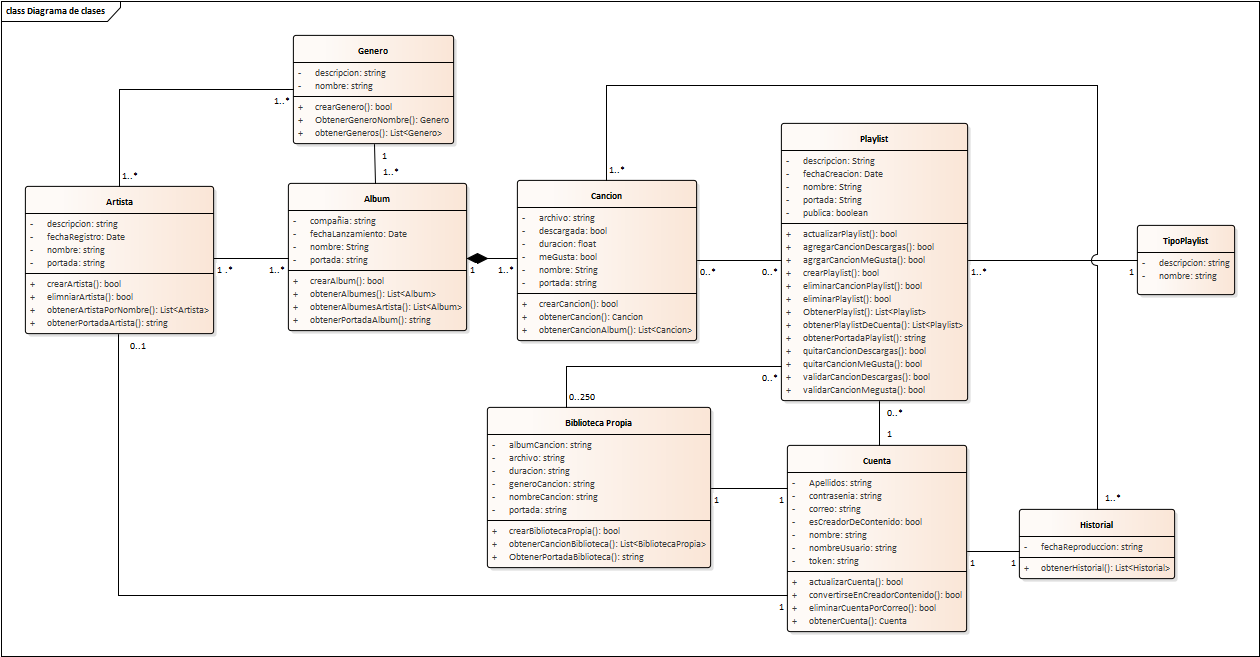
**Interoperabilidad:**

La arquitectura del sistema constará de distintos componentes que se comunicarán entre sí, con la habilidad poder intercambiar información y utilizarla.

## Modelo de dominio preliminar.



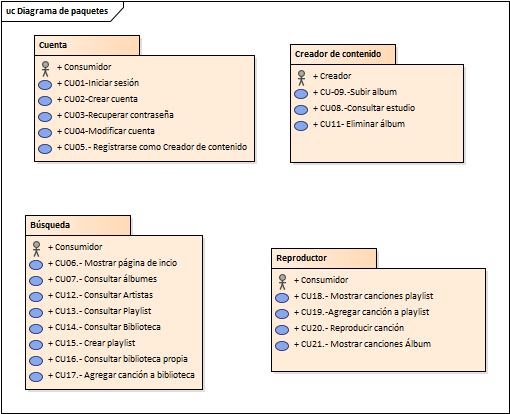
## Diagrama de clases



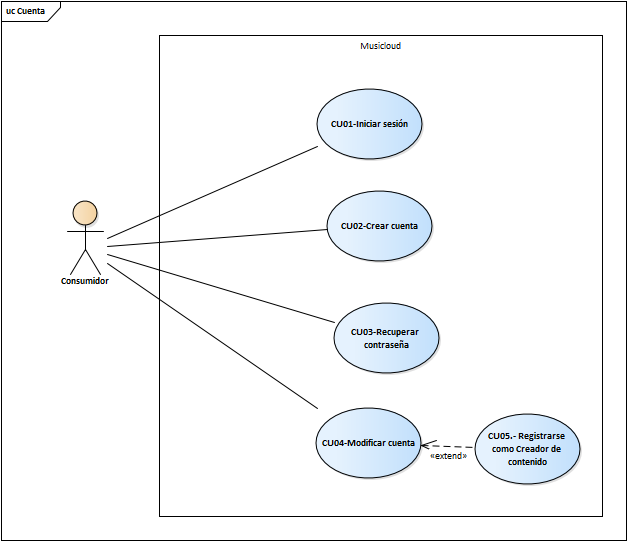
## Diagrama de casos de uso

### Diagrama de paquetes

A continuación, se describe el diagrama de paquetes de casos de uso, los cuales están divididos por cuatro características principales del sistema como lo son; cuenta, Creador de contenido, Búsqueda y Reproductor.

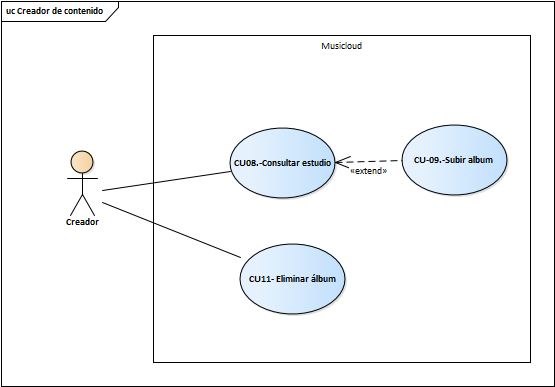


### Diagrama de casos de uso de Cuenta

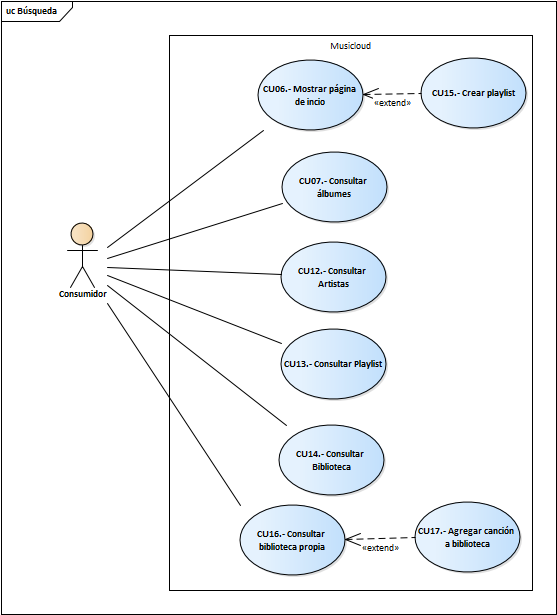


### Diagrama de casos de uso de Reproductor

### Diagramas de casos de uso de Creador de contenido



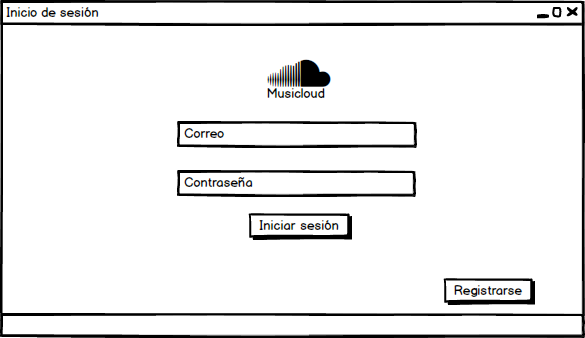
### Diagrama de casos de uso de búsqueda



## CU01.- Iniciar sesión

### Prototipo de CU01

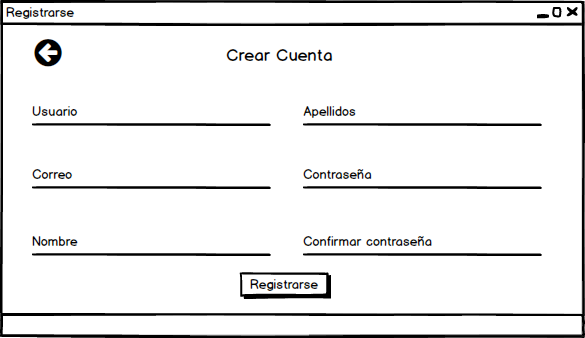
Se trata de la pantalla de iniciar sesión donde el usuario ingresará su correo y contraseña para poder acceder al sistema.



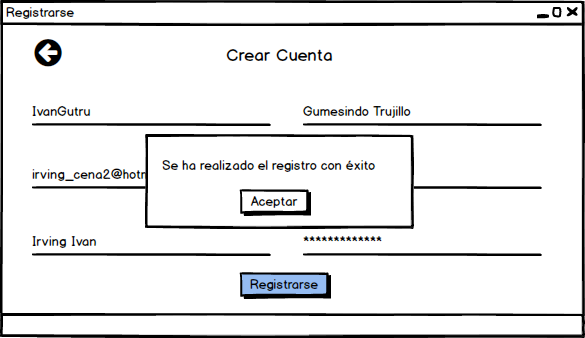
## CU02.- Crear Cuenta

### Prototipo de CU02

Este prototipo muestra el formulario para registrarse al sistema.

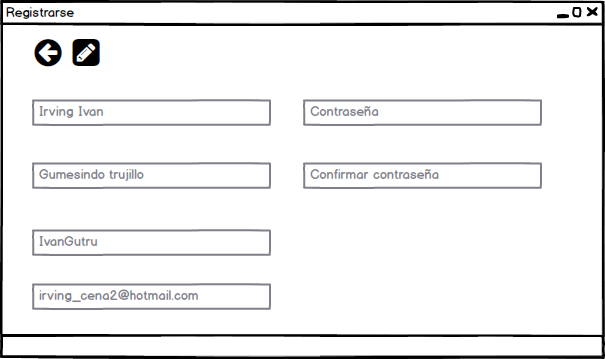


Muestra que todos los datos son correctos y que se ha registrado la cuenta en el sistema.

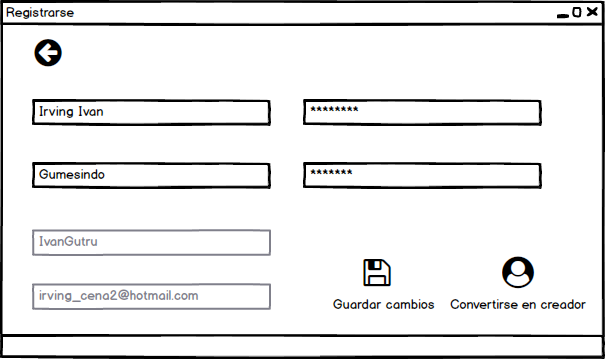


## CU04 Modificar cuenta

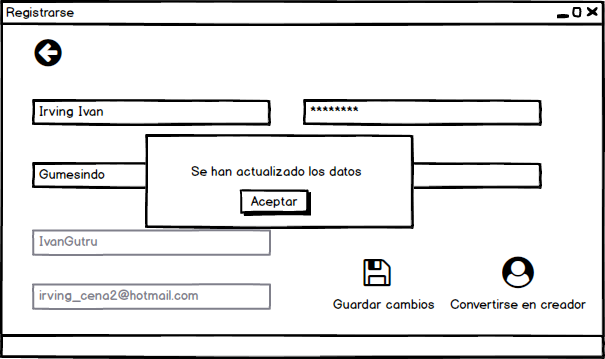
### Prototipo CU04

Permite habilitar los campos de información de la cuenta para poder ser modificados si el usuario lo desea.

Al habilitar los campos, los únicos que no se podrán modificar son el nombre de usuario y el correo electrónico, se habilitan los botones de guardar y convertirse en creador de contenido.



Muestra que la actualización de la cuenta fue realizada con éxito.



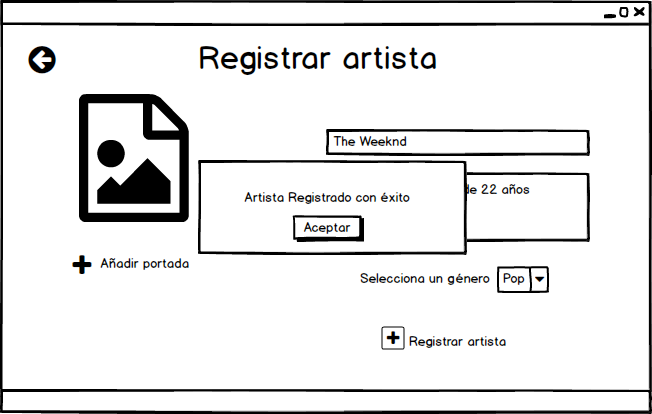
## CU05 Convertirse en creador de contenido

### Prototipo CU05

Después de dar clic en “Convertirse en creador de contenido”, se mostrará la siguiente pantalla, donde se pide información del artista que se registrará y administrarás para poder subir contenido al sistema.



Muestra que el registro se ha realizado con éxito.



## CU06 Mostrar página de inicio

### Página principal



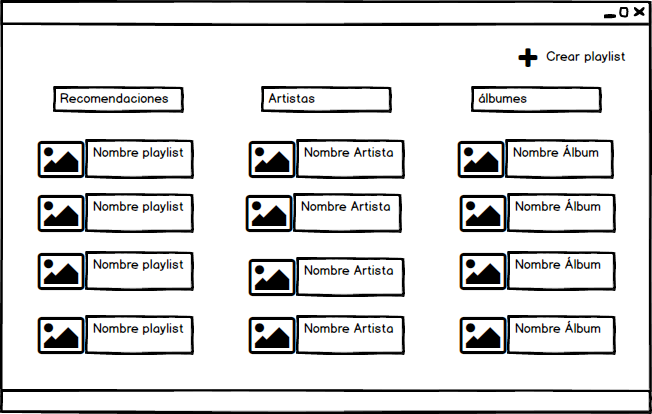
Esta pantalla representa la ventana principal, es decir, será la base para todos los siguientes prototipos, donde se irán mostrando dentro de ella, para ser precisos será en el centro gris.

Cuenta con un menú lateral con los apartados de Home, Álbumes, Artistas, Playlist, Biblioteca y en caso de ser creador de contenido, Mi estudio.

Otro aspecto importante a considerar la parte inferior, la cual se encargará de realizar las funcionalidades del reproductor.

### Prototipo CU06

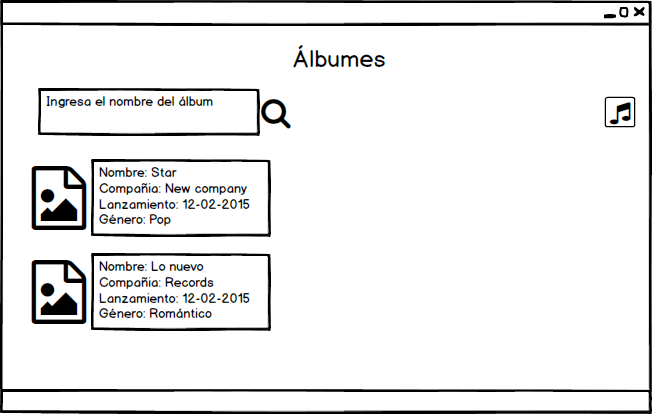
El siguiente prototipo resulta de seleccionar “Home” de la página principal. Se mostrarán las playlist que han sido creadas por la cuenta administradora del sistema, también muestra los últimos álbumes y artistas que han sido registrados.



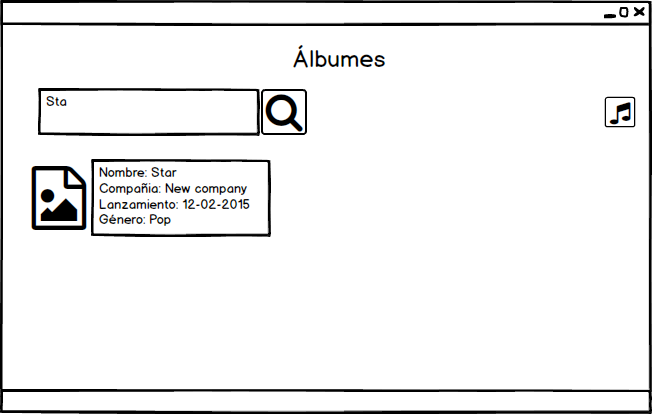
## Cu07 Mostrar álbumes

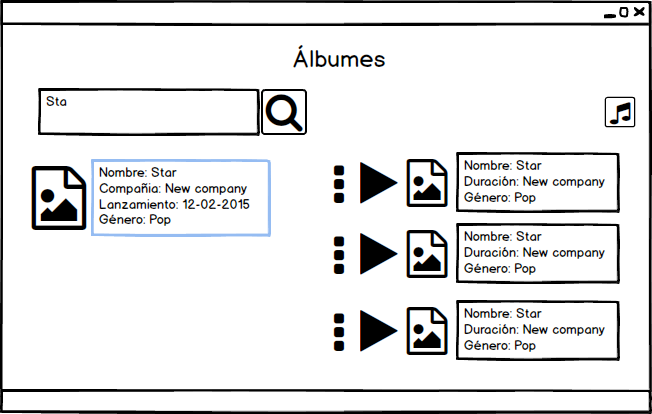
### Prototipo CU07

El siguiente prototipo resulta de seleccionar “Álbumes” de la página principal.



Al inicializar este caso de uso, se mostrarán por default algunos álbumes, también se podrán buscar todos los artistas que están registrados por medio del buscador, se ingresa el nombre y se pulsa buscar.



Después de filtrar alguna búsqueda o no, se da doble clic al álbum y se mostrarán las canciones que contiene como se muestra a continuación. Desde aquí ya se puede reproducir o gestionar que hacer con las canciones.

## CU08.- Consultar estudio

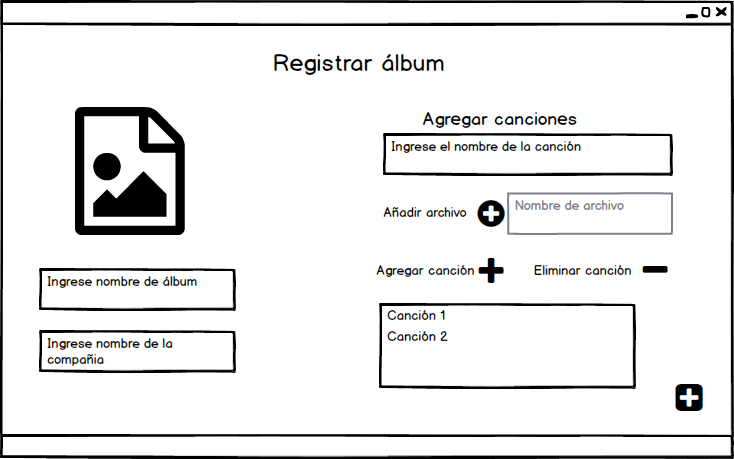
### Prototipo de CU08

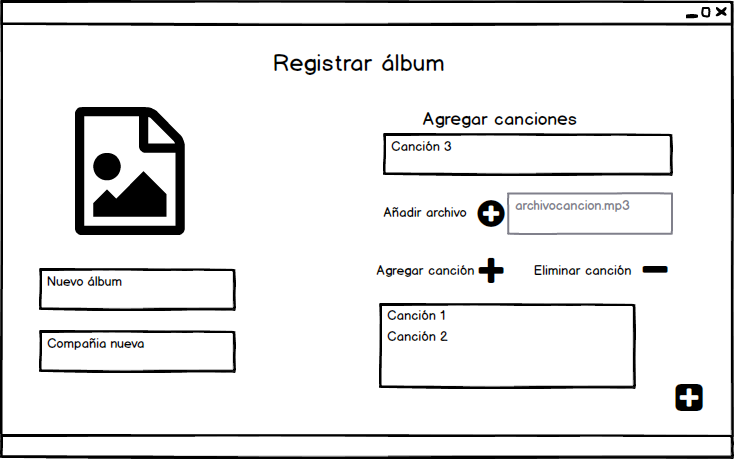
El siguiente prototipo resulta de seleccionar “Mi estudio” de la página principal. Este prototipo muestra la gestión de tu artista, dónde podrás consultar tu discografía e incluso subir un nuevo álbum.

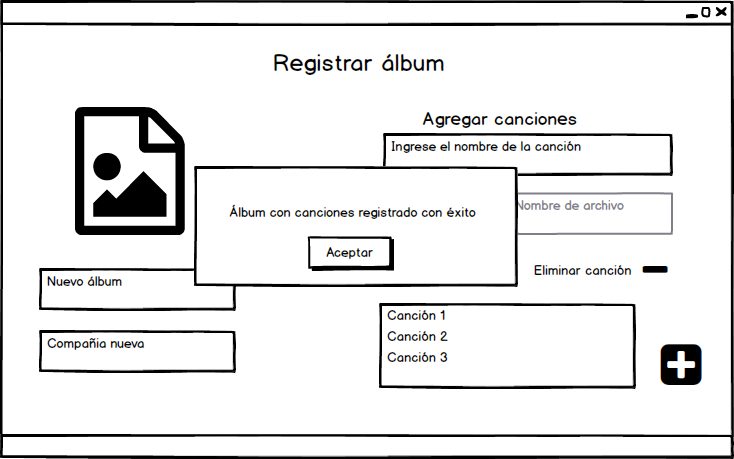


## CU09.- Subir álbum

### Prototipo CU09



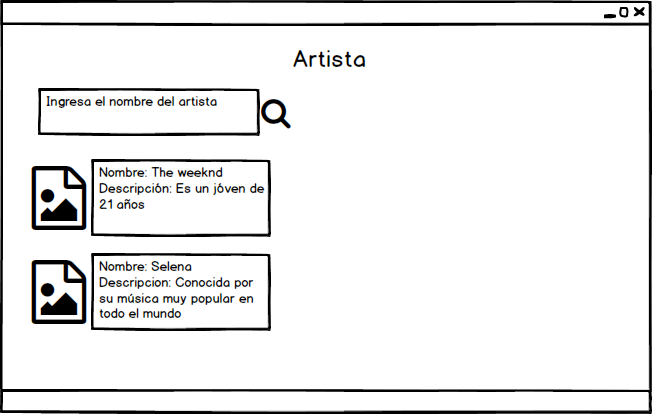
Primero se muestran todos los campos del álbum y de la canción, se va agregando canción por canción, con la opción de eliminar alguna de ser necesario, se debe ingresar al menos una, una vez que se da registrar álbum se guarda este mismo junto con todas las canciones que se hayan agradado a la lista.

Confirmación de registro de álbum.

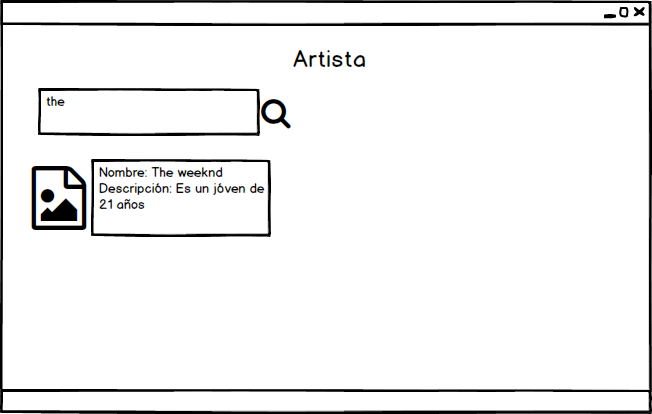
## CU12.- Consultar artistas

### Prototipo CU12

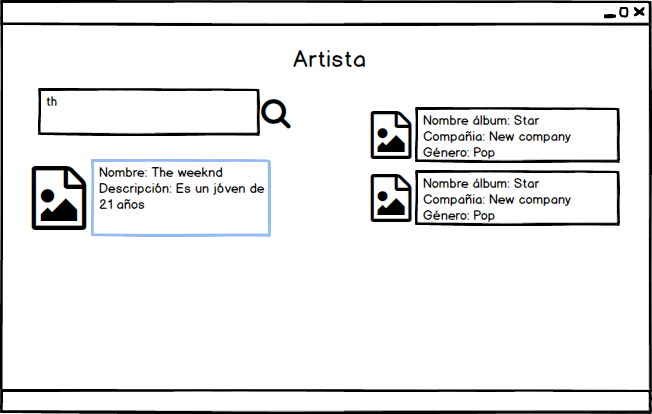
El siguiente prototipo resulta de seleccionar “Artista” de la página principal.



Después de mostrar los artistas por default se puede hacer una búsqueda personalizada dónde basta con poner alguna palabra de referencia para buscar y mostrar a los artistas cuyo nombre coinciden con lo ingresado.



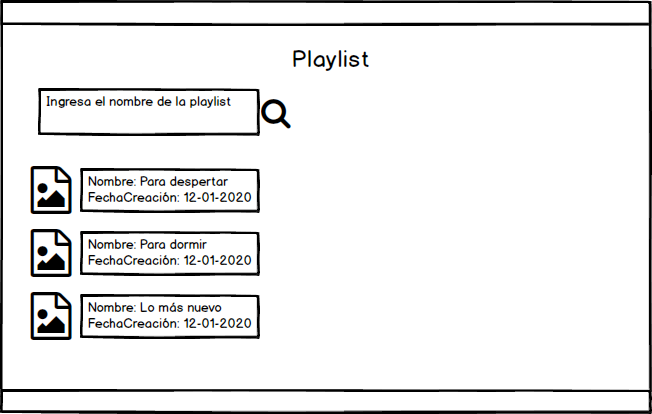
Después de mostrar al artista y dar doble clic sobre él, se mostraran sus álbumes para así poder acceder a la lista de canciones que tiene este último.

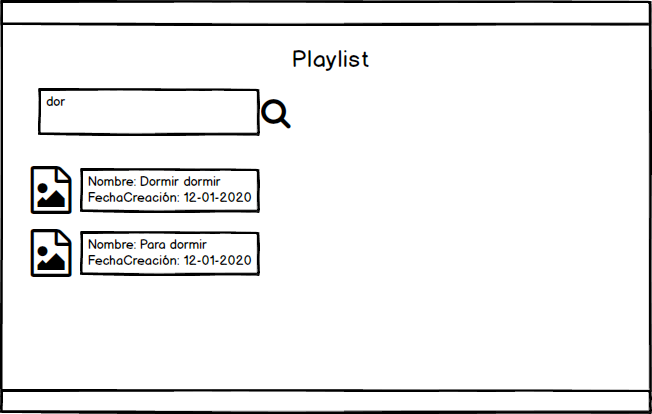


## CU13.- Consultar playlist

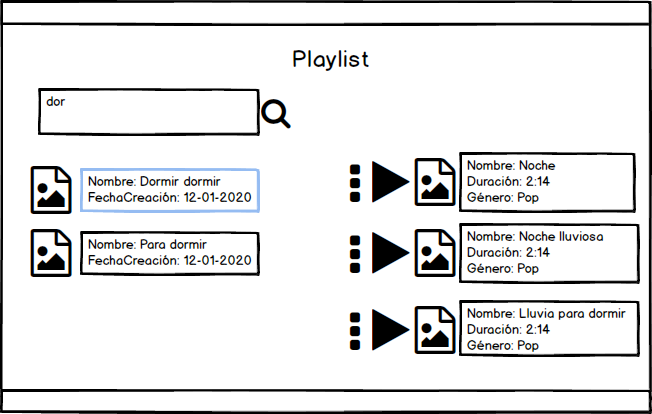
### Prototipo CU13

El siguiente prototipo resulta de seleccionar “Playlist” de la página principal. Se muestran las playlist públicas por default.



Se puede hacer una búsqueda personalizada ingresado alguna palabra de referencia en el campo de búsqueda.

Después de buscar y seleccionar una playlist, en esta misma pantalla se mostrarán las canciones que contiene para la gestión de la canción.

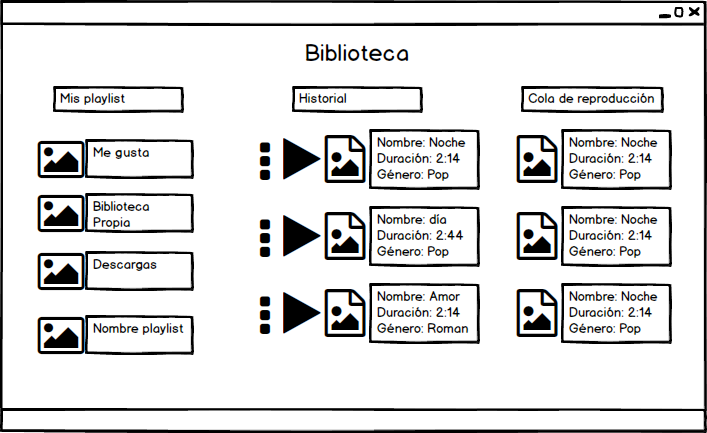


## CU14.- Consultar Biblioteca

### Prototipo CU14

El siguiente prototipo resulta de seleccionar “Biblioteca” de la página principal. En este caso de uso se muestran las playlist por default, “Me gusta”, “Descargas” y “Biblioteca propia”, además de aquellas que vas creando ya sean privadas o públicas.

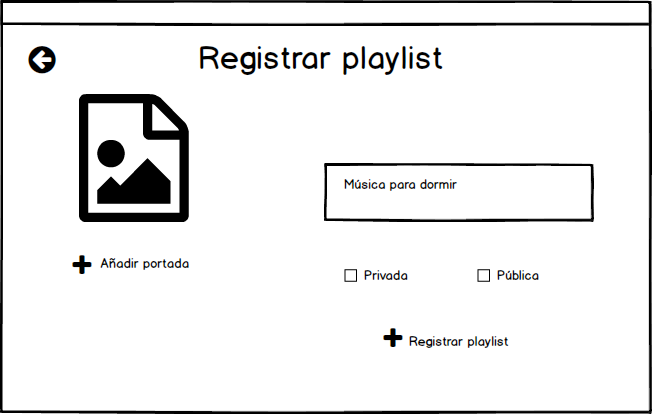
También se muestra el historial de reproducción donde va mostrando por fecha de reproducción aquellas canciones que se han escuchado. Por último, se muestra la cola de reproducción actual con las canciones que seguirán a continuación de lo que estás escuchando.



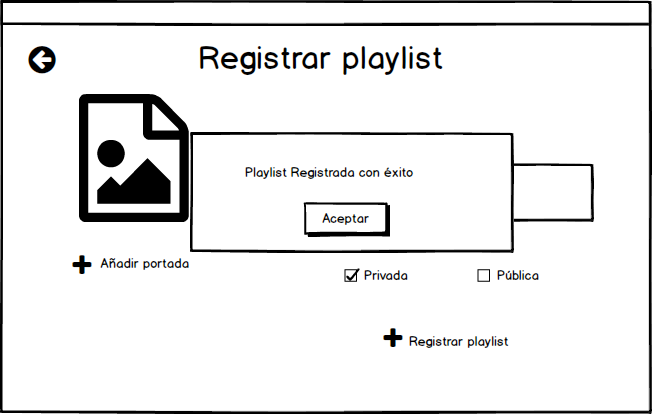
## CU15 Crear Playlist

### Prototipo CU15

Aquí se podrán crear tus playlist, nombrándolas, agregando alguna imagen de portada y estableciendo su visibilidad.

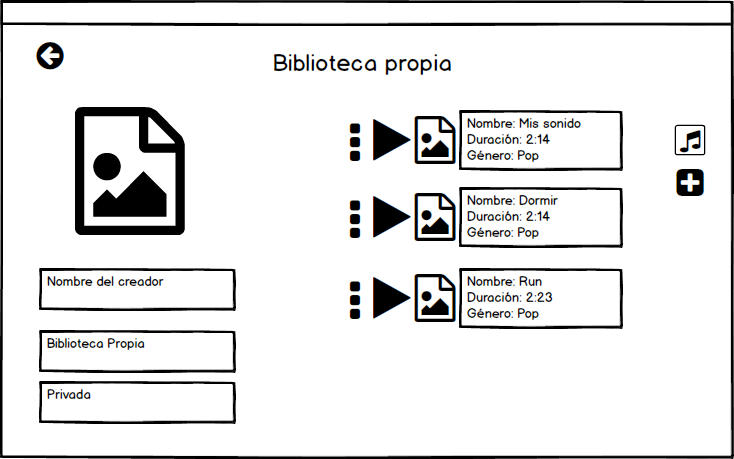


Se muestra el mensaje de éxito de la creación de la playlist.



## CU16 Consultar Biblioteca Propia

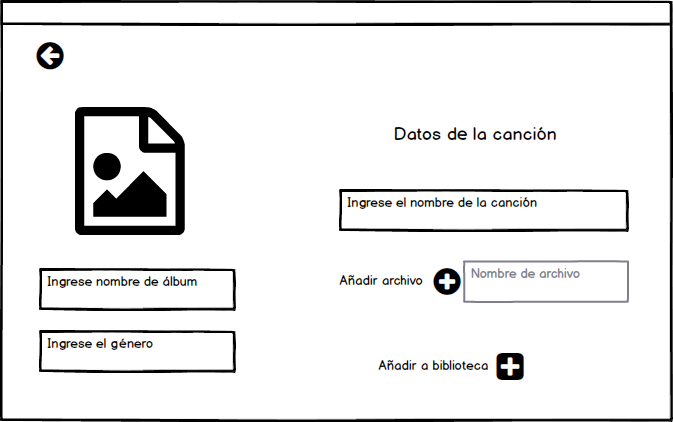
### Prototipo CU16

En esta pantalla se mostrarán las canciones que tu mismo subes al servidor, las cuales solo tu podrás ver y gestionar, teniendo la posibilidad de agregar hasta 250 canciones.

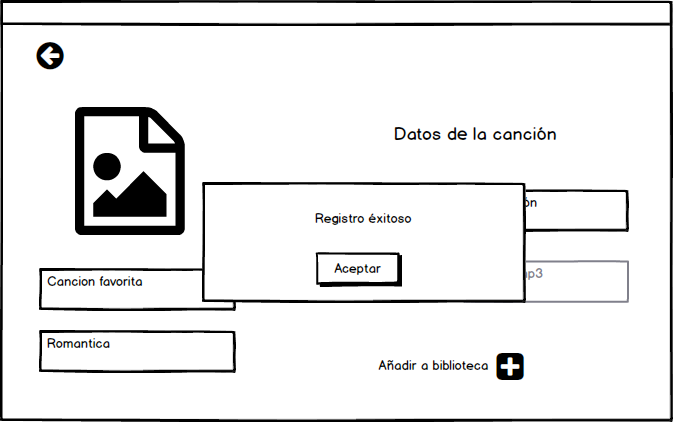
## CU17 Agregar canción a biblioteca

### Prototipo CU17

En esta pantalla se agregarán las canciones a la biblioteca propia, dónde se irán registrando una por una, se piden los datos como nombre del álbum, género al que pertenece, una imagen de portada, el nombre de la canción, así como el archivo de audio.



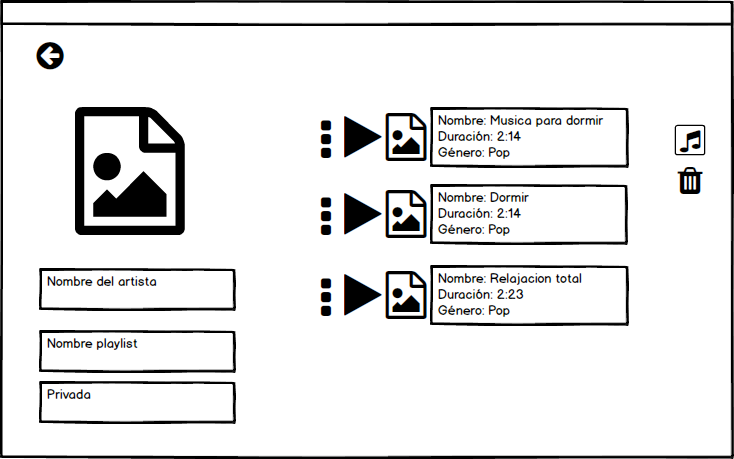
Mensaje de confirmación de que se ha registrado una canción en la biblioteca.



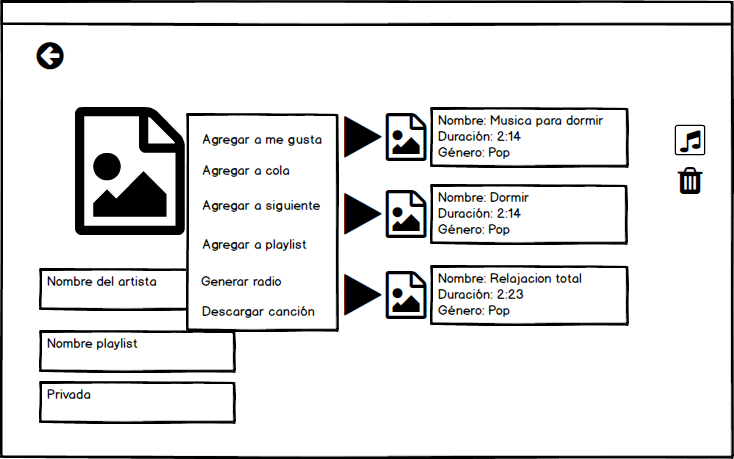
## CU18 Mostrar canciones playlist

### Prototipo CU18

Este prototipo se muestra cada vez que se accede al contenido de cualquier playlist que sea creada por el mismo usuario agregando la funcionalidad de eliminar alguna canción de esta misma playlist, de tratarse de una playlist pública de la cual no eres dueño está funcionalidad no estará disponible.



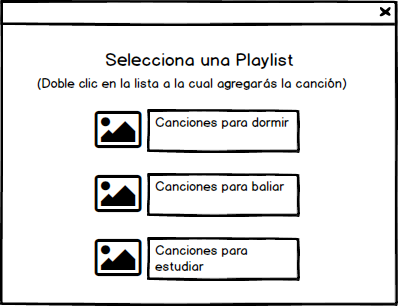
Este caso de uso tiene como flujos alternos las funcionalidades de “Agregar a cola”, “Agregar a siguiente”, “Descargar canción”, “Generar radio” y “Agregar a me gusta”.



## CU19.- Agregar canción a playlist

### Prototipo CU19

En esta pantalla se muestra la funcionalidad de agregar una canción previamente seleccionada a una playlist que ya se haya creado con anterioridad.



Mensaje de confirmación de éxito.

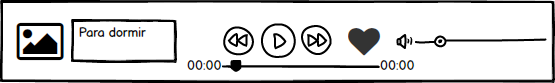


## CU20 Reproducir canción

### Prototipo CU20

Este prototipo forma parte de la página principal, se va a inicializar siempre que se pulse en reproducir en cualquiera de los casos de uso dónde se muestren las canciones y se puedan reproducir, carga la información de la canción, así como el archivo de audio.

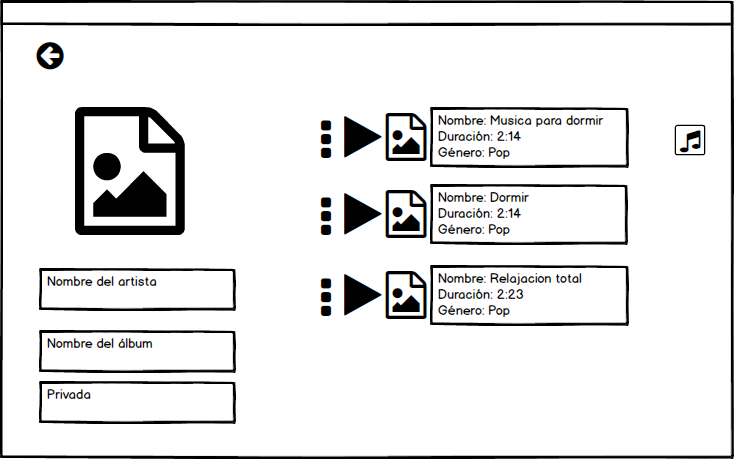
Se podrán realizar las funcionalidades de reproducir, anterior y siguiente canción, así como adelantar y subir el volumen.



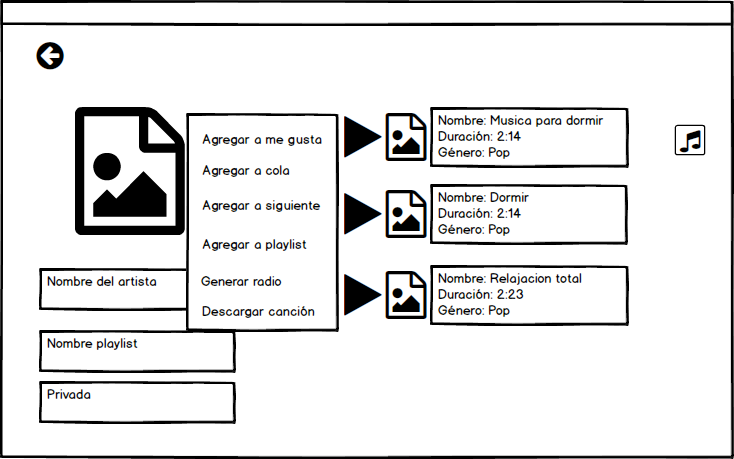
## CU21.- Mostrar canciones álbum

Este prototipo surge de seleccionar un álbum, después de ello se mostrarán las canciones para su reproducción o una de las funcionalidades en las cuales podrá gestionar las acciones con estas canciones.

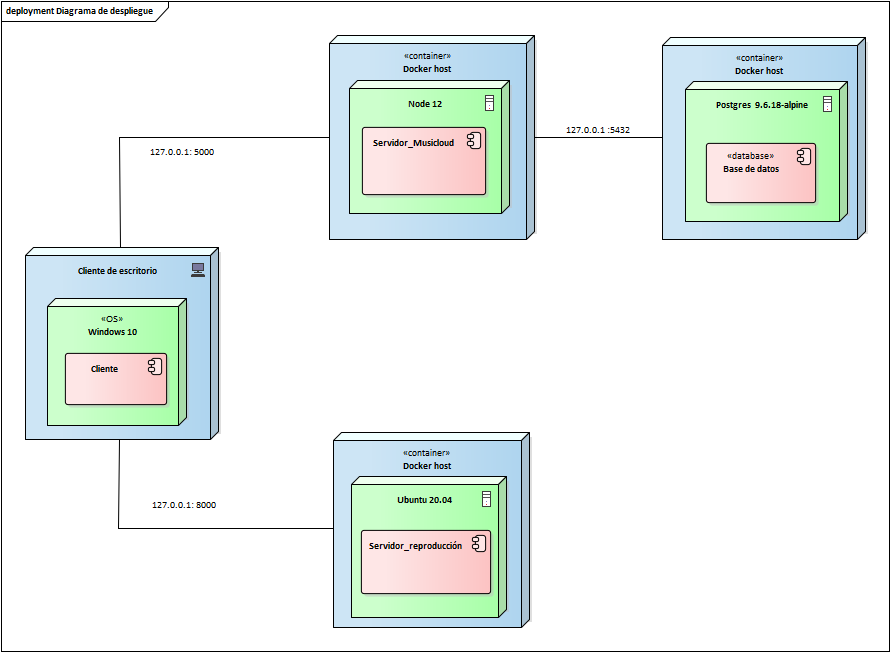
### Prototipo CU21



Al igual que el mostrar canciones de playlist, de aquí pueden surgir los flujos de “Agregar a me gusta”, “Agregar a cola”, “Agregar a siguiente”, “Agregar a playlist”, “Descargar canción” y “Generar Radio”.

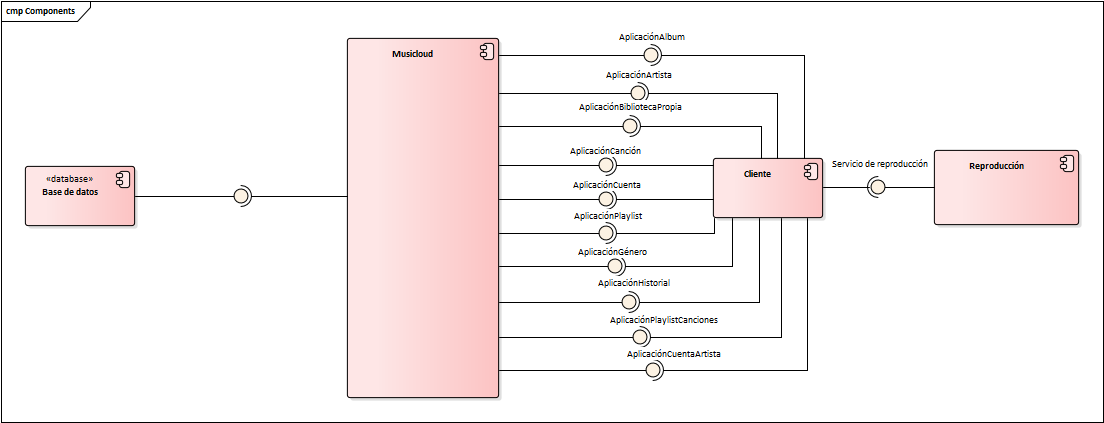


## Diagrama de despliegue

El siguiente diagrama de despliegue muestra como están distribuidos los servidores, como estos se comunican con el cliente de escritorio. El servidor\_Musicloud se encuentra dentro de un contenedor cuya imagen es de Node 12, este es el encargado de hacer todas las solicitudes de lectura y escritura con la base de datos, el cliente se encuentra una computadora de escritorio con sistema operativo Windows 10, el segundo servidor llamado “Servidor\_reproducción”, es el encargado de proveer los archivos de audio al cliente para la reproducción en línea. La base de datos se encuentra en un contenedor alpine aprovisionado con postgres, por lo cual nuestra base de datos se encuentra escrita en PostgreSQL.

## Diagrama de componentes

El siguiente diagrama de componentes muestra las interfaces provistas por el servidor Musicloud, cada interfaz contiene los endpoints que serán expuestos y así ser consumidos por el cliente, también se muestra la interfaz provista del servicio de reproducción, por último, la interfaz provista de la base de datos hacía nuestra API REST de Musicloud.



## Diseño de API con Canvas

A continuación, se busca identificar la lista completa de las metas que deben cumplir las clases de usuario en Musicloud por medio de una tabla de seis columnas; ¿Quién? ¿Qué? ¿Cómo?, Entrada, Salida y meta. Como se muestra en la ilustración 1. (Lauret, 2019)

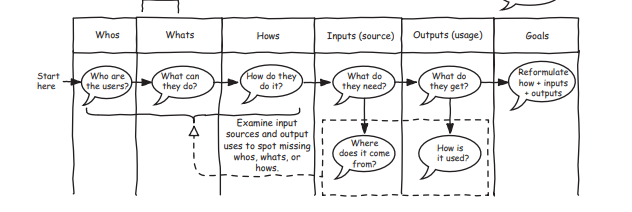


Ilustración 1 Tabla de metas de una API

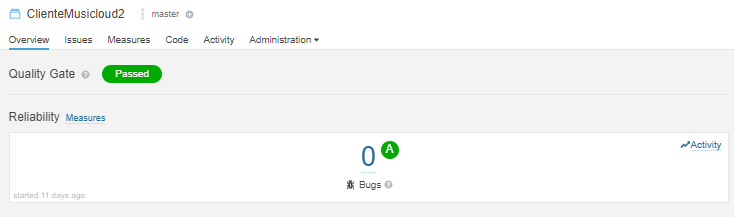
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ¿Quién? | ¿Qué? | ¿Cómo? | Entrada | Salida | Metas |
| Consumidor de música | Reproducir música | Buscar canciones | Catálogo de canciones, consulta (Por usuario) | Lista de Canciones | Buscar canciones usando una consulta |
| Buscar playlist | Catálogo de playlist, consulta (por usuario) | Lista de Playlist | Buscar playlist usando una consulta |
| Añadir canción a cola | Cola de reproducción, canción (por usuario) |  | Añadir canción a cola |
| Añadir canción a “Me gusta” | Catálogo de “Me gusta”, canción (por usuario) |  | Añadir canción a la playlist de “Me gusta” |
| Descargar música | Canción (por usuario) |  | Descargar canción |
| Administrar playlist | Agregar playlist | Catálogo de playlist, playlist (por usuario) |  | Agregar playlist al catálogo de playlist |
| Eliminar playlist | Catálogo de playlist, playlist (por usuario) |  | Eliminar playlist del catálogo |
| Actualizar playlist | Playlist, tipo (por usuario) |  | Actualizar datos de playlist |
| Buscar playlist | Catálogo de playlist, consulta (por usuario) | Lista de playlist | Buscar playlist usando una consulta |
| Añadir canción a playlist | Playlist, canción (Por usuario) |  | Añadir canción a playlist |
| Buscar contenido | Buscar Artista | Catálogo de artistas, consulta (por usuario) | Lista de artistas | Buscar artista usando una consulta |
| Buscar álbum | Catálogo de álbumes, consulta (por usuario) | Lista de álbumes | Buscar álbum usando una consulta |
| Gestionar biblioteca | Añadir canción a biblioteca | Biblioteca, canción (por usuario) |  | Agregar canción a biblioteca |
| Eliminar canción de biblioteca | Biblioteca, canción (por usuario) |  | Eliminar canción de biblioteca |
| Consultar biblioteca | Biblioteca, consulta (por usuario) | Biblioteca | Mostrar biblioteca |
| Gestionar cuenta | Crear cuenta | Campos del formulario (por usuario) |  | Crear cuenta |
| Modificar cuenta | Cuenta, campos del formulario (por usuario) |  | Modificar datos de la cuenta |
| Recuperar contraseña | Correo (por usuario) | Link para cambiar contraseña | Recuperar contraseña |
| Iniciar sesión | Catálogo de cuentas, cuenta (por usuario) |  | Iniciar sesión |
| Creador de contenido | Gestionar discografía | Mostrar discografía | Catálogo de Discografía, consulta (Por usuario) | Discografía de creador | Mostrar discografía de creador de contenido |
| Añadir álbum | Catálogo de discografía, Álbum (por usuario) |  | Añadir álbum a discografía |
| Añadir canciones | Álbum, canción (por usuario) |  | Añadir canción a álbum |
| Eliminar canción | Álbum, canción (por usuario) |  | Eliminar canción de álbum |
| Dar de baja | Eliminar cuenta | Cuenta (por usuario) |  | Eliminar cuenta de creador de contenido |

# Anexos

## Análisis estático de código con SonarQube

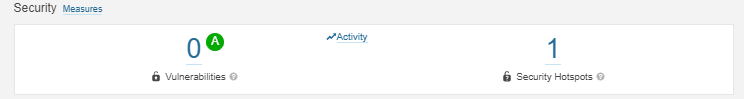
### Confiabilidad

Como se muestra a continuación el proyecto no presenta algún bug que pueda afectar a la confiabilidad de este.



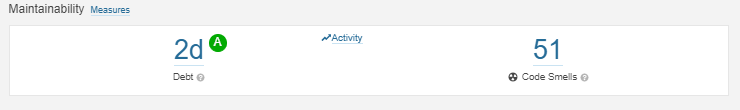
### Seguridad

Como se muestra a continuación se presenta un Security Hotspots, este surge por el hecho de usar la clase Random, lo cuál considero que en realidad no afecta a la seguridad del sistema dado que la clase es utilizada para generar un número aleatorio para el nombre del archivo de las canciones por lo cuál no afecta al proyecto.



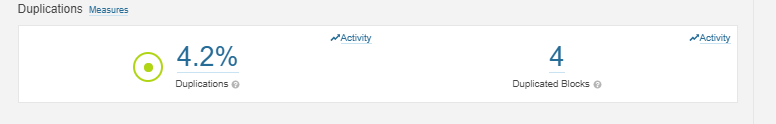
### Mantenibilidad

A continuación, podemos ver que nos muestra 2 días de deuda técnica a consecuencia de los 51 code smells, los cuáles en su mayoría son causados por manejar las Excepciones de manera general, decidí dejarlo así dado que hasta el momento fue la manera que encontré de manejar aquellos errores que surgen al hacer peticiones con el servidor.



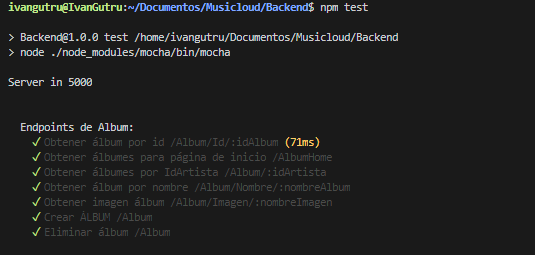
### Duplicación

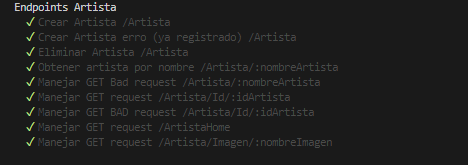
Hace referencia a las pages CrearPLaylist y CrearArtista, dado que en ambas se utiliza este código duplicado para subir las portadas de las playlist y artistas respectivamente. Considero que debe haber una forma de combatir esto tal vez con una clase auxiliar, pero por cuestiones de tiempo decidí dejarlo así.

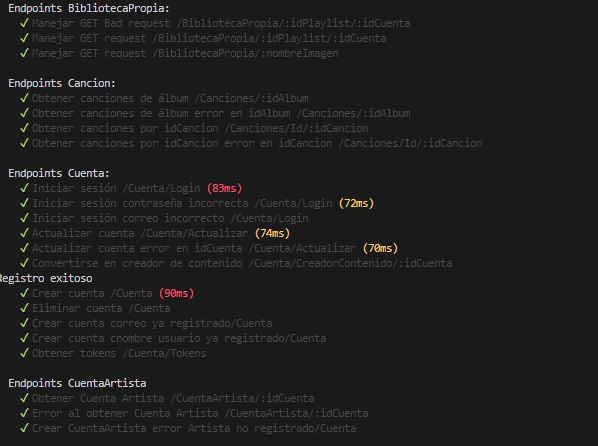


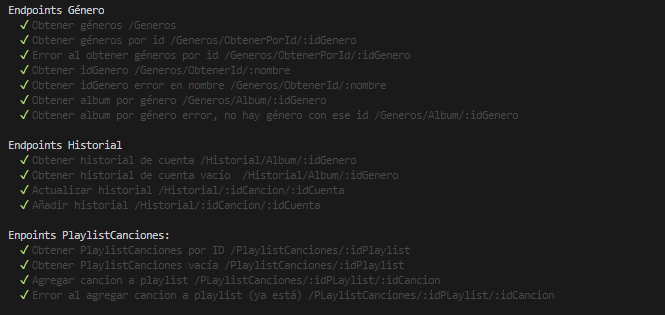
## Pruebas del servidor

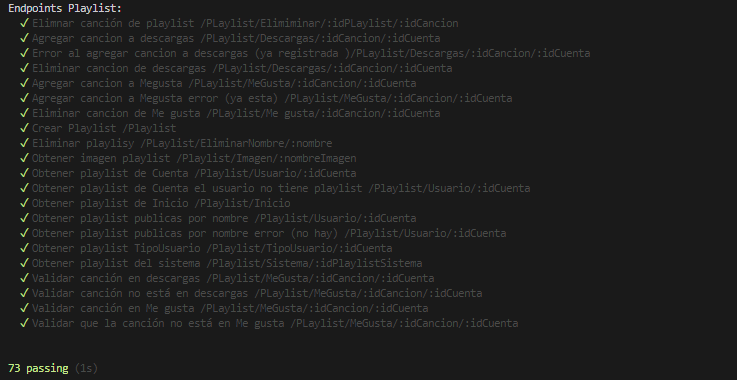
Las pruebas del servidor fueron realizadas con ayuda de Mocha, framework de JavaScript con ayuda de la librería Chai HTTP extensión de Chai. A continuación, se muestran todas las pruebas de todos los endpoints que se realizaron para el servidor.











Dando como resultado 73 pruebas las cuales salieron positivas.

# Referencias

Lauret, A. (2019). *The Design of Web APIs.* United States of America: Manning Publications Co.

https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework