

Documento para almacenar los diferentes requisitos y artefactos para el  
Proyecto de SpotyMe

# Documentación de SpotyMe

Ricardo Moguel Sánchez – S18012183  
Cesar Sergio Martínez Palacios – S18012143

---

# Tabla de Contenidos

<b>1. Introducción</b>	<b>4</b>
1.1. Problema Plantado	4
1.2. Objetivo	4
<b>2. Requerimientos</b>	<b>5</b>
2.1. Contexto	5
2.1.1. Modelo del Dominio	5
2.1.2. Modelo de Contexto	6
2.2. Clases de Usuario	7
2.3. Modelo de Casos de Uso	8
2.4. Prototipo de UI (Baja Fidelidad)	9
2.4.1. CU-A-1: Gestionar solicitudes de canciones	9
2.4.2. CU-A-2: Banear usuario	9
2.4.3. CU-U-1: Registrar cuenta	10
2.4.4. CU-U-2: Actualizar cuenta	11
2.4.5. CU-U-3: Agregar canción personal	11
2.4.6. CU-U-4: Gestionar biblioteca privada	12
2.4.7. CU-U-5: Gestionar lista de reproducción	12
2.4.8. CU-U-6: Publicar canción	13
2.4.9. CU-U-7: Buscar canción	13
2.4.10. CU-U-8: Consultar canción	14
2.5. Requisitos Funcionales	15
2.6. Requisitos No Funcionales	16
<b>3. Diseño</b>	<b>17</b>
3.1. Diseño arquitectónico	17
3.1.1. Vista de casos de uso	17
3.1.2. Vista de implementación	18
3.1.3. Vista de procesos	22
3.1.4. Vista de despliegue	25
3.2. Modelo de Datos	26
3.2.1. Base de Datos de Cuentas	26
3.2.2. Base de datos Library	26
3.3. Descripciones de Casos de Uso	27

3.3.1.	Casos de Uso para Administrador .....	27
3.3.2.	Casos de Uso para Usuario.....	30
4.	Construcción .....	45
4.1.	Selección justificada de pila tecnológica .....	45
4.1.1.	Aplicación de Cliente Rico .....	45
4.1.2.	Aplicación de servicios en API.....	46
4.2.	Estándar de Codificación.....	47
4.3.	Reportes de Análisis Estático de Código.....	47
4.4.	Prácticas de Construcción Realizadas .....	50
5.	Pruebas.....	51
5.1.	Plan de Pruebas para la API.....	51
5.1.1.	Introducción .....	51
5.1.2.	Contexto .....	51
5.1.3.	Registro de Riesgos .....	52
5.2.	Procedimiento de Prueba .....	53
5.2.1.	Subprocesos de Prueba .....	54
5.2.2.	Formato de Casos de Prueba .....	54
5.2.3.	Criterios de Finalización .....	55
5.2.3.1.	Casos de uso asociados .....	56
5.3.	Resultados del Plan de Pruebas .....	56
5.3.1.	Resultados de Pruebas del Servicio de Cuentas del API.....	56
5.3.2.	Resultados de Pruebas del Servicio de Biblioteca Pública del API .....	66
5.3.3.	Resultados de Pruebas del Servicio de Biblioteca Privada del API .....	76
5.3.4.	Resultados de Pruebas del Servicio de Streaming del API .....	84
5.4	Conclusiones de Pruebas .....	87
6.	Estrategia de despliegue.....	88
6.1.	Despliegue de Servidor .....	88
6.2.	Despliegue de Clientes.....	89
7.	Conclusiones .....	89
8.	Referencias.....	90

## ***Tabla de Figuras***

Figura 1 Modelo del Dominio de SpotyMe .....	5
Figura 2 Modelo de Contexto de SpotyMe .....	6
Figura 3 Modelo de Actores de SpotyMe .....	7
Figura 4 Diagrama de Casos de Uso .....	8
Figura 5 Prototipo de Lista de Solicitudes.....	9
Figura 6 Prototipo de Banear Usuario .....	9
Figura 7 Prototipo de Inicio de Sesión .....	10
Figura 8 Prototipo de Registro de Usuario.....	10
Figura 9 Prototipo de Actualización de Usuario .....	11
Figura 10 Prototipo de Agregación de Canción .....	11
Figura 11 Prototipo de Biblioteca Personal .....	12
Figura 12 Prototipo de Lista de Reproducción .....	12
Figura 13 Prototipo de Subir Canción Pública.....	13
Figura 14 Prototipo de Buscar Canción.....	13
Figura 15 Prototipo de Detalles Canción .....	14
Figura 16 Diagrama UML de casos de uso.....	17
Figura 17 Diagrama de Componentes de Cuentas .....	19
Figura 18 Diagrama de Componentes de Biblioteca Pública .....	19
Figura 19 Diagrama de Componentes de Biblioteca Privada .....	20
Figura 20 Diagrama de Componentes de Streaming .....	20
Figura 21 Diagrama de Componentes de API.....	21
Figura 22 Diagrama de Proceso de petición GET .....	22
Figura 23 Diagrama de Proceso de petición View .....	22
Figura 24 Diagrama de proceso de petición POST .....	23
Figura 25 Diagrama de Proceso de petición LogIn .....	23
Figura 26 Diagrama de Proceso de petición PUT .....	24
Figura 27 Diagrama de Despliegue de SpotyMe .....	25
Figura 28 Diagrama E-R de base de datos de Cuentas .....	26
Figura 29 Diagrama E-R de base de datos de Bibliotecas.....	26
Figura 30 Captura de sobrevista del análisis estático.....	48
Figura 31 Captura de bugs del análisis estático .....	48
Figura 32 Captura de vulnerabilidades del análisis estático .....	49
Figura 33 Captura de code smells del análisis estático .....	49

# **1. Introducción**

## **1.1. Problema Plantado**

Para las experiencias educativas de Desarrollo de sistemas en red y Desarrollo de aplicaciones se busca crear una aplicación de escritorio y una web app que soliciten funcionalidades a un servidor por medio de microservicios para un cliente con el cual el usuario interactúa. Se pretende desarrollar un sistema de gestión y reproducción de música para abarcar las tecnologías requeridas en ambas experiencias educativas.

El sistema de gestión de música denominado “SpotyMe” debe permitir a un usuario ingresar al sistema y tener la habilidad de subir canciones a una biblioteca privada y reproducir canciones de esta misma biblioteca. Adicionalmente tendrá una biblioteca pública a la cual los usuarios pueden subir música de su biblioteca privada después de verificación por un administrador. Finalmente, todos los usuarios deben tener acceso a consultar y reproducir música de la biblioteca pública.

## **1.2. Objetivo**

El objetivo de la aplicación SpotyMe es brindar microservicios que trabajen en conjunto dentro de un servidor multihilo para crear un sistema de gestión de música con un front-end para aplicación web y de escritorio. Para el cliente web app y de escritorio de SpotyMe se creará un registro de cuentas con un apodo, correo electrónico y contraseña. Una vez registrados, los usuarios podrán acceder a las funcionalidades de reproducción de música después de un inicio de sesión para la biblioteca pública y privada. Un usuario podrá subir música solo a su biblioteca privada y podrá solicitar subida a la biblioteca pública que tendrá que ser aprobada por un administrador antes de ser guardada a esta biblioteca.

## 2. Requerimientos

### 2.1. Contexto

En esta sección se muestran los modelos que representan el contexto del sistema de gestión de música a elaborar.

#### 2.1.1. Modelo del Dominio

En la siguiente figura se muestra el modelo del dominio para un sistema de gestión de música.

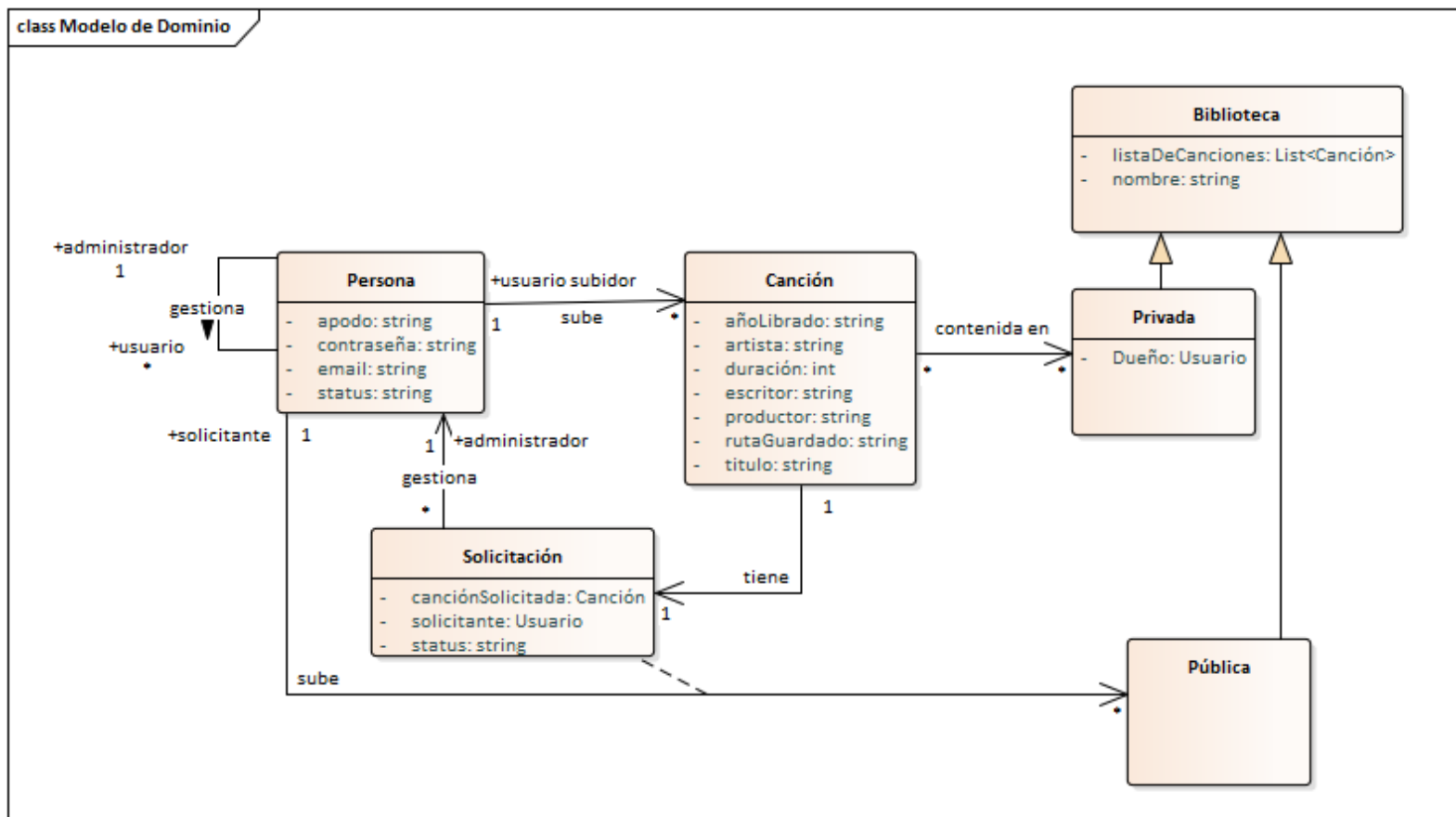


Figura 1 Modelo del Dominio de SpotyMe

### 2.1.2. Modelo de Contexto

En la siguiente figura se muestra el modelo de contexto para el sistema de gestión de música SpotyMe.

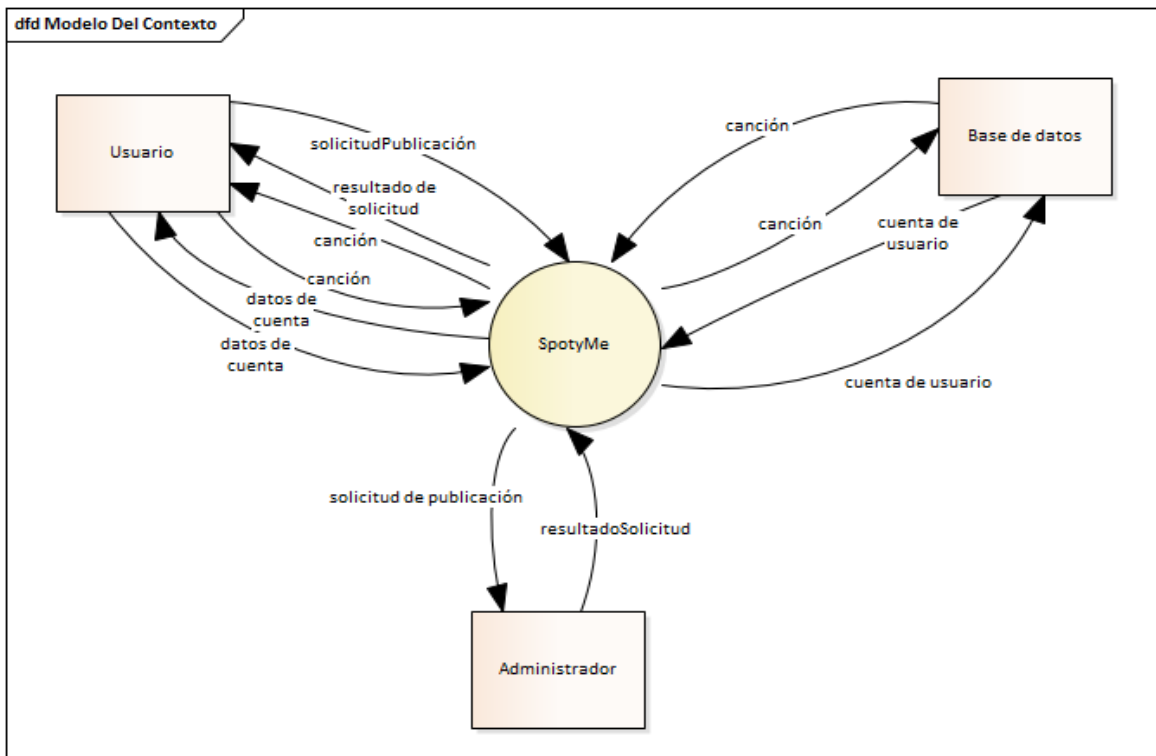


Figura 2 Modelo de Contexto de SpotyMe

## 2.2. Clases de Usuario

En la siguiente figura se muestra el modelo de actores identificados para el sistema SpotyMe

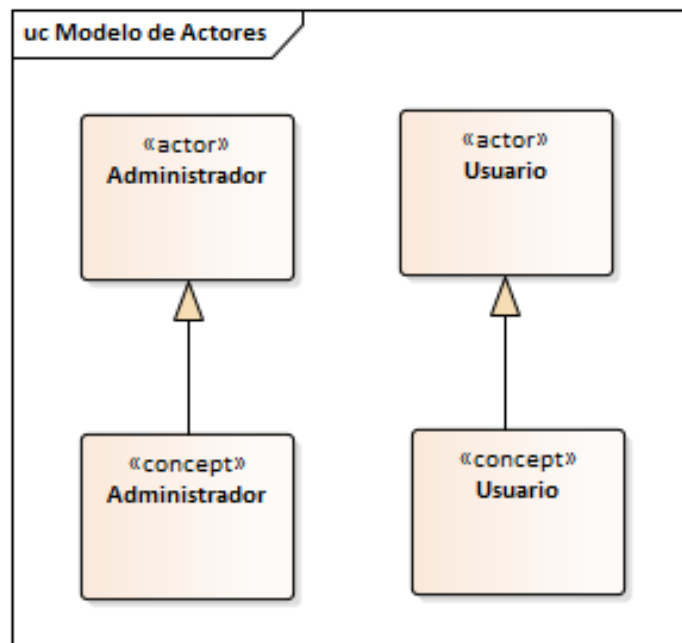


Figura 3 Modelo de Actores de SpotyMe

### Administrador

El administrador está encargado de aprobar o rechazar solicitudes para subir canciones a la biblioteca pública. Adicionalmente tiene la funcionalidad de negarle el acceso al sistema a un usuario en el caso de realizar solicitudes de material ofensivo a la biblioteca pública.

### Usuario

El usuario puede registrar una cuenta para acesar el sistema SpotyMe. Una vez ingresado con cuenta tiene a su disponibilidad la funcionalidad de consultar canciones en la biblioteca pública, subir, actualizar y eliminar música en su biblioteca privada. Adicionalmente tiene la funcionalidad de realizar una solicitud para subir una canción a la biblioteca publica para que otros usuarios puedan escucharla.



## 2.3. Modelo de Casos de Uso

En la siguiente figura se encuentra la primera propuesta de casos de uso para los dos diferentes actores identificados.

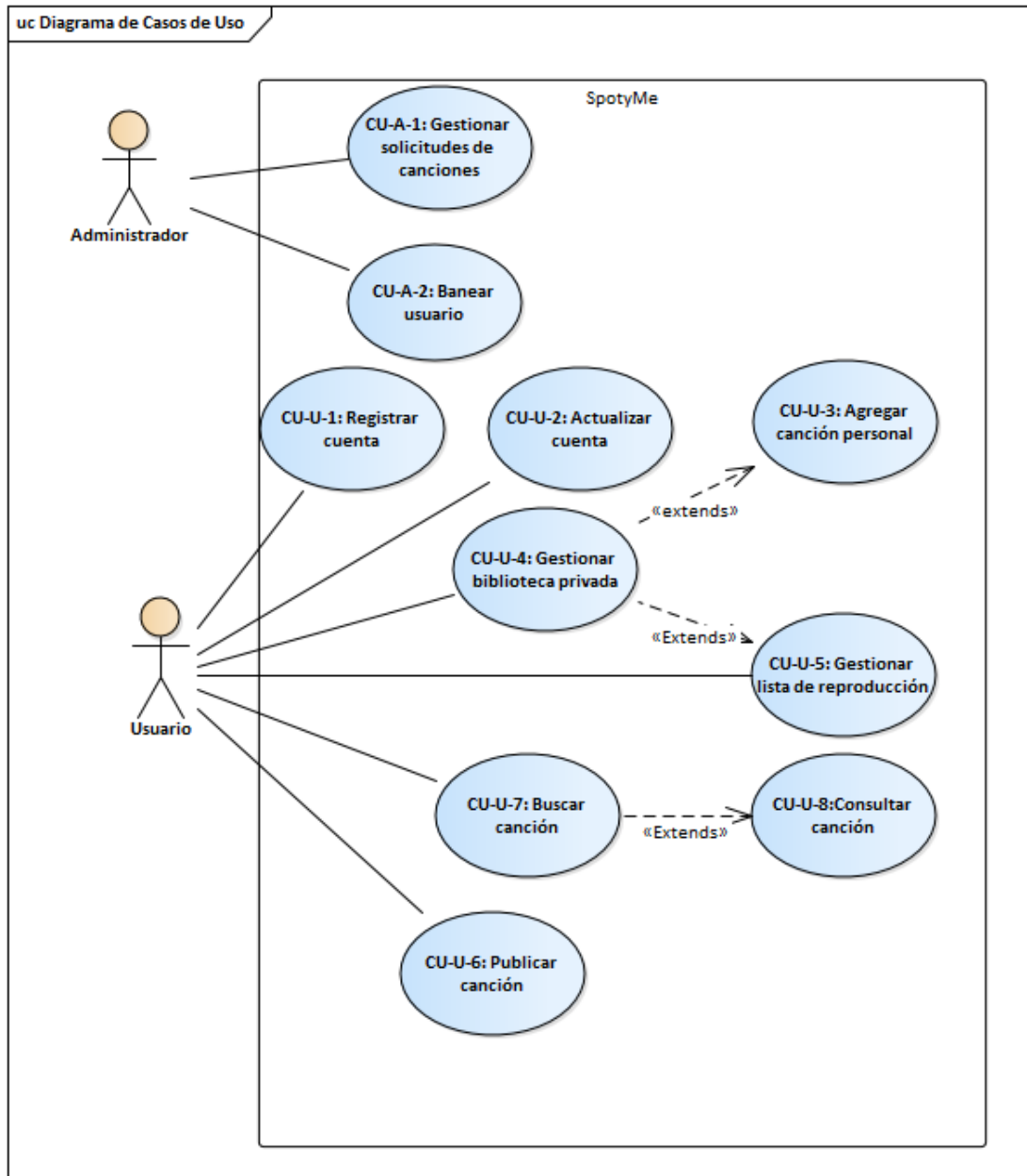


Figura 4 Diagrama de Casos de Uso

## 2.4. Prototipo de UI (Baja Fidelidad)

En esta sección se muestran los prototipos de interfaz gráfica usados en los diferentes casos de usos identificados.

### 2.4.1. CU-A-1: Gestionar solicitudes de canciones

En la siguiente figura se muestra el prototipo de la ventana Lista de Solicitudes de Aprobación para el caso de uso CU-A-1: Gestionar solicitudes de canciones.



Figura 5 Prototipo de Lista de Solicitudes

### 2.4.2. CU-A-2: Banear usuario

En la siguiente figura se muestra el prototipo de la ventana BanearUsuario para el caso de uso CU-A-2: Banear usuario.

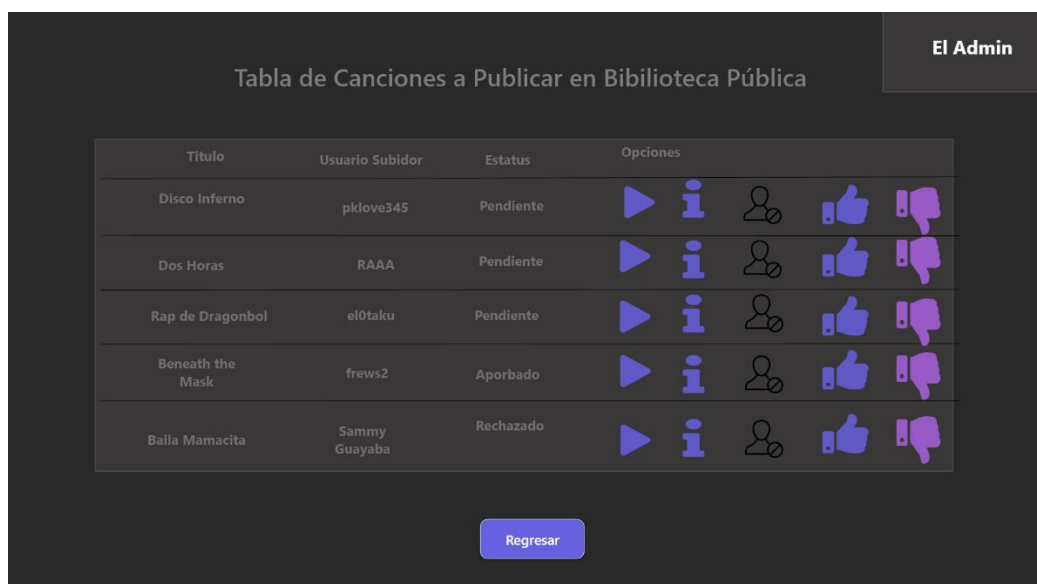
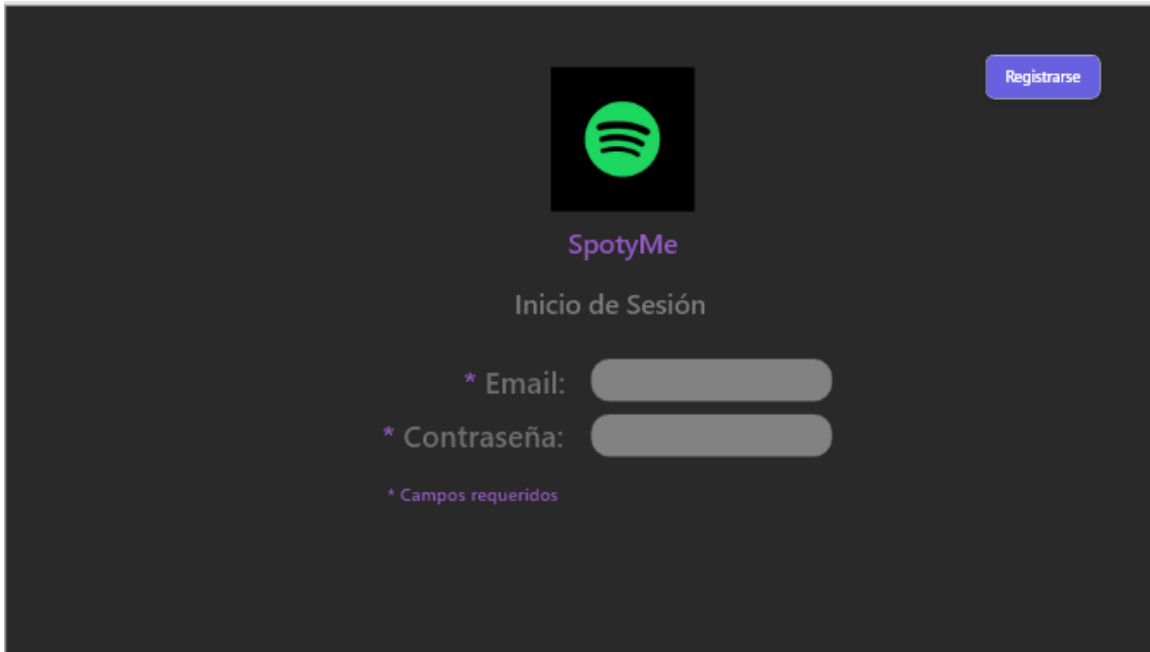


Figura 6 Prototipo de Banear Usuario

### 2.4.3. CU-U-1: Registrar cuenta

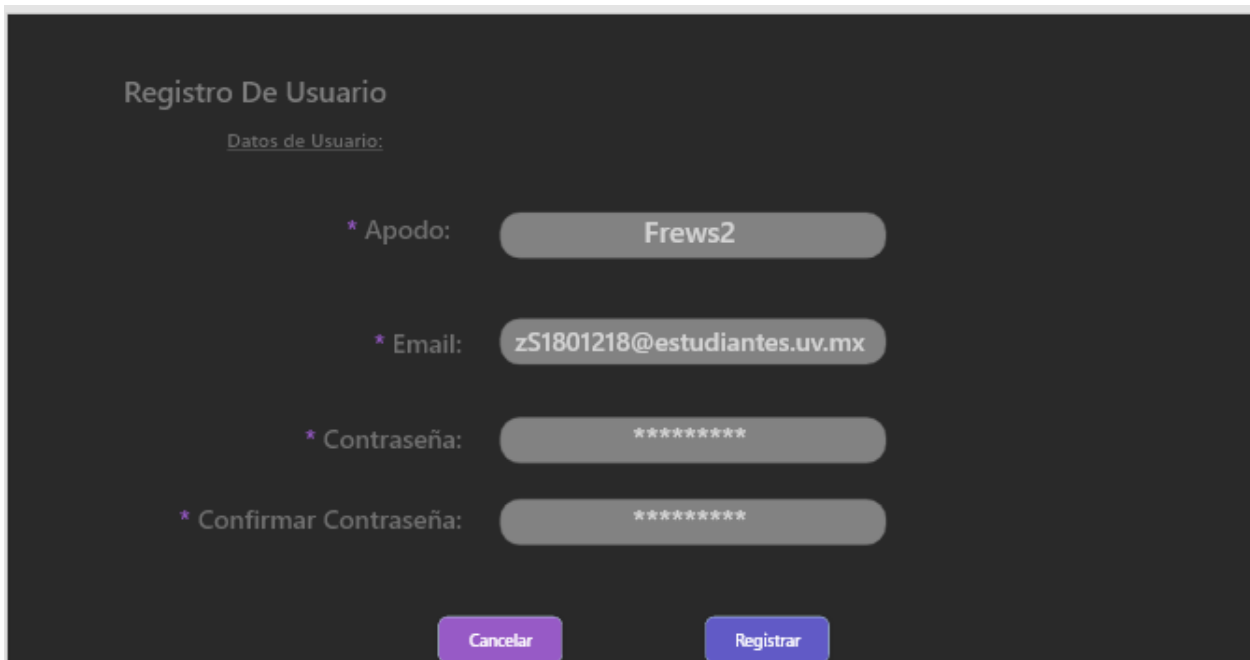
En la siguiente figura se muestra el prototipo de la ventana Inicio De Sesión para el caso de uso CU-U-1: Registrar cuenta.



El prototipo de la ventana 'Inicio De Sesión' para 'SpotyMe' tiene un fondo oscuro. En la parte superior central hay un logo verde con tres líneas horizontales. A la derecha superior hay un botón rectangular azul con el texto 'Registrarse'. Debajo del logo, el texto 'SpotyMe' aparece en color verde. Justo debajo, el título 'Inicio de Sesión' está en un gris más claro. Los campos de entrada están centrados: '\* Email:' con un campo gris, '\* Contraseña:' con un campo gris, y '\* Campos requeridos' en un color verde más tenue. Los campos de entrada tienen un efecto de sombra y un borde sutil.

Figura 7 Prototipo de Inicio de Sesión

En la siguiente figura se muestra el prototipo de la ventana Registro De Usuario.



El prototipo de la ventana 'Registro De Usuario' tiene un fondo oscuro. El título 'Registro De Usuario' está en un gris claro. Debajo de él, el subtítulo 'Datos de Usuario:' es en un verde más tenue. Los campos de entrada están distribuidos verticalmente: '\* Apodo:' con el valor 'Frews2' en un campo gris; '\* Email:' con el valor 'zS1801218@estudiantes.uv.mx' en un campo gris; '\* Contraseña:' con caracteres de relleno '\*\*\*\*\*' en un campo gris; y '\* Confirmar Contraseña:' con caracteres de relleno '\*\*\*\*\*' en un campo gris. En la parte inferior, hay dos botones rectangulares azules: 'Cancelar' a la izquierda y 'Registrar' a la derecha. Los campos de entrada tienen un efecto de sombra y un borde sutil.

Figura 8 Prototipo de Registro de Usuario

#### 2.4.4. CU-U-2: Actualizar cuenta

En la siguiente figura se muestra el prototipo de la ventana Actualización De Usuario para el caso de uso CU-U-2: Actualizar cuenta.

El prototipo de la ventana 'Actualización De Usuario' tiene un fondo oscuro. En la esquina superior derecha hay un botón 'Frews2'. El título 'Actualización De Usuario' está en la parte superior izquierda. Debajo del título, el texto 'Datos de Usuario:' precede a cuatro campos de entrada. El primer campo, etiquetado '\* Apodo:', contiene el texto 'Frews2'. El segundo, '\* Email:', contiene 'zS1801218@estudiantes.uv.mx'. Los dos campos siguientes, '\* Contraseña:' y '\* Confirmar Contraseña:', muestran caracteres ocultos por asteriscos. En la parte inferior, hay dos botones: 'Cancelar' a la izquierda y 'Actualizar más datos' a la derecha.

Figura 9 Prototipo de Actualización de Usuario

#### 2.4.5. CU-U-3: Agregar canción personal

En la siguiente figura se muestra el prototipo de la ventana Agregación De Canción para el caso de uso CU-U-3: Agregar canción personal.

El prototipo de la ventana 'Subir Canción A Mi Biblioteca' tiene un fondo oscuro. En la esquina superior derecha hay un botón 'Frews2'. El título 'Subir Canción A Mi Biblioteca' está en la parte superior izquierda. Debajo del título, el texto 'Datos de Canción:' precede a un formulario con varias etiquetas y valores. Las etiquetas incluyen 'Titulo:', 'Artista:', 'Duración:', 'Año Librado:', 'Album:', 'Género', 'Escrito por:' y 'Producido por:'. Los valores correspondientes son 'Disco Inferno', 'The Trammps', '3:34', '1976', 'The Best Of The Trammps', 'Disco', 'Leroy Green, Ron Kersey, Tyrone Kersey' y 'Earl Young, Norman Harris, Ronnie Baker'. Debajo del formulario, hay un botón 'Adjuntar Canción' con un ícono de archivo. En la parte inferior, hay dos botones: 'Cancelar' a la izquierda y 'Subir' a la derecha.

Figura 10 Prototipo de Agregación de Canción

#### 2.4.6. CU-U-4: Gestionar biblioteca privada

En la siguiente figura se muestra el prototipo de la ventana Biblioteca Personal para el caso de uso CU-U-4: Gestionar biblioteca privada.

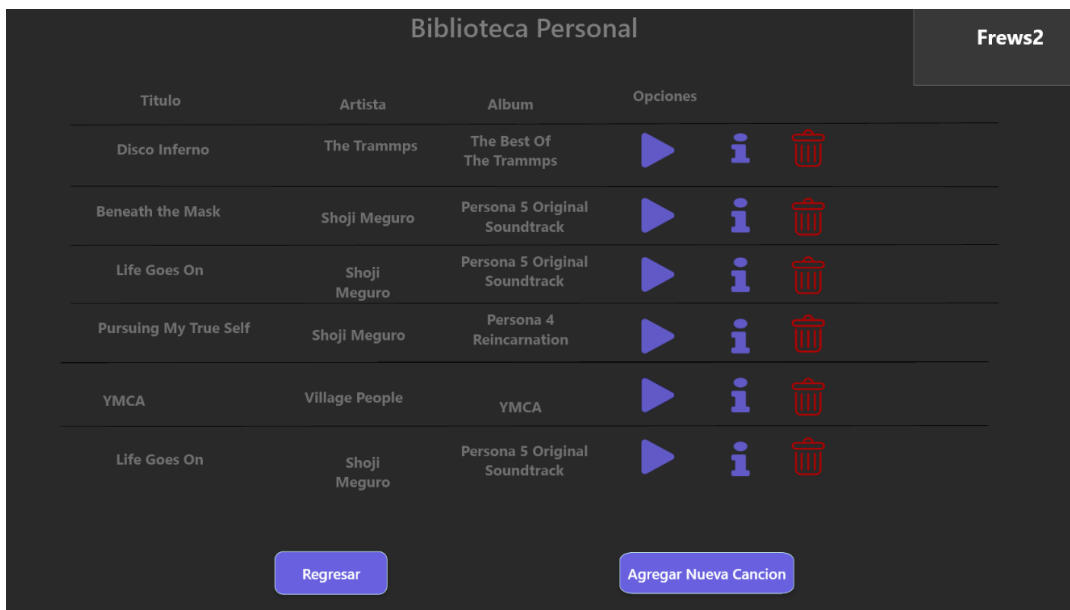


Figura 11 Prototipo de Biblioteca Personal

#### 2.4.7. CU-U-5: Gestionar lista de reproducción

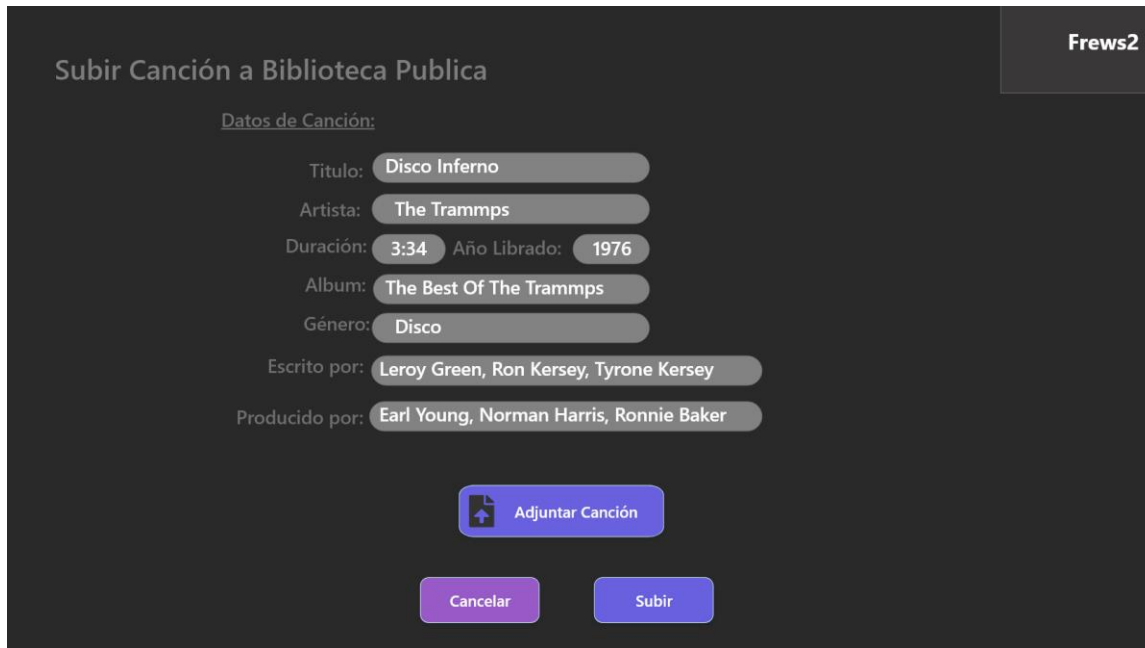
En la siguiente figura se muestra el prototipo de la ventana Lista De Reproducción para el caso de uso CU-U-5: Gestionar lista de reproducción.



Figura 12 Prototipo de Lista de Reproducción

#### 2.4.8. CU-U-6: Publicar canción

En la siguiente figura se muestra el prototipo de la ventana Subir Canción Pública para el caso de uso CU-U-6: Publicar canción.



Subir Canción a Biblioteca Publica

Frews2

Datos de Canción:

Título: Disco Inferno

Artista: The Tramps

Duración: 3:34 Año Librado: 1976

Album: The Best Of The Tramps

Género: Disco

Escrito por: Leroy Green, Ron Kersey, Tyrone Kersey

Producido por: Earl Young, Norman Harris, Ronnie Baker

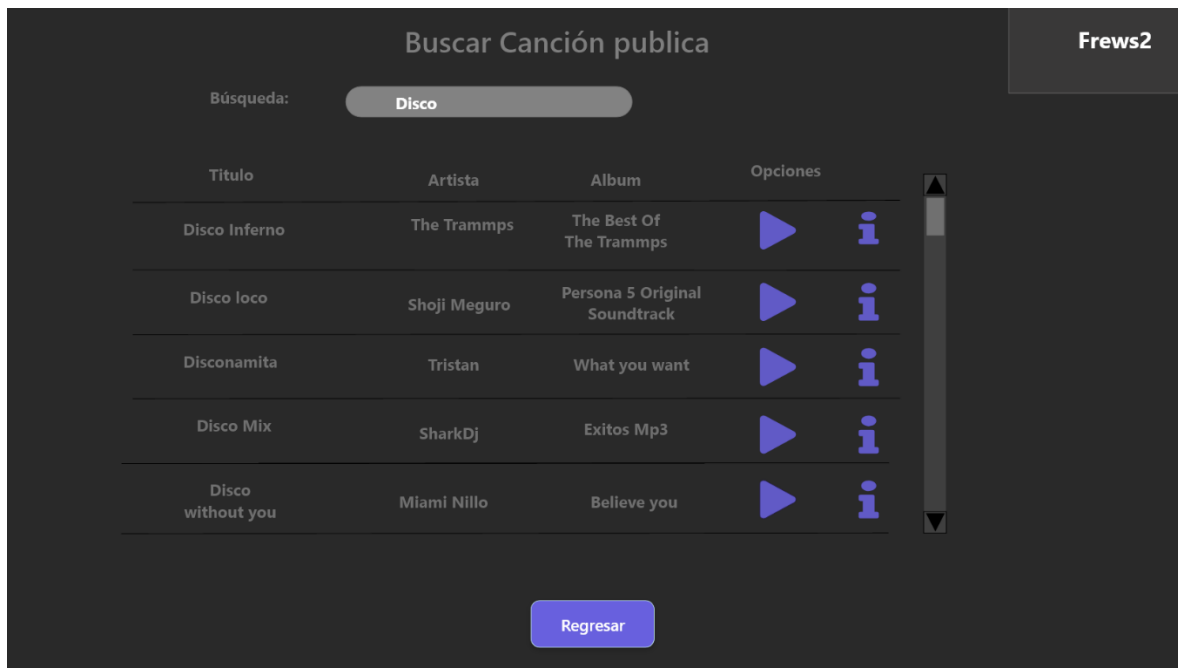
Adjuntar Canción

Cancelar Subir

Figura 13 Prototipo de Subir Canción Pública

#### 2.4.9. CU-U-7: Buscar canción

En la siguiente figura se muestra el prototipo de la ventana Buscar Canción para el caso de uso CU-U-7: Buscar canción.



Buscar Canción publica

Frews2

Búsqueda: Disco

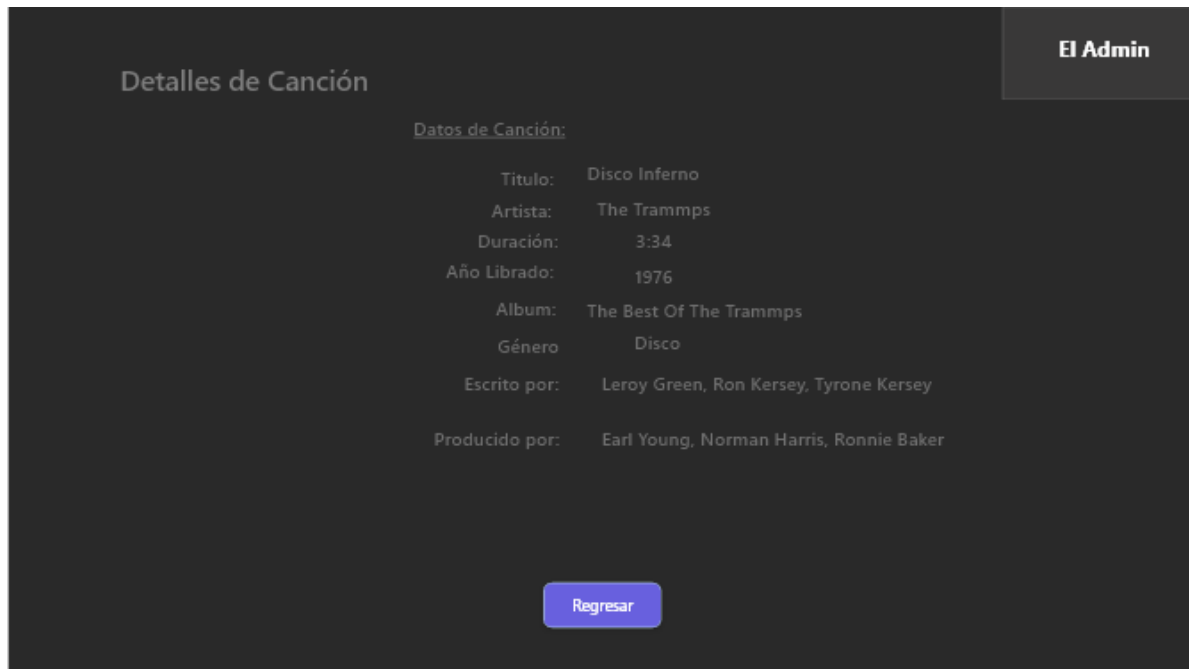
Título	Artista	Album	Opciones
Disco Inferno	The Tramps	The Best Of The Tramps	▶ i
Disco loco	Shoji Meguro	Persona 5 Original Soundtrack	▶ i
Disconamita	Tristan	What you want	▶ i
Disco Mix	SharkDj	Exitos Mp3	▶ i
Disco without you	Miami Nillo	Believe you	▶ i

Regresar

Figura 14 Prototipo de Buscar Canción

#### 2.4.10. CU-U-8: Consultar canción

En la siguiente figura se muestra el prototipo de la ventana Detalles Canción para el caso de uso CU-U-8: Consultar canción.



El prototipo de la ventana 'Detalles de Canción' presenta un encabezado con el título 'Detalles de Canción' a la izquierda y un botón 'El Admin' a la derecha. El contenido principal muestra los datos de la canción en un formato de lista de pares. Los datos son: Título: Disco Inferno, Artista: The Trammps, Duración: 3:34, Año Librado: 1976, Album: The Best Of The Trammps, Género: Disco, Escrito por: Leroy Green, Ron Kersey, Tyrone Kersey, and Producido por: Earl Young, Norman Harris, Ronnie Baker. En la parte inferior central hay un botón 'Regresar'.

Detalles de Canción		El Admin
<u>Datos de Canción:</u>		
Título:	Disco Inferno	
Artista:	The Trammps	
Duración:	3:34	
Año Librado:	1976	
Album:	The Best Of The Trammps	
Género:	Disco	
Escrito por:	Leroy Green, Ron Kersey, Tyrone Kersey	
Producido por:	Earl Young, Norman Harris, Ronnie Baker	
<a href="#">Regresar</a>		

Figura 15 Prototipo de Detalles Canción

## 2.5. Requisitos Funcionales

En la siguiente table se muestran los requisitos funcionales para el sistema distribuido SpotyMe.

Tabla 1 Requisitos Funcionales

Identificador de requisito	Descripción de requisito
RF-01	El sistema debe permitir al Usuario registrar una cuenta para poder acceder a las bibliotecas
RF-02	El sistema debe permitir al Usuario actualizar la información de su cuenta
RF-03	El sistema debe permitir al Usuario agregar canciones personales a una biblioteca privada
RF-04	El sistema debe permitir al Usuario gestionar una lista de reproducción generada por el sistema
RF-05	El sistema debe permitir al usuario gestionar las canciones encontradas en su biblioteca privada
RF-06	El sistema debe permitir al Usuario generar una solicitud de publicación de canción
RF-07	El sistema debe permitir al Usuario buscar y reproducir canciones de la biblioteca publica
RF-08	El sistema debe permitir al Usuario consultar la información de las canciones de la biblioteca publica
RF-09	El sistema debe permitir al Administrador Gestionar las solicitudes de publicación de canción de los Usuarios
RF-10	El sistema debe permitir a los Usuarios reproducir las canciones de su biblioteca privada



## 2.6. Requisitos No Funcionales

En la siguiente table se muestran los requisitos no funcionales para el sistema distribuido SpotyMe.

Tabla 2 Requisitos No Funcionales

Identificador de requisito	Descripción de requisito
RNF-01	El sistema debe iniciar a reproducir una canción seleccionada en no más de 5 segundos un 90% del tiempo.
RNF-02	El sistema debe realizar la búsqueda de una canción en no más de 3 segundos el 85% del tiempo
RNF-03	El sistema debe proteger con derechos de acceso la autorización de canciones a publicar en la biblioteca publica
RNF-04	Un usuario debe poder realizar la funcionalidad de un caso de uso en no más de 2 minutos el 90% del tiempo
RNF-05	El sistema debe proteger el acceso a funcionalidades con un inicio de sesión
RNF-06	Un usuario debe poder registrar su cuenta en sistema en no más de 3 minutos el 99% del tiempo. El sistema debe permitir al Usuario generar una solicitud de publicación de canción
RNF-07	El sistema debe de cargar la biblioteca privada de un usuario en no más de 5 minutos el 80% del tiempo
RNF-08	El usuario debe de poder subir una canción en no más de 5 minutos el 90% del tiempo.

### 3. Diseño

#### 3.1. Diseño arquitectónico

##### 3.1.1. Vista de casos de uso

En esta sección se puede ver el diagrama UML de casos de uso a implementar en el sistema SpotyMe.

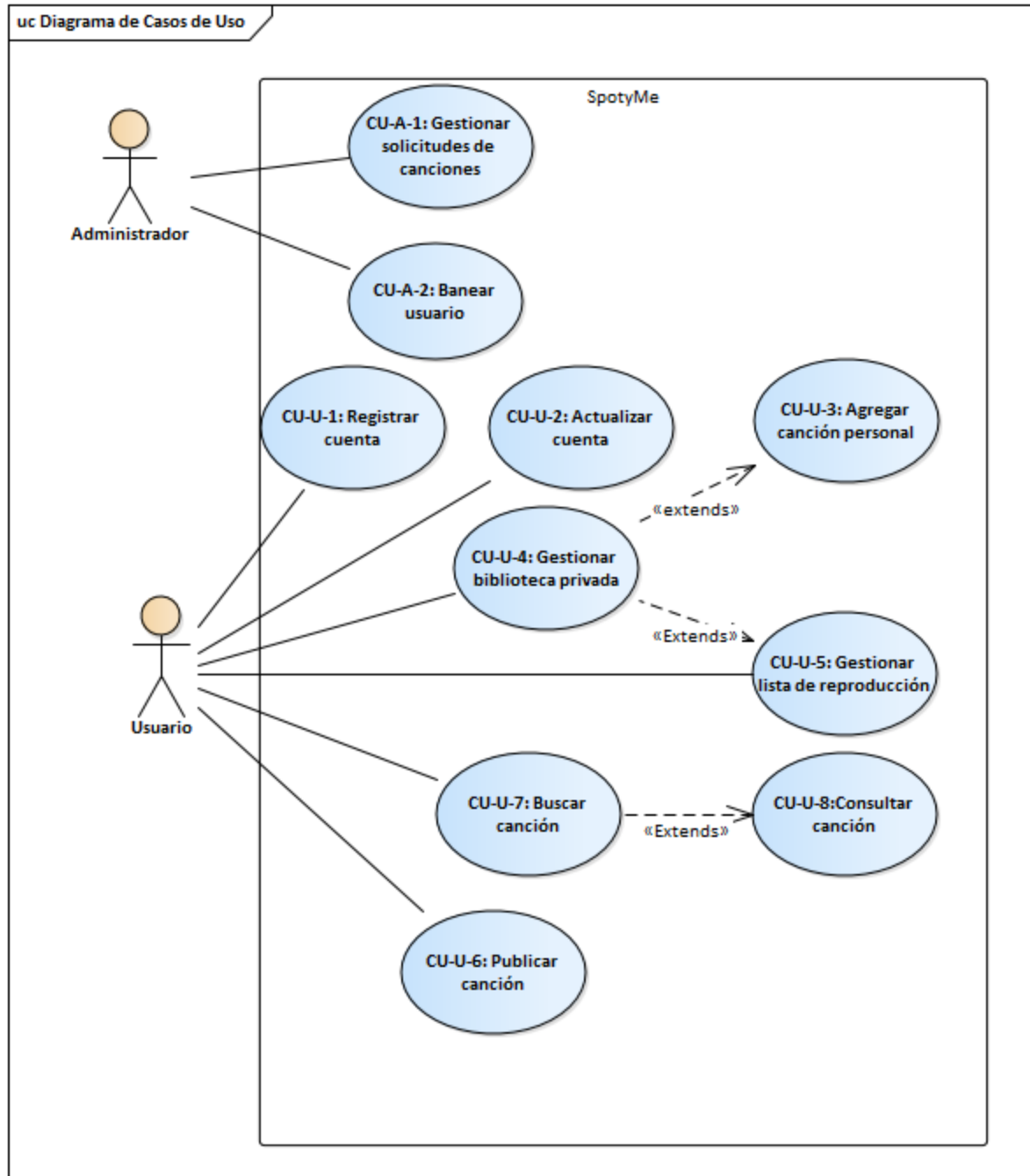


Figura 16 Diagrama UML de casos de uso

### 3.1.2. Vista de implementación

En esta sección se puede ver el diagrama UML de componentes basado en los microservicios a implementar para cumplir con los requisitos del sistema SpotyMe.

#### 3.1.2.1. Tabla de Microservicios

En la siguiente tabla se muestran los microservicios que conforman el sistema SpotyMe.

Tabla 3 Microservicios de SpotyMe

Servicio	Caso de Uso Involucrados	Descripción de Servicio
Microservicio de Biblioteca pública	CU-U-6: Publicar canción CU-A-1: Gestionar solicitudes de canciones	En este servicio se encuentra la funcionalidad necesaria para solicitar la publicación de una canción de un usuario y evaluar la solicitud por el administrador.
Microservicio de Cuentas	CU-A-2: Banear Usuario CU-U-1: Registrar cuenta CU-U-2: Actualizar cuenta	En este servicio se encuentra la funcionalidad para crear una cuenta para acceso al sistema y modificar datos personales de usuario para darle acceso al sistema.
Servicio de Streaming	CU-U-7: Buscar canción CU-U-8: Consultar canción	En este servicio se brinda la funcionalidad para consultar de la base de datos canciones de la biblioteca pública.
Microservicio de Biblioteca Privada	CU-U-3: Agregar canción personal CU-U-4: Gestionar lista de reproducción CU-U-5: Gestionar biblioteca privada	En este servicio se brinda la funcionalidad para consultar de la base de datos canciones de la biblioteca privada de un usuario y realizar la gestión de estas.

### 3.1.2.2. Diagrama de Microservicio de Cuentas

En la siguiente figura se muestra el diagrama UML de componentes del Microservicio de Cuentas.

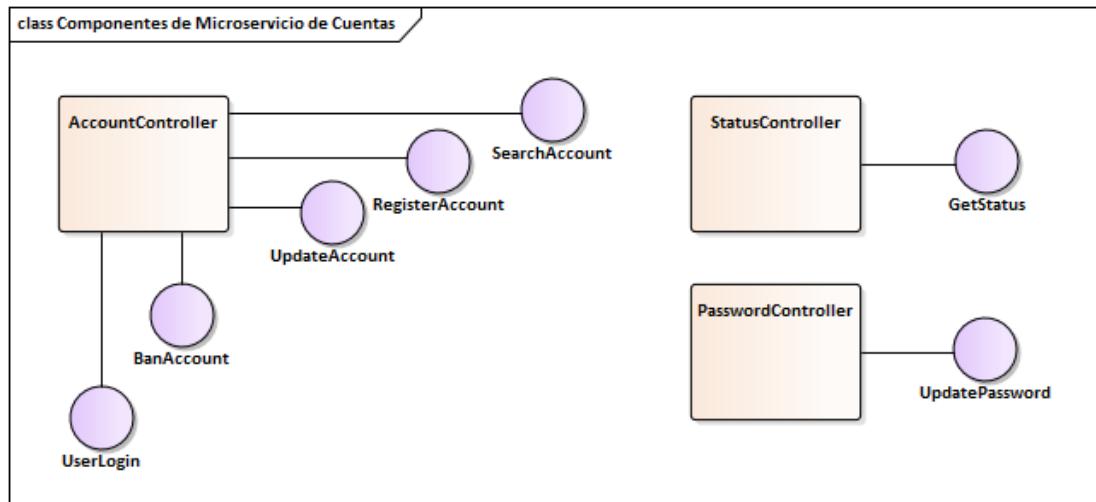


Figura 17 Diagrama de Componentes de Cuentas

### 3.1.2.3. Diagrama de Microservicio de Biblioteca Pública

En la siguiente figura se muestra el diagrama UML de componentes del Microservicio de Biblioteca Pública.

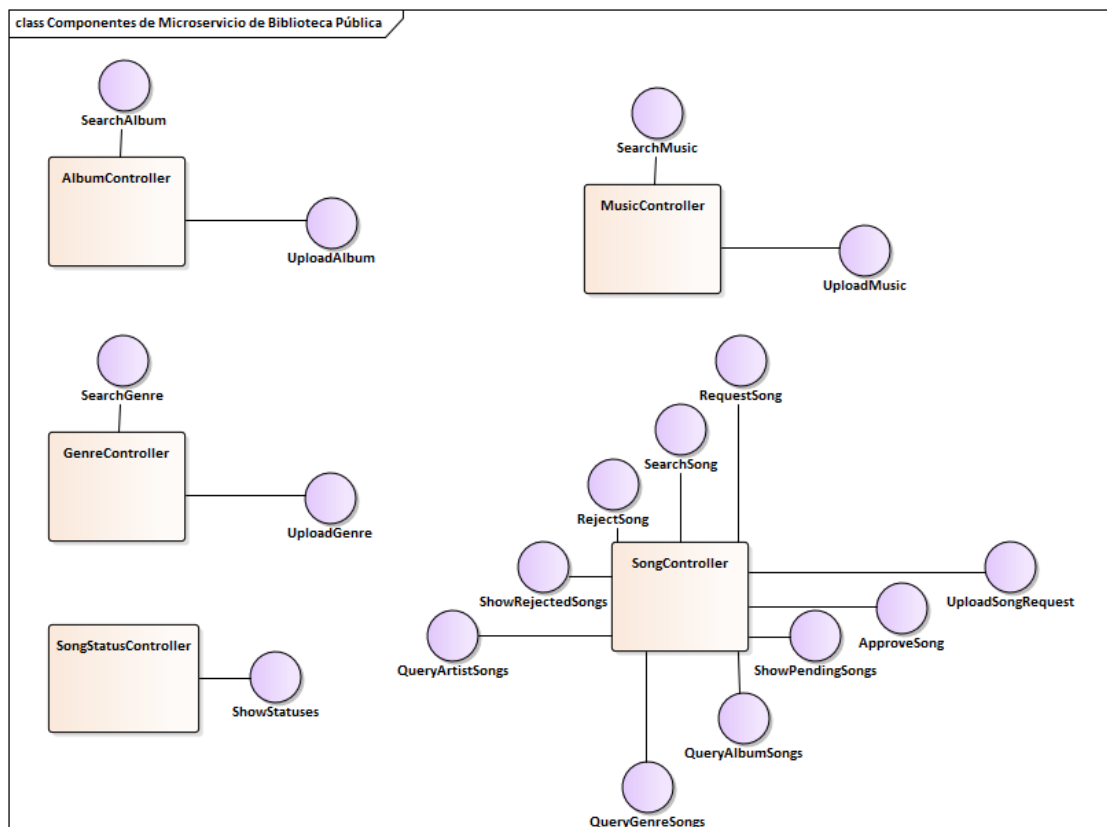


Figura 18 Diagrama de Componentes de Biblioteca Pública

### 3.1.2.4. Diagrama de Microservicio de Biblioteca Privada

En la siguiente figura se muestra el diagrama UML de componentes del Microservicio de Biblioteca Privada.

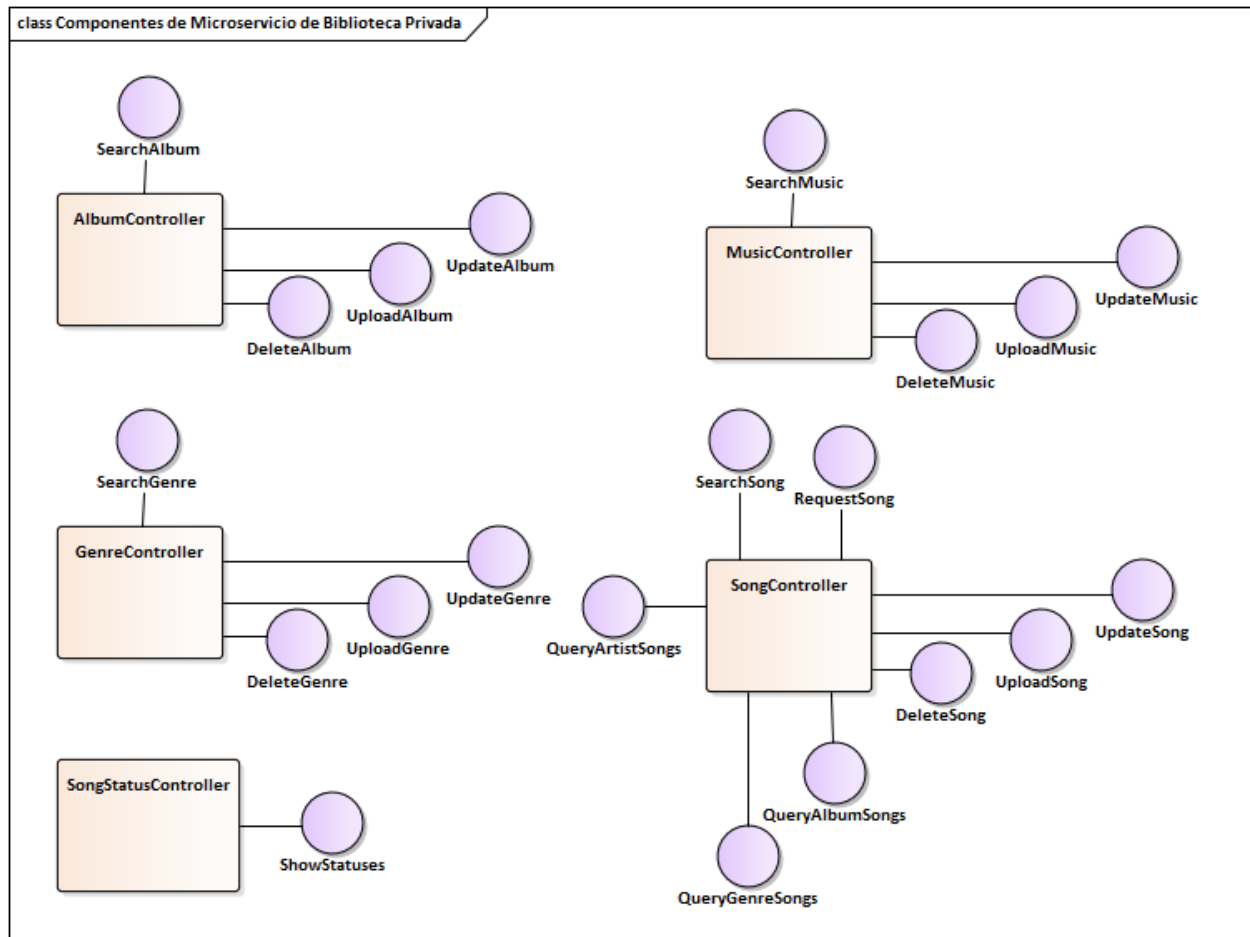


Figura 19 Diagrama de Componentes de Biblioteca Privada

### 3.1.2.5. Diagrama de Microservicio de Streaming

En la siguiente figura se muestra el diagrama UML de componentes del Microservicio de Streaming.

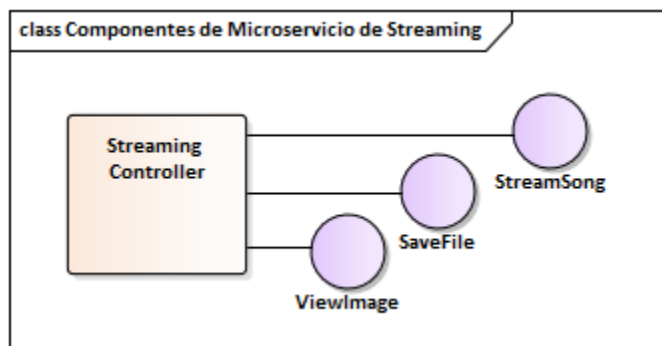


Figura 20 Diagrama de Componentes de Streaming

### 3.1.2.6. Diagrama de API

En la siguiente figura se muestra el diagrama UML de componentes de la API.

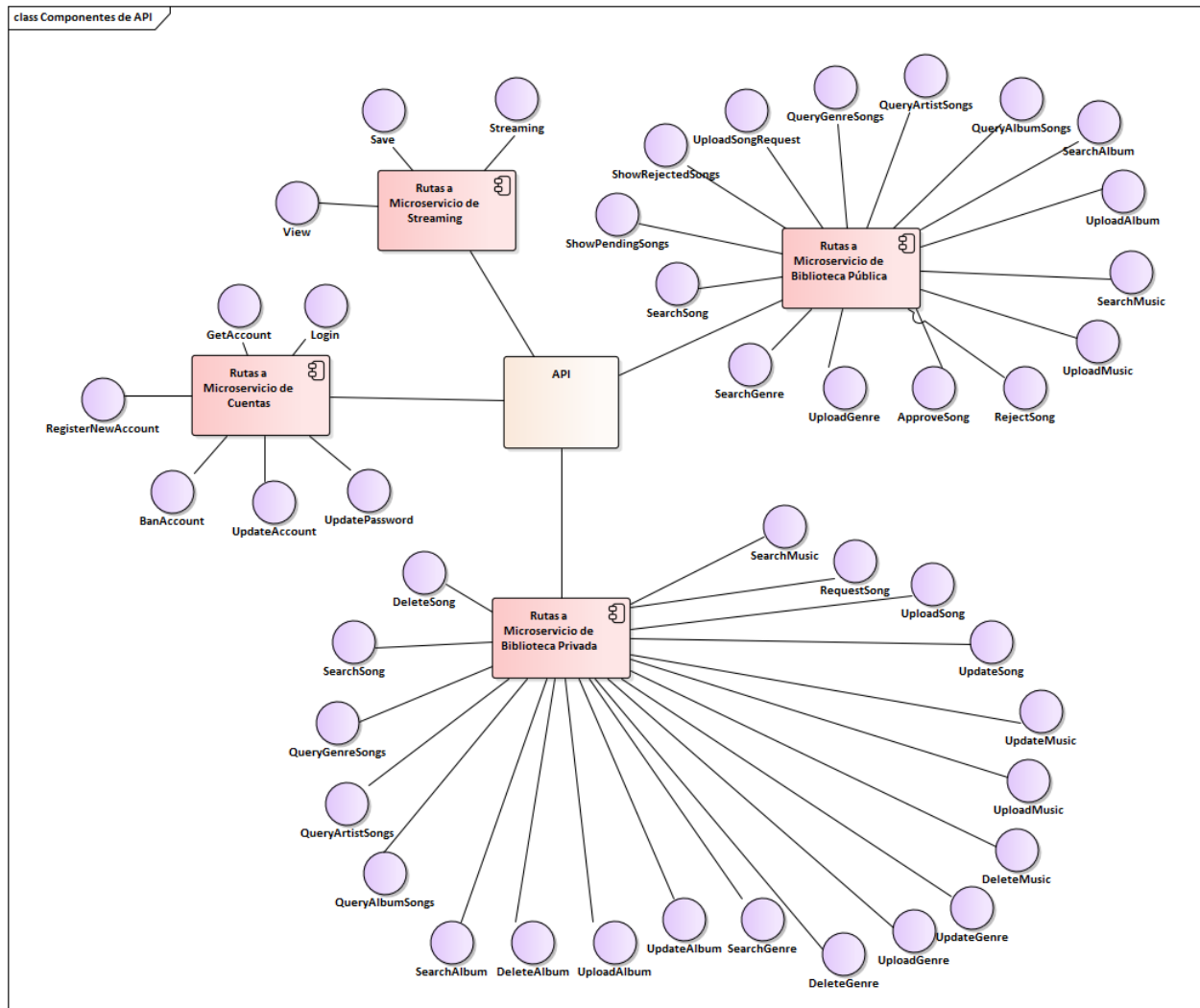


Figura 21 Diagrama de Componentes de API

### 3.1.3. Vista de procesos

En esta sección se pueden ver los diagramas UML de secuencia para los procesos que forman parte de la funcionalidad del sistema SpotyMe.

La siguiente figura muestra el proceso de una petición GET HTTP para los métodos de consulta de datos con excepción del método View del Microservicio de Streaming.

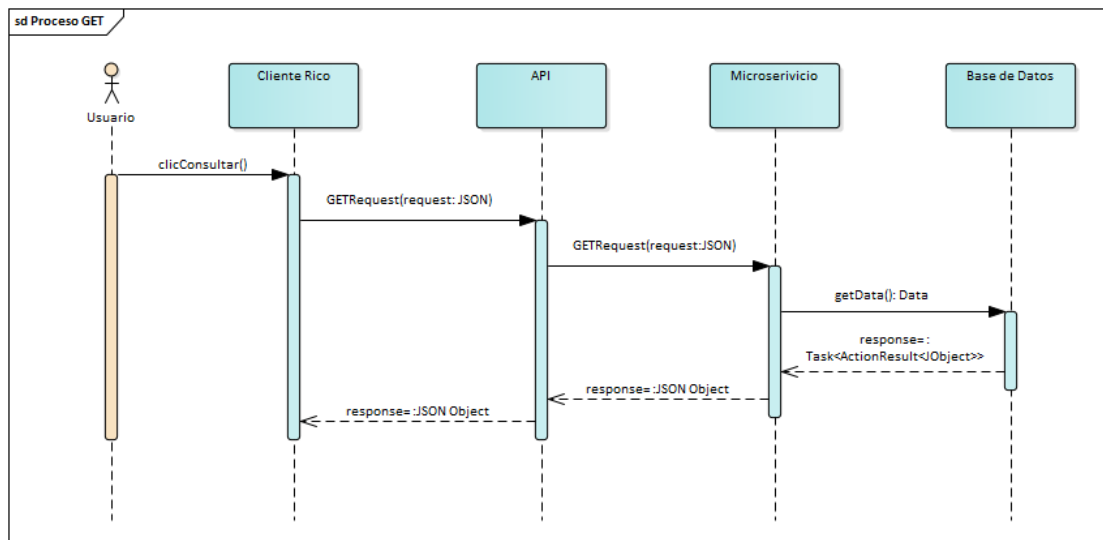


Figura 22 Diagrama de Proceso de petición GET

La siguiente figura muestra el proceso para la petición GET View para consultar imágenes almacenadas en el contenedor del Microservicio de Streaming.

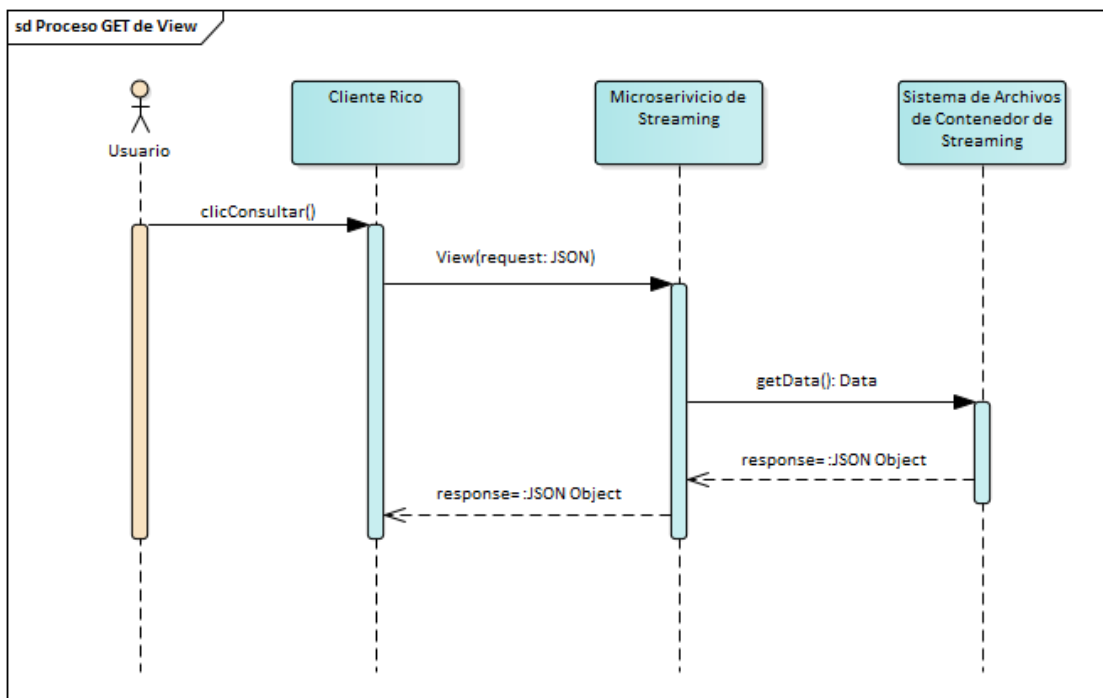


Figura 23 Diagrama de Proceso de petición View

La siguiente figura muestra el proceso de una petición POST HTTP para todos los métodos utilizados excepto el método Login de la API para la creación de datos.

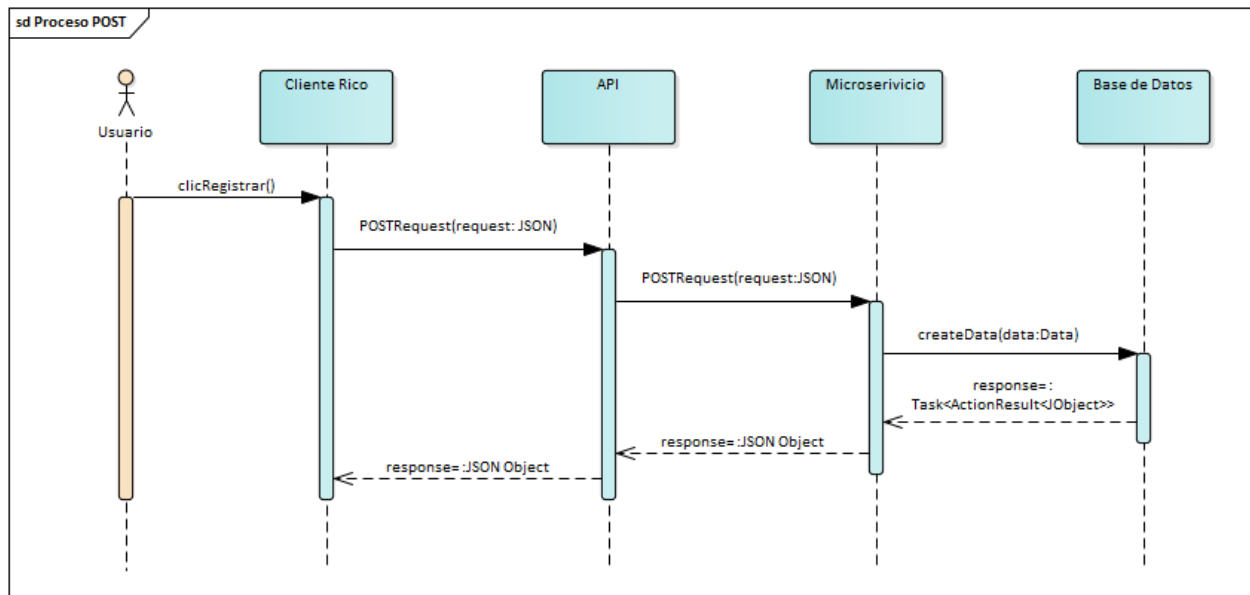


Figura 24 Diagrama de proceso de petición POST

La siguiente figura muestra el proceso de la petición POST Login que consulta datos para verificar que exista una cuenta antes de iniciar una sesión en el sistema.

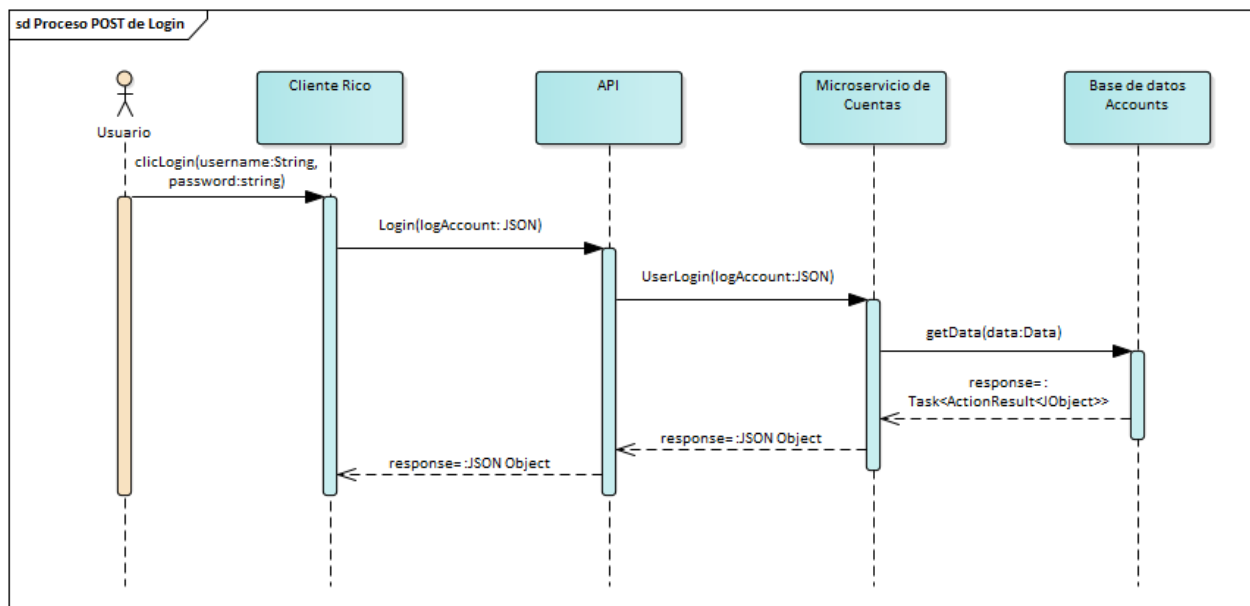


Figura 25 Diagrama de Proceso de petición Login



La siguiente figura muestra el proceso de una petición PUT HTTP para todos los métodos utilizados. Estos métodos actualizan datos de entidades registradas por métodos POST.

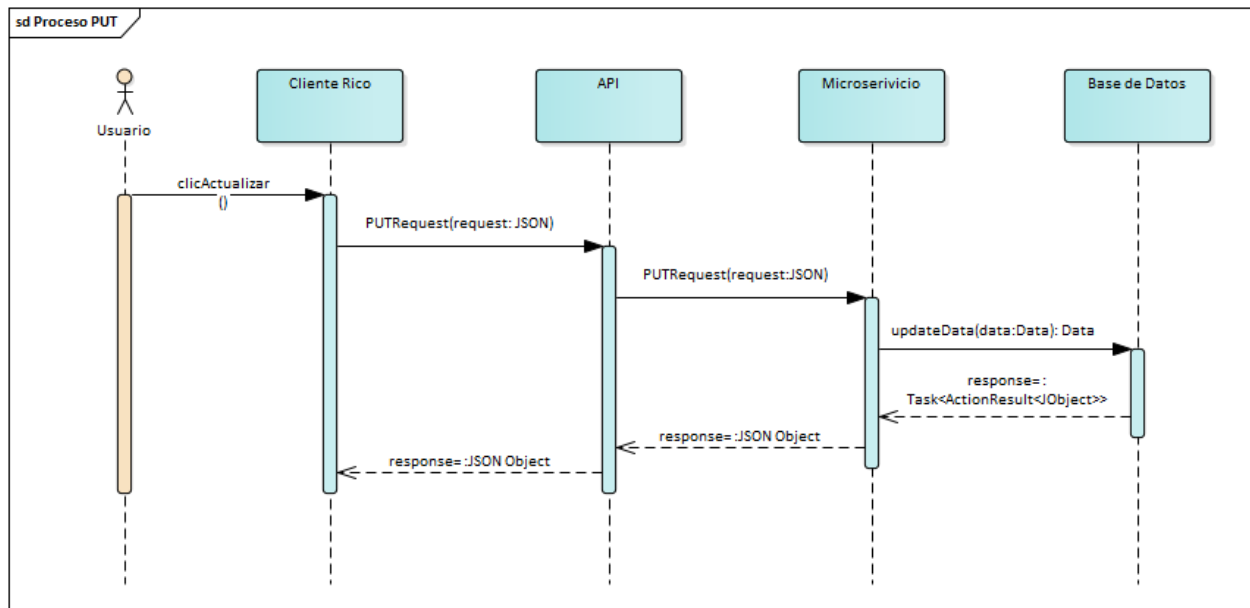


Figura 26 Diagrama de Proceso de petición PUT

### 3.1.4. Vista de despliegue

En esta sección se puede ver el diagrama UML de despliegue para el ambiente necesario para correr el sistema SpotyMe.

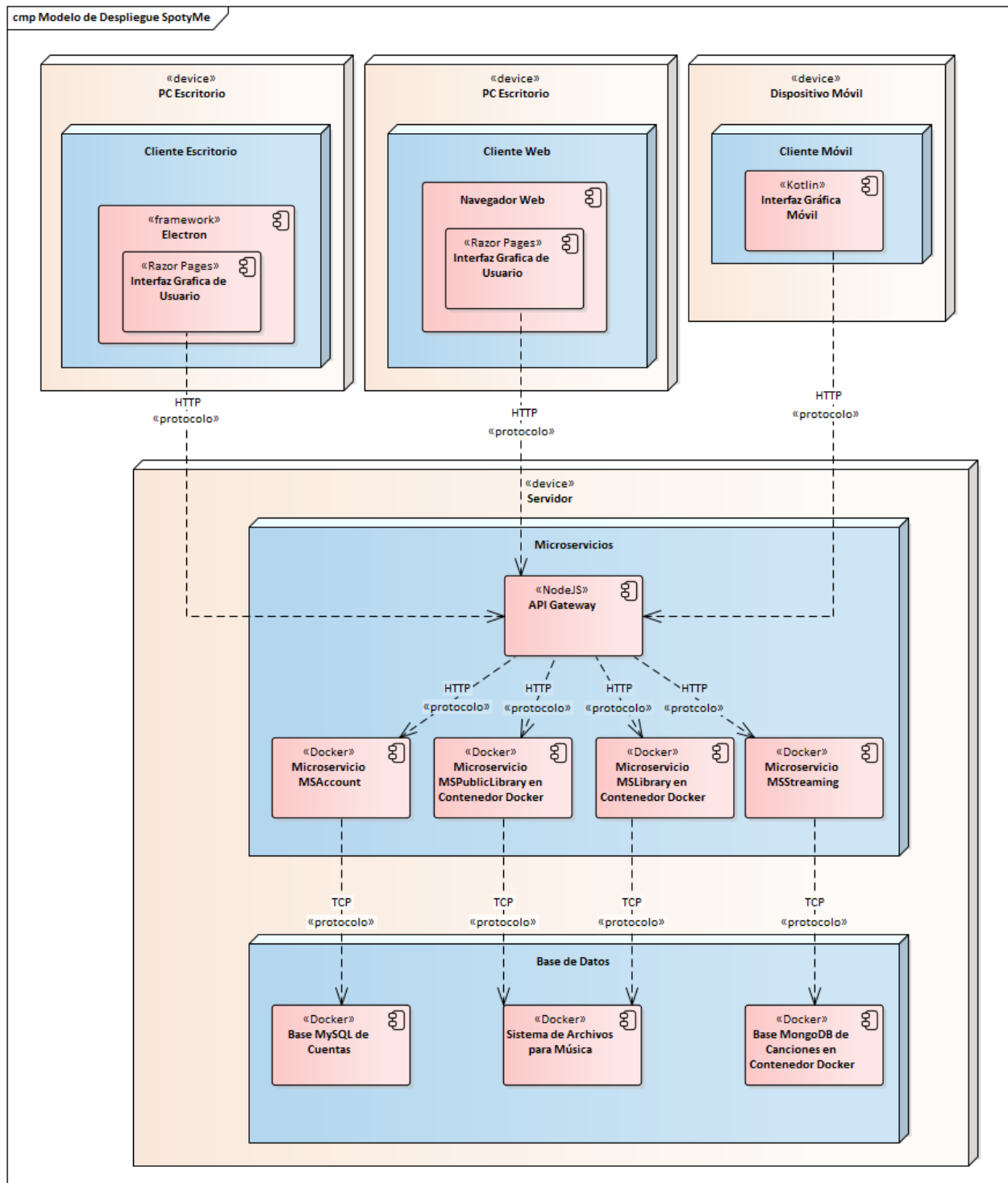


Figura 27 Diagrama de Despliegue de SpotyMe

### 3.2. Modelo de Datos

En esta sección se pueden ver los diagramas Entidad-Relación para las bases de datos utilizadas por el sistema SpotyMe.

#### 3.2.1. Base de Datos de Cuentas

En la siguiente figura se muestra el modelo entidad-relación para la base de datos Accounts en MySQL del microservicio de Cuentas

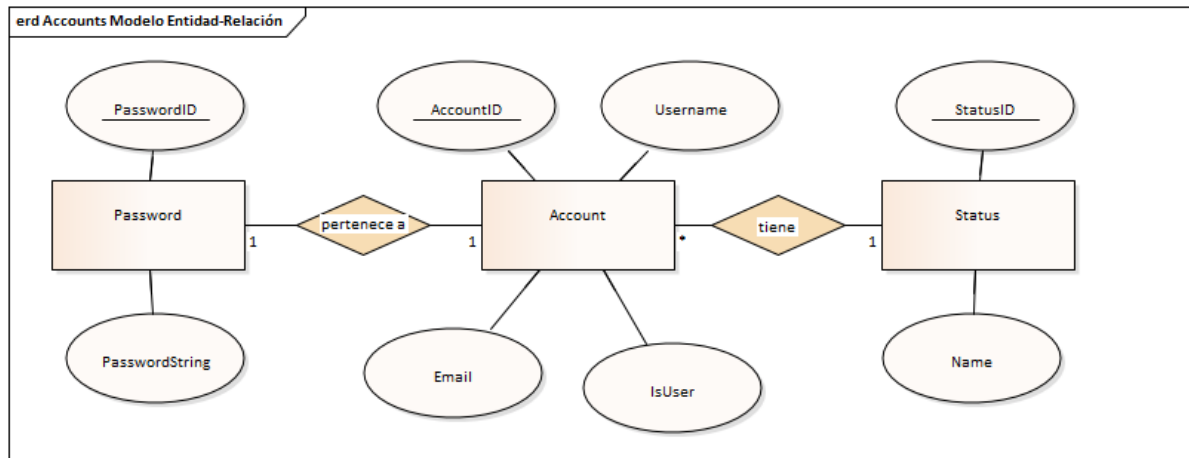


Figura 28 Diagrama E-R de base de datos de Cuentas

#### 3.2.2. Base de datos Library

En la siguiente figura se muestra el modelo entidad-relación para la base de datos en MongoDB del microservicio de Biblioteca Privada y Biblioteca Pública.

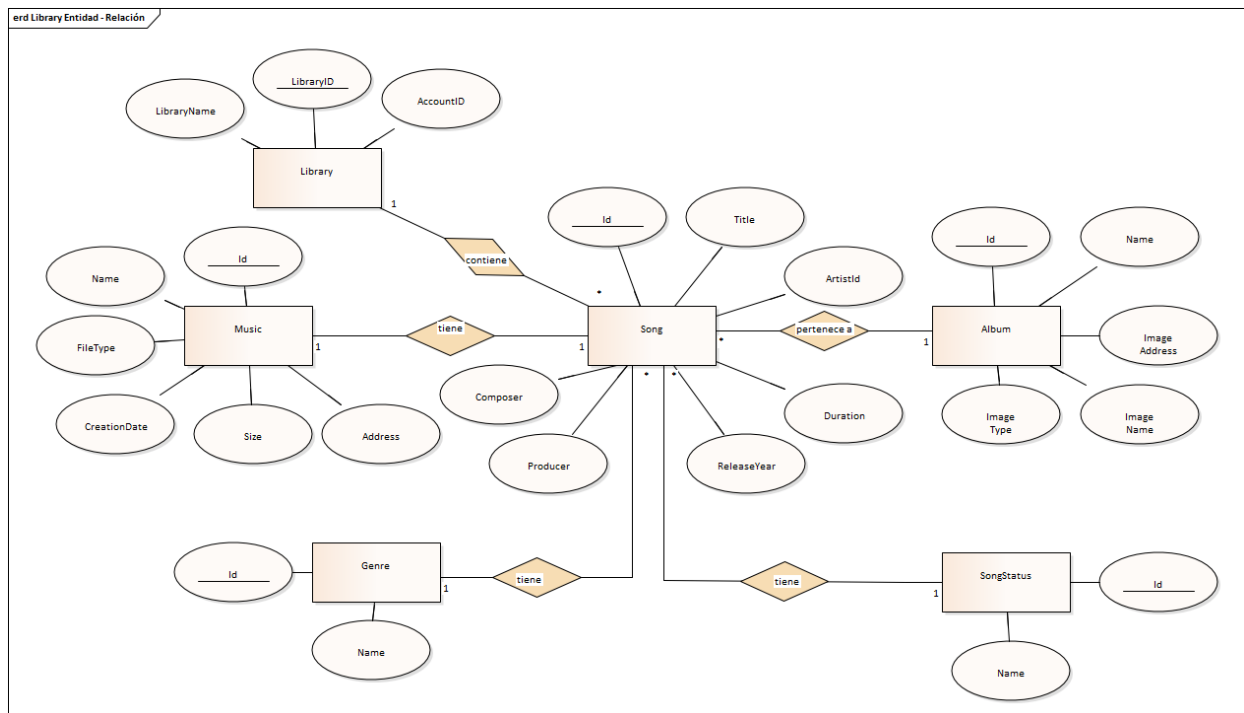


Figura 29 Diagrama E-R de base de datos de Bibliotecas

### 3.3. Descripciones de Casos de Uso

En la siguiente sección se encuentran las descripciones de los casos de uso identificados en el diagrama UML de casos de uso.

#### 3.3.1. Casos de Uso para Administrador

En primer lugar, la siguiente tabla muestra la descripción para el caso de uso del actor Administrador CU-A-1: Gestionar solicitudes de canciones.

<b>ID:</b>	CU-A-01
<b>Nombre:</b>	Gestionar solicitudes de canciones
<b>Autor(es):</b>	Ricardo Moguel Sánchez
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Administrador</b> visualiza una tabla con datos de Solicitudes de Usuarios para archivos a aprobar su publicación en biblioteca dentro del Sistema Distribuido SpotyMe(SM).
<b>Actor(es):</b>	<b>Administrador</b>
<b>Frecuencia de uso:</b>	Aproximadamente 4 veces cada semana
<b>Disparador:</b>	El <b>Administrador</b> hace clic el botón “Consultar” bajo la sección: SOLICITUDES en su ventana principal.
<b>Precondiciones:</b>	PRE-01: Debe haber al menos 1 instancia de Solicitud de Aprobación de Canción registrado dentro del sistema.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"><li>1. El SM muestra la ventana “Lista De Solicitudes de Aprobación” y la plantilla <b>datos_solicitud</b> se muestra los iconos “Play”, “Detalles”, y “Aprobar” y “Rechazar” para las solicitudes con estatus “Pendiente”. Además de un botón “Regresar” debajo de la tabla (Ver EX1)</li><li>2. El <b>Administrador</b> hace clic el botón “Regresar”. (Ver FA 2.1, FA2.2, FA2.3, FA2.4)</li><li>3. El SM cierra la ventana “Lista De Solicitudes de Aprobación”</li><li>4. Termina el caso de uso.</li></ol>
<b>Flujos Alternos:</b>	<p>FA 2.1: El <b>Administrador</b> desea reproducir una canción.</p> <ol style="list-style-type: none"><li>1. El <b>Administrador</b> hace clic el botón “Play” de una entrada en la tabla.</li><li>2. El SM abre un menú “Reproducción de Canción” con el botón “Stop” y reproduce la canción seleccionada.</li><li>3. El <b>Administrador</b> hace clic el botón “Stop”</li><li>4. El SM cierra la ventana “Reproducción de Canción” y se regresa al paso 1 del flujo normal.</li></ol> <p>FA 2.2: El <b>Administrador</b> desea consultar una canción.</p> <ol style="list-style-type: none"><li>1. El <b>Administrador</b> hace clic el botón “Detalles” de una entrada en la tabla.</li><li>2. El SM muestra una ventana “Detalles de Canción” con el formulario <b>datos_cancion</b> lleno con la información de la canción seleccionada y debajo el botón “Regresar”. (EX1)</li></ol>

	<ol style="list-style-type: none"> <li>3. El <b>Administrador</b> hace clic el botón “Regresar”</li> <li>4. El SM cierra la ventana “Detalles de Canción” y se regresa al paso 1 del flujo normal.</li> </ol> <p>FA 2.3: El <b>Administrador</b> desea aprobar una canción.</p> <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> hace clic el botón “Aprobar” de una entrada en la tabla.</li> <li>2. El SM cambia el atributo estatus a “Aprobado” de la solicitud seleccionada y la guarda en la base de datos, cierra “Confirmar Aprobar” y se muestra la interfaz gráfica “Éxito Rechazar” con el mensaje: “La solicitud fue rechazada exitosamente.” y el botón “Ok”. (Ver EX1)</li> <li>3. El <b>Administrador</b> hace clic el botón “OK”</li> <li>4. El SM cierra la ventana “Éxito Rechazar” y se regresa al paso 1 del flujo normal.</li> </ol> <p>FA 2.4: El <b>Administrador</b> desea rechazar una canción.</p> <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> hace clic el botón “Rechazar” de una entrada en la tabla.</li> <li>2. El SM muestra la interfaz gráfica: “Confirmar Rechazar” con los botones “Si” y “No” y un campo de texto vacío.</li> <li>3. El <b>Administrador</b> ingresa datos en el campo de texto y hace clic el botón “SI”. (Ver FA2.5)</li> <li>4. El SM verifica el campo de texto y si es válido cambia el atributo estatus a “Rechazado” y guarda en la base de datos el cambio a la solicitud seleccionada y se muestra la interfaz gráfica “Éxito Rechazar” y el botón “Ok”. (Ver EX1)</li> <li>5. El <b>Administrador</b> hace clic el botón “OK”</li> <li>6. El SM cierra la ventana “Éxito Rechazar” y se regresa al paso 1 del flujo normal.</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no pudo conectarse a la base de datos y no realizo la acción sobre la base de datos.</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde.” y el botón “Ok” en una interfaz gráfica “Error Conexión”.</li> <li>2. El <b>Administrador</b> hace clic el botón “Ok”</li> <li>3. El SM cierra “Error Conexión” y “Lista De Solicitudes de Aprobación”</li> <li>4. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	<p>POST-1: El SM muestra la ventana: “Éxito Aprobar” o “Éxito Rechazar”</p> <p>POST=2: El SM cambia el atributo Estatus a “Aprobado” o “Rechazado”</p>
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	No Aplica
<b>Prioridad:</b>	Alta
<b>datos_solicitud</b>	Estatus, Título de Canción, Usuario Subidor

<b>datos_cancion:</b>	Título, Género, Escritor, Duración, Álbum, Artista, Año, Productor
-----------------------	--

En la siguiente tabla se muestra la descripción para el siguiente caso de uso del actor Administrador CU-A-2: Banear usuario.

<b>ID:</b>	CU-A-02
<b>Nombre:</b>	Banear usuario
<b>Autor(es):</b>	Cesar Sergio Martinez Palacios
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El <b>Administrador</b> desea <u>banear</u> a un usuario por mal uso del servicio de publicación de canciones
<b>Actor(es):</b>	<b>Administrador</b>
<b>Frecuencia de uso:</b>	Aproximadamente 2 veces cada mes
<b>Disparador:</b>	El <b>Administrador</b> hace clic el botón “Consultar” bajo la sección: SOLICITUDES en su ventana principal.
<b>Precondiciones:</b>	PRE-01: Debe haber al menos 1 instancia de Solicitud de Aprobación de Canción registrado dentro del sistema.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Datos de Usuario” con los botones “Banear” y “Regresar”</li> <li>2. El <b>Administrador</b> hace clic el botón “Banear”. (Ver FA 2.1)</li> <li>3. EL SM muestra una ventana de confirmación de acción con las opciones “Si” y “No”</li> <li>4. El <b>Administrador</b> selecciona la opción “Si”</li> <li>5. El SM actualiza el estado del USUARIO,regresa al CU-A-01 y Muestra ventana de éxito(Ver EX1)</li> <li>6. Termina el caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	FA 2.1: El <b>Administrador</b> desea reproducir una canción. <ol style="list-style-type: none"> <li>1. El <b>Administrador</b> selecciona la opción “Regresar”</li> <li>2. El SM regresa al CU-A-01</li> </ol>
<b>Excepciones:</b>	EX1- El SM no pudo conectarse a la base de datos y no realizo la acción sobre la base de datos. <ol style="list-style-type: none"> <li>5. El SM muestra el mensaje “Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde.” y el botón “Ok” en una interfaz gráfica “Error Conexión”.</li> <li>6. El <b>Administrador</b> hace clic el botón “Ok”</li> <li>7. El SM cierra “Error Conexión” y “Lista De Solicitudes de Aprobación”</li> <li>8. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	POST-1: El SM actualiza la BD y cambia el estado del USUARIO
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	No Aplica
<b>Prioridad:</b>	Alta

### 3.3.2. Casos de Uso para Usuario

En primer lugar, la siguiente tabla muestra la descripción para el primer caso de uso del actor Usuario CU-U-1: Registrar cuenta.

<b>ID:</b>	CU-U-1
<b>Nombre:</b>	Registrar cuenta
<b>Autor(es):</b>	Ricardo Moguel Sánchez
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El Sistema Distribuido de SpotyMe(SM) crea una CUENTA y el <b>Usuario</b> ingresa datos en una plantilla que lo guarda como atributos para que el Usuario tenga acceso a la funcionalidad del sistema.
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 5 veces cada semana
<b>Disparador:</b>	Se inicia la aplicación
<b>Precondiciones:</b>	PRE-01: Debe haber una conexión de internet activa.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana "Inicio De Sesión" con el botón "Registrarse" en la parte derecha superior, los campos: "Email" y "Contraseña" y el botón "Iniciar Sesión".</li> <li>2. El <b>Usuario</b> hace clic el botón "Registrarse" en la ventana "Inicio De Sesión". (Ver FA 2.1)</li> <li>3. El SM muestra la ventana "Registro de Usuario" con una plantilla conteniendo los campos: "Apodo", "Email" y "Contraseña". Debajo de la plantilla se muestran los botones "Registrar" y "Cancelar"</li> <li>4. El <b>Usuario</b> ingresa datos en el formulario y hace clic el botón "Registrarse". (Ver FA 4.1)</li> <li>5. El SM verifica que los datos en cada campo tengan información y que dicha información este correcta. Si esta correcto se crea un CUENTA con los atributos ingresados, guarda la información en la base de datos y muestra la interfaz gráfica: "Éxito Registrar" con el mensaje "Fuiste registrado exitosamente." y el botón "Ok". (Ver FA 5.1, EX1)</li> <li>6. El <b>Usuario</b> hace clic el botón "Ok".</li> <li>7. El SM cierra la ventana "Éxito Registrar" y "Registro de Usuario".</li> <li>8. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1: El SM inicia sesión con los datos ingresados.</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón "Iniciar Sesión"</li> <li>2. El SM verifica que los datos de usuario y contraseña se encuentren un USUARIO en la base de datos". Si se encuentra se muestra la ventana principal de Usuario "Principal Usuario". De lo contrario se muestra los campos de usuario y contraseña con resalte rojo y el mensaje "Usuario o contraseña incorrecta. Verifique sus datos para iniciar sesión" debajo del campo de contraseña en rojo y se regresa al paso 1 del flujo normal.</li> </ol>

	<p>FA 4.1: Cancelar registrar el Usuario</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón "Cancelar"</li> <li>2. El SM muestra el mensaje "¿Seguro desea cancelar?" y los botones "Sí" y "No" en la interfaz gráfica "Confirmar Cancelar".</li> <li>3. Si el <b>Usuario</b> hace clic el botón "Sí", termina el caso de uso, si no, cierra la interfaz gráfica "Confirmar Cancelar" y regresa al paso 1 del flujo normal.</li> </ol> <p>FA 5.1: El SM detecta campos con datos inválidos</p> <ol style="list-style-type: none"> <li>1. El SM muestra una interfaz gráfica "Error Validación" con el mensaje "La información en uno o varios campos es incorrecta. Por favor verifica la información." con el botón "Ok".</li> <li>2. El <b>Usuario</b> hace clic el botón "Ok".</li> <li>3. El SM cierra "Error Validación" y regresa al paso 1 del flujo normal con la información ya ingresada.</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no pudo conectarse a la base de datos y no guardo la información.</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje "Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde." y el botón "Ok" en una interfaz gráfica "Error Conexión".</li> <li>2. El <b>Usuario</b> hace clic el botón "Ok"</li> <li>3. El SM cierra "Error Conexión" y "Registro de Usuario"</li> <li>4. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	<p>POST-1: El SM muestra el mensaje "Fuiste registrado exitosamente".</p> <p>POST-2: Se guarda una CUENTA de Usuario dentro de la base de datos.</p>
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	No aplica
<b>Prioridad:</b>	Alta
<b>datos de Usuario</b>	Apodo, Email, Contraseña



La siguiente tabla muestra la descripción del caso de uso CU-U-2: Actualizar cuenta

<b>ID:</b>	CU-U-2
<b>Nombre:</b>	Actualizar cuenta
<b>Autor(es):</b>	Ricardo Moguel Sánchez
<b>Fecha de creación:</b>	11 de marzo de 2021
<b>Fecha de actualización:</b>	
<b>Descripción:</b>	El Sistema Distribuido de SpotyMe(SM) modifica una CUENTA y el <b>Usuario</b> ingresa datos nuevos en una plantilla para mantener actualizados los datos de él dentro del sistema.
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 2 veces cada semana
<b>Disparador:</b>	El <b>Usuario</b> hace clic el botón “Cuenta” en el pequeño menú que tiene de lado izquierdo en la ventana principal.
<b>Precondiciones:</b>	PRE-01: Debe haber una conexión de internet activa.
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Datos de Cuenta” con el botón “Actualizar” en la parte derecha superior, y la plantilla <b>datos_usuario</b> y el botón “Regresar”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Actualizar”. (Ver FA 2.1)</li> <li>3. El SM muestra la ventana “Actualización de Usuario” con una plantilla conteniendo los campos: “Apodo”, “Email” y “Contraseña”. Debajo de la plantilla se muestran los botones “Actualizar mis datos” y “Cancelar”</li> <li>4. El <b>Usuario</b> ingresa datos en el formulario y hace clic el botón “Actualizar mis datos”. (Ver FA 4.1)</li> <li>5. El SM verifica que los datos en cada campo tengan información correcta. Si esta correcto se guarda los datos de la CUENTA con los atributos ingresados en la base de datos y muestra la interfaz gráfica: “Éxito Actualizar” con el botón “Ok”. (Ver FA 7.1, EX1)</li> <li>6. El <b>Usuario</b> hace clic el botón “Ok”.</li> <li>7. El SM cierra la ventana “Éxito Actualizar” y “Actualización de Usuario”.</li> <li>8. Termina caso de uso.</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1: El Usuario desea Salir de actualización de cuenta</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Regresar”</li> <li>2. El SM cierra la ventana “Datos de Cuenta”</li> <li>3. Termina caso de uso.</li> </ol> <p>FA 4.1: Cancelar actualizar el Usuario</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Cancelar”</li> <li>2. El SM cierra la interfaz gráfica “Actualización de Usuario” y regresa al paso 1 del flujo normal.</li> </ol> <p>FA 7.1: El SM detecta campos con datos inválidos</p> <ol style="list-style-type: none"> <li>2. El SM muestra una interfaz gráfica “Error Validación” con el mensaje “La información en uno o varios campos es</li> </ol>

	<p>incorrecta. Por favor verifica la información.” con el botón “Ok”.</p> <p>3. El <b>Usuario</b> hace clic el botón “Ok”.</p> <p>4. El SM cierra “Error Validación” y regresa al paso 1 del flujo normal con la información ya ingresada.</p>
<b>Excepciones:</b>	<p>EX1- El SM no pudo conectarse a la base de datos y no guardo la información.</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde.” y el botón “Ok” en una interfaz gráfica “Error Conexión”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM cierra “Error Conexión” y “Registro de Usuario”</li> <li>4. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	<p>POST-1: El SM muestra el mensaje “Se actualizo exitosamente tus datos de cuenta”.</p> <p>POST-2: Se actualiza la CUENTA en la base de datos del usuario ingresado.</p>
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	No aplica
<b>Prioridad:</b>	Media
<b>datos_de_Usuario</b>	Apodo, Email, Contraseña

La siguiente tabla muestra la descripción del caso de uso CU-U-3: Agregar canción personal

<b>ID:</b>	CU-U-3
<b>Nombre:</b>	Agregar canción personal
<b>Autor(es):</b>	Ricardo Moguel Sánchez
<b>Fecha de creación:</b>	11 de marzo del 2021
<b>Fecha de actualización:</b>	No aplica
<b>Descripción:</b>	El <b>Usuario</b> desea subir una canción a su biblioteca personal.
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 1 vez a la semana
<b>Disparador:</b>	El <b>Usuario</b> hace clic el botón “Agregar canción” en la ventana “Biblioteca privada”
<b>Precondiciones:</b>	[No aplica]
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Agregación de canción” con las opciones “Subir”, “Adjuntar Canción”, “Cancelar” y el formulario <b>datos_cancion</b></li> <li>2. El <b>Usuario</b> ingresa la información solicitada y selecciona la opción “Adjuntar Canción” (Ver FA 2.1)</li> <li>3. El SM abre el explorador de archivos del sistema operativo con los botones “Abrir” y “Cancelar” (Ver EX1)</li> <li>4. El <b>Usuario</b> selecciona el archivo .mp3 de la canción a y hace clic “Abrir” (Ver FA4.1)</li> <li>5. El SM verifica que el archivo es de tamaño no más de 3 MB y carga el archivo y activa el botón “Subir” (Ver EX2)</li> <li>6. El <b>Usuario</b> selecciona la opción “Subir” (Ver FA2.1)</li> <li>7. El SM verifica que los datos de canción son válidos. Si son correctos se crea una instancia de CANCION y la relaciona a la CUENTA del <b>Usuario</b> y lo guarda en la base de datos (Ver FA7.1, EX3)</li> <li>8. Termina el caso de uso</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1: El Usuario desea Salir de “Agregación de Canción”</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Cancelar”</li> <li>2. El SM cierra la ventana “Agregación de Canción”</li> <li>3. Termina caso de uso</li> </ol> <p>FA 4.1: El Usuario desea Salir del explorador de archivos</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Cancelar”</li> <li>2. El SM cierra el explorador de archivos y regresa a paso 1 del flujo normal con los datos ingresados</li> </ol> <p>FA 5.1: El SM detecta archivo invalido</p> <ol style="list-style-type: none"> <li>1. El SM muestra una interfaz gráfica “Archivo Invalido” con el mensaje “ERROR: El archivo rebasó el límite de tamaño de 30 MB. Por favor verifica la información.” con el botón “Ok”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”.</li> </ol>

	<p>3. El SM cierra “Error Validación” y regresa al paso 1 del flujo normal con la información ya ingresada.</p> <p>FA 7.1: El SM detecta campos con datos inválidos</p> <ol style="list-style-type: none"> <li>1. El SM muestra una interfaz gráfica “Error Validación” con el mensaje “La información en uno o varios campos es incorrecta. Por favor verifica la información.” con el botón “Ok”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”.</li> <li>3. El SM cierra “Error Validación” y regresa al paso 1 del flujo normal con la información ya ingresada.</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no abrir el explorador de archivos</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar abrir el explorador. Intente de nuevo más tarde.” y el botón “Ok”</li> <li>2. El Usuario hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 1 del flujo normal</li> </ol> <p>EX2- El SM no pudo abrir el archivo</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar leer el archivo, verifique el formato o Intente más tarde.” y el botón “Ok”</li> <li>2. El Usuario hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 1 del flujo normal.</li> </ol> <p>EX1- El SM no pudo conectarse a la base de datos y no guardo la información.</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde.” y el botón “Ok” en una interfaz gráfica “Error Conexión”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM cierra “Error Conexión” y “Agregación de Canción”</li> <li>4. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	No identificadas
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	Extiende de CU-U-4: Gestionar biblioteca privada
<b>Prioridad:</b>	Media

La siguiente tabla muestra la descripción del caso de uso CU-U-4: Gestionar biblioteca privada

<b>ID:</b>	CU-U-4
<b>Nombre:</b>	Gestionar biblioteca privada
<b>Autor(es):</b>	César Sergio Martínez Palacios
<b>Fecha de creación:</b>	10 de marzo del 2021
<b>Fecha de actualización:</b>	[No aplica]
<b>Descripción:</b>	El <b>Usuario</b> explora y modifica las canciones que se encuentran en su repositorio o biblioteca privados y que solamente él puede observar
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 1 vez al día
<b>Disparador:</b>	El <b>Usuario</b> hace clic el botón “Biblioteca privada” en el pequeño menú que tiene de lado izquierdo en la ventana principal.
<b>Precondiciones:</b>	[No aplica]
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Biblioteca privada” con la opción “Agregar canción” y una lista de canciones privadas con la opción “Eliminar” (Ver FA1.1 y EX1)</li> <li>2. El <b>Usuario</b> selecciona la opción “Eliminar” de una canción (Ver FA 2.1 y FA2.2)</li> <li>3. El SM muestra una ventana de confirmación y las opciones “Si” y “No”</li> <li>4. El <b>Usuario</b> selecciona la opción “Si”(Ver EX3)</li> <li>5. Termina el caso de uso</li> </ol>
<b>Flujos Alternos:</b>	<p>FA1.1 Aun no ha registrado una canción en la biblioteca privada</p> <ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Biblioteca privada” con la opción “Agregar canción”</li> <li>2. El <b>Usuario</b> selecciona la opción “Agregar canción”</li> <li>3. El SM extiende al CU-U-3: Agregar canción personal</li> </ol> <p>FA2.1 El <b>Usuario</b> quiere agregar una canción</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> selecciona la opción “Agregar”</li> <li>2. El SM extiende al CU-U-3: Agregar canción personal</li> </ol> <p>FA2.2 El <b>Usuario</b> reproduce una canción</p> <ol style="list-style-type: none"> <li>1. El Usuario selecciona una canción</li> <li>2. El SM comienza a reproducir la canción seleccionada y regresa al paso 1 del flujo normal</li> </ol>

<b>Excepciones:</b>	EX1- El SM no pudo cargar la biblioteca privada y no mostro las canciones <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar cargar la biblioteca. Intente de nuevo más tarde.” y el botón “Ok”</li> <li>2. El Usuario hace clic el botón “Ok”</li> <li>3. El SM regresa a la ventana principal y termina el caso de uso</li> </ol>
<b>Postcondiciones:</b>	No identificadas
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	Extiende a: CU-U-3: Agregar canción personal Extiende a: CU-U-5: Gestionar lista de reproducción
<b>Prioridad:</b>	Media

La siguiente tabla muestra la descripción del caso de uso CU-U-5: Gestionar lista de reproducción

<b>ID:</b>	CU-U-5
<b>Nombre:</b>	Gestionar lista de reproducción
<b>Autor(es):</b>	Ricardo Moguel Sánchez
<b>Fecha de creación:</b>	11 de marzo del 2021
<b>Fecha de actualización:</b>	[No aplica]
<b>Descripción:</b>	El <b>Usuario</b> explora y modifica las canciones que se encuentran en su lista de canciones a reproducir.
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 1 vez al día
<b>Disparador:</b>	El <b>Usuario</b> hace clic el botón “Ver Lista de reproducción” en el pequeño menú que tiene de lado izquierdo en la ventana principal.
<b>Precondiciones:</b>	No aplica
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Lista de Reproducción” que muestra una lista de canciones privadas con las opciones “Regresar” y “Play”, “Quitar” para cada canción (Ver EX1)</li> <li>2. El <b>Usuario</b> selecciona la opción “Quitar” de una canción (Ver FA 2.1 y FA2.2)</li> <li>3. El SM muestra una ventana de confirmación y las opciones “Si” y “No”</li> <li>4. El <b>Usuario</b> selecciona la opción “Si” (Ver EX2)</li> <li>5. El SM elimina la lista de la colección de listas a reproducir</li> <li>6. Termina el caso de uso</li> </ol>
<b>Flujos Alternos:</b>	<p>FA 2.1: El Usuario desea Salir de “Lista de Reproducción”</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Regresar”</li> <li>2. El SM cierra la ventana “Lista de Reproducción”</li> <li>3. Termina caso de uso</li> </ol> <p>FA 2.2: El <b>Usuario</b> desea reproducir una canción.</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> hace clic el botón “Play” de una entrada en la tabla.</li> <li>3. El SM comienza a reproducir la canción seleccionada y regresa al paso 1 del flujo normal</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no pudo cargar la lista de reproducción y no mostro las canciones.</p> <ol style="list-style-type: none"> <li>4. El SM muestra el mensaje “Ocurrió un fallo al intentar cargar la biblioteca. Intente de nuevo más tarde.” y el botón “Ok”</li> <li>5. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>6. El SM regresa a la ventana principal</li> <li>7. termina el caso de uso</li> </ol>

	<p>EX2- El SM no pudo conectarse a la base de datos y no guardo el cambio en la lista de reproducción.</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar conectarse a la base de datos. Intente de nuevo más tarde.” y el botón “Ok” en una interfaz gráfica “Error Conexión”.</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM cierra “Error Conexión” y “Lista de Reproducción”</li> <li>4. Termina el caso de uso.</li> </ol>
<b>Postcondiciones:</b>	No identificadas
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	Extiende de: CU-U-4: Gestionar biblioteca privada
<b>Prioridad:</b>	Alta



La siguiente tabla muestra la descripción del caso de uso CU-U-6: Publicar canción

<b>ID:</b>	CU-U-6
<b>Nombre:</b>	Publicar canción
<b>Autor(es):</b>	César Sergio Martínez Palacios
<b>Fecha de creación:</b>	10 de marzo del 2021
<b>Fecha de actualización:</b>	[No aplica]
<b>Descripción:</b>	El <b>Usuario</b> desea publicar una canción y generara una solicitud para publicar su canción
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 1 vez al mes
<b>Disparador:</b>	El <b>Usuario</b> hace clic el botón “Publicar canción” en el pequeño menú que tiene de lado izquierdo en la ventana principal.
<b>Precondiciones:</b>	[No aplica]
<b>Flujo Normal:</b>	<p>9. El SM muestra la ventana publicación de canción” con las opciones “Subir”, “Agregar canción” y el formulario <b>datos_cancion</b></p> <p>10. El <b>Usuario</b> ingresa la información solicitada y selecciona la opción “Agregar canción”</p> <p>11. El SM abre el explorador de archivos del sistema operativo (Ver EX1)</p> <p>12. El <b>Usuario</b> selecciona el archivo de la canción a subir (Ver FA4.1)</p> <p>13. El SM carga el archivo y activa la opción “Subir” (Ver EX2)</p> <p>14. El <b>Usuario</b> selecciona la opción “Subir” (Ver FA6.1)</p> <p>15. El SM manda el archivo para aprobación, despliega mensaje de éxito y regresa al menú principal (Ver EX3)</p> <p>16. Termina el caso de uso</p>

<b>Flujos Alternos:</b>	<p>FA4.1 El <b>Usuario</b> cancela la operación</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> selecciona la opción “Cancelar” y cierra el explorador de archivos</li> <li>2. El SM regresa al paso 1 del flujo normal</li> </ol> <p>FA6.1 El <b>Usuario</b> cancela la operación</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> selecciona la opción “Cancelar operación”</li> <li>2. El SM muestra una ventana de confirmación con las opciones “Si” y “No”</li> <li>3. El <b>Usuario</b> selecciona la opción “Si”</li> <li>4. Termina el caso de uso</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no abrir el explorador de archivos</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar abrir el explorador. Intente de nuevo más tarde.” y el botón “Ok”</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 1 del flujo normal</li> </ol> <p>EX2- El SM no pudo abrir el archivo</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al intentar leer el archivo, verifique el formato o Intente más tarde.” y el botón “Ok”</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 1 del flujo normal</li> </ol> <p>EX3- El SM no pudo enviar la canción al administrador</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al conectar a red, Intente más tarde.” y el botón “Ok”</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 5 del flujo normal</li> </ol>
<b>Postcondiciones:</b>	Se crea una nueva instancia de CANCION con el estado “En aprobación”
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	No aplica
<b>Prioridad:</b>	Alta
<b>datos_cancion:</b>	<p>Título, Genero  Archivo, Autor  Duración, Álbum  Caratula, Año  Editora</p>

La siguiente tabla muestra la descripción del caso de uso CU-U-7: Buscar canción

<b>ID:</b>	CU-U-7
<b>Nombre:</b>	Buscar canción
<b>Autor(es):</b>	César Sergio Martínez Palacios
<b>Fecha de creación:</b>	10 de marzo del 2021
<b>Fecha de actualización:</b>	[No aplica]
<b>Descripción:</b>	El <b>Usuario</b> busca una canción en específico en las bibliotecas públicas y privadas
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 5 veces al día
<b>Disparador:</b>	El <b>Usuario</b> hace clic el buscador superior en la ventana principal
<b>Precondiciones:</b>	[No aplica]
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra una lista de canciones escuchadas recientemente con las opciones “Play” y “Detalles”</li> <li>2. El <b>Usuario</b> ingresa el nombre de la canción a buscar (Ver FA2.1)</li> <li>3. El SM muestra una lista de canciones con el nombre ingresado relacionado (Ver FA3.1 y EX1)</li> <li>4. El <b>Usuario</b> selecciona la canción a reproducir en el botón “Play”(Ver FA 4.1)</li> <li>5. El SM reproduce la canción escogida y cierra el buscador (Ver EX2)</li> <li>6. Termina el caso de uso</li> </ol>
<b>Flujos Alternos:</b>	<p>FA2.1 El <b>Usuario</b> cancela la operación</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> selecciona la opción “Cancelar” y cierra el buscador</li> <li>2. El SM regresa al menú principal y termina caso de uso</li> </ol> <p>FA3.1 El SM no encuentra una canción relacionada al nombre ingresado</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje” No existe una canción con ese nombre relacionado”</li> <li>2. Regresa al paso 2 del flujo normal</li> </ol> <p>FA4.1 El <b>Usuario</b> consulta la información de la canción</p> <ol style="list-style-type: none"> <li>1. El <b>Usuario</b> selecciona la opción “Detalles” de una canción</li> <li>2. El SM extiende al CU-C8</li> </ol>

<b>Excepciones:</b>	<p>EX1- El SM no pudo acceder a la base de datos a encontrar la canción</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al conectar a red, Intente más tarde.” y el botón “Ok”</li> <li>2. El Usuario hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 1 del flujo normal</li> </ol> <p>EX1- El SM no pudo reproducir la canción</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un error a la hora de reproducir la canción, Intente más tarde.” y el botón “Ok”</li> <li>2. El Usuario hace clic el botón “Ok”</li> <li>3. El SM regresa al paso 3 del flujo normal</li> </ol>
<b>Postcondiciones:</b>	No identificadas
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	Extiende a: CU-U-8: Consultar canción
<b>Prioridad:</b>	Media

La siguiente tabla muestra la descripción del caso de uso CU-U-8: Consultar canción

<b>ID:</b>	CU-U-8
<b>Nombre:</b>	Consultar canción
<b>Autor(es):</b>	César Sergio Martínez Palacios
<b>Fecha de creación:</b>	11 de marzo del 2021
<b>Fecha de actualización:</b>	[No aplica]
<b>Descripción:</b>	El <b>Usuario</b> desea consultar la información que contiene una canción seleccionada
<b>Actor(es):</b>	<b>Usuario</b>
<b>Frecuencia de uso:</b>	Aproximadamente 1 vez al DIA
<b>Disparador:</b>	El <b>Usuario</b> selecciona la opción “Detalles” de una canción en el CU-U-7: Buscar canción
<b>Precondiciones:</b>	[No aplica]
<b>Flujo Normal:</b>	<ol style="list-style-type: none"> <li>1. El SM muestra la ventana “Datos de canción” con el formulario <b>datos_cancion</b> lleno con la información de la canción y las opciones “Play” y “Regresar” (Ver EX1)</li> <li>2. El <b>Usuario</b> consulta los datos y selecciona la opción “Regresar” (Ver FA2.1)</li> <li>3. El SM regresa al CU-U-7: Buscar Canción</li> </ol>

<b>Flujos Alternos:</b>	<p>FA2.1 El <b>Usuario</b> reproduce la canción</p> <ol style="list-style-type: none"> <li>3. El <b>Usuario</b> selecciona la opción “Play”</li> <li>4. El SM comienza a reproducir la canción y regresa al paso 1 del flujo normal</li> </ol>
<b>Excepciones:</b>	<p>EX1- El SM no pudo obtener la información solicitada</p> <ol style="list-style-type: none"> <li>1. El SM muestra el mensaje “Ocurrió un fallo al conectarse a la red, Intente de nuevo más tarde.” y el botón “Ok”</li> <li>2. El <b>Usuario</b> hace clic el botón “Ok”</li> <li>3. El SM regresa al CU-U-7: Buscar Canción</li> </ol>
<b>Postcondiciones:</b>	No identificadas
<b>Reglas de negocio:</b>	No identificadas
<b>Incluye:</b>	No aplica
<b>Extiende:</b>	Extiende de: CU-U-7: Buscar canción
<b>Prioridad:</b>	Alta
<b>datos_cancion:</b>	<p>Título, Genero  Archivo, Autor  Duración, Álbum  Caratula, Año  Editora</p>

## **4. Construcción**

### **4.1. Selección justificada de pila tecnológica**

En esta sección se encuentran las descripciones y justificaciones de las tecnologías empleadas para el sistema SpotyMe.

#### **4.1.1. Aplicación de Cliente Rico**

Para el Cliente rico se utilizan 3 diferentes tecnologías para implementar las diferentes versiones del cliente rico.

Para el cliente web se utiliza Razor Pages. Para el cliente en aplicación de escritorio se utiliza Electron y para el cliente Android se utiliza Kotlin.

#### **Razor Pages**

Razor Pages es una implementación alternativa al patrón Model View Controller de ASP.NET para la creación de páginas web. Se utilizan paginas tipo CSHTML en donde se declara el comportamiento de una página. Esto simplifica el proceso de desarrollo web y debido a que el sistema SpotyMe es nuestra primera aplicación web se decidió utilizar Razor Pages para ser más productivos debido al uso de un Page Model en C# en el que tenemos más experiencia en. Razor Pages simplifica el desarrollo web debido a que en vez de utilizar múltiples paginas para un modelo, vista y controlador, se emplea una sola página controladora CSHTML para contener código HTML y enlaces URL para llamadas a la API. Para realizar peticiones a la API también se utiliza AJAX en JavaScript para construir peticiones HTTP para consumir recursos de la API.

#### **Electron**

Se utilizó el framework Electron, que habilita la construcción de aplicaciones de escritorio con el uso de tecnologías de desarrollo web para adaptar la aplicación web a una aplicación de escritorio con la interfaz grafica del cliente web existente. Debido a que Razor Pages es una tecnología web compatible con Electron.NET una extensión de Electron para tecnologías web de ASP.NET se decidió aplicarla para no tener que desarrollar una aplicación GUI de escritorio y tener que reescribir código para este cliente. En vez se utilizó Electron que transforma el código fuente de la aplicación web existente. La ventaja principal de usar Electron fue disminuir drásticamente el tiempo de desarrollo de interfaces graficas independientes entre el cliente web y el cliente de escritorio para poder en vez enfocarse en desarrollar con calidad la capa lógica compartida por ambos clientes que se conecta con la API.

#### **Kotlin**

Kotlin es un lenguaje de programación para desarrollo móvil que sirve como alternativa a Java para aplicaciones móvil en Android. El lenguaje es compatible e interoperable con Java, esto significa que código Kotlin y código Java pueden ser utilizados para la misma aplicación sin problemas grandes. Debido a nuestra experiencia previa desarrollando en Java consideramos adecuado el uso de Kotlin para desarrollo del cliente móvil debido a

la similitud entre los lenguajes y facilidad de aprendizaje. Adicionalmente se usó Kotlin debido a la facilidad de desarrollar vistas en Android y realizar llamadas a una API con el uso de Kotlin/JS que facilita llamadas entre código Kotlin en el cliente y JavaScript, el lenguaje usado para la API.

#### **4.1.2. Aplicación de servicios en API**

Para el servidor se crearon microservicios en .NET y en Node JS que se conectan a bases de datos relacionales en MySQL y bases de datos de documentos en MongoDB. Para conectar los microservicios se creó un Application Programming Interface (API) en Node Js. Cada base de datos, microservicio y la API se encuentra dentro de un contenedor de Docker.

### **C#**

El lenguaje C# es un lenguaje de programación orientado a objetos para desarrollo de aplicaciones en Windows. Debido a nuestra profunda familiaridad con este lenguaje, se desarrollaron todos los microservicios en C# con excepción del Microservicio de Streaming. Las principales ventajas de C# aparte de la familiaridad con el lenguaje, es el manejo de llamadas asíncronas para peticiones HTTP con los métodos Async y Await. También se utilizó una biblioteca JSON para formatear las respuestas de peticiones HTTP en objetos JSON con información personalizada de acuerdo con los requisitos de cada microservicio.

### **MySQL**

Se utilizó una imagen de MySQL, un sistema de gestión de bases de datos relacionales para persistir una base de datos “Accounts” con la información de cuentas de usuarios dentro de un contenedor de Docker. Con el sistema de gestión se logró proteger el acceso a la base de datos de la información personal de usuarios en el Microservicio de Cuentas con un usuario y contraseña otorgada por MySQL para conectarse a la base de datos. Debido a que MySQL es compatible con Docker y tiene medidas de seguridad se decidió utilizar el sistema de gestión para persistir datos.

### **MongoDB**

MongoDB es un sistema de gestión de bases de datos de documentos. Se utilizó para persistir los metadatos de canciones, álbumes y géneros de música de la biblioteca publica y de la biblioteca privada. Debido a que MongoDB es compatible con Docker, se empleó el uso de una imagen de MongoDB para crear dos diferentes contenedores, uno contiene la base de datos de canciones de la biblioteca pública y el otro encapsula la base de datos de la biblioteca privada. Ambos contenedores están protegidos con un usuario y contraseña otorgado por el sistema de gestión de MongoDB.

Finalmente, se decidió utilizar MongoDB debido a la estructura de los documentos en las bases de datos en estructuras BSON que es compatible con JSON, utilizado en los microservicios simplifica las operaciones CRUD sobre la base de datos desde nuestros microservicios escritos en C#. Las medidas de seguridad y la facilidad y flexibilidad de

gestionar los metadatos de canciones en una base de datos de documentos en contraste a una base de datos relacional son las razones principales para seleccionar MongoDB para persistir metadatos de canciones en el sistema SpotyMe.

## **Node.js**

Node.js es un entorno de tiempo de ejecución para ejecutar código JavaScript fuera de un web browser. Se utiliza Node.js para crear servidores y el back-end de aplicaciones web. Se utilizó Node.js para crear una API RESTful y el Microservicio de Streaming. Se usó Node.js para la API debido a la facilidad de crear un servidor de respuestas HTTP con Express, una biblioteca de Node.js que facilita el manejo de peticiones HTTP. Se utilizó esta tecnología también en el Microservicio de Streaming junto con la biblioteca Express-Upload para persistir archivos por medio de peticiones HTTP dentro del sistema de archivos dentro de un contenedor de Docker.

## **Docker**

Se utilizó Docker para crear contenedores que encapsulan los diferentes microservicios, bases de datos y la API. Los contenedores de Docker sirven como pequeñas máquinas virtuales que ejecutan el código que contienen en una máquina virtual de Linux. Con el uso de contenedores se puede levantar el servidor del sistema de SpotyMe en cualquier plataforma ya sea MacOS, Linux o Windows. Gracias al encapsulamiento de los Microservicios y API, se facilita el mantenimiento de los diferentes servicios ya que se puede modificar un microservicio durante el despliegue de la API y los demás microservicios al desactivar un solo contenedor y gestionar el programa que contiene sin desactivar el servidor y el servicio que ofrece a clientes.

### **4.2. Estándar de Codificación**

El estándar de codificación se encuentra en un documento separado que puede ser acesado [aquí](#). También se encuentra en la rama Main del repositorio del proyecto de SpotyMe.

### **4.3. Reportes de Análisis Estático de Código**

En esta sección se pueden ver los reportes de análisis estático realizados para cada microservicio y para la API.

Después de usar la herramienta SonarQube para realizar un análisis estático del código de la API y los microservicios, se reveló que el código es aceptable de acuerdo con la herramienta. En la siguiente captura se ve una sobrevista del análisis estático.



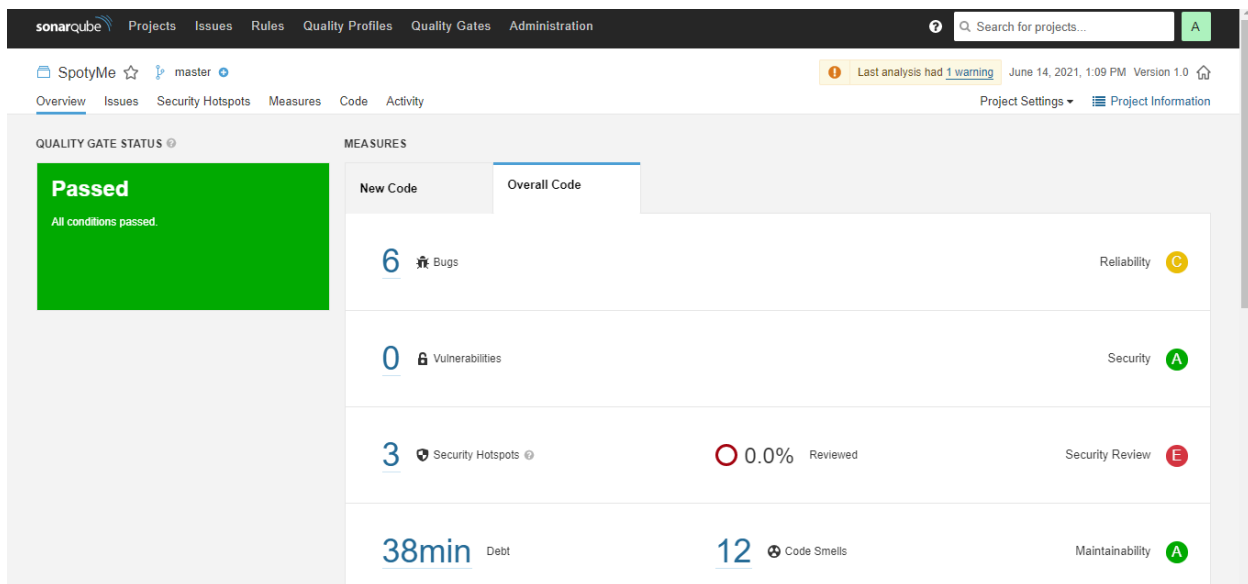


Figura 30 Captura de sobrevista del análisis estático

En la siguiente captura se demuestra los 6 bugs identificados. Los bugs encontrados en la API son irrelevantes ya que se encuentran en los paquetes importados de node\_modules utilizados para el manejo de transmisión de canciones.

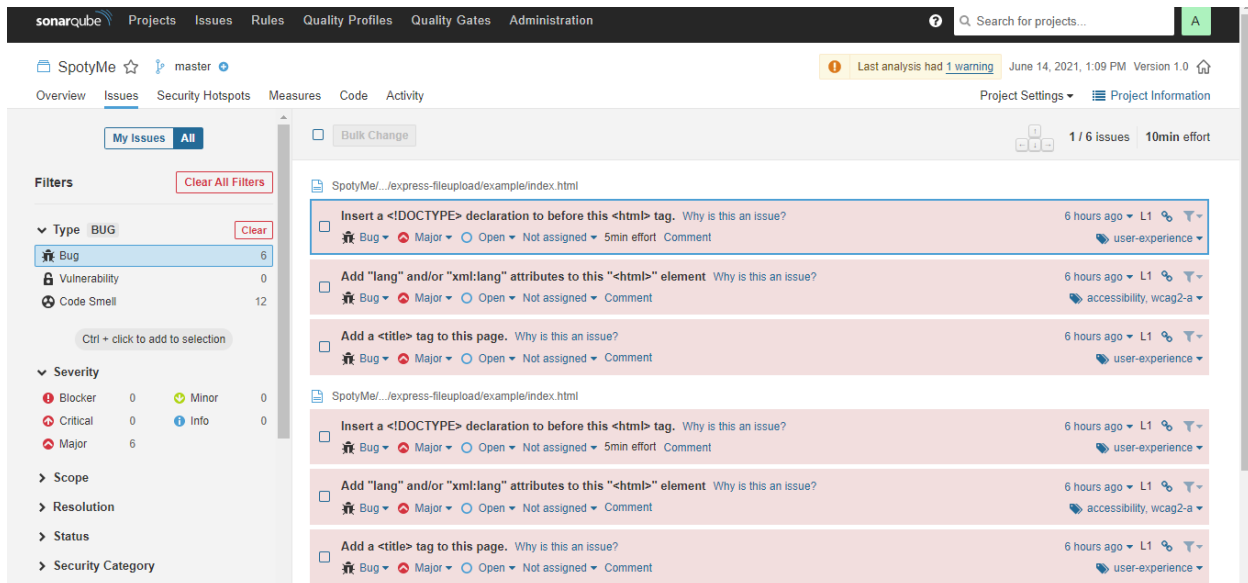


Figura 31 Captura de bugs del análisis estático

En la siguiente captura se muestran las áreas de vulnerabilidad de seguridad. Se puede ver que las 3 vulnerabilidades encontradas son de bajo peligro.

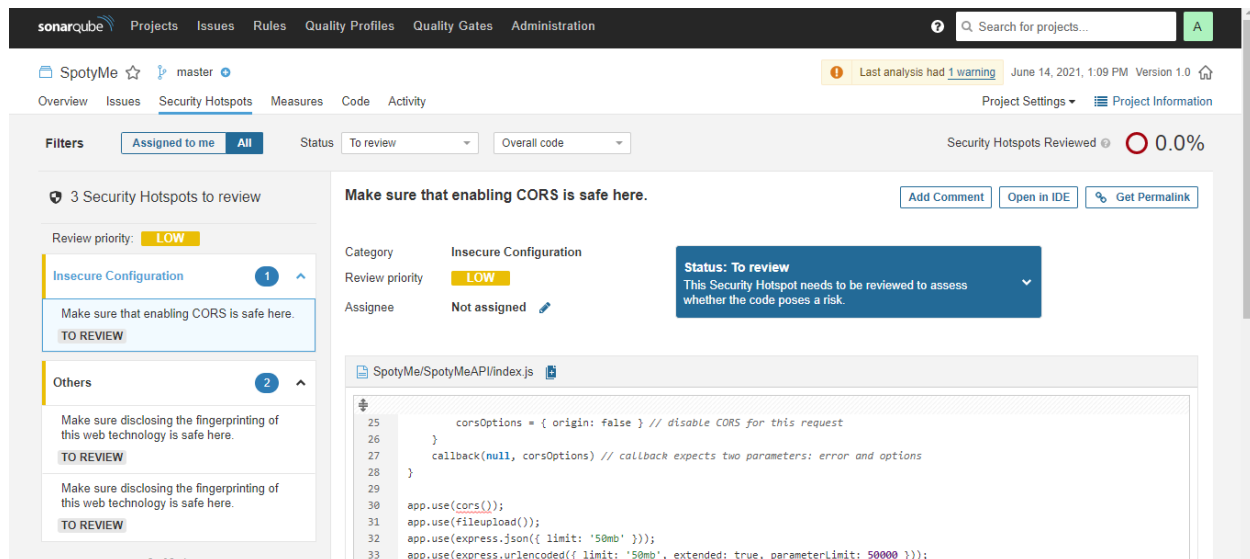


Figura 32 Captura de vulnerabilidades del análisis estático

En la siguiente captura se puede ver los *code smells* o malas practicas de codificación identificadas. Se puede ver que las 12 malas prácticas identificadas por la herramienta son de baja severidad.

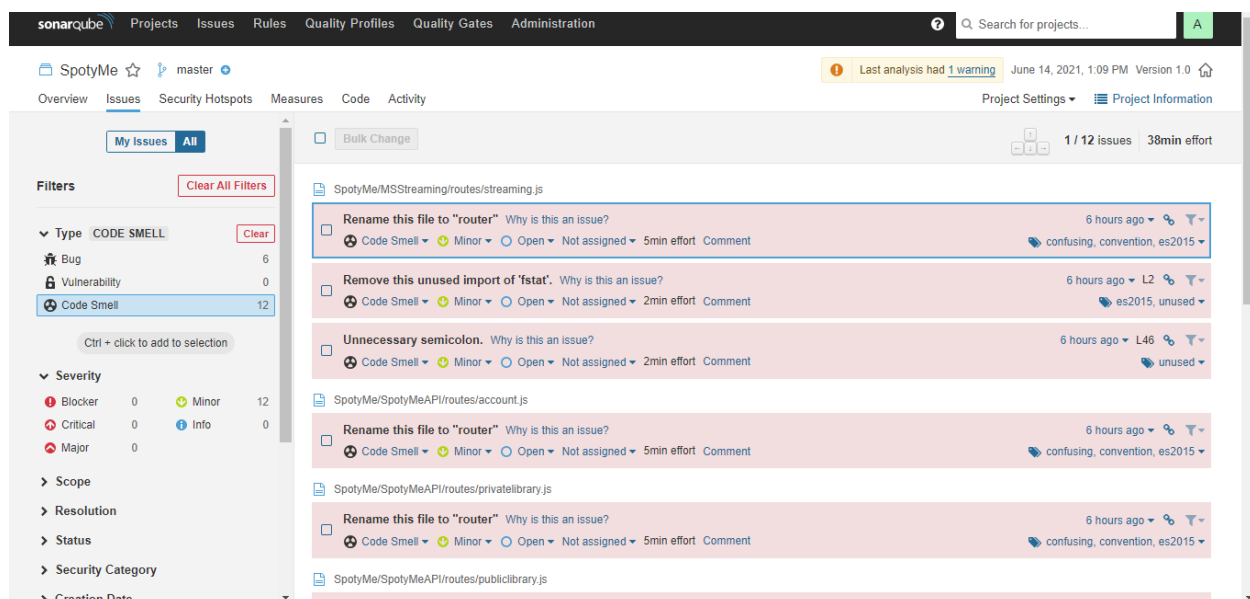


Figura 33 Captura de code smells del análisis estático

#### 4.4. Prácticas de Construcción Realizadas

Para el back-end del sistema se empleó una API RESTful que redirige peticiones HTTP a los Microservicios. Aunque existen varios tipos de métodos para peticiones HTTP, para el desarrollo de la API y microservicios se utilizaron los métodos POST, para realizar operaciones de creación de datos, PUT, para realizar operaciones de actualización de datos, y GET para realizar consultas de datos en bases de datos conectadas a los microservicios.

Una API RESTful, utiliza peticiones HTTP para la comunicación a los contenedores de Microservicios para acceder y utilizar recursos encontrados en los contenedores que contienen las bases de datos asociados a los microservicios. Para coordinar las conexiones entre contenedores y crear contenedores se utiliza un archivo YAML llamado Docker-compose y archivos Dockerfile para cada Microservicio.

Para construir el servidor entonces. Se levanta el servicio con el Docker-compose que contiene instrucciones para asignar variables de entorno que asignan la información de conexión y un puerto a exponer para cada contenedor. Los contenedores son creados con ayuda de un archivo Dockerfile encontrado en la carpeta de cada Microservicio del proyecto. Incluso la API contiene un Dockerfile para levantarla dentro de un contenedor. Una vez que los contenedores de Docker son creados y se ejecuta el programa que contienen, exponen un puerto asignado en el Docker-compose para recibir peticiones HTTP.

Las peticiones HTTP permitidas son:

##### **GET**

El método GET solicita un recurso. Las peticiones tipo GET realizan una consulta por datos existentes en el sistema. Para construir una petición GET se pasa por parámetro en la URL de la petición un query que representara un tipo de dato y una información a consultar por. Por ejemplo:

GET / http://localhost:8083/audio/streaming?address=/spotyme/specialist.mp3

##### **POST**

El método POST crea un registro al enviar datos a guardar en un recurso. Para construir una petición POST se pasa por parámetro en el body de la petición un objeto JSON que contiene los datos de una entidad a guardar. En el header o cabecera de la petición Content-Type se indica el tipo del contenido. Para peticiones al Microservicio de Streaming se usaba el Content-Type MP3 o JPG y un cuerpo de tipo multipart/form-data. En los demás microservicios se identificaba un Content-Type de application/json para identificar que el cuerpo es un objeto JSON dentro de la cabecera.

##### **PUT**

El método PUT 'pone' o cambia los datos actuales en un recurso con los datos de la petición para realizar una actualización. Se utilizan los mismos parámetros que las peticiones POST para construir la petición POST. Como no se permite actualizar archivos en el sistema. Los métodos PUT siempre tenían un Content-Type de application/json debido a que todas nuestras peticiones PUT llevaban un objeto JSON en el cuerpo.

## **5. Pruebas**

### **5.1. Plan de Pruebas para la API**

#### **5.1.1. Introducción**

El plan de prueba tiene como objetivo proporcionar información y un marco necesario para planificar y realizar los procesos de prueba que son necesarios para la API del sistema SpotyMe y los microservicios conectados a ella. Se divide en partes iniciando con el contexto, donde se dan los antecedentes del proyecto a probar, cuáles serán los elementos para probar, define el alcance de las pruebas, los supuestos y limitaciones para el esfuerzo. En la parte de riesgos se especifican los riesgos relacionados al producto y el proyecto. La sección de estrategia de prueba define los subprocesos que se ejecutarán, los documentos generados por las actividades, las técnicas de prueba, cuáles son los criterios de finalización, los requisitos de datos necesarios y del entorno para los casos de prueba.

#### **5.1.2. Contexto**

En esta sección se detallan los elementos de la API a probar y el alcance de las pruebas.

##### **5.1.2.1. Elementos Que Probar**

La API del sistema SpotyMe se divide entre las funcionalidades otorgadas por los diferentes microservicios con las que se comunica. Las pruebas realizadas se enfocan en estos diferentes microservicios los cuales son:

##### **1. Microservicio de Cuentas**

o En este microservicio se realiza la gestión de cuentas y el inicio de sesión al sistema y el registro de nuevas cuentas para habilitar el uso de las demás funcionalidades del sistema por usuarios.

##### **2. Microservicio de Biblioteca Pública**

o En este microservicio se realiza la solicitud de publicación de una canción y búsqueda de canciones por parte de usuarios y la evaluación de solicitudes por el actor Administrador.

##### **3. Microservicio de Biblioteca Privada**

o En este microservicio se brinda la funcionalidad para consultar de la base de datos canciones de la biblioteca privada de un usuario y realizar la gestión de estas.

##### **4. Microservicio de Streaming**

o En este microservicio se realiza el guardado de canciones e imágenes y la visualización de imágenes y consulta de archivos mp3 de canciones.

##### **5.1.2.2. Alcance de la prueba**

Para facilitar las distintas tareas que surgen en los microservicios para escuchar canciones en el sistema SpotyMe, el plan de prueba busca probar dicha funcionalidad

para garantizar que el código escrito sea capaz de satisfacer las necesidades del Usuario y el Administrador.

El alcance de nuestras pruebas está tomando en cuenta los distintos microservicios existentes dentro del sistema. Para cada microservicio se busca probar la funcionalidad correcta conforme a los requisitos funcionales del sistema y los casos de uso identificados.

#### **5.1.2.3. Suposiciones y limitaciones**

El plan de pruebas está estimado con una fecha máxima marcada hasta el 15 de junio del año 2021 por lo que el alcance del plan de pruebas será adaptado para cumplir con el tiempo que se tiene. El plan será ejecutado por los mismos desarrolladores del sistema por lo que al tener un conocimiento previo del software se espera que el plan sea más fluido, respecto a la disponibilidad del personal se debe de tener en cuenta que el tiempo de los desarrolladores al contar con otras actividades.

Se utilizará el estándar ISO-29119 el cual nos otorga una estructura a seguir que garantiza que las pruebas realizadas tengan todo lo necesario para ser optimas en su ejecución.

#### **5.1.3. Registro de Riesgos**

En esta sección se muestran los riesgos del producto y proyecto asociados a la ejecución del plan de pruebas y el impacto analizado en el caso de la manifestación de dichos riesgos.

##### **5.1.3.1. Riesgos del Producto**

En esta sección se muestran los riesgos asociados a probar la API.

Tabla 4 Riesgos del producto de SpotyMe

ID	Nombre del Riesgo	Descripción del Riesgo
1	Importaciones de bibliotecas externas o internas a la solución del código, no son soportados	Al trasladar el proyecto al ambiente de prueba, importes a bibliotecas no son compatibles y causan conflictos.
2	Perdida de conexión a contenedor de base de datos o microservicio	Se pierde la conexión al servidor de la base de datos durante pruebas.
3	Archivos persistidos son demasiados grandes	Hay problemas al acomodar textos muy largos ingresados en campos de texto de la capa GUI.
4	No existen puertos disponibles para los contenedores	Puertos utilizados por contenedores son utilizados por otros recursos de una computadora.
5	Respuestas HTTP erróneas	Al realizar una petición a un microservicio se retorna una respuesta de código 200

		cuando debe de ser de código 400 o viceversa.
6	Consulta a base de datos demora mucho	Al realizar consultas a la base de datos remota el tiempo de respuesta es muy largo lo que causa que se retrase la ejecución de pruebas.
7	Tiempo de respuesta muy largo	El sistema se ralentiza, causa que el tiempo de respuesta sea muy largo y se causa una falla debido a estrés excesivo.

### 5.1.3.2. Riesgos del Proyecto

En esta sección se muestran los riesgos asociados al equipo de pruebas.

Tabla 5 Riesgos del proyecto de SpotyMe

ID	Nombre del Riesgo	Descripción del Riesgo
1	Falta de miembros de equipo	No hay suficientes miembros en el equipo de pruebas para ejecutar completamente el plan de prueba
2	Falta de experiencia y conocimiento	Los miembros disponibles carecen de los conocimientos y experiencia necesaria para cumplir las actividades detalladas en el plan de prueba.
3	Falta de familiarización con herramientas de seguimiento de pruebas	No hay suficiente familiarización con las herramientas de seguimiento de pruebas para utilizar las herramientas efectivamente.
4	Se expira la licencia de la base de datos remota	Se expira la licencia de la base de datos remota o se pierde el acceso a la base de datos remota alojada en un servidor de terceros.
5	Tiempo excesivo dedicado a reportes de incidencias	Tiempo excesivo dedicado a reportes de faltas identificadas durante pruebas en los reportes de incidencias.
6	No se ejecuta completamente el plan de pruebas.	No todas las actividades y técnicas de prueba determinadas en el plan de pruebas son ejecutadas y finalizadas.

## 5.2. Procedimiento de Prueba

### **5.2.1. Subprocesos de Prueba**

Para la realización de las pruebas del Sistema SpotyMe se identificaron los siguientes sub-procesos, los cuales se enumeran en el orden que serán realizados. Las pruebas realizadas fueron separadas por los Microservicios consumidos por la API y a continuación se muestra la organización de las pruebas por cada uno de ellos:

#### **Métodos GET**

Caso de Uso UML asociado

Casos de Prueba asociado

Enlace a CURL de prueba

#### **Métodos POST**

Caso de Uso UML asociado

Casos de Prueba asociado

Enlace a CURL de prueba

#### **Métodos PUT**

Caso de Uso UML asociado

Casos de Prueba asociado

Enlace a CURL de prueba

Con esta separación de pruebas se logrará cubrir un 100% de métodos HTTP diseñados e implementados en la API al igual que los flujos alternos de los métodos HTTP como una respuesta fallida. Adicionalmente queda mencionar que para todas las pruebas se elaboraran en base a el método de pruebas basados en casos de uso.

### **5.2.2. Formato de Casos de Prueba**

En esta sección se muestra la plantilla utilizada para realizar los casos de prueba basado en el formato especificado en el estándar ISO/ IEC/IEEE 29919.

Cada caso de prueba sigue el siguiente formato:

<b>Sistema:</b> [1]	<b>Unidad:</b> [2]
---------------------	--------------------

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
[3]	[4]	[5]	[6]
Entradas	Resultados esperados	Resultados observados	Resultado del caso de prueba
[7]	[8]	[9]	[10]
Diseño	Aplicó	Fecha	
[11]	[12]	[13]	

A continuación, se muestra la especificación para el llenado de los casos de prueba.

[1] Indicar el nombre del sistema que será sometido a la prueba.

[2] Indicar el nombre de la unidad a programación a probar.

[3] Indicar el identificador o clave, que permite distinguir de manera única el caso de uso de otros casos. Las herramientas automatizadas pueden generar estos controles de identificación, o puede incluir una clave si la prueba se genera de manera manual. Este identificador no puede cambiarse o reasignarse, durante el proceso de prueba pues será usado con propósitos de trazabilidad.

[4] Describir brevemente el objetivo que persigue el caso de prueba.

[5] Definir la prioridad de este caso para la prueba, los casos de mayor prioridad se ejecutarán antes que los casos de menor prioridad.

[6] Describir el estado requerido del entorno de prueba y cualquier restricción especial relacionada con la ejecución del caso de prueba.

[7] Especificar las acciones necesarias para conducir el programa bajo prueba a un estado en el que el resultado esperado se pueda comparar con los resultados reales. La descripción debe facilitar el entendimiento de quien realiza la prueba.

[8] Especificar las salidas esperadas y el comportamiento requerido del programa bajo prueba, en respuesta a las entradas que se le das las precondiciones indicadas.

[9] Indicar las salidas observadas y el comportamiento del programa bajo prueba, al ejecutar el caso de prueba.

### 5.2.3. Criterios de Finalización

El sistema a probar cuenta con un total de 10 casos de uso, de los cuales todos están implementados en la API. Se tomó la decisión de que el criterio de finalización para las pruebas fuera el cubrir al menos 7 casos de uso, lo cual nos daría una cobertura aproximada del 100% de todos los flujos de funcionalidades accesibles a los clientes en los Microservicios ofertados.



### 5.2.3.1. Casos de uso asociados

En la siguiente tabla se muestran los casos de uso a probar y los microservicios asociados a ellos.

Servicio	Caso de Uso Por Probar	Descripción de Servicio
Microservicio de Biblioteca pública	CU-U-6: Publicar canción CU-A-1: Gestionar solicitudes de canciones	En este servicio se encuentra la funcionalidad necesaria para solicitar la publicación de una canción de un usuario y evaluar la solicitud por el administrador.
Microservicio de Cuentas	CU-A-2: Banear Usuario CU-U-1: Registrar cuenta CU-U-2: Actualizar cuenta	En este servicio se encuentra la funcionalidad para crear una cuenta para acceso al sistema y modificar datos personales de usuario para darle acceso al sistema.
Servicio de Streaming	CU-U-7: Buscar canción CU-U-8: Consultar canción	En este servicio se brinda la funcionalidad para transmitir canciones e imágenes y guardarlas en un contenedor
Microservicio de Biblioteca Privada	CU-U-3: Agregar canción personal CU-U-4: Gestionar lista de reproducción CU-U-5: Gestionar biblioteca privada	En este servicio se brinda la funcionalidad para consultar de la base de datos canciones de la biblioteca privada de un usuario y realizar la gestión de estas.

## 5.3. Resultados del Plan de Pruebas

En esta sección se muestran los resultados de los casos de prueba ejecutados sobre los métodos HTTP de la API.

### 5.3.1. Resultados de Pruebas del Servicio de Cuentas del API

En esta sección se muestran los resultados de las pruebas de métodos HTTP para el servicio de Cuentas de la API

#### Métodos GET

##### GetAccount

Caso de uso asociado: CU-U-1: Registrar cuenta:

Caso: Consultar cuenta sin params

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#consultar-cuentas-sin-params>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-GetAccount-01	Comprobar que sea exitosa el flujo: Consultar cuentas	Alta	Debe existir cuentas de usuarios
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Params Vacios	Objeto JSON con mensaje "Account found" y arreglo de cuentas	Objeto JSON con mensaje "Account found" y arreglo de cuentas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Caso: Consultar cuenta con username

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Resultados-de-Pruebas#consultar-cuenta-con-username>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-GetAccount-02	Comprobar que sea exitosa el flujo: Consultar cuenta específica	Media	Debe existir cuentas de usuarios
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Username = Pklove345	Objeto JSON con mensaje "Account found" y arreglo de cuentas bajo el username consultado	Objeto JSON con mensaje "Account found" y arreglo de cuentas bajo el username consultado	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos POST

### Login

Caso de uso asociado: CU-U-1: Registrar cuenta:

Caso: Login con administrador

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#login-con-administrador>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-login-01	Comprobar que sea exitosa el flujo: Iniciar sesión como Administrador	Alta	Debe existir una cuenta de administrador registrada
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Username = correoAdmin@hotmail.com Password = 4105751805	Objeto JSON con mensaje "Welcome admin"	Objeto JSON con mensaje "Welcome admin"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Login con usuario

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#login-con-usuario>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-login-02	Comprobar que sea exitosa el flujo: Iniciar sesión como Usuario	Alta	Debe existir una cuenta de usuario registrada
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Username = delmanclen@hotmail.com Password = -5118730917	Objeto JSON con mensaje "Login successful"	Objeto JSON con mensaje "Login successful"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Login con usuario no existente

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#password-incorrecto>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-login-03	Comprobar que se muestre error en el flujo: username correcto con contraseña incorrecta	Alta	Debe existir una cuenta de registrada
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Username = delmanclen@hotmail.com Password = 4105751805	Objeto JSON con mensaje "Incorrect password. Please try again"	Objeto JSON con mensaje "Incorrect password. Please try again"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Login con usuario no existente

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#login-con-usuario-no-existente>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-login-04	Comprobar que se muestre error en el flujo: username no existe	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Username = ddd Password = dd	Objeto JSON con mensaje ERROR: This username is not registered. Please try a different one	Objeto JSON con mensaje ERROR: This username is not registered. Please try a different one	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## RegistrarCuenta

Caso de uso asociado: CU-U-1: Registrar cuenta:

Caso: Registrar cuenta exitosa

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Resultados-de-Pruebas/edit#registro-con-%C3%A9xito>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-registerAccount-01	Comprobar que sea exitosa el flujo: Registrar cuenta valida	Alta	No existe un usuario registrado con el mismo nombre de usuario y correo.
Entrada	Resultado Esperado	Resultado Obtenido	Estado
<pre>"Username": "faraon", "Email": "superchidoRAA", "IsUser": 1, "StatusId": 1, "Status": {   "StatusId": 1,   "Name": "Activo" }, "Passwords": [ { "PasswordId": "82f92dfb- dba9-4985-b295- 301f2741ddf6", "PasswordString": "Activo" } ]</pre>	Objeto JSON con mensaje "Account registered successfully"	Objeto JSON con mensaje "Account registered successfully"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Registrar cuenta con mismo email

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#registrar-cuenta-con-mismo-email>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-registerAccount-02	Comprobar que se muestre error en el flujo: Registrar la misma cuenta por segunda vez	Alta	No existe un usuario registrado con el mismo nombre de usuario y correo.
Entrada	Resultado Esperado	Resultado Obtenido	Estado
<pre>"Username": "faraon", "Email": "superchidoRAA", "IsUser": 1, "StatusId": 1, "Status": {   "StatusId": 1,</pre>	Objeto JSON con mensaje "ERROR: This email is already used for another account. Please use a different one"	Objeto JSON con mensaje "ERROR: This email is already used for another account. Please use a different one"	Pasó

<pre>       "Name": "Activo"     },     "Passwords": [       {         "PasswordId": "82f92dfb-         dba9-4985-b295-         301f2741ddf6",         "PasswordString": "Activo"       }     ] </pre>			
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Registrar cuenta sin datos

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#registro-con-cuerpo-vac%C3%ADo>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-registerAccount-03	Comprobar que se muestre un mensaje indicando éxito igual a falso en el flujo: Registrar cuenta sin datos	Alta	No existe un usuario registrado con el mismo nombre de usuario y correo.
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje " Object reference not set to an instance of an object."	Objeto JSON con mensaje " Object reference not set to an instance of an object."	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos PUT

### UpdateAccount

Caso de Uso asociado: CU-U-2: Actualizar cuenta:

Caso: Actualización de email con éxito

Enlace a CURL: [https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas/\\_edit#actualizaci%C3%B3n-de-datos-con-%C3%A9xito](https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas/_edit#actualizaci%C3%B3n-de-datos-con-%C3%A9xito)

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-updateAccount-01	Comprobar que existe el flujo: Éxito al actualizar cuenta.	Alta	No existe un usuario registrado con el mismo correo.
Entrada	Resultado Esperado	Resultado Obtenido	Estado
{ "AccountId": "86c8bf6-678a-4f3c-a103-4e7b02e411e6", "Username": "faraonNUEVO", "Email": "nuevocambiado@gmail.com" }	Objeto JSON con mensaje "Account update successful"	Objeto JSON con mensaje "Account update successful"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Caso: Actualizar con email duplicado

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas/edit#actualizar-con-email-duplicado>

## Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-updateAccount-02	Comprobar que existe el flujo de error: No se puede utilizar un correo registrado previamente	Alta	Existe un usuario registrado con el mismo correo.
Entrada	Resultado Esperado	Resultado Obtenido	Estado
{ "AccountId": "86c8bf6-678a-4f3c-a103-4e7b02e411e6", "Username": "faraon", "Email": " zS18012143@estudiantes.uv.mx"	Objeto JSON con mensaje "ERROR: This email is already used for another account. Please use a different one"	Objeto JSON con mensaje "ERROR: This email is already used for another account. Please use a different one"	Pasó

<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## BanAccount

Caso de uso asociado: CU-A-2: Banear Usuario

Caso: Banear con éxito

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#banear-con-exito>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-banAccount-01	Comprobar que existe el flujo: Éxito al banear cuenta.	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado
accountId=da43f900-890f-469e-90b3-18cdfacce74b	Objeto JSON con mensaje "Account ban successful"	Objeto JSON con mensaje "ERROR: The account to ban could not be found."	Falló
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Cuenta no encontrada

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#cuenta-no-encontrada>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-banAccount-02	Comprobar que existe el flujo: No se encontró cuenta	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado
accountId=444	Objeto JSON con mensaje "ERROR: The account to ban could not be found."	Objeto JSON con mensaje "ERROR: The account to ban could not be found."	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	



Caso: Banear cuenta baneada

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#banear-cuenta-baneada>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-banAccount-03	Comprobar que existe el flujo: Error no se puede banear una cuenta dos veces	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado
accountId=da43f900-890f-469e-90b3-18cdfacce74b	Objeto JSON con mensaje "ERROR: This account has already been banned."	Objeto JSON con mensaje "ERROR: The account to ban could not be found."	Falló
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Banear cuenta administrador

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Resultados-de-Pruebas#banear-cuenta-administrador>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-banAccount-04	Comprobar que existe el flujo: Éxito al banear cuenta.	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado
accountId=537a412b-e084-474a-a68d-705982b61802	Objeto JSON con mensaje "ERROR: Administrator account cannot be banned."	Objeto JSON con mensaje "ERROR: The account to ban could not be found."	Falló
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

**UpdatePassword**

Caso de uso asociado: CU-U-2: Actualizar cuenta:

Caso: Actualizar password

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Resultados-de-Pruebas#passowrd-actualizada>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-updatePassword-01	Comprobar que existe el flujo: Éxito al actualizar password.	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado
{ "PasswordId": "d034ddb8-97c5-45e0-9bab-bd46f040f092",  "PasswordString": "RA4", "OwnerId": "8fa5e2b8-9d8a-444d-9901-81f0b8ad3dbd" }	Objeto JSON con mensaje "Password update successful"	Objeto JSON con mensaje "Password update successful"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: No se encuentra el password

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Resultados-de-Pruebas#no-se-encuentra-el-password>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Cuentas
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-updatePassword-02	Comprobar que existe el flujo: No se encuentra el password a actualizar	Alta	Existe un usuario registrado
Entrada	Resultado Esperado	Resultado Obtenido	Estado

{ "PasswordId": "52374be9-a9a6-4dda-b3d6-4b736b7a9006",  "PasswordString": "RA4", "OwnerId": "6e879fb7-c573-4d47-928a-ec6d24d6df04" }	Objeto JSON con mensaje " Could not locate password to update"	Objeto JSON con mensaje " Could not locate password to update"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

### 5.3.2. Resultados de Pruebas del Servicio de Biblioteca Pública del API

En esta sección se muestran los resultados de las pruebas basadas en caso de uso del servicio de biblioteca pública.

#### Métodos POST

##### UploadGenre

Caso de uso asociado: CU-U-6: Publicar canción

Caso: Registrar genero

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BABlica#registrar-genero>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadGenre-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Genero Prueba" }'	Objeto JSON con mensaje " registered successfully"	Objeto JSON con mensaje " registered successfully"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

##### UploadAlbum

Caso de uso asociado: CU-U-6: Publicar canción

Caso: Registrar album

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#registrar-album>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadAlbum-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Prueba Album", "ImageAddress": "pruebaAlbum/imagenPrueba", "ImageName": "imagenPrueba", "ImageType": ".jpg" }'	Objeto JSON con mensaje "registered successfully"	Objeto JSON con mensaje "registered successfully"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## UploadMusic

Caso de uso asociado: CU-U-6: Publicar canción

Caso: Registrar metadato de archivo

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#registrar-metadato-de-archivo-mp3>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadMusic-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Prueba Album", "ImageAddress": "pruebaAlbum/imagenPrueba", "ImageName": "imagenPrueba", "ImageType": ".mp3" }'	Objeto JSON con mensaje "registered successfully"	Objeto JSON con mensaje "registered successfully"	Pasó

"ImageType": ".jpg"			
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## UploadSongRequest

Caso de uso asociado: CU-U-6: Publicar canción

Caso: Registrar solicitud de canción

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#registrar-solicitud-de-cancion>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadSongRequest-01	Comprobar que existe el flujo: Éxito al guardar	Media	Se debe haber registrado la canción y se necesita saber su Id
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Title": "The Poem For Everyone\'s Souls", "UploaderId": "3f835f14-8111-4a1d-8385-2871743290e6", "ArtistId": "88f96978-6f13-483c-8a46-10b3e4d52962", "MultimediaId": "437f223f-1f82-4a1c-b3b7-04bdc4188e0a", "Composer": "Shigeru Meguro", "Producer": "Atlus", "Duration": "420", "ReleaseYear": "2009", "AlbumId": "fccdffbd-cca1-4b9b-8882-5067d8da2060", "GenreId": "5", "StatusId": "" }'	Objeto JSON con mensaje "Song request registered successfully"	Objeto JSON con mensaje "Song request registered successfully"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos PUT

### ApproveSong

Caso de uso asociado: CU-A-1: Gestionar solicitudes de canciones

Caso: Aceptar solicitud

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#aceptar-solicitud>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ApproveSong-01	Comprobar que existe el flujo: Éxito al aprobar solicitud	Media	Se debe haber registrado la solicitud y se necesita conocer su Id
Entrada	Resultado Esperado	Resultado Obtenido	Estado
songId=f078f4ea-7663-4c78-9141-db5df4d8997a	Objeto JSON con mensaje "Song approved successfully"	Objeto JSON con mensaje "Song approved successfully"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Negar aceptar solicitud rechazada

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#negar-aceptar-solicitud-rechazada>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ApproveSong-02	Comprobar que existe el flujo: Error no se puede aprobar solicitud rechazada	Media	Se debe haber rechazada una solicitud y se necesita conocer su Id
Entrada	Resultado Esperado	Resultado Obtenido	Estado
songId= cf8341f4-fe14-4761-be66-f24b517e68d3	Objeto JSON con mensaje "ERROR: This song has been rejected and cannot be approved."	Objeto JSON con mensaje "ERROR: This song has been rejected and cannot be approved."	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## RejectSong

Caso de uso asociado: CU-A-1: Gestionar solicitudes de canciones

Caso: Rechazar solicitud

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#rechazar-solicitud>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-RejectSong-01	Comprobar que existe el flujo: Éxito al rechazar solicitud	Media	Se debe haber registrado la solicitud y no debe haber sido aprobada y se necesita conocer su Id
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
songId= cf8341f4-fe14-4761-be66-f24b517e68d3	Objeto JSON con mensaje "Song rejected successfully"	Objeto JSON con mensaje "Song rejected successfully"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Negar rechazar solicitud aprobada

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#negar-rechazar-solicitud-aprobada>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-RejectSong-02	Comprobar que existe el flujo: Error no se puede rechazar una solicitud porque ya esta aprobada.	Media	Se debe haber aprobado la solicitud y no debe haber sido aprobada y se necesita conocer su Id
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
songId= f078f4ea-7663-4c78-9141-db5df4d8997a	Objeto JSON con mensaje "ERROR: This song has been approved and cannot be rejected."	Objeto JSON con mensaje "ERROR: This song has been approved and cannot be rejected."	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos GET

### SearchGenre

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Consultar géneros

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-generos-1>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ShowGenre-01	Comprobar que existe el flujo: Buscar generos	Media	Deben existir registros de la entidad
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Genre found" y lista de entidades consultadas	Objeto JSON con mensaje "Genre found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

### SearchAlbum

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Consultar álbumes

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-albumes>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ShowAlbum-01	Comprobar que existe el flujo: Buscar Albumes	Media	Deben existir registros de la entidad
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	



Caso: Consultar álbum específico

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-album-especifico>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ShowAlbum-01	Comprobar que existe el flujo: Buscar Album por nombre	Media	Deben existir registros de la entidad
Entrada	Resultado Esperado	Resultado Obtenido	Estado
name=Album	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## SearchMusic

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Mostrar metadatos de archivos mp3

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-metadatos-de-archivos-mp3>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchMusic-01	Comprobar que existe el flujo: Buscar datos de archivo de canción aprobada	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Ninguno	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Mostrar metadatos de archivo mp3 específico

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-metadatos-de-archivo-mp3-especifico>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchMusic-02	Comprobar que existe el flujo: Buscar datos de archivo de canción aprobada específica	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
name=Prueba	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## SearchSong

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Mostrar datos de canciones

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-datos-de-canciones>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchSong-01	Comprobar que existe el flujo: Buscar datos de canciones aprobadas	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Ninguno	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Mostrar de canción específica

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-datos-de-canci%C3%B3n-especifica>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
---------------------------------------	----------	-----------	----------------

CP-SearchSong-02	Comprobar que existe el flujo: Buscar datos de canción aprobada específica	Media	Deben existir canciones aprobadas
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
name=poem	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## ShowPendingSongs

Caso de uso asociado: CU-U-01 Gestionar lista de solicitudes

Caso: Mostrar solicitudes pendientes

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-solicitudes-pendientes>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- ShowPendingSongs-01	Comprobar que existe el flujo: Buscar datos de canciones pendientes	Media	Deben existir canciones no aprobadas o rechazadas
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## ShowRejectedSongs

Caso de uso asociado: CU-U-01 Gestionar lista de solicitudes

Caso: Mostrar solicitudes rechazadas

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-solicitudes-rechazadas>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
---------------------------------------	----------	-----------	----------------

CP- ShowRejectedSongs-01	Comprobar que existe el flujo: Buscar datos de canciones rechazadas	Media	Deben existir canciones rechazadas
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryGenreSongs

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Mostrar canciones de un genero

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-canciones-de-un-genero>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- QueryGenreSongs-01	Comprobar que existe el flujo: Buscar canciones asociadas a un genero	Media	Deben existir canciones aprobadas y se debe conocer la id del genero a consultar por
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
genreId=5	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryAlbumSongs

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Mostrar canciones de un genero

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-datos-de-canciones-de-un-album>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
---------------------------------------	----------	-----------	----------------

CP- QueryAlbumSongs-01	Comprobar que existe el flujo: Busca canciones asociadas a un album	Media	Deben existir canciones aprobadas y se debe conocer la id del album a consultar por
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
albumId=f0a8f3a8-8952-46c4-911f-6a7ddb3dccc6	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "No song was found"	Falló
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryArtistSongs

Caso de uso asociado: CU-U-07 Buscar canción

Caso: Mostrar canciones de un artista

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-P%C3%BAblica#mostrar-datos-de-canciones-de-un-artista>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Pública
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- QueryArtistSongs-01	Comprobar que existe el flujo: Buscar canciones asociadas a un artista	Media	Deben existir canciones aprobadas y se debe conocer la id del artista a consultar por
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
artistId=88f96978-6f13-483c-8a46-10b3e4d52962	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

### 5.3.3. Resultados de Pruebas del Servicio de Biblioteca Privada del API

En esta sección se muestran los resultados de las pruebas basadas en caso de uso del servicio de biblioteca privada.

## Métodos POST

### UploadGenre

Caso de uso asociado: CU-U-3: Agregar canción personal

Caso: Registrar genero

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#registrar-genero>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadGenre-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Genero Prueba" '	Objeto JSON con mensaje "registered successfully"	Objeto JSON con mensaje "registered successfully"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## UploadAlbum

Caso de uso asociado: CU-U-3: Agregar canción personal

Caso: Registrar album

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#registrar-album>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadAlbum-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Prueba Album", "ImageAddress": "pruebaAlbum/imagenPrueba", "ImageName": "imagenPrueba", "ImageType": ".jpg" '	Objeto JSON con mensaje "registered successfully"	Objeto JSON con mensaje "registered successfully"	Pasó

}			
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## UploadMusic

Caso de uso asociado: CU-U-3: Agregar canción personal

Caso: Registrar metadato de archivo

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#registrar-metadato-de-archivo-mp3>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadMusic-01	Comprobar que existe el flujo: Éxito al guardar	Media	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Name": "Prueba Album", "ImageAddress": "pruebaAlbum/imagenPrueba", "ImageName": "imagenPrueba", "ImageType": ".jpg" }'	Objeto JSON con mensaje " registered successfully"	Objeto JSON con mensaje " registered successfully"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## UploadSong

Caso de uso asociado: CU-U-3: Agregar canción personal

Caso: Registrar canción

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#registrar-cancion>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadSongRequest-01	Comprobar que existe el flujo: Éxito al guardar	Media	Se debe haber registrado la canción y se necesita saber su Id

Entrada	Resultado Esperado	Resultado Obtenido	Estado
--data-raw '{ "Id": "", "Title": "The Poem For Everyone\'s Souls", "UploaderId": "3f835f14-8111-4a1d-8385- 2871743290e6", "ArtistId": "88f96978- 6f13-483c-8a46-10b3e4d52962", "MultimediaId": "437f223f-1f82-4a1c-b3b7- 04bdc4188e0a", "Composer": "Shigeru Meguro", "Producer": "Atlus", "Duration": "420", "ReleaseYear": "2009", "AlbumId": "fccdffbd- cca1-4b9b-8882-5067d8da2060", "GenreId": "5", "StatusId": "" }'	Objeto JSON con mensaje "Song registered successfully"	Objeto JSON con mensaje "Song registered successfully"	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos PUT

### RequestSong

Caso de uso asociado: CU-U-6: Publicar canción

Caso: Registrar solicitud de canción

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#registrar-solicitud-en-local>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-UploadSongRequest-01	Comprobar que existe el flujo: Marcar como solicitado en biblioteca privada una canción	Media	Se debe haber registrado la canción y se necesita saber su Id
Entrada	Resultado Esperado	Resultado Obtenido	Estado
songId: "437f223f-1f82-4a1c- b3b7-04bdc4188e0a",	Objeto JSON con mensaje "Song request registered successfully"	Objeto JSON con mensaje "Song not found"	Falló
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	



## Métodos GET

### SearchGenre

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Consultar géneros

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-generos>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ShowGenre-01	Comprobar que existe el flujo: Buscar generos	Media	Deben existir registros de la entidad
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Genre found" y lista de entidades consultadas	Objeto JSON con mensaje "Genre found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

### SearchAlbum

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Consultar álbumes

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-albumes>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-ShowAlbum-01	Comprobar que existe el flujo: Buscar Albumes	Media	Deben existir registros de la entidad
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
Ninguno	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Objeto JSON con mensaje "Album found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## SearchMusic

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Mostrar metadatos de archivos mp3

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-metadatos-de-archivos-mp3>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchMusic-01	Comprobar que existe el flujo: Buscar datos de archivo de canción aprobada	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Ninguno	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Objeto JSON con mensaje "Music found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## SearchSong

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Mostrar datos de canciones

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-datos-de-canciones>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchSong-01	Comprobar que existe el flujo: Buscar datos de canciones aprobadas	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
Ninguno	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Mostrar de canción específica

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-datos-de-canci%C3%B3n-especifica>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-SearchSong-02	Comprobar que existe el flujo: Buscar datos de canción aprobada específica	Media	Deben existir canciones aprobadas
Entrada	Resultado Esperado	Resultado Obtenido	Estado
name=poem	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Objeto JSON con mensaje "Song found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryGenreSongs

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Mostrar canciones de un genero

Enlace a CURL: <https://github.com/BaezCrdrumUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-canciones-de-un-genero>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- QueryGenreSongs-01	Comprobar que existe el flujo: Buscar canciones asociadas a un genero	Media	Deben existir canciones aprobadas y se debe conocer la id del genero a consultar por
Entrada	Resultado Esperado	Resultado Obtenido	Estado
genreId=5	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryAlbumSongs

Caso de uso asociado: CU-U-5: Gestionar biblioteca privada

Caso: Mostrar canciones de un genero

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-datos-de-canciones-de-un-album>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- QueryAlbumSongs-01	Comprobar que existe el flujo: Busca canciones asociadas a un album	Media	Deben existir canciones aprobadas y se debe conocer la id del album a consultar por
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
albumId=f0a8f3a8-8952-46c4-911f-6a7ddb3dccc6	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "No song was found"	Falló
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## QueryArtistSongs

Caso de uso asociado: CU-U-05: Gestionar biblioteca privada

Caso: Mostrar canciones de un artista

Enlace a CURL: <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Biblioteca-Privada#mostrar-datos-de-canciones-de-un-artista>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Biblioteca Privada
-------------------------	--

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP- QueryArtistSongs-01	Comprobar que existe el flujo: Buscar canciones asociadas a un artista	Media	Deben existir canciones aprobadas y se debe conocer la id del artista a consultar por
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
artistId=88f96978-6f13-483c-8a46-10b3e4d52962	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Objeto JSON con mensaje "Songs found" y lista de entidades consultadas	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

### 5.3.4. Resultados de Pruebas del Servicio de Streaming del API

En esta sección se muestran los resultados de las pruebas basadas en caso de uso del servicio de biblioteca privada.

#### Métodos POST

##### Save

Caso de uso asociado: CU-U-8: Consultar canción

Caso: Guardar archivo mp3

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#guardar-archivo-mp3>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-Save-01	Comprobar que existe el flujo: Éxito al guardar mp3	Alta	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
album=albumPrueba --form 'archivo=@"C:/Users/parke/Downloads/look.mp3"'	Objeto JSON con mensaje "Se guardo el archivo en directorio "	Objeto JSON con mensaje "Se guardo el archivo en directorio "	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Guardar archivo jpg

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#guardar-archivo-jpg>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-Save-02	Comprobar que existe el flujo: Éxito al guardar jpg	Alta	Ninguna

Entrada	Resultado Esperado	Resultado Obtenido	Estado
album=albumPrueba --form 'archivo=@"/C:/Users/parke/Pictures/gulag.meme.jpg"'	Objeto JSON con mensaje "Se guardo el archivo en directorio "	Objeto JSON con mensaje "Se guardo el archivo en directorio "	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Guardar sin archivo

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#guardar-sin-archivo>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-Save-03	Comprobar que exista un flujo de error de guardar sin archivo	Baja	Ninguna
Entrada	Resultado Esperado	Resultado Obtenido	Estado
album=albumPrueba	Objeto JSON con mensaje "ERROR: No se encuentra un archivo "	Objeto JSON con mensaje de excepción	Falló
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## Métodos GET

### Streaming

Caso de uso asociado: CU-U-8: Consultar canción

Caso: Transmitir archivo valido

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#transmitir-archivo-valido>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-Streaming-01	Comprobar que existe el flujo: Transmitir mp3	Alta	Debe haberse registrado el archivo previamente

Entrada	Resultado Esperado	Resultado Obtenido	Estado
address=albumPrueba/look.mp3	Objeto JSON con arreglo de bytes de mp3	Objeto JSON con arreglo de bytes de mp3	Pasó
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Mostrar imagen no existente

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#transmitir-archivo-no-existente>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-Streaming-02	Comprobar que existe el flujo: Error no se encontró el mp3	Alta	No debe haberse registrado el archivo previamente
Entrada	Resultado Esperado	Resultado Obtenido	Estado
address=albumPrueba/aaa.mp3	Objeto JSON con mensaje "No se pudo reproducir la canción"	Nunca se retorna un Objeto JSON	Falló
Diseño	Aplicó	Fecha	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## View

El método VIEW es acezado directamente del microservicio y no pasa por la API la petición

Caso de uso asociado: CU-U-8: Consultar canción

Caso: Mostrar imagen valida

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#mostrar-imagen-valida>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-View-01	Comprobar que existe el flujo:	Alta	Debe haberse registrado la imagen previamente

	Transmitir imagen de album		
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
path=albumPrueba/gulag.meme&type=.jpg	Objeto JSON con arreglo de bytes de la imagen	Objeto JSON con arreglo de bytes de la imagen	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

Caso: Mostrar imagen no existente

Enlace a CURL: <https://github.com/BaezCrdmUV/proyecto-equipob/wiki/Pruebas-de-Servicio-de-Streaming#ver-imagen-no-existente>

Caso de Prueba:

<b>Sistema:</b> SpotyMe	<b>Unidad:</b> Microservicio de Streaming
-------------------------	---

Identificador Único CASO DE PRUEBA	Objetivo	Prioridad	Precondiciones
CP-View-02	Comprobar que existe el flujo: Error no se encontró la imagen	Alta	No debe haberse registrado la imagen previamente
<b>Entrada</b>	<b>Resultado Esperado</b>	<b>Resultado Obtenido</b>	<b>Estado</b>
path=imagenPrueba&type=.jpg	Objeto JSON con mensaje "No se pudo encontrar la imagen en ruta/node/app/spotyme/imagenPrueba.jpg"	Objeto JSON con mensaje "No se pudo encontrar la imagen en ruta/node/app/spotyme/imagenPrueba.jpg"	Pasó
<b>Diseño</b>	<b>Aplicó</b>	<b>Fecha</b>	
Ricardo Moguel Sánchez	Ricardo Moguel Sánchez	16/06/2021	

## 5.4 Conclusiones de Pruebas

Después de ejecutar el plan de pruebas se identificaron defectos en la API. Existen ciertas excepciones que no son manejadas. En particular cuando los body de peticiones POST están vacíos, se arroja una excepción que es propagada al cliente en vez de un mensaje indicando que ocurrió un error.

Adicionalmente las siguientes peticiones HTTP tienen defectos:

falla la petición PUT Ban Account en la API

Al intentar transmitir una canción no encontrada nunca se retorna una respuesta indicando que no se logro encontrar un archivo mp3 a transmitir.

No se encuentra canciones por álbum en el método QueryAlbumSongs de la biblioteca privada y biblioteca publica



## 6. Estrategia de despliegue

En esta sección se detalla la estrategia de despliegue para el sistema SpotyMe en dos partes: el Servidor y los Clientes

### 6.1. Despliegue de Servidor

Para desplegar el servidor se utilizan imágenes del API y los diferentes Microservicios y bases de datos. El Servidor se levanta con el uso de un archivo YAML que construye los contenedores de la API, Microservicios y bases de datos. Existe un contenedor para la API. Adicionalmente un contenedor para cada Microservicio y cada base de datos. Los contenedores de los microservicios se conectan por protocolo TCP a los contenedores de la base de datos de Cuenta, Biblioteca Privada y Biblioteca por los puertos que exponen.

El contenedor `db_Accounts` contiene una base de datos relacional de MySQL expuesta en el puerto 3310 en donde se guardan las cuentas de usuarios para gestionar acceso al sistema SpotyMe. El contenedor `db_public_library` contiene una base de datos de documentos en mongoDB expuesta en el puerto 3311 donde se guardan los metadatos de canciones disponibles al público. y el contenedor `db_private_library` contiene una base de datos de documentos en mongoDB expuesta en el puerto 3312 que contiene metadatos de canciones personales de los usuarios. El contenedor `Cuenta` encapsula el Microservicio de Cuentas expuesto en el puerto 8080 y contiene una variable de entorno que guarda la conexión al contenedor `db_Accounts`. El contenedor `Public` encapsula el Microservicio de Biblioteca Pública expuesto en el puerto 8081 y contiene una variable de entorno que guarda la conexión al contenedor `db_public_library`. El contenedor `Private` encapsula el Microservicio de Biblioteca Privada expuesto en el puerto 8082 y contiene una variable de entorno que guarda la conexión al contenedor `db_private_library`. El contenedor `Streaming` encapsula el Microservicio de Streaming expuesto en el puerto 8083 y no utiliza variables de entorno ya que persiste canciones e imágenes en el sistema de archivos del contenedor.

Por último, el contenedor `main_api_spotyme` encapsula la API expuesta en el puerto 4000 y se comunica con los contenedores de los demás microservicios por medio de las 4 variables de entorno que contiene. Estas variables de entorno guardan la URL a los 4 microservicios conectados. El contenedor de la API se conecta por el protocolo HTTP a los contenedores de los Microservicios para consumir recursos persistidos en los contenedores de las bases de datos. Los archivos de canciones e imágenes son persistidos dentro del contenedor del Microservicio de Streaming y el servicio de transmisión de imágenes que contiene es consumido directamente de los clientes sin pasar por la API mientras el servicio de transmisión de música si pasa por la API. El contenedor de la API escucha por peticiones HTTP por parte de clientes en el puerto 4000. Al recibir una petición, crea un objeto JSON basado en la petición recibida y conforme la ruta de la petición, la API lo dirige al microservicio adecuado. El Microservicio realiza operaciones de gestión de datos incluyendo creación, actualización y consulta de datos. El microservicio retorna una respuesta en objeto JSON a la API. Al recibir una

respuesta de un Microservicio, la API retorna una respuesta HTTP con código de 200 en caso de éxito y 400 en caso de error al cliente que solicito un recurso o servicio.

## **6.2. Despliegue de Clientes**

El sistema SpotyMe brinda servicios a tres diferentes clientes ricos que se conectan al Servidor desplegado. Estos clientes son: una aplicación web, una aplicación de escritorio, y una aplicación móvil para Android. A través de las aplicaciones primero se registra una cuenta y se inicia sesión para poder accezar los servicios de gestión de canciones del sistema.

La aplicación web creada con el modelo Razor Pages de ASP.NET se corre desde un ejecutable en una computadora de escritorio. Después, desde un navegador web se ingresa a la URL expuesta por el Servidor para acceso la página principal de inicio de sesión. Desde esta página se puede consumir recursos y servicios expuestos por la API por medio de peticiones HTTP al servidor.

La aplicación de escritorio es una adaptación del cliente web ajustada con el uso de Electron.NET una extensión del framework Electron para correr la aplicación en la plataforma Windows 10 por medio de un programa ejecutable. Una vez corriendo el ejecutable inicia la aplicación con las mismas páginas que el cliente web. La aplicación de escritorio consume recursos de la API de SpotyMe en el Servidor por medio de peticiones HTTP Post, PUT y GET. La aplicación guarda la conexión a la API en una variable de entorno que dirige al puerto 4000 en donde la API escucha por peticiones HTTP.

Finalmente, la aplicación móvil codificada en Kotlin para el sistema operativo móvil Android 10 consume recursos del Servidor a través del protocolo HTTP al estar conectado a la misma red inalámbrica de Wifi que el Servidor. Una vez ejecutada la aplicación a través de la red se conecta al servidor para consumir recursos. La aplicación móvil solo tiene acceso al servicio de cuentas para registrar un usuario e iniciar sesión para comprobar un registro exitoso.

## **7. Conclusiones**

Después de la creación de un API basado en microservicios y 3 diferentes clientes ricos se lograron aprender mucho sobre las tecnologías y prácticas de DevOps. Con el uso de archivos YAML se implementó la integración continua ya que después de compilar con el comando Docker-compose del proyecto se construían los contenedores de Docker para los servicios. Se lograba también el despliegue continuo ya que los contenedores exponían puertos para consumir recursos y desde el contenedor del API se lograba probar cambios en la funcionalidad de los microservicios. El contenedor con la API consumía recursos de otros contenedores en donde estaba los microservicios y bases de datos. Con el uso de frameworks como Electron y Razor Pages se logró desarrollar

para web utilizando C# un lenguaje de programación con el que ya teníamos experiencia. Adicionalmente aprendimos a utilizar JavaScript para codificar la API y HTML y CSS para la aplicación web y Kotlin para programar en Android.

## **8. Referencias**

Wiki de las pruebas de la API <https://github.com/BaezCrdrmUV/proyecto-equipob/wiki>

Repositorio del back-end <https://github.com/BaezCrdrmUV/proyecto-equipob>

Repositorio del front-end <https://github.com/Pklove3459/ClienteWebSpotyMe>