



# UNIVERSIDAD DE GRANADA

DISEÑO Y DESARROLLO DE SISTEMAS DE INFORMACIÓN

## **Diseño de un sistema de información para la gestión y reproducción de música.**

*Entrega final.*

*Darío Abad Tarifa*

*Juan Francisco Díaz Moreno*

*Pedro Domínguez López*

*Javier Sáez de la Coba*

Curso 2018-2019

17 de enero de 2019

# Índice

|   |           |
|---|-----------|
| <b>1. Descripción del problema a resolver.</b>  | <b>2</b>  |
| <b>2. Análisis de requisitos.</b>               | <b>2</b>  |
| 2.1. Requisitos de datos . . . . .              | 2         |
| 2.2. Requisitos funcionales . . . . .           | 8         |
| 2.3. Validación cruzada de requisitos . . . . . | 11        |
| <b>3. Diseño.</b>                               | <b>13</b> |
| 3.1. Esquema de caja negra. . . . .             | 13        |
| 3.2. Esquema almacén DFD0. . . . .              | 13        |
| 3.3. Esquemas externos del DFD0. . . . .        | 14        |
| 3.4. Esquema almacén D (ER0). . . . .           | 14        |
| 3.5. Refinamientos. . . . .                     | 14        |
| 3.5.1. Módulo social. . . . .                   | 14        |
| 3.5.2. Módulo escuchar canciones. . . . .       | 16        |
| 3.5.3. Unión de módulos DFD1. . . . .           | 20        |
| 3.5.4. Unión de módulos DFD2. . . . .           | 21        |
| 3.6. Esquema entidad relación. . . . .          | 22        |
| 3.7. Paso a tablas. . . . .                     | 22        |
| <b>4. Implementación.</b>                       | <b>23</b> |
| 4.1. Diseño físico de la base de datos. . . . . | 23        |
| 4.1.1. Sentencias SQL. . . . .                  | 24        |
| 4.2. Implementación modulo social. . . . .      | 27        |
| 4.2.1. Implementación del disparador. . . . .   | 27        |
| 4.2.2. Implementación de un proceso. . . . .    | 28        |

# **1. Descripción del problema a resolver.**

Una empresa de streaming de audio quiere rehacer su plataforma de gestión de música (todo el servicio excepto el propio streaming de música). Para ello requiere que puedan haber cuentas de oyentes y de artistas.

El sistema ofrece un servicio de musica en streaming, es decir permite que un usuario pueda buscar una cancion, un album o una lista de musica en la base de datos para posteriormente escucharla.

Las listas funcionan como si de un directorio se tratase, el usuario las gestiona como quiere y en ellas guarda sus canciones favoritas o canciones segun un tema especifico.

El sistema ofrece ademas diferentes opciones de carácter social para facilitar la relación entre amigos y los artistas con sus oyentes. Un usuario puede seguir a otro usuario (sea artista o no) para estar al corriente de lo que escucha proporcionando el nombre del usuario al que quiere seguir. Un usuario podrá recomendar una canción a otro usuario proporcionando el nombre de este y el identificador de la canción que quiere recomendar. Un usuario podrá solicitar un resumen sobre las canciones que escucha su lista de amigos. Un usuario podrá ver las recomendaciones que le llegan de sus amigos. Además, los usuarios oyentes pueden valorar las canciones que escuchan. Por último podrán visualizar las canciones mejor valoradas por un usuario proporcionando su nombre de usuario.P

## **2. Análisis de requisitos.**

### **2.1. Requisitos de datos**

RD1. Identificador de la canción, álbum o lista para escuchar:

- Elección (canción, álbum o lista)
- Identificador

RD2. Datos de álbum almacenado:

- Nombre del álbum
- Nombre del artista
- Fecha de introducción
- Identificador del álbum
- Nombre del álbum
- Nombre del artista

- Fecha de introducción
- Número de canciones
- Duración

RD3. Datos de canción almacenada:

- Identificador de la canción
- Nombre de la canción
- Nombre del artista
- Nombre del álbum
- Estilo de la canción
- Duración de la canción
- Fecha de introducción
- Archivo de audio
- Número de reproducciones
- Valoración media

RD4. Datos de lista almacenada:

- Identificador de la lista
- Nombre de la lista
- Canciones que contiene
- Duración de la lista
- Fecha de creacion
- Usuario al que pertenece
- Seguidores

RD5. Identificador de la nueva lista creada:

- Identificador de la lista

RD6. Nombre para la búsqueda de una canción, álbum o lista:

- Nombre

RD7. Lista con los posibles identificadores según el resultado de la búsqueda:

- Identificador
- Tipo: canción, álbum, lista
- Nombre

- Usuario o artista
- Fecha de publicación o de creación
- Duracionr

RD8. Datos para crear una lista nueva:

- Usuario
- Fecha

RD9. Valoración de una canción:

- Identificador de la canción
- Identificador

RD10. Datos nuevo álbum:

- Nombre del álbum
- Nombre del artista
- Fecha de introducción

RD11. Datos nueva canción:

- Nombre de la canción
- Nombre del artista
- Nombre del álbum
- Estilo de la canción
- Duración de la canción
- Fecha de introducción
- Archivo de audio

RD12. Identificador de la canción introducida:

- Identificador de la canción

RD13. Identificador de la canción para ver sus estadísticas:

- Identificador de la canción

RD14. Reproduccion de una lista:

- Audio de las canciones de la lista

RD15. Estadísticas de la canción:

- Nombre de la canción

- Número de reproducciones
- Valoración media

RD16. Identificador de la canción para añadirla a canciones destacadas:

- Identificador de la canción

RD17. Lista de las canciones destacadas actualizada:

- Canciones destacadas

RD18. Identificador de la canción o álbum para eliminarlo:

- Elección (canción o álbum)
- Identificador

RD19. Datos de registro del usuario:

- Nombre de usuario
- Correo electrónico
- Contraseña
- Nombre
- Apellidos
- Tipo de usuario (oyente, artista)
- Dirección

RD20. Datos usuario almacenado:

- Identificador
- Nombre de usuario
- Correo electrónico
- Contraseña
- Nombre
- Apellidos
- Tipo de usuario (oyente, artista)
- Dirección

RD21. Contraseña nueva:

- Identificador de usuario
- Contraseña nueva

RD22. Datos perfil usuario:

- Nombre de usuario
- Correo electrónico
- Contraseña
- Nombre
- Apellidos
- Dirección

RD23. Identificador del usuario a eliminar:

- Identificador del usuario

RD24. Datos de un usuario:

- Nombre de usuario

RD25. Datos de un amigo:

- Nombre de usuario
- Canción que está escuchando

RD26. Datos de artista

- Nombre de usuario
- Nombre artístico
- Número de canciones subidas
- Álbumes subidos

RD27. Recomendación saliente

- Nombre de usuario de salida
- Cuerpo de mensaje
- Nombre de usuario de entrada

RD28. Recomendación entrante

- Nombre de usuario de entrada
- Cuerpo de mensaje
- Nombre de usuario de salida

RD29. Recomendación almacenada.

- Nombre de usuario que recomienda
- Cuerpo de mensaje

- Nombre de usuario que es recomendado

RD30. Datos de canción que está siendo escuchada

- Identificador de canción
- Usuario que la está escuchando

RD31. Datos del usuario actual

- Identificador de usuario

RD32. Datos de canciones mejor valoradas

- Identificadores de canciones
- Identificador de usuario que ha valorado

RD33. Datos de usuario que ha realizado valoraciones

- Nombre de usuario

RD34. Valoración dada por un usuario a una canción

- identificador de canción
- Valoración

RD35. Datos modificar estado de canción en lista.

- Identificador de canción
- Identificador de la lista de reproducción
- Operación a realizar

RD36. Identificador de la lista a eliminar

- Identificador de la lista de reproducción

RD37. Datos canciones que estan siendo escuchadas

- Identificador de la cancion
- Identificador de usuario que escucha la cancion

RD38. Datos recomendaciones almacenadas

- Identificador del usuario que recomienda
- Identificador del usuario que recibe la recomendación
- Identificador de la canción que es recomendada

RD39. Datos canciones mejor valoradas



- Nombre de la canción
- Nombre del artista
- Valoración

## 2.2. Requisitos funcionales

RF1. Escuchar una canción: un usuario introduce el identificador de la canción, álbum o lista que quiere escuchar..

- Entrada: RD1.
- Consulta: RD2., RD3., RD4.
- Salida: RD14.

RF2. Buscar una canción: a través del nombre de una canción, álbum o lista se elige el identificador que se considere el adecuado con los criterios de búsqueda .

- Entrada: RD6.
- Consulta: RD2., RD3., RD4.
- Salida: RD7.

RF3. Crear listas de reproducción: un usuario puede crear una lista donde añade canciones y las tiene todas en un mismo lugar. .

- Entrada: RD8.
- Almacena: RD4.
- Salida: RD5.

RF4. Añadir canciones a una lista de reproducción: un usuario puede añadir o eliminar las canciones de su lista a través del identificador..

- Entrada: RD35.
- Consulta: RD4.
- Actualiza: RD4.

RF5. Borrar lista: un usuario puede eliminar su lista..

- Entrada: RD36.
- Consulta: RD4.
- Actualiza: RD4.

RF6. Valorar canciones: un usuario puede valorar una canción..

- Entrada: RD34.
- Consulta: RD9.
- Actualiza: RD9.

RF7. Crear álbum: un artista registra en el sistema un nuevo álbum..

- Entrada: RD10.
- Almacena: RD2.
- Salida: ninguna

RF8. Crear canción: un artista introduce en el sistema una nueva canción..

- Entrada: RD11.
- Consulta: RD2.
- Actualiza: RD3.
- Salida: RD12.

RF9. Ver estadísticas de canción: un artista consulta las estadísticas de una de sus canciones buscándola con su identificador..

- Entrada: RD13.
- Consulta: RD12.
- Salida: RD15.

RF10. Destacar canciones individuales: un artista selecciona una canción a través de su identificador y la añade a su lista de canciones destacadas..

- Entrada: RD16.
- Consulta: RD3.
- Salida: RD17.

RF11. Borrar canciones: un artista introduce en el sistema si quiere eliminar una canción o un álbum y el identificador de su elección y el sistema elimina los datos asociados al mismo..

- Entrada: RD18.
- Consulta: RD2. RD3.
- Actualiza: RD2. RD3.

RF12. Crear usuarios. El sistema guarda la información de un nuevo usuario .

- Entrada: RD19.
- Almacena: RD20.

RF13. Recuperar contraseña: .

- Entrada: RD21.
- Actualiza: RD20.

RF14. Modificar perfil. El sistema guarda la información modificada por el usuario. .

- Entrada: RD22.
- Almacena: RD20.

RF15. Borrar usuarios. Eliminar a un usuario del sistema..

- Entrada: RD23.
- Actualiza: RD20.

RF16. Seguir usuario. Se crea una relación entre dos usuarios, el seguidor estará al tanto de las acciones del usuario al que sigue.

- Entrada: RD24.
- Almacenado: RD26.

RF17. Recomendar canciones a amigos. Un usuario envía a través de un mensaje una canción a otro usuario.

- Entrada: RD27.
- Almacenado: RD29.

RF18. Ver lo que están escuchando los amigos. Se muestra al usuario una lista con las canciones que están escuchando sus amigos en ese momento.

- Entrada: RD25.
- Consulta: RD37.
- Salida: RD30.

RF19. Ver recomendaciones entrantes. Se muestra al usuario una lista con las recomendaciones que le han hecho sus amigos.

- Entrada: RD31.
- Consulta: RD38.
- Salida: RD28.

RF20. Canciones mejor valoradas por un usuario. Un usuario solicita una lista de las canciones mejor valoradas proporcionando el identificador de otro usuario.

- Entrada: RD33.
- Consulta: RD32.
- Salida: RD39.

### 2.3. Validación cruzada de requisitos

| RF    | Entrada | Manejo          | Salida |
|-------|---------|-----------------|--------|
| RF1.  | RD1.    | RD2.,RD3.,RD4.  | RD14.  |
| RF2.  | RD6.    | RD4.            | RD7.   |
| RF3.  | RD8.    | RD4.            | RD5.   |
| RF4.  | RD5.    | RD4.            |        |
| RF5.  | RD36.   | RD4.            |        |
| RF6.  | RD5.    | RD9.            | RD9.   |
| RF7.  | RD10.   | RD2.            |        |
| RF8.  | RD11.   | RD2.,RD3.       | RD12.  |
| RF9.  | RD13.   | RD12.           | RD15.  |
| RF10. | RD16.   | RD3.,RD10.      | RD17.  |
| RF11. | RD18.   | RD2.,RD3.,RD10. |        |
| RF12. | RD19.   | RD20.           |        |
| RF13. | RD21.   | RD20.           |        |
| RF14. | RD22.   | RD20.           |        |
| RF15. | RD23.   | RD20.           |        |
| RF16. | RD24.   | RD25.,RD26.     |        |
| RF17. | RD27.   | RD29.           |        |
| RF18. | RD25.   | RD37.           | RD30.  |
| RF19. | RD31.   | RD38.           | RD28.  |
| RF20. | RD33.   | RD32.           | RD39.  |

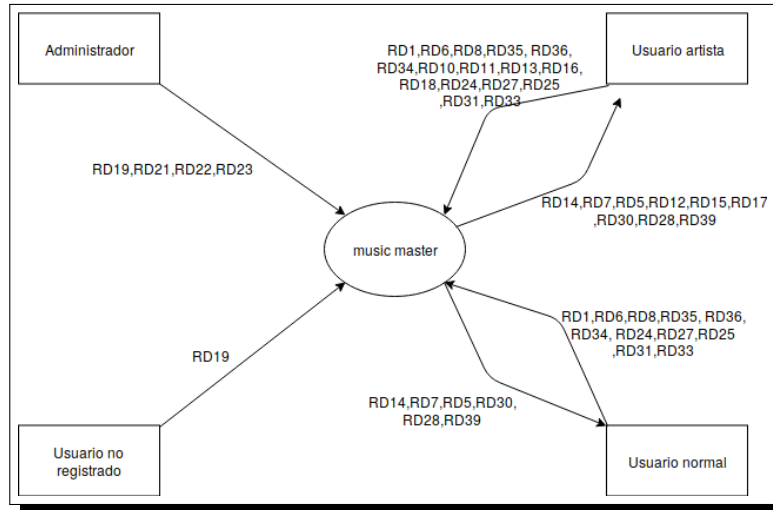
Cuadro 1: Requisitos funcionales

| RD    | Entrada | Manejo                     | Salida |
|-------|---------|----------------------------|--------|
| RD1.  | RF1.    |                            |        |
| RD2.  |         | RF1.,RF2.,RF7.,RF8.,RF11.  |        |
| RD3.  |         | RF1.,RF2.,RF8.,RF10.,RF11. |        |
| RD4.  |         | RF1.,RF2.,RF3.,RF4.,RF5.   |        |
| RD5.  |         |                            | RF3.   |
| RD6.  | RF2.    |                            |        |
| RD7.  |         |                            | RF2.   |
| RD8.  | RF3.    |                            |        |
| RD9.  |         | RF6.                       |        |
| RD10. | RF7.    |                            |        |
| RD11. | RF8.    |                            |        |
| RD12. |         | RF9.                       | RF8.   |
| RD13. | RF9.    |                            |        |
| RD14. |         |                            | RF1.   |
| RD15. |         |                            | RF9.   |
| RD16. | RF10.   |                            |        |
| RD17. |         |                            | RF10.  |
| RD18. | RF11.   |                            |        |
| RD19. | RF12.   |                            |        |
| RD20. |         | RF12.,RF13.,RF14.,RF15.    |        |
| RD21. | RF13.   |                            |        |
| RD22. | RF14.   |                            |        |
| RD23. | RF15.   |                            |        |
| RD24. | RF16.   |                            |        |
| RD25. | RF18.   |                            |        |
| RD26. |         | RF16.                      |        |
| RD27. | RF17.   |                            |        |
| RD28. |         |                            | RF19.  |
| RD29. |         | RF17.                      |        |
| RD30. |         |                            | RF18.  |
| RD31. | RF19.   |                            |        |
| RD32. |         | RF20.                      |        |
| RD33. | RF20.   |                            |        |
| RD34. | RF6.    |                            |        |
| RD35. | RF4.    |                            |        |
| RD36. | RF5.    |                            |        |
| RD37. |         | RF18.                      |        |
| RD38. |         | RF19.                      |        |
| RD39. |         |                            | RF20.  |

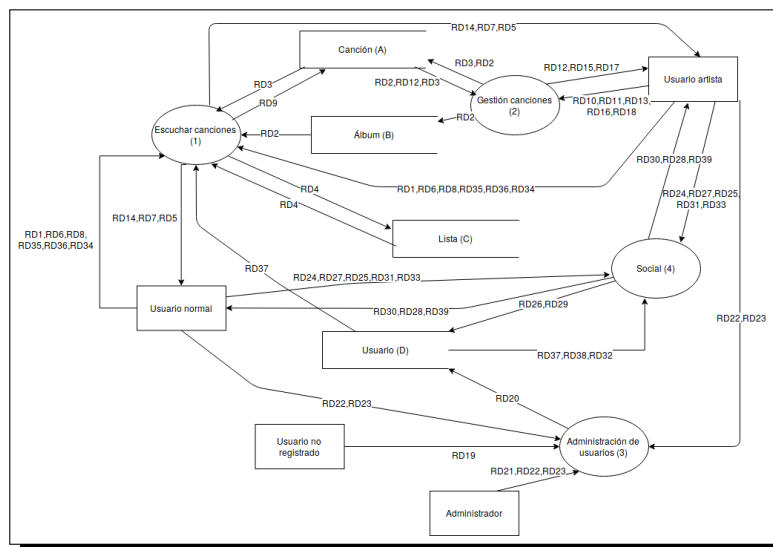
Cuadro 2: Requisitos de datos

### 3. Diseño.

#### 3.1. Esquema de caja negra.



#### 3.2. Esquema armazón DFD0.



### 3.3. Esquemas externos del DFD0.

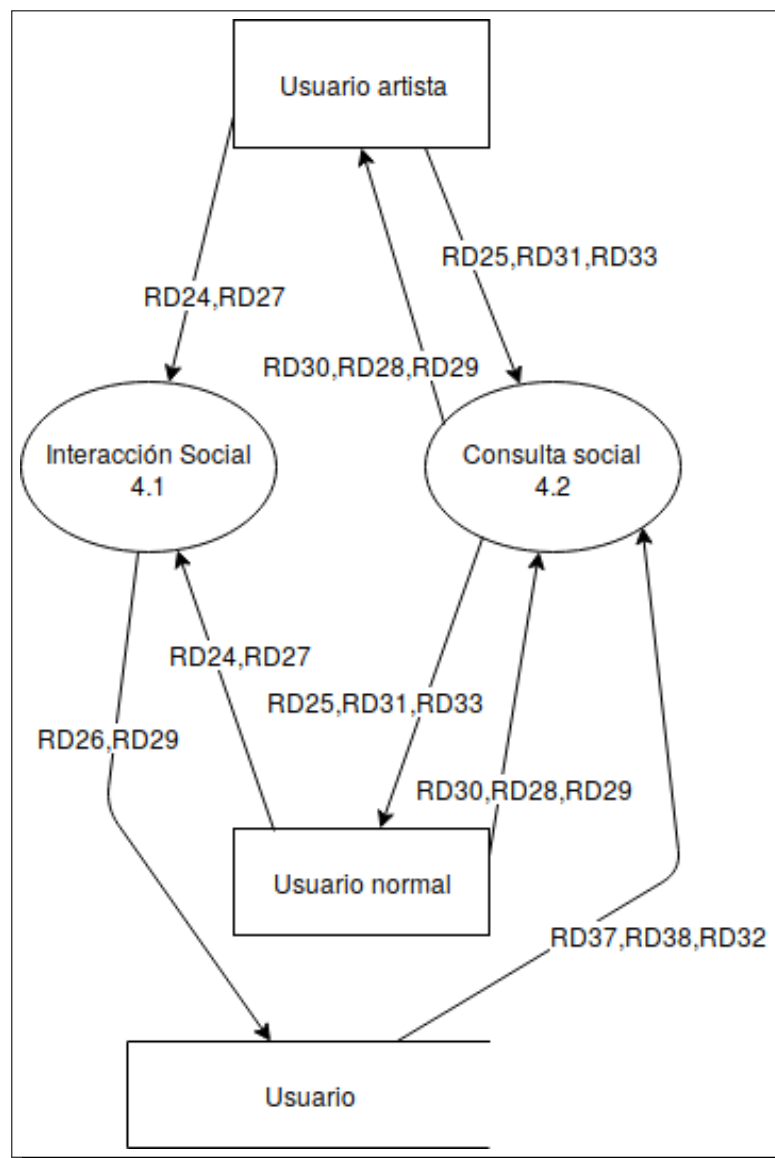
### 3.4. Esquema almacén D (ER0).

Unión de los esquemas externos.

### 3.5. Refinamientos.

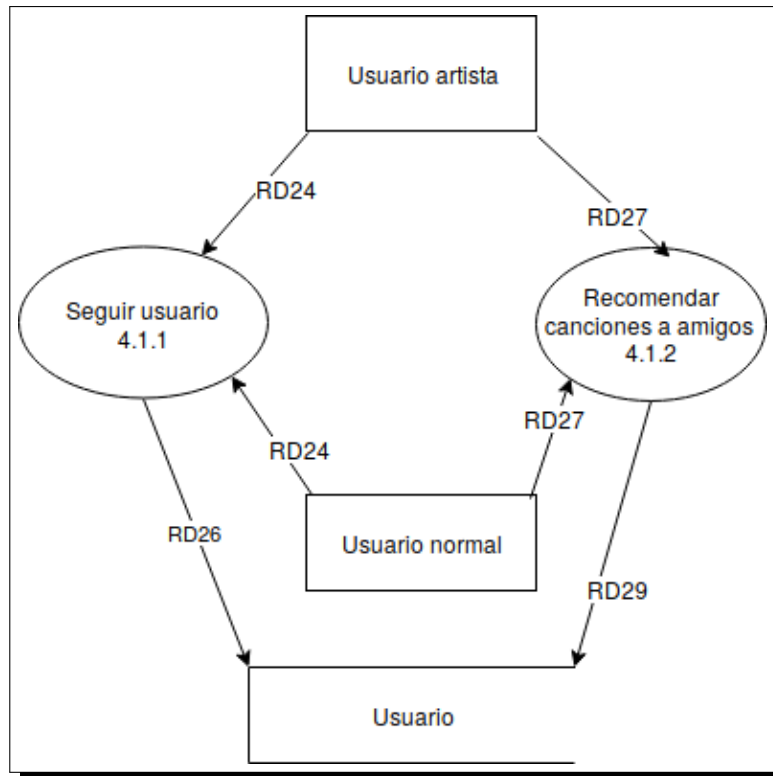
#### 3.5.1. Módulo social.

##### 3.5.1.1. DFD1



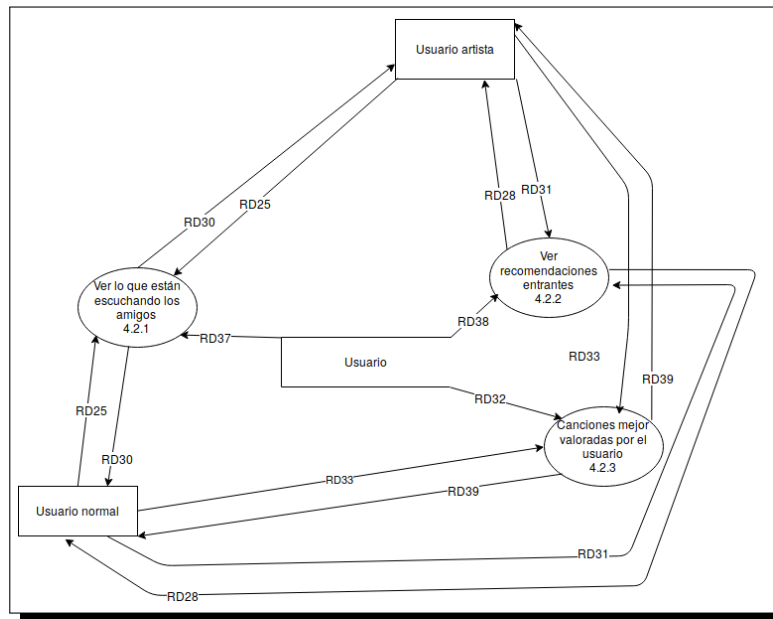
### 3.5.1.2. DFD2

#### 3.5.1.2.1. Primer plano de refinamiento.

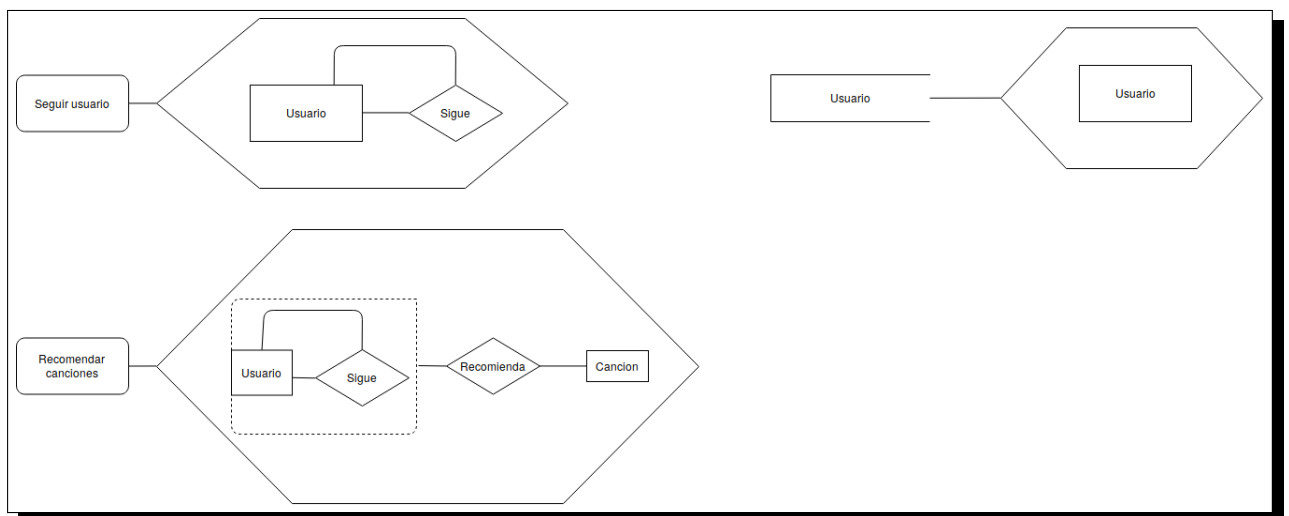




### 3.5.1.2.2. Segundo plano de refinamiento.

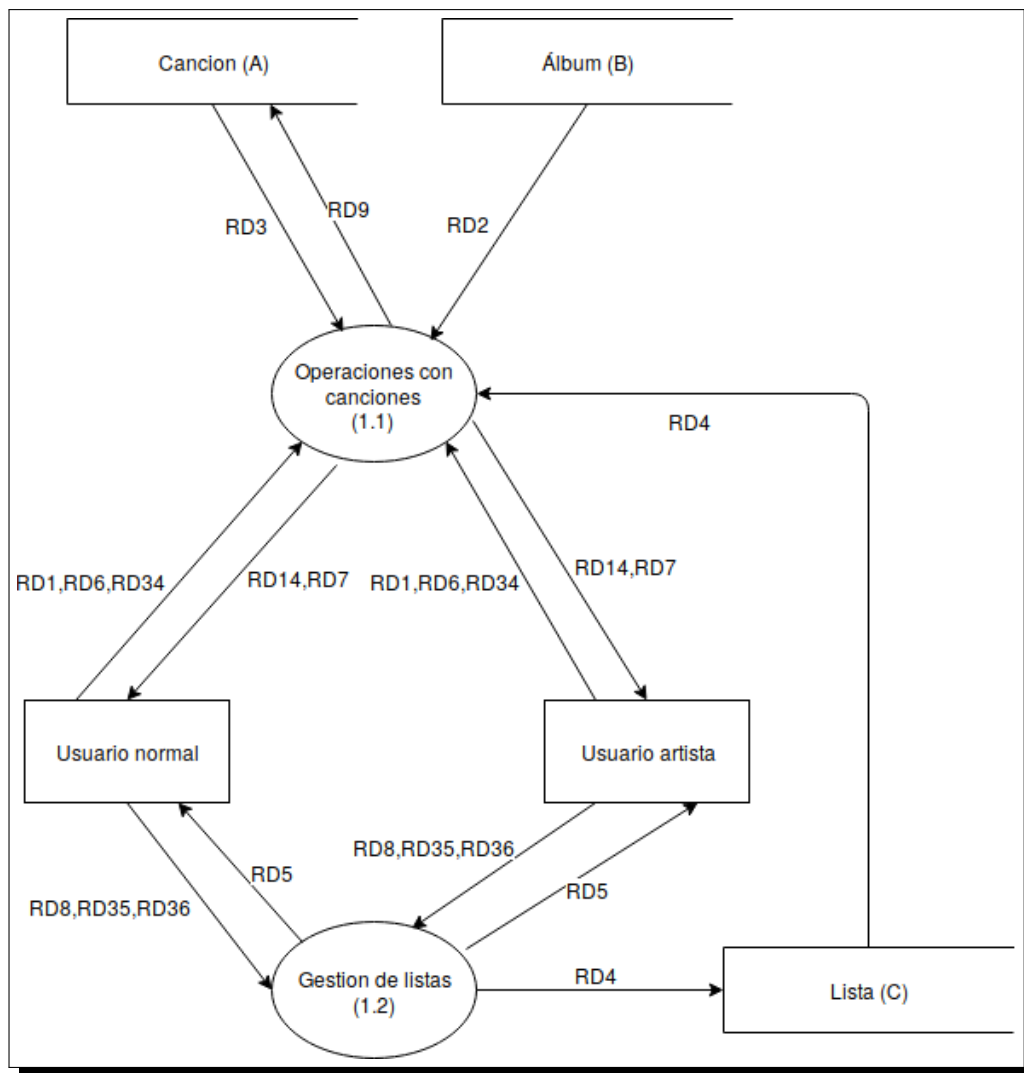


### 3.5.1.3. Esquema externo.



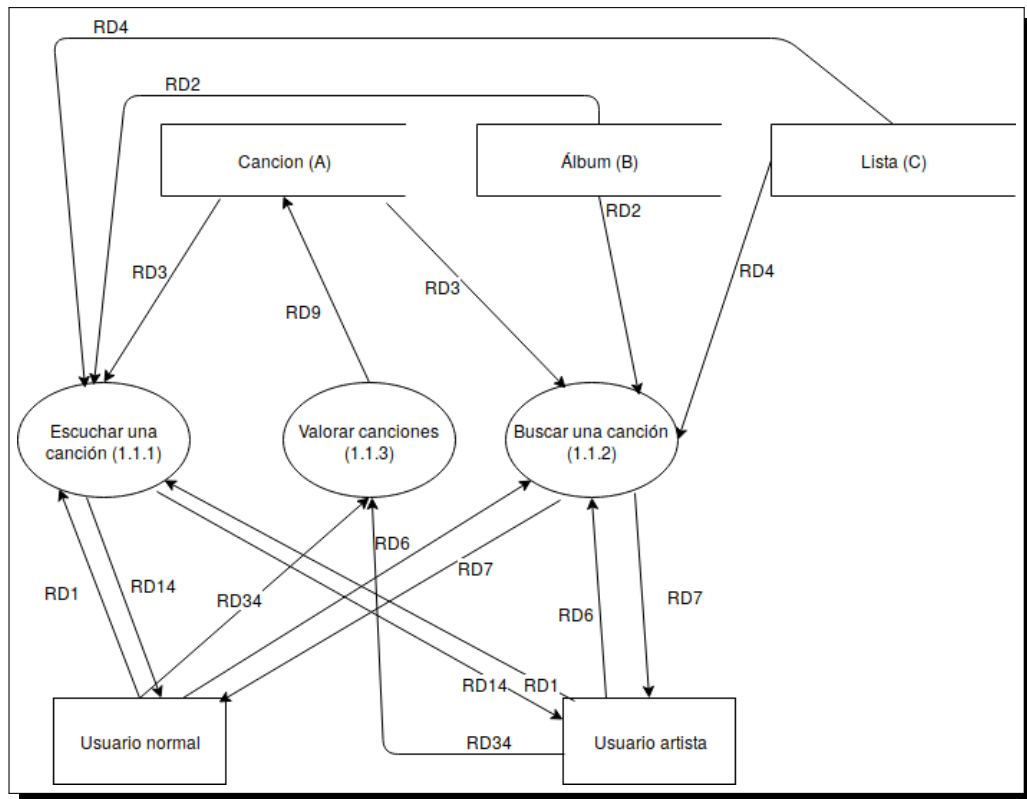
## 3.5.2. Módulo escuchar canciones.

### 3.5.2.1. DFD1

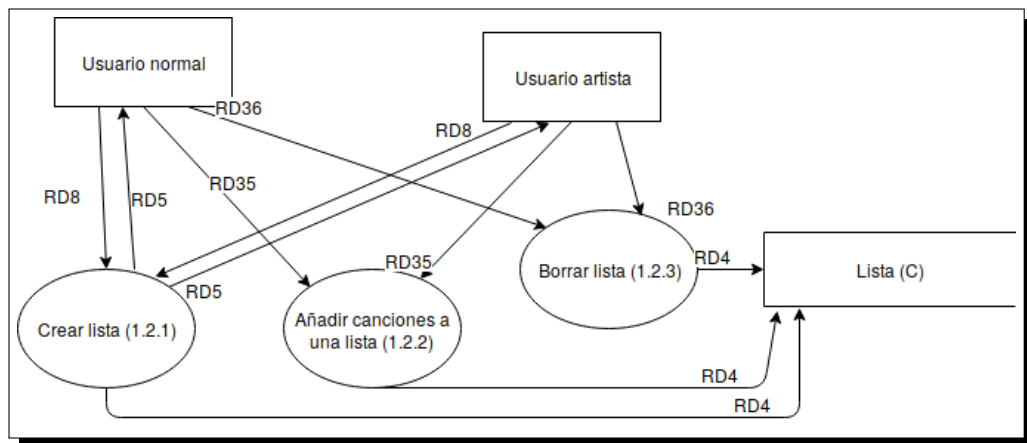


### 3.5.2.2. DFD2

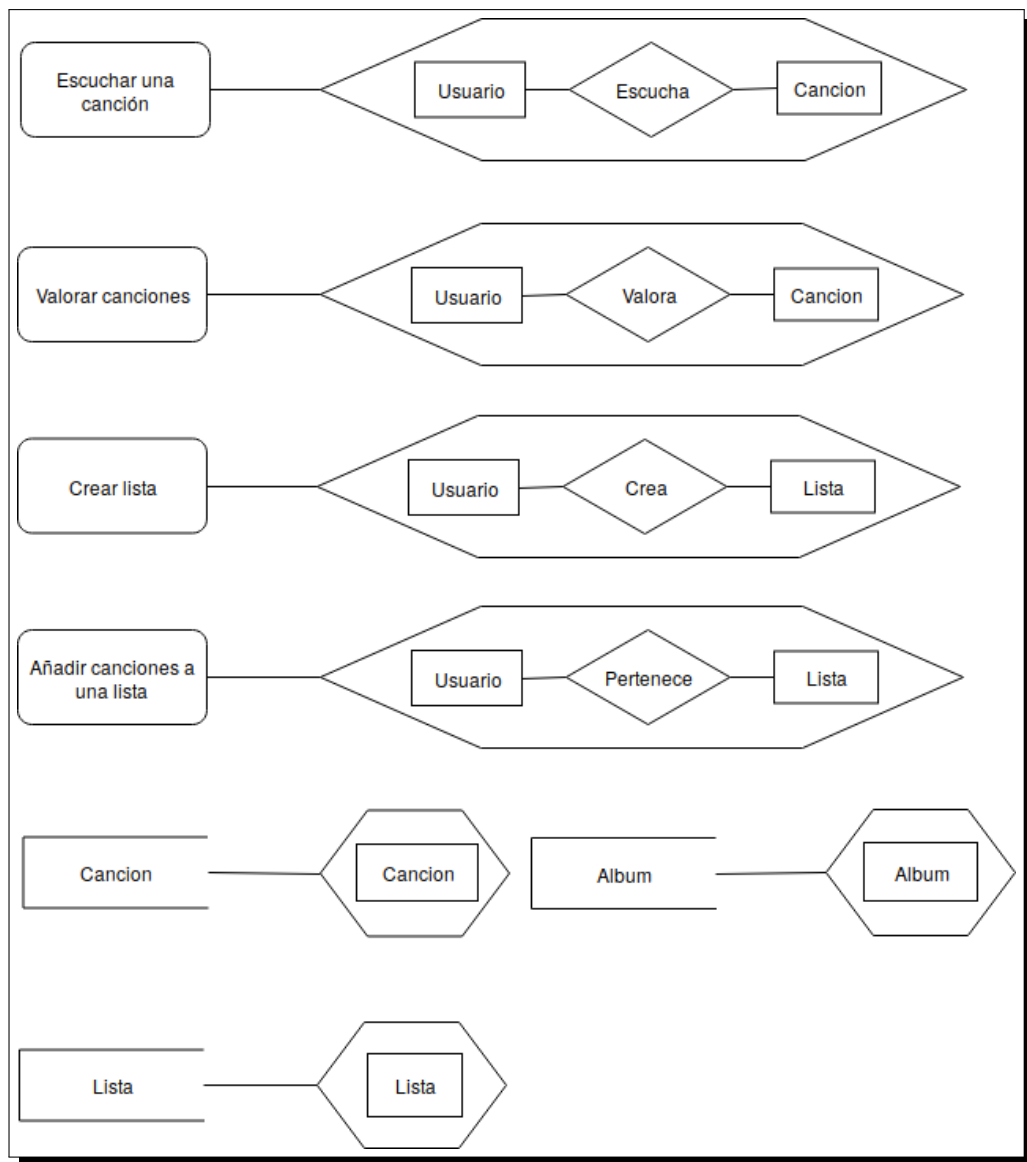
#### 3.5.2.2.1. Primer plano de refinamiento.



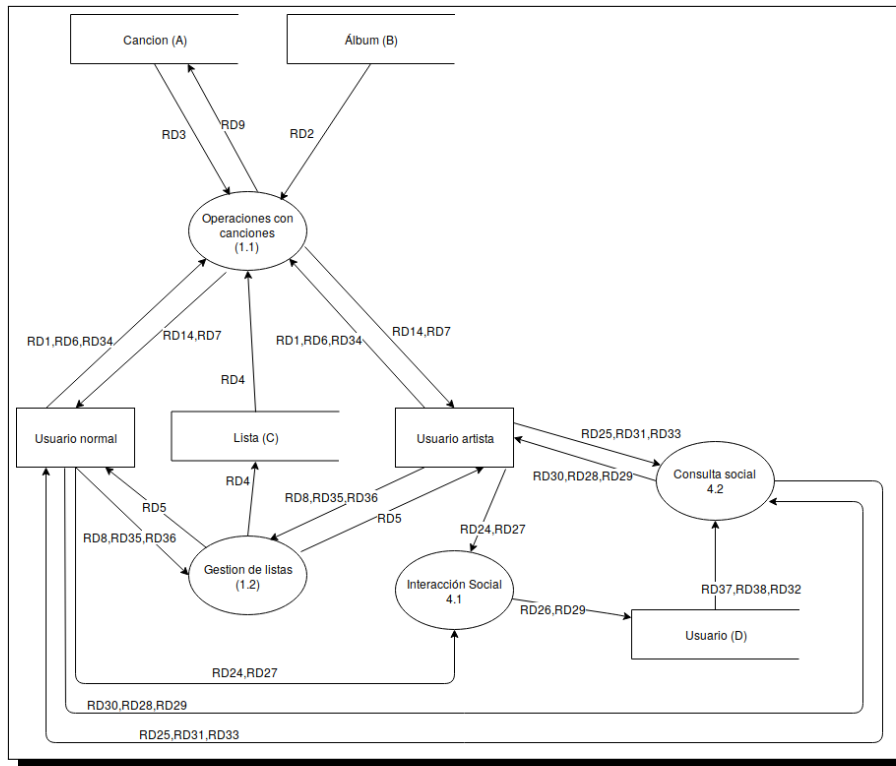
### 3.5.2.2. Segundo plano de refinamiento.



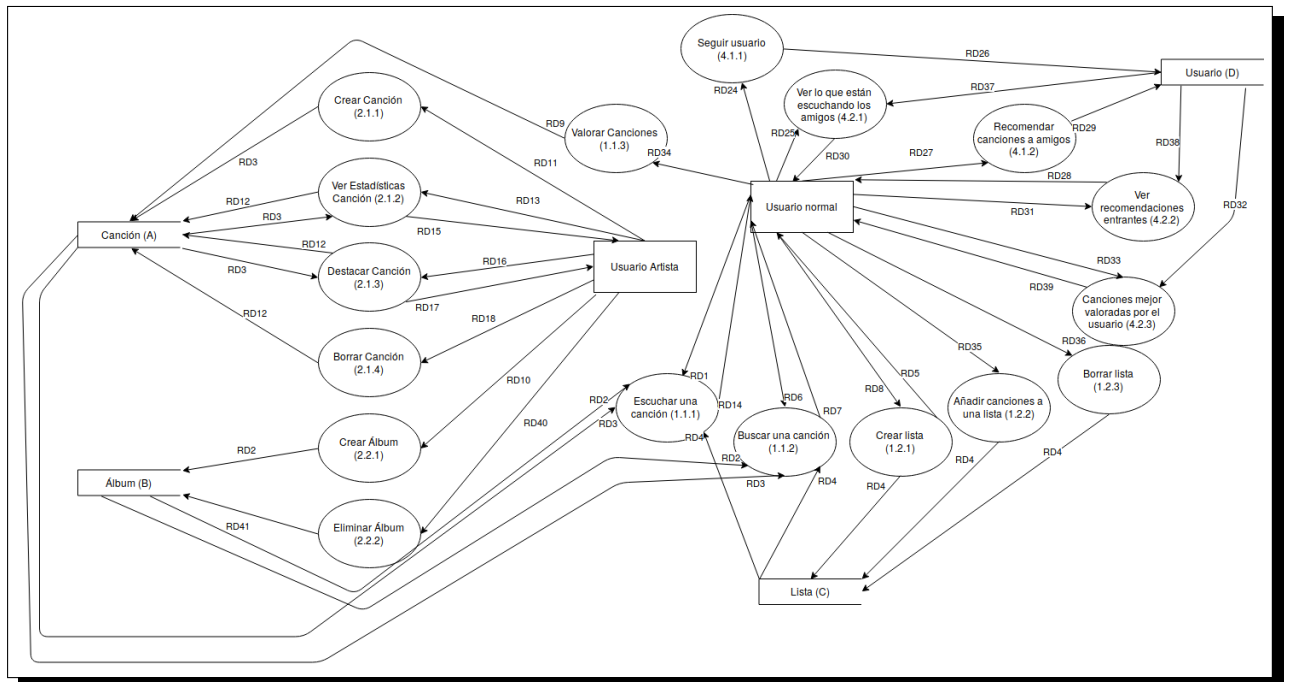
### 3.5.2.3. Esquema externo.



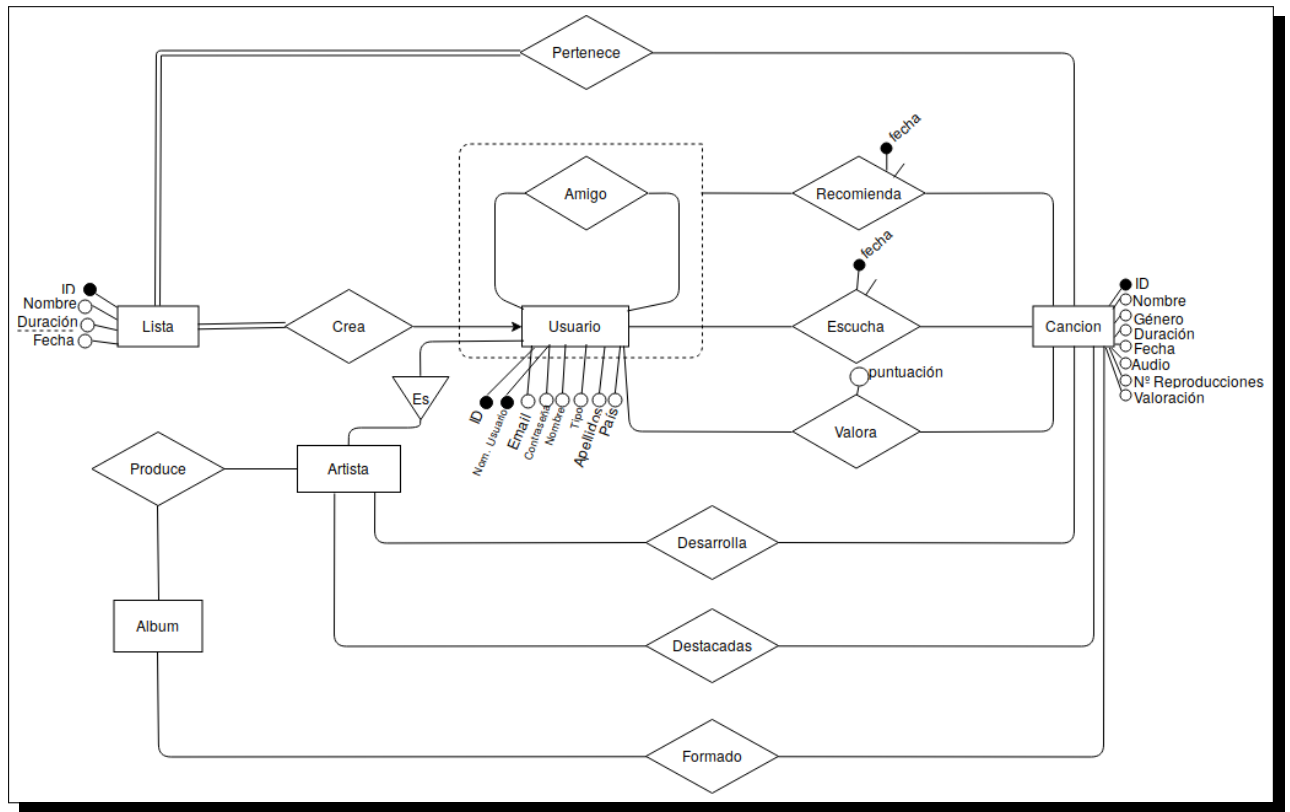
### 3.5.3. Unión de módulos DFD1.



### 3.5.4. Unión de módulos DFD2.



### 3.6. Esquema entidad relación.



### 3.7. Paso a tablas.

Tabla usuario.

Clave primaria: ID usuario.

Resto de atributos: nombre usuario, nombre artistico, email, contraseña, nombre, apellidos, tipo, pais.

Tabla canción.

Clave primaria: ID cancion.

Resto de atributos: nombre, genero, duración, fecha, ruta audio, numero de reproducciones, valoración.

Tabla lista.

Clave primaria: ID lista.

Resto de atributos: nombre, fecha, duración.

Tabla amigo.

Clave primaria: ID usuario1, ID usuario2. Ambas como claves externas a ID usuario de la tabla usuario.

Tabla recomienda.

Clave primaria: ID usuario1, ID usuario2, ID cancion, fecha.

ID usuario1 e ID usuario2 como claves externas a la tabla amigo. ID cancion como clave externa a la tabla cancion.

Tabla escucha.

Clave primaria: ID usuario1, ID usuario2, fecha.

ID usuario1 e ID usuario2 como claves externas a la tabla cancion.

Tabla valora.

Clave primaria: ID usuario, ID cancion.

ID usuario como clave externa a la tabla usuario. ID cancion como clave externa a la tabla cancion.

Resto de atributos: puntuación.

Tabla pertenece.

Clave primaria: ID lista, ID cancion.

ID lista como clave externa a la tabla lista. ID cancion como clave externa a la tabla cancion.

Tabla crea.

Clave primaria: ID lista, ID usuario.

ID lista como clave externa a la tabla lista. ID usuario como clave externa a la tabla usuario.

## **4. Implementación.**

### **4.1. Diseño físico de la base de datos.**



#### 4.1.1. Sentencias SQL.

```
create table usuario(  
    id_usuario char(8) constraint id_usuario_no_nulo not null  
    constraint id_usuario_primaria primary key,  
    nombre_usuario varchar2(15)  
        constraint nombre_usuario_no_nulo not null  
    constraint nombre_usuario_candidata unique,  
    email_u varchar2(40) constraint email_no_nulo not null,  
    contrasenia_u varchar2(40) constraint contrasenia_no_nula not null,  
    nombre_per varchar2(15) constraint nombre_no_nulo not null,  
    apellidos_per varchar2(30),  
    tipo varchar2(7) constraint tipo_correcto  
        check (tipo in ( 'artista ', 'usuario ')),  
    pais varchar2(20),  
    nombre_artistico varchar2(50));  
  
create table cancion(  
    id_cancion char(8) constraint id_cancion_no_nulo not null  
    constraint id_cancion_primaria primary key,  
    nombre_cancion varchar2(50)  
        constraint nombre_cancion_no_nulo not null,  
    genero varchar2(20) constraint genero_no_nulo not null,  
    duracion_seg number(3) constraint duracion_seg_no_nula not null,  
    fecha_c date constraint fecha_cancion_no_nula not null,  
    ruta_audio varchar2(100),  
    num_repro number(10),  
    valoracion number(1,2));  
  
create table lista(  
    id_lista char(8) constraint id_lista_no_nulo not null  
    constraint id_lista_primaria primary key,  
    nombre_lista varchar2(50) constraint nombre_lista_no_nulo not null,  
    fecha_l date constraint fecha_lista_no_nula not null,  
    duracion_seg number(6));
```

```

create table pertenece(
    id_lista ,
    id_cancion ,
    constraint pertenece_clave_primaria primary key (id_lista ,id_cancion)
    constraint id_lista_ext_pertenece foreign key(id_lista)
        references lista(id_lista),
    constraint id_cancion_ext_pertenece foreign key(id_cancion)
        references cancion(id_cancion));

```

```

create table crea(
    id_usuario ,
    id_lista ,
    constraint crea_clave_primaria primary key (id_usuario ,id_lista),
    constraint id_usuario_ext_crea foreign key(id_usuario)
        references usuario(id_usuario),
    constraint id_lista_ext_crea foreign key(id_lista)
        references lista(id_lista));

```

```

CREATE TABLE amigo
(
    id_usuario1 ,
    id_usuario2 ,
    CONSTRAINT clave_primaria PRIMARY KEY (id_usuario1 ,id_usuario2),
    CONSTRAINT id_usuario1_ext_pertenece FOREIGN KEY(id_usuario1)
    REFERENCES usuario(id_usuario),
    CONSTRAINT id_usuario2_ext_pertenece FOREIGN KEY(id_usuario2)
    REFERENCES usuario(id_usuario)
);

```

```

create table recomienda(
    id_usuario1 ,
    id_usuario2 ,
    id_cancion ,
    fecha date not null ,
    constraint clave_pri
        primary key (id_usuario1 ,id_usuario2 ,id_cancion ,fecha) ,
    foreign key (id_usuario1 ,id_usuario2)
        references amigo(id_usuario1 ,id_usuario2) ,
    foreign key (id_cancion) references cancion(id_cancion)
);

```

```
create table valora(  
    id_usuario ,  
    id_cancion ,  
    puntuacion number(1,1) ,  
    primary key (id_usuario ,id_cancion) ,  
    foreign key (id_usuario) references usuario(id_usuario) ,  
    foreign key (id_cancion) references cancion(id_cancion)  
);
```

```
create table escucha(  
    id_usuario ,  
    id_cancion ,  
    fecha date not null ,  
    primary key (id_usuario ,id_cancion ,fecha) ,  
    foreign key (id_usuario) references usuario(id_usuario) ,  
    foreign key (id_cancion) references cancion(id_cancion)  
);
```

## 4.2. Implementación modulo social.

### 4.2.1. Implementación del disparador.

Este disparador controla la inserción de una tupla en la tabla recomendación. Es decir comprueba una condición cuando un usuario recomienda una canción a otro usuario. Dicha condición consiste en que un usuario no puede recomendar una canción si no la ha escuchado previamente.

Básicamente, el funcionamiento consiste en comprobar la tabla escucha (donde reside el registro de reproducciones que han realizado los usuarios) y mirar si el usuario que recomienda ha reproducido la canción que se recomienda canción. Si no encuentra ningún resultado, levanta una excepción.

```
create or replace TRIGGER dario_trigg
BEFORE INSERT ON recomienda
FOR EACH ROW
DECLARE
no_escuchada EXCEPTION;
existe number;
BEGIN
SELECT count(*) INTO existe FROM escucha
WHERE id_cancion = :new.id_cancion and id_usuario = :new.id_usuario1;
    IF (existe = 0) THEN
        RAISE no_escuchada;
    END IF;
EXCEPTION
WHEN no_escuchada THEN
    RAISE_APPLICATION_ERROR(-20000,'No_has_escuchado_la_cancion');
END;
```

#### 4.2.2. Implementación de un proceso.

El proceso acordado para la implementación consiste mostrar que están escuchando mis amigos. Debido a la dificultad de mostrar lo que se está reproduciendo en un instante, se decidió mostrar las reproducciones realizadas “hoy”, es decir, en un día completo.

El proceso ha sido implementado en Python y la interfaz es línea de comandos (CLI), debido a la falta de tiempo y el poco conocimiento a cerca de interfaces gráficas.

La conexión a la base de datos se realiza mediante el paquete “cx\_Oracle”, que nos permite realizar las consultas y manejar los datos consultados.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import os
import cx_Oracle

def menu():
    print ("Opciones disponibles:")
    print ("\t1 – Ver que han escuchado mis amigos hoy.")
    print ("\t0 – Salir")

def mostrar_info_amigo(info_amigo):
    for row in info_amigo:
        print "Usuario: " + row[0]
        print "Nombre: " + row[1] + " " + row[2]
    print "_____ "

#####

conn_str = 'USUARIO/PASSWORD@HOSTNAME:PORT/SERVICENAME'
conn = cx_Oracle.connect(conn_str)

my_id = 'UN000002' # Hipot tico usuario logeado en el sistema
hoy = '17/09/2017' # Hipot cica fecha actual
```

```

c = conn.cursor()

while True:
    menu()

    opcionMenu = str(input("Selecciona una acci n: "))

    if opcionMenu == '1':
        #Listamos los amigos
        c.execute("SELECT id_usuario2 FROM amigo
                    WHERE id_usuario1 = '%s'" % (my_id))
        id_amigos = c.fetchall()

        contador = 1
        info_amigo = []
        #MOSTRAR LISTA DE AMIGOS
        print "_____ "
        for ID in id_amigos:
            c.execute("SELECT nombre_usuario, nombre_per, apellidos_per
                        FROM usuario WHERE id_usuario = '%s'" % (ID))
            info_amigo = c.fetchall()
            print "Amigo " + str(contador)
            mostrar_info_amigo(info_amigo)
            contador += 1

        opcionMenu = str(input("Selecciona un amigo: "))
        num_usuario = int(opcionMenu)-1

        #RECUPERAR ID AMIGO SELECCIONADO
        c.execute("SELECT id_usuario2 FROM amigo
                    WHERE id_usuario1 = '%s'" % (my_id))
        id_amigos = c.fetchall()
        amigo = id_amigos[num_usuario]

```

```

#Recuperamos nombre de amigo
c.execute("SELECT nombre_usuario FROM usuario
          WHERE id_usuario = '%s'" % (amigo[0]))
nombre_amigo = c.fetchone()

#Consultamos las canciones escuchadas hoy por el amigo
#seleccionado
c.execute("SELECT * FROM escucha WHERE id_usuario = '%s'
          and fecha = to_date('%s', 'DD/MM/YYYY ')" % (amigo[0], hoy))
canciones = c.fetchall()

#Mostramos lista de canciones escuchadas por el amigo
#seleccionado
print "\n—————"
print "Lista de canciones escuchadas hoy por %s:"
                                     % (nombre_amigo)

for cancion in canciones:
    c.execute("SELECT nombre_cancion FROM cancion
              WHERE id_cancion = '%s'" % (cancion[1]))
    nombre_cancion = c.fetchone()[0]
    print "— " + nombre_cancion

print "\n\n"

if opcionMenu == '0':
    conn.close()
    break

```