

# Administración y Diseño de Bases de Datos:

## Proyecto de la asignatura

### ULL Music Database

Carla Cristina Olivares Rodríguez  
[\(alu0101120218@ull.edu.es\)](mailto:alu0101120218@ull.edu.es)

Bruno Lorenzo Arroyo Pedraza  
[\(alu0101123677@ull.edu.es\)](mailto:alu0101123677@ull.edu.es)

Dana Belén Choque Zárate  
[\(alu0101328348@ull.edu.es\)](mailto:alu0101328348@ull.edu.es)



## Índice:

1. Objetivos.	2
2. Descripción el contexto de la base de datos	2
3. Modelo entidad-relación	4
4. Grafo relacional	8
5. Descripción de la base de datos en SQL	9
6. Carga de datos	26
7. Testing de consultas de la base de datos	29
8. Implementación de la API REST	33
9. Conclusiones	40



## 1. Objetivos.

El objetivo del proyecto es el diseño, creación e implementación de una base de datos para una aplicación destinada a escuchar y compartir música entre la comunidad de la Universidad de La Laguna (ULL). La aplicación recibe el nombre de ULL Music, y la base de datos es, ULL Music Database.

El propósito de la base de datos es almacenar el contenido fundamental de la aplicación, esto son la música, los usuarios, los artistas, etc...

## 2. Descripción el contexto de la base de datos

La aplicación ULL Music está destinada a almacenar música para ser compartida entre **usuarios**. La gestión de esta biblioteca de música requiere almacenar las distintas **canciones** que escuchan los usuarios.

Los **usuarios** tienen un ID de usuario, su nombre completo (formado por nombre y apellido), identificación o DNI, username o nombre de usuario (en el contexto de la Universidad existen distintos tipos de usuarios, por lo tanto catalogamos este atributo como “username” y no como, por ejemplo, “alu”), correo electrónico, fecha de registro en la aplicación (que será asignada automáticamente con la fecha del sistema).

Por otro lado, las **canciones** se componen por un ID de canción, el nombre de la canción, su año de salida y la duración de la misma.

Además los **usuarios** pueden crear **listas de canciones** o **playlist** para almacenar sus **canciones** favoritas, pudiendo también compartirlas con otros **usuarios**, dando lugar a que puedan existir varios propietarios por **lista de canciones**. Las **listas de canciones** están conformadas por un ID de lista, el nombre de la lista, una descripción y la duración total de la misma.

Los **usuarios** tienen también la posibilidad de insertar **comentarios** en las **canciones**, estos están formados por un ID del comentario, el contenido y la fecha del mismo.

Las **canciones** pertenecen a **autores**, los **autores** están conformados por un ID de autor, el nombre del autor y la discográfica a la que pertenecen (pudiendo ser esta característica nula). Los autores pueden ser **grupos** (conjuntos de autores, conformados por el nombre de cada uno de los integrantes) o **artistas solistas** (conformados por el nombre del artista).



Los **autores** tienen **álbumes** formados por sus **canciones**, estos **álbumes** tienen un ID de álbum, el nombre del álbum, la duración del mismo (este atributo deriva de la suma de la duración de todas las canciones que lo conforman) y el año de salida.

Las **canciones**, **autores** y **álbumes** son de un **género** concreto, los **géneros** están formados por un ID de género y el nombre del mismo.

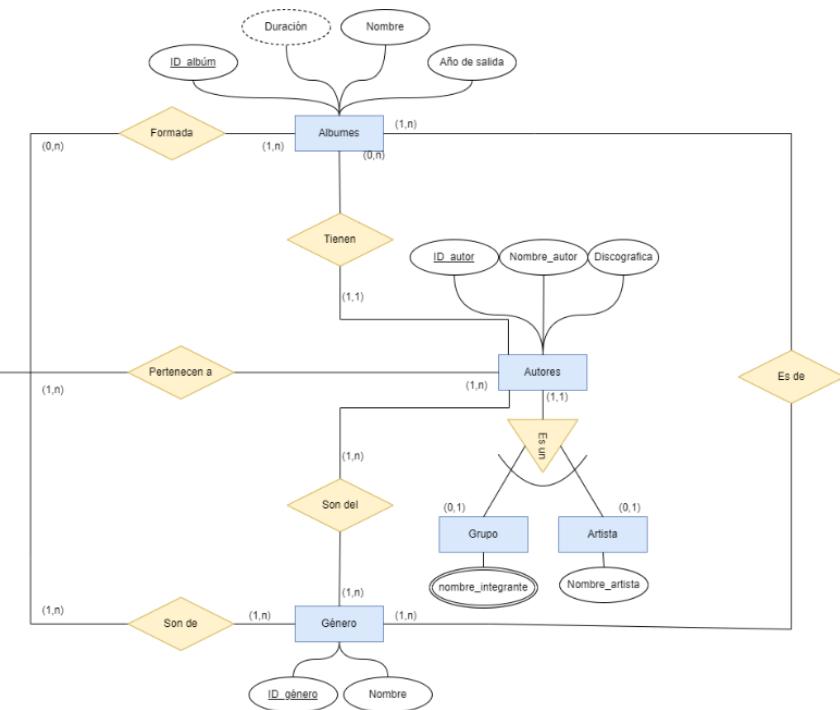
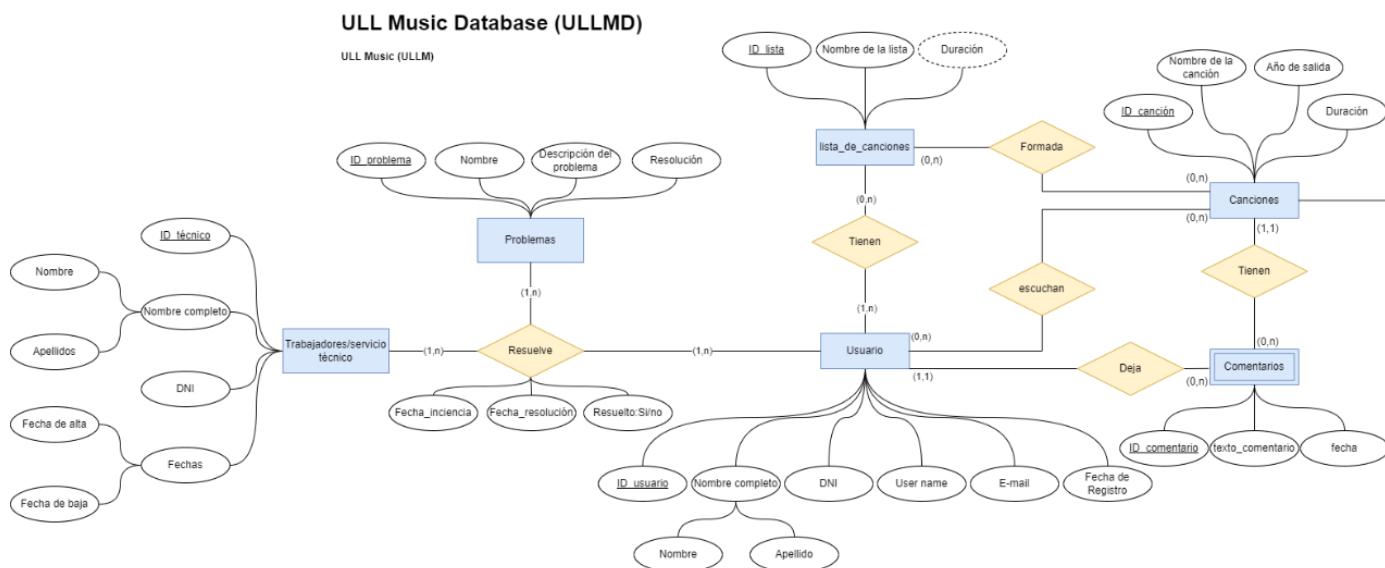
La aplicación requiere de un servicio técnico que será organizado de la siguiente manera: los **trabajadores** de este servicio resolverán los **problemas** del **usuario**, debemos tener en cuenta la fecha en la que se generó la incidencia y la fecha de resolución de la misma, además de si está resuelto o no.

Los **trabajadores** son entidades formadas por un ID técnico, un nombre completo (conformado por un nombre y apellidos), una identificación o DNI, y un conjunto de fechas que almacenan la fecha de alta y la fecha de baja del trabajador.

Los **problemas** contienen un ID del problema, un nombre para identificarlo, una descripción y un campo para comentar acerca de la resolución del problema



### 3. Modelo entidad-relación





## Descripción del modelo entidad-relación

El diagrama entidad-relación, está compuesto por las entidades fuertes y débiles. Cada una de estas entidades contiene sus correspondientes atributos, ya sean simples, compuestas, derivados o multivaluados.

En cuanto a las relaciones simples (dos entidades) y triples, se tiene relaciones del tipo 1:N, N:M, 0:N, 1:1. Además, se tiene relaciones de tipo herencia exclusiva.

En nuestro proyecto final, se identifican los siguientes componentes del diagrama entidad-relación:

- **Entidades fuertes**

**Trabajadores/servicio técnico:** contiene atributos simples como id\_técnico y dni; y atributos compuestos como nombre completo (contiene nombre y apellidos) y fechas (contiene fecha de alta y fecha de baja).

**Problemas:** contiene atributos normales como id\_problema, nombre, descripción del problema y resolución.

**Usuarios:** contiene atributos simples como id\_usuario, dni, username, e-mail, fecha de registro, y atributos compuestos como nombre completo (nombre y apellidos).

**Lista de canciones:** contiene atributos simples como id\_lista y nombre de la lista. Además, esta entidad tiene un atributo derivado como la duración. Ya que la duración de la lista de canciones se puede calcular a partir de los valores de los atributos relacionados.

**Canciones:** está formado por atributos simples como id\_canción, nombre de la canción, año de salida y duración.

**Álbumes:** contiene atributos simples como id\_album, nombre y año salida. Además, tiene un atributo derivado, que es la duración.

**Autores:** Está compuesto por atributos simples como el id\_autor, nombre autor y discografía.

**Grupo:** contiene un único atributo que es nombre\_integrante, este atributo es multivaluado. Ya que puede tomar distintos valores.

**Artista:** tiene un único atributo simple como nombre\_artista.

**Género:** contiene atributos simples como id\_género y nombre.



- **Entidades débiles**

**Comentarios:** es una entidad débil ya que depende de la entidad usuario. Está compuesto por id\_comentario, texto\_comentario y fecha.

- **Relaciones**

**Trabajadores, problemas y usuarios:** existe una relación triple del tipo 1:n donde vemos que: un trabajador resuelve varios problemas a varios usuarios, varios usuarios tienen problemas y esos problemas lo resuelven varios trabajadores, y un o varios problemas que tienen los usuarios lo resuelven varios trabajadores.

**Lista de canciones y usuario:** existe una relación 0:n (considerando que al crear una lista de canciones puede estar vacía al principio) y 1:n. La primera relación es cuando un usuario crea una lista vacía inicialmente y puede llegar a "n" canciones. La segunda relación es que la lista de canciones puede tener un usuario (persona que creó la lista) o varios, si la lista es colaborativa.

**Lista de canciones y canciones:** existe una relación m:n donde las canciones no está en una lista, o está en varias playlist, o la lista no tiene canciones (inicialmente) o tiene varias.

**Usuario y canciones:** existe una relación m:n en la que ningún o varios usuarios escuchan canciones.

**Canciones y álbumes:** existe una relación 1:n en donde una canción o varias canciones no pueden pertenecer a un álbum o varios álbumes. También, hay una relación 1:n en donde los álbumes puede estar formada por una canción (en tal caso, sería un sencillo) o por varias.

**Canciones y autores:** existe una relación m:n en la que una o varias canciones pueden pertenecer a uno o varios autores (colaboraciones).

**Canciones y género:** existe una relación m:n donde una o varias canciones puede pertenecer a uno o varios géneros.

**Álbumes y autores:** existen dos tipos de relaciones, la relación 0:n indica que un autor puede no tener (ser un autor que esté comenzando) o tener varios álbumes; la relación 1:1 indica que un álbum pertenece a un autor.



**Álbumes y género:** existe una relación 1:n donde uno o varios álbumes pueden pertenecer a uno o varios géneros. Al igual que en uno o varios géneros pueden contener uno o varios álbumes.

**Autores y género:** existe una relación 1:n donde uno o varios autores pueden pertenecer a uno o varios géneros. Del mismo modo, varios géneros pueden contener uno o varios autores.

**Usuario y comentarios:** existen dos tipos de relaciones, 1:1 y 1:n. El primero, indica que un comentario está asociado a un usuario. El segundo, indica que un usuario puede dejar uno o varios comentarios.

**Canciones y comentarios:** existen dos tipos de relaciones, 1:1 y 0:n. La relación 1:1 indica que un comentario está en una canción. La segunda relación, 0:n, indica que una canción puede tener ningún comentario o varios.

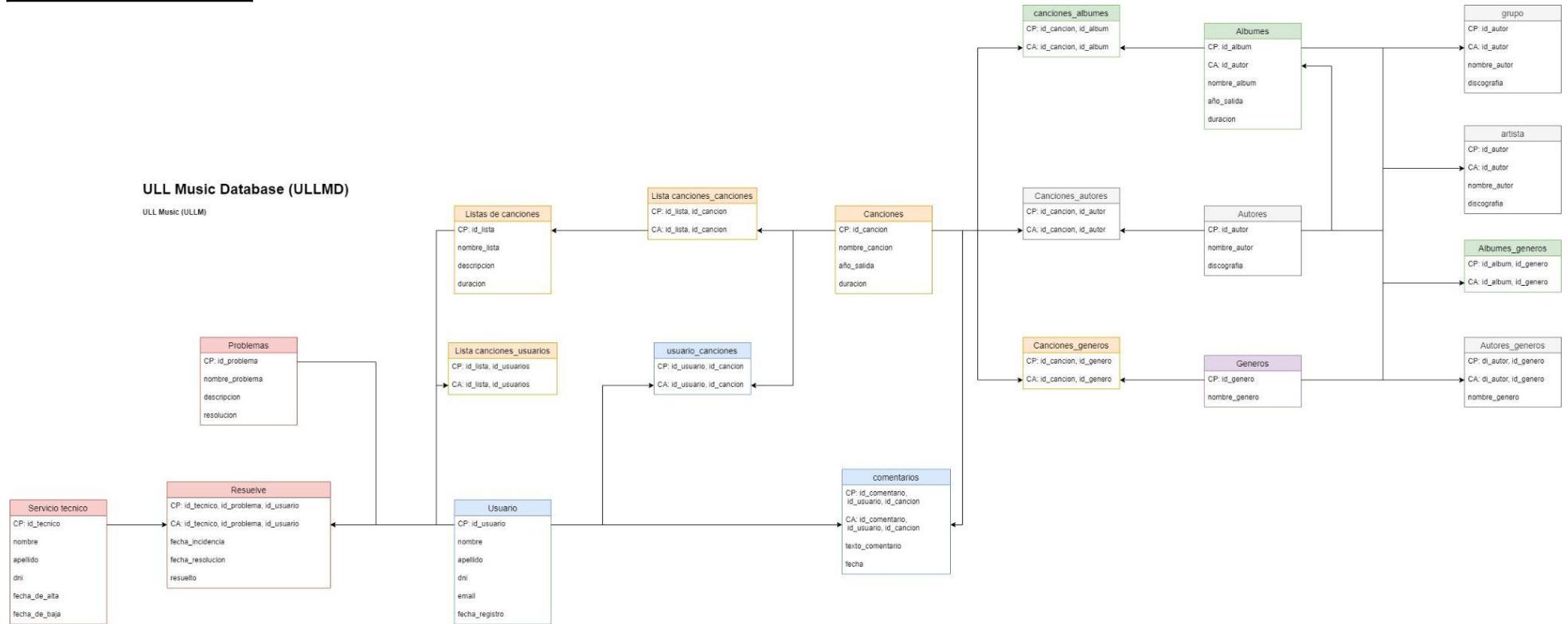
- Herencia exclusiva

Los autores pueden ser un grupo formado por integrantes o un artista, pero nunca las dos cosas.



## 4. Grafo relacional

**Leyenda:**  
CP: Clave privada  
CA: Clave ajena





## 5. Descripción de la base de datos en SQL

El archivo [ull\\_music.sql](#) contiene la implementación de la base de datos. Para la ejecución del mismo fichero, desde psql realizamos el siguiente comando:

```
=# \i ull_music.sql
```

- Inicialización de las tablas

Al inicializar las tablas, podemos distinguir entre las tablas principales y las generadas por las relaciones entre estas tablas.

La tabla **servicio\_tecnico** es una tabla principal que representa a un técnico como trabajador, con sus datos básicos de usuario, considerando que la identificación o DNI es un parámetro obligatorio y almacenando también una fecha de alta y de baja que representan la duración del puesto de empleo del técnico.

```
CREATE TABLE servicio_tecnico (
    id_tecnico SERIAL NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    dni VARCHAR(20) NOT NULL UNIQUE,
    fecha_de_alta DATE DEFAULT CURRENT_TIMESTAMP,
    fecha_de_baja DATE NULL,
    PRIMARY KEY (id_tecnico)
);
```

(Imagen 1. Creación de la tabla servicio\_tecnico)



La tabla **problema** es una tabla principal que representa los problemas que deben solucionar los técnicos, aportando una descripción del mismo además de información acerca de su resolución.

```
CREATE TABLE problema (
    id_problema SERIAL NOT NULL,
    nombre_problema VARCHAR(50) NOT NULL UNIQUE,
    descripcion VARCHAR(1000) NOT NULL,
    resolucion VARCHAR(1000) NOT NULL,
    PRIMARY KEY (id_problema)
);
```

(Imagen 2. Creación de la tabla problema)

La tabla **usuarios** es una tabla principal que representa la información de un usuario de la aplicación. En esta tabla podemos destacar el atributo de la fecha de registro del usuario, un atributo que se asignará automáticamente a la fecha que corresponda cuando sea creado el usuario.

```
CREATE TABLE usuarios (
    id_usuario SERIAL NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NOT NULL,
    dni VARCHAR(20) NOT NULL UNIQUE,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(50) NOT NULL UNIQUE,
    fecha_registro DATE DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id_usuario)
);
```

(Imagen 3. Creación de la tabla usuarios)



La tabla **canciones** es una tabla principal que representa una canción de la base de datos, contiene los atributos que mencionamos anteriormente en la [descripción del contexto de la base de datos](#).

```
CREATE TABLE canciones (
    id_cancion SERIAL NOT NULL,
    nombre_cancion VARCHAR(50) NOT NULL,
    año_salida INT NOT NULL,
    duracion FLOAT NOT NULL,
    PRIMARY KEY (id_cancion)
);
```

(Imagen 4. Creación de la tabla canciones)

La tabla **listas\_de\_canciones** es una tabla principal que representa una lista de reproducción o playlist de un usuario. La duración de la lista será el sumatorio de todas las canciones que contenga (como podremos ver en los disparadores más adelante).

```
CREATE TABLE listas_de_canciones (
    id_lista SERIAL NOT NULL,
    nombre_lista VARCHAR(50) NOT NULL,
    descripcion VARCHAR(1000) NULL,
    duracion FLOAT NOT NULL,
    PRIMARY KEY (id_lista)
);
```

(Imagen 5. Creación de la listas\_de\_canciones)



La tabla **álbumes** es una tabla principal que representa álbumes de canciones, que como mencionamos anteriormente son conjuntos de canciones que pertenecen a artistas. Al igual que las listas de canciones la duración total del conjunto de canciones se calculará con un disparador cada vez que se agregue una canción al álbum.

```
CREATE TABLE albumes (
    id_album SERIAL NOT NULL,
    id_autor INT NOT NULL,
    nombre_album VARCHAR(50) NOT NULL UNIQUE,
    año_salida INT NOT NULL,
    duracion FLOAT NOT NULL,
    PRIMARY KEY (id_album),
    CONSTRAINT fk_id_autor
        FOREIGN KEY (id_autor)
        REFERENCES autores (id_autor)
        ON DELETE CASCADE
);
```

(Imagen 6. Creación de la tabla *álbumes*)

La tabla **generos** es una tabla principal que representa los géneros musicales a los que pertenecen las canciones, artistas y álbumes.

```
CREATE TABLE generos (
    id_genero SERIAL NOT NULL,
    nombre_genero VARCHAR(50) NOT NULL UNIQUE,
    PRIMARY KEY (id_genero)
);
```

(Imagen 7. Creación de la tabla *generos*)



La tabla **comentarios** es una tabla principal que representa un comentario dejado por un usuario en una canción. Al igual que la fecha de registro del usuario, la fecha del comentario se asigna automáticamente con la fecha del sistema a la hora de crearse el comentario.

```
CREATE TABLE comentarios (
    id_comentario SERIAL NOT NULL,
    id_usuario INT NOT NULL,
    id_cancion INT NOT NULL,
    texto_comentario VARCHAR(1000) NOT NULL,
    fecha_comentario DATE DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY
(id_comentario, id_usuario, id_cancion),
CONSTRAINT fk_id_usuario
    FOREIGN KEY (id_usuario)
    REFERENCES usuarios (id_usuario)
    ON DELETE CASCADE,
CONSTRAINT fk_id_cancion
    FOREIGN KEY (id_cancion)
    REFERENCES canciones (id_cancion)
    ON DELETE CASCADE
);
```

(Imagen 8. Creación de la tabla comentarios)

La tabla **autores** es la última de las tablas principales, esta representa a un autor y la información básica de nombre y discografía.



```
CREATE TABLE autores (
    id_autor SERIAL NOT NULL,
    nombre_autor VARCHAR(50) NOT NULL UNIQUE,
    discografia VARCHAR(1000) NULL,
    PRIMARY KEY (id_autor)
);
```

(Imagen 9. Creación de la tabla autores)

Las tablas **grupo** y **artista** son tablas que nacen de una herencia: el autor de una canción puede ser un grupo o un artista solista, cada uno de estos con un atributo básico con el correspondiente nombre.

```
CREATE TABLE grupo (
    id_autor INT NOT NULL,
    nombre_integrante VARCHAR(100) NOT NULL,
    PRIMARY KEY (id_autor, nombre_integrante),
    CONSTRAINT fk_id_autor
        FOREIGN KEY (id_autor)
        REFERENCES autores (id_autor)
        ON DELETE CASCADE
);

CREATE TABLE artista (
    id_autor INT NOT NULL,
    nombre_artista VARCHAR(100) NOT NULL,
    PRIMARY KEY (id_autor),
    CONSTRAINT fk_id_autor
        FOREIGN KEY (id_autor)
        REFERENCES autores (id_autor)
        ON DELETE CASCADE
);
```

(Imagen 10. Creación de las tabla grupo y artista)



La tabla **resuelve** es una de las tablas que representa la relación entre las tablas principales, esta cumple la función de relacionar **servicio\_técnico** y **problema**, agregando un valor booleano al final para indicar que el problema ha sido resuelto o no.

```
CREATE TABLE resuelve (
    id_tecnico INT NOT NULL,
    id_problema INT NOT NULL,
    id_usuario INT NOT NULL,
    fecha_incidencia DATE NOT NULL,
    fecha_resolucion DATE NOT NULL,
    resuelto BOOLEAN NOT NULL,
    PRIMARY KEY
(id_tecnico, id_problema, id_usuario),
    CONSTRAINT fk_id_tecnico
        FOREIGN KEY (id_tecnico)
            REFERENCES servicio_tecnico (id_tecnico)
            ON DELETE CASCADE,
    CONSTRAINT fk_id_problema
        FOREIGN KEY (id_problema)
            REFERENCES problema (id_problema)
            ON DELETE CASCADE,
    CONSTRAINT fk_id_usuario
        FOREIGN KEY (id_usuario)
            REFERENCES usuarios (id_usuario)
            ON DELETE CASCADE
);
```

(Imagen 11. Creación de la tabla resuelve)



El resto de tablas representan el resto de [relaciones](#) entre tablas, la estructura que tienen sigue el mismo patrón que tiene, por ejemplo, **usuarios\_canciones** como podemos ver a continuación:

```
CREATE TABLE usuarios_canciones (
    id_usuario INT NOT NULL,
    id_cancion INT NOT NULL,
    PRIMARY KEY (id_usuario, id_cancion),
    CONSTRAINT fk_id_usuario
        FOREIGN KEY (id_usuario)
        REFERENCES usuarios (id_usuario)
        ON DELETE CASCADE,
    CONSTRAINT fk_id_cancion
        FOREIGN KEY (id_cancion)
        REFERENCES canciones (id_cancion)
        ON DELETE CASCADE
);
```

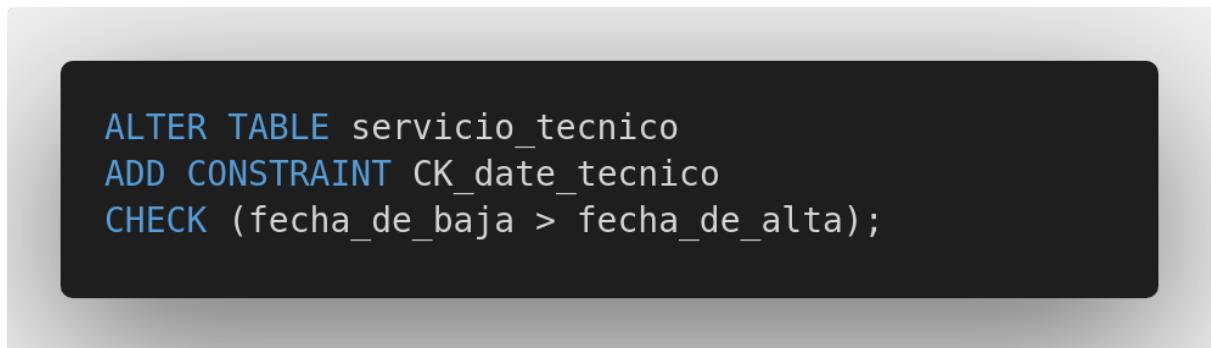
(Imagen 12. Creación de la tabla usuarios\_canciones)

Este patrón consiste en relacionar las claves principales (los ID) de las tablas implicadas en la relación.



- Checks/assertions

Para la definición de los checks definimos diferentes restricciones según lo pudiera requerir la tabla, en este caso podemos observar una restricción para asegurarnos de que la fecha de baja de un **técnico** siempre sea mayor que su fecha de alta:



(Imagen 13. check para la fecha de baja de un técnico)

Procedemos a intentar insertar fechas que no cumplen este requisito para generar un fallo:



(Imagen 14. actualizar un técnico para el check)

Podemos observar que, al intentar proporcionarle a la base de datos una información no válida por sus restricciones, obtenemos un error:

```
cur.execute("UPDATE servicio_tecnico SET nombre = %s, apellido = %s, dni = %s, fecha_de_alta = %s, fecha_de_baja = %s WHERE
own, id_technician)
psycopg2.errors.CheckViolation: new row for relation "servicio_tecnico" violates check constraint "ck_date_tecnico"
DETAIL: Failing row contains (1, Roberto, Mascarpone Olivares, 12345678P, 2023-01-12, 2023-01-02).
```

(Imagen 15. comprobación de CK\_date\_tecnico)



Con la siguiente restricción queremos asegurarnos de que la duración de las **canciones** siempre sea mayor que cero:

```
ALTER TABLE canciones
ADD CONSTRAINT CK_duration_canciones
CHECK (duracion > 0);
```

(Imagen 16. check para la duración de las canciones)

Comprobamos el correcto funcionamiento de la restricción intentando proporcionarle un dato erróneo a la base de datos:

The screenshot shows a form titled "Agregar una canción". It has three input fields: "nombre\_cancion" with value "Call me by your name", "año\_salida" with value "2022", and "duracion" with value "-2.00". Below the form is a button labeled "Añadir una canción".

(Imagen 17. agregar una canción para el check)

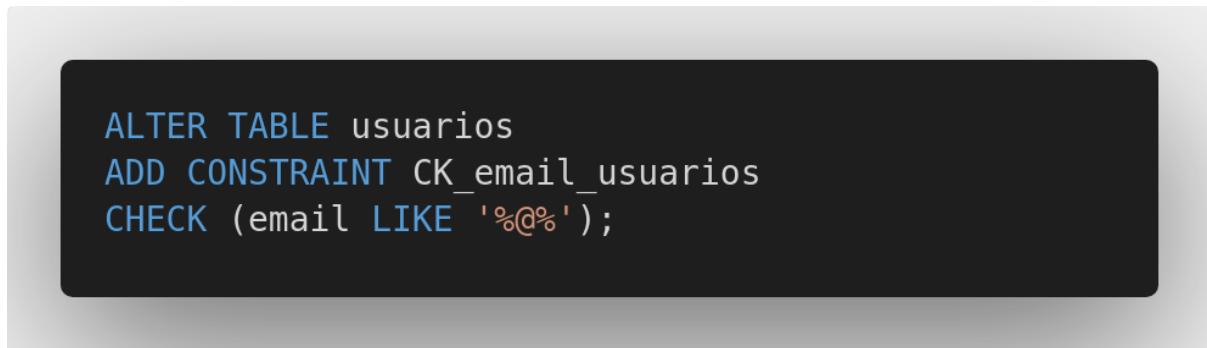
Recibimos el esperado fallo ya que la información no cumple con las restricciones:

```
psycopg2.errors.CheckViolation: new row for relation "canciones" violates check constraint
DETAIL:  Failing row contains (23, Call me by your name, 2022, -2).
```

(Imagen 18. comprobación de CK\_duration\_canciones)



La siguiente restricción consiste en asegurarnos de que el formato del correo electrónico del **usuario** sea correcto, es decir, de la forma “[usuario@correo.es](mailto:usuario@correo.es)“.



(Imagen 19. check para comprobación del email)

Como en las anteriores restricciones procedemos a intentar generar un fallo proporcionando a la base de datos información no soportada por las restricciones:



(Imagen 20. agregar un usuario para el check)

Observamos que el fallo se produce, asegurando que las restricciones se aplican:

```
File: /home/bruno/Programacion/dbsa_proyecto_final/app.py , line 33, in insert
    cur.execute("INSERT INTO usuarios (nombre, apellido, dni, username, email) VALUES (%s, %s, %s,
psycopg2.errors.CheckViolation: new row for relation "usuarios" violates check constraint "ck_email_usuarios"
DETAIL: Failing row contains (5, Eduardo, Castelier, 12345671A, educer, educer.com, 2023-01-12).
```

(Imagen 21. comprobación de CK\_email\_usuarios)



### - Disparadores

En el caso de los disparadores hemos desarrollado distintas acciones que necesitamos para construir correctamente nuestra base de datos. Podemos observar que el siguiente disparador se basa en realizar un sumatorio de la duración de las **canciones** para definir la duración total de las **listas de canciones** cuando se hace una inserción, eliminación y actualización:

```
CREATE FUNCTION duration_list() returns trigger AS $$  
BEGIN  
    UPDATE listas_de_canciones  
    SET duracion = (SELECT ROUND(SUM(duracion)::numeric, 2)  
                    FROM canciones  
                   WHERE id_cancion IN (SELECT id_cancion  
                                         FROM lista_canciones_canciones  
                                        WHERE id_lista = NEW.id_lista))  
    WHERE id_lista = NEW.id_lista;  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER duration_list AFTER INSERT OR UPDATE OR DELETE ON  
    lista_canciones_canciones  
FOR EACH ROW EXECUTE PROCEDURE duration_list();
```

(Imagen 22. Disparador para actualizar la duración de las listas de canciones)

Para comprobar su funcionamiento insertamos canciones nuevas en una lista de reproducción vacía y observamos su cambio (en este caso la lista “Indie Rock” (7) que tiene una duración total de 0 minutos):

id_lista	nombre_lista	descripción	duracion
7	Indie rock	Uma lista de reproducción con algunos de los grandes éxitos del indie rock	0
1	Clásicos del rock	Uma lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos	23.63
2	Hits del pop	Uma lista de reproducción con algunas de las canciones más populares del pop actual	4.7
3	Lo mejor del hip hop	Uma lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos	14.99
4	Éxitos del K-Pop	Uma lista de reproducción con algunas de las mejores canciones de K-pop	6.38
5	Electrónica para bailar	Uma lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar	6.59
6	Lo más nuevo del reggaetón	Uma lista de reproducción con las canciones más recientes y populares del reggaetón	9.71
(7 rows)			

(Imagen 23. Estado de la tabla antes del disparador)



Agregamos la canción con ID = 1 a la lista con ID = 7:

(Imagen 24. Inserción para activar el disparador)

Al ejecutarse el disparador por la operación de inserción la lista de reproducción “Indie Rock” pasa a tener una duración total de 3.39 minutos al haberle agregado una única canción con esa duración:

id_lista	nombre_lista	descripcion	duracion
1	Clásicos del rock	Uma lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos	23.63
2	Hits del pop	Uma lista de reproducción con algunas de las canciones más populares del pop actual	4.7
3	Lo mejor del hip hop	Uma lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos	14.99
4	Éxitos del K-Pop	Uma lista de reproducción con algunas de las mejores canciones de K-pop	6.38
5	Electrónica para bailar	Uma lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar	6.59
6	Lo más nuevo del reggeatón	Uma lista de reproducción con las canciones más recientes y populares del reggeatón	9.71
7	Indie rock	Uma lista de reproducción con algunos de los grandes éxitos del indie rock	3.39

(Imagen 25. Estado de la tabla después del disparador)

Para la duración total del álbum, calculamos la suma de las duraciones de las canciones que pertenecen al álbum. Para ello, creamos el siguiente disparador. En la función, se actualiza el atributo “duracion” de la tabla albumes, mediante la suma de la duración de las canciones que devuelve la consulta. Además, esta consulta devolverá el resultado con dos decimales. El disparador funcionará después de insertar, actualizar o borrar una canción del álbum.

```
CREATE FUNCTION duration_album() returns trigger AS $$  
BEGIN  
    UPDATE albumes  
    SET duracion = (SELECT ROUND((SUM(duracion))::numeric, 2)  
                    FROM canciones  
                   WHERE id_cancion IN (SELECT id_cancion  
                                         FROM canciones_albumes  
                                         WHERE id_album = NEW.id_album))  
                   WHERE id_album = NEW.id_album;  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER duration_album AFTER INSERT OR UPDATE OR DELETE ON  
    canciones_albumes  
FOR EACH ROW EXECUTE PROCEDURE duration_album();
```

(Imagen 26. Disparador para actualizar la duración de los álbumes de canciones)



A continuación, en la columna “duracion” tendremos la suma de las canciones iniciales antes del disparador.

id_album	id_autor	nombre_album	año_salida	duracion
1	1	ALL YOU CAN EAT	2022	8.78
2	2	PLANET HER	2021	4.7
3	3	UN VERANO SIN TI	2022	9.71
4	4	EL MADRILEÑO	2021	6.21
5	5	ORIGIN OF SYMMETRY	2001	14.53
6	6	<  °_° >	2022	6.59
7	7	RED MOON	2022	6.38
8	8	DEATH OF A BACHELOR	2016	9.1

(Imagen 27. Estado de la tabla antes del disparador)

Como podemos observar, al insertar una canción al álbum de id 1, vemos que se actualiza la duración de dicho álbum.

Agregar una canción a un álbum

6			1	
1				

Añadir una canción a un álbum

(Imagen 28. Inserción para activar el disparador)

id_album	id_autor	nombre_album	año_salida	duracion
2	2	PLANET HER	2021	4.7
3	3	UN VERANO SIN TI	2022	9.71
4	4	EL MADRILEÑO	2021	6.21
5	5	ORIGIN OF SYMMETRY	2001	14.53
6	6	<  °_° >	2022	6.59
7	7	RED MOON	2022	6.38
8	8	DEATH OF A BACHELOR	2016	9.1
1	1	ALL YOU CAN EAT	2022	12.96

(Imagen 29. Estado de la tabla después del disparador)



Vemos en este disparador como nuestro objetivo es que se eliminen las **canciones** que no tienen **autores** cuando detectemos que se está eliminando algo en la tabla **canciones\_autores** que los relaciona:

```
CREATE FUNCTION delete_song() returns trigger AS $$  
BEGIN  
    DELETE FROM canciones  
    WHERE id_cancion = OLD.id_cancion  
    AND NOT EXISTS (SELECT * FROM canciones_autores WHERE  
    id_cancion = OLD.id_cancion);  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER delete_song AFTER DELETE ON canciones_autores  
FOR EACH ROW EXECUTE PROCEDURE delete_song();
```

(Imagen 30. Disparador para eliminar una canción sin autores)

Aquí podemos observar el estado en el que se encontraba la tabla **canciones** antes de la modificación que haremos para comprobar el funcionamiento del disparador:

```
ull_music=# SELECT * FROM canciones;  
id_cancion | nombre_cancion | año_salida | duracion  
-----+-----+-----+-----  
1 | Que te vaya mal | 2022 | 3.39  
2 | Los Gatos | 2022 | 3.16  
3 | Gratiné | 2022 | 2.23  
4 | Get Into It | 2021 | 2.18  
5 | Woman | 2021 | 2.52  
6 | Ojitos lindos | 2022 | 4.18  
7 | Neverita | 2022 | 2.53  
8 | Me Fui De Vacaciones | 2022 | 3  
9 | Cuando Olvidaré | 2021 | 3.09  
10 | Los Tontos | 2021 | 3.12  
11 | Bliss | 2001 | 4.11  
12 | New Born | 2001 | 6.03  
13 | Megalomania | 2001 | 4.39  
14 | Lone Digger | 2015 | 3.49  
15 | Wonderland | 2015 | 3.1  
16 | Egotistic | 2018 | 3.16  
17 | Sleep In The Car | 2018 | 3.22  
18 | Emperors New Clothes | 2016 | 2.38  
19 | Golden Days | 2016 | 4.14  
20 | Victorious | 2016 | 2.58  
(20 rows)
```

(Imagen 31. Estado de la tabla antes del disparador)



Procedemos a eliminar el **autor** con ID = 1:

The screenshot shows a light purple rectangular form with a white border. At the top center, it says "Eliminar un autor musical". Below that is a dropdown menu with the number "1" selected. At the bottom right is a button labeled "Eliminar autor musical".

(Imagen 32. Eliminación para activar el disparador)

Las 3 primeras **canciones** en la imagen del estado anterior de la tabla corresponden al autor que eliminamos, como podemos observar se han eliminado esas tres entradas:

```
ull_music=# SELECT * FROM canciones;
+-----+-----+-----+-----+
| id_cancion | nombre_cancion | año_salida | duracion |
+-----+-----+-----+-----+
| 4 | Get Into It | 2021 | 2.18 |
| 5 | Woman | 2021 | 2.52 |
| 6 | Ojitos lindos | 2022 | 4.18 |
| 7 | Neverita | 2022 | 2.53 |
| 8 | Me Fui De Vacaciones | 2022 | 3 |
| 9 | Cuando Olvidaré | 2021 | 3.09 |
| 10 | Los Tontos | 2021 | 3.12 |
| 11 | Bliss | 2001 | 4.11 |
| 12 | New Born | 2001 | 6.03 |
| 13 | Megalomania | 2001 | 4.39 |
| 14 | Lone Digger | 2015 | 3.49 |
| 15 | Wonderland | 2015 | 3.1 |
| 16 | Egotistic | 2018 | 3.16 |
| 17 | Sleep In The Car | 2018 | 3.22 |
| 18 | Emperors New Clothes | 2016 | 2.38 |
| 19 | Golden Days | 2016 | 4.14 |
| 20 | Victorious | 2016 | 2.58 |
+-----+-----+-----+-----+
(17 rows)
```

(Imagen 33. Estado de la tabla después del disparador)



En el siguiente disparador, tiene como objetivo eliminar una lista de canciones cuando ésta no tiene usuarios. En la función **delete\_list**, borramos aquella lista de canciones con un id y sea una lista que no tenga usuarios. El disparador funcionará cuando sea después de una eliminación de usuarios en la tabla **lista\_de\_canciones\_usuarios**.

```
CREATE FUNCTION delete_list() RETURNS trigger AS $$  
BEGIN  
    DELETE FROM listas_de_canciones  
    WHERE id_lista = OLD.id_lista  
    AND NOT EXISTS (SELECT * FROM lista_canciones_usuarios WHERE  
    id_lista = OLD.id_lista);  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER delete_list AFTER DELETE ON lista_canciones_usuarios  
FOR EACH ROW EXECUTE PROCEDURE delete_list();
```

(Imagen 34. Disparador para eliminar una lista de canciones sin usuarios)

En la siguiente imagen, vemos el estado inicial antes de que se elimine una lista de canciones de la tabla **listas\_de\_canciones**.

id_lista	nombre_lista	descripcion	duracion
7	Indie rock	Una lista de reproducción con algunos de los grandes éxitos del indie rock	0
1	Clásicos del rock	Una lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos	23.63
2	Hits del pop	Una lista de reproducción con algunas de las canciones más populares del pop actual	4.7
3	Lo mejor del hip hop	Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos	14.99
4	Éxitos del K-Pop	Una lista de reproducción con algunas de las mejores canciones de K-pop	6.38
5	Electrónica para bailar	Una lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar	6.59
6	Lo más nuevo del reggaetón	Una lista de reproducción con las canciones más recientes y populares del reggaetón	9.71

(Imagen 35. Estado de la tabla antes del disparador)

En este ejemplo, vemos que eliminamos un usuario de la lista de canciones con ID 1. Como esta lista sólo contiene un único usuario, vemos que en la siguiente consulta, dicha lista ha sido eliminada de la tabla **listas\_de\_canciones**.

Eliminar un usuario de una lista de canciones

1  2

(Imagen 36. Eliminación para activar el disparador)



id_lista	nombre_lista	descripcion	duracion
7	Indie rock	Uma lista de reproducción con algunos de los grandes éxitos del indie rock	0
2	Hits del pop	Uma lista de reproducción con algunas de las canciones más populares del pop actual	4.7
3	Lo mejor del hip hop	Uma lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos	14.99
4	Éxitos del K-Pop	Uma lista de reproducción con algunas de las mejores canciones de K-pop	6.38
5	Electrónica para bailar	Uma lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar	6.59
6	Lo más nuevo del reggaetón	Uma lista de reproducción con las canciones más recientes y populares del reggaetón	9.71

(Imagen 37. Estado de la tabla después del disparador)

## 6. Carga de datos

En el mismo archivo [ull\\_music.sql](#), el último apartado del fichero contiene la carga de datos en las tablas. A continuación veremos algunas de las inserciones.

Inserción de técnicos en la tabla **servicio\_técnico**:

```
INSERT INTO servicio_tecnico (nombre, apellido, dni, fecha_de_alta, fecha_de_baja) VALUES
('Roberto', 'Mascarpone Olivares', '12345678P', '02-01-2022', NULL),
('Juan', 'Perez Reverte', '12345679A', '02-01-2022', NULL),
('Andrés', 'González González', '12345678B', '02-01-2022', NULL);
```

(Imagen 38. Carga de datos de servicio\_tecnico)

Inserción de datos a la tabla **problema**:

```
INSERT INTO problema (nombre_problema, descripcion, resolucion) VALUES
('No se puede iniciar sesión', 'No se puede iniciar sesión en la aplicación', 'Comprobar que las credenciales de acceso están bien'),
('No se puede reproducir una canción', 'No se puede reproducir una canción', 'Reiniciar la aplicación'),
('No se puede reproducir un álbum', 'No se puede reproducir un álbum', 'Reiniciar la aplicación'),
('No se puede reproducir una lista de reproducción', 'No se puede reproducir una lista de reproducción', 'Reiniciar la aplicación');
```

(Imagen 39. Carga de datos de problema)

Inserción de usuarios a la tabla **usuarios**:

```
INSERT INTO usuarios (nombre, apellido, dni, username, email, fecha_registro) VALUES
('Bruno Lorenzo', 'Arroyo Pedraza', '12345678C', 'brunolarroyo', 'bruno@gmail.com', '02-01-2022'),
('Carla Cristina', 'Olivares Rodríguez', '12345679D', 'carlacolivares', 'carla@gmail.com', '02-01-2022'),
('Dana Belén', 'Choque Zárate', '12345678E', 'danabelenchoque', 'dana@gmail.com', '02-01-2022'),
('Miguel Herradores', 'Tanaisu', '12345678Z', 'miguelisus', 'miguel@gmail.com', '02-01-2022'),
('Micaela Belén', 'Pedraza Zuminich', '12345678V', 'zuminica', 'micaela@gmail.com', '02-01-2022');
```

(Imagen 40. Carga de datos de usuarios)



### Inserción de canciones a la tabla **canciones**:

```
INSERT INTO canciones (nombre_cancion, año_salida, duracion) VALUES
    ('Que te vaya mal', 2022, 3.39), -- ALL YOU CAN EAT (COMIDA PARA LLEVAR)
    ('Los Gatos', 2022, 3.16), -- ALL YOU CAN EAT (COMIDA PARA LLEVAR)
    ('Gratiné', 2022, 2.23), -- ALL YOU CAN EAT (COMIDA PARA LLEVAR)
    ('Get Into It', 2021, 2.18), -- PLANET HER (DOJA CAT)
    ('Woman', 2021, 2.52), -- PLANET HER (DOJA CAT)
    ('Ojitos lindos', 2022, 4.18), -- UN VERANO SIN TI (BAD BUNNY)
    ('Neverita', 2022, 2.53), -- UN VERANO SIN TI (BAD BUNNY)
    ('Me Fui De Vacaciones', 2022, 3.00), -- UN VERANO SIN TI (BAD BUNNY)
    ('Cuando Olvidaré', 2021, 3.09), -- EL MADRILEÑO (C. TANGANA)
    ('Los Tontos', 2021, 3.12), -- EL MADRILEÑO (C. TANGANA)
    ('Bliss', 2001, 4.11), -- ORIGIN OF SYMMETRY (MUSE)
    ('New Born', 2001, 6.03), -- ORIGIN OF SYMMETRY (MUSE)
    ('Megalomania', 2001, 4.39), -- ORIGIN OF SYMMETRY (MUSE)
    ('Lone Digger', 2015, 3.49), -- <|º_º|> (CARAVAN PALACE)
    ('Wonderland', 2015, 3.10), -- <|º_º|> (CARAVAN PALACE)
    ('Egotistic', 2018, 3.16), -- RED MOON (MAMAMOO)
    ('Sleep In The Car', 2018, 3.22), -- RED MOON (MAMAMOO)
    ('Emperors New Clothes', 2016, 2.38),
    -- (DGATDeñFDAYBACH2016, (4Añ14); AT ÐBÄTBIÐEÐOÀ BACHELOR (PANIC! AT THE DISCO)
    ('Victorious', 2016, 2.58); -- DEATH OF A BACHELOR (PANIC! AT THE DISCO)
```

(Imagen 41. Carga de datos de canciones)

### Inserción de **listas de canciones**. Podemos observar que las listas se crean con una duración de cero minutos:

```
INSERT INTO listas_de_canciones (nombre_lista,descripcion, duracion) VALUES
    ('Clásicos del rock', 'Una lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos', 0),
    ('Hits del pop', 'Una lista de reproducción con algunas de las canciones más populares del pop actual', 0),
    ('Lo mejor del hip hop', 'Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos', 0),
    ('Éxitos del K-Pop', 'Una lista de reproducción con algunas de las mejores canciones de K-pop', 0),
    ('Electrónica para bailar', 'Una lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar', 0),
    ('Lo más nuevo del reggaetón', 'Una lista de reproducción con las canciones más recientes y populares del reggaetón', 0),
    ('Indie rock', 'Una lista de reproducción con algunos de los grandes éxitos del indie rock', 0);
```

(Imagen 42. Carga de datos de lista\_de\_canciones)

### Inserción de autores (nombre de un grupo o artista) a la tabla **autores**:



```
INSERT INTO autores (nombre_autor, discografia) VALUES
('Motherflowers', NULL),
('Doja Cat', 'RCA Records'),
('Bad Bunny', 'Rimas Entertainment'),
('C. Tangana', 'Sony Music'),
('MUSE', 'Warner Records'),
('Caravan Palace', 'Le Plan Recordings'),
('MAMAMOO', 'RBW Entertainment'),
('PANIC! AT THE DISCO', 'Atlantic Records');
```

(Imagen 43. Carga de datos de servicio\_tecnico)

Inserción de **álbumes**. Al igual que las listas de canciones, la duración total del álbum se crea por defecto a 0 minutos.

```
INSERT INTO albumes (id_autor, nombre_album, año_salida, duracion) VALUES
(1, 'ALL YOU CAN EAT', 2022, 0),
(2, 'PLANET HER', 2021, 0),
(3, 'UN VERANO SIN TI', 2022, 0),
(4, 'EL MADRILEÑO', 2021, 0),
(5, 'ORIGIN OF SYMMETRY', 2001, 0),
(6, '<|º_º|>', 2022, 0),
(7, 'RED MOON', 2022, 0),
(8, 'DEATH OF A BACHELOR', 2016, 0);
```

(Imagen 44. Carga de datos de álbumes)

Inserción de los géneros musicales a la tabla **generos**:



```
INSERT INTO generos (nombre_genero) VALUES
('Rock'),
('Pop'),
('Jazz'),
('Metal'),
('Clásica'),
('Reggae'),
('Salsa'),
('Folk'),
('Blues'),
('Rap'),
('Electrónica'),
('Funk'),
('Soul'),
('Country'),
('Reggaeton'),
('Disco'),
('Indie'),
('Punk'),
('Hip Hop'),
```

(Imagen 45. Carga de datos de géneros)

El resto de inserciones se pueden visualizar en el repositorio GitHub en el archivo [ull\\_music.sql](#).

## 7. Testing de consultas de la base de datos

Se han realizado diversas consultas para testear la base de datos. Varias de ellas incorporadas en la API REST asociada.

- Autor de la canción más escuchada

```
SELECT nombre_autor, nombre_cancion
FROM
(SELECT id_autor, id_cancion
FROM canciones_autores
WHERE id_cancion = (SELECT id_cancion
FROM usuarios_canciones
GROUP BY id_cancion
ORDER BY count(id_usuario)
DESC LIMIT 1)) AS t1
```



#### INNER JOIN

```
(SELECT * FROM autores) AS t2  
USING (id_autor)  
INNER JOIN  
(SELECT * FROM canciones) AS t3  
USING (id_cancion);
```

nombre_autor	nombre_cancion
Bad Bunny	Me Fui De Vacaciones
(1 row)	

(Imagen 46. Resultado de la consulta)

- Usuario con mayor cantidad de listas de canciones

```
SELECT * FROM usuarios  
WHERE id_usuario = (  
SELECT id_usuario FROM  
lista_canciones_usuarios  
GROUP BY id_usuario  
ORDER BY count(id_lista) DESC LIMIT 1);
```

id_usuario	nombre	apellido	dni	username	email	fecha_registro
3	Dana Belén	Choque Zárate	12345678E	danabelenchoque	dana@gmail.com	2022-01-02
(1 row)						

(Imagen 47. Resultado de la consulta)

- Problema más común

```
SELECT * FROM problema  
WHERE id_problema = (  
SELECT id_problema FROM  
resuelve  
GROUP BY id_problema  
ORDER BY count(id_usuario) DESC LIMIT 1);
```

id_problema	nombre_problema	descripcion	resolucion
2	No se puede reproducir una canción	No se puede reproducir una canción	Reiniciar la aplicación
(1 row)			

(Imagen 48. Resultado de la consulta)



- Top 10 géneros con más canciones

```
SELECT id_genero, nombre_genero, canciones
FROM generos
INNER JOIN
(SELECT id_genero, count(id_cancion) as canciones FROM
canciones_generos
GROUP BY id_genero
ORDER BY canciones DESC LIMIT 10) AS t1
USING (id_genero)
ORDER BY canciones DESC;
```

id_genero	nombre_genero	canciones
19	Hip Hop	6
1	Rock	5
11	Electrónica	4
2	Pop	4
15	Reggaeton	3
32	Kpop	2
10	Rap	1
7	Salsa	1
6	Reggae	1
20	Alternativa	1

(Imagen 49. Resultado de la consulta)

- Listas de canciones, con sus respectivos usuarios y canciones

```
SELECT id_lista, nombre_lista, descripcion, duracion, usuarios, canciones
FROM (SELECT id_lista, nombre_lista, descripcion, string_agg(nombre_cancion, ', ') AS
canciones
FROM listas_de_canciones
INNER JOIN lista_canciones_canciones USING (id_lista)
INNER JOIN canciones USING (id_cancion)
GROUP BY id_lista) AS t1
INNER JOIN (SELECT id_lista, duracion, string_agg(nombre, ', ') AS usuarios
FROM listas_de_canciones
INNER JOIN lista_canciones_usuarios USING (id_lista)
INNER JOIN usuarios USING (id_usuario)
GROUP BY id_lista) AS t1 USING (id_lista)
ORDER BY id_lista ASC;
```



id_lista	nombre_lista	descripcion	duracion	usuarios
<b>canciones</b>				
2	Hits del pop	Una lista de reproducción con algunas de las canciones más populares del pop actual	4.7	Dana Belén, Bruno Lorenzo
3	Lo mejor del hip hop	Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos	14.99	Carla Cristina
4	Los Tontos	Cuando Olvidaré, L		
5	Éxitos del K-Pop	Una lista de reproducción con algunas de las mejores canciones de K-pop	6.38	Dana Belén
6	n The Car	Egotistic, Sleep I		
7	Electrónica para bailar	Una lista de reproducción con algunas de las mejores canciones de la música electrónica para bailar	6.59	Bruno Lorenzo
8	Land	Lone Digger, Wonde		
9	Lo más nuevo del reggaetón	Ojitos lindos, Nev		
10	erita, Me Fui De Vacaciones			
11	(5 rows)			

(Imagen 49. Resultado de la consulta)

#### - Álbumes y canciones que los conforman

```

SELECT id_album, id_autor, nombre_album, albumes.año_salida, albumes.duracion,
string_agg(nombre_cancion, ', ')
FROM albumes
INNER JOIN canciones_albumes USING (id_album)
INNER JOIN canciones USING (id_cancion)
GROUP BY id_album
ORDER BY id_album ASC;

```

id_album	id_autor	nombre_album	año_salida	duracion	string_agg
2	2	PLANET HER	2021	4.7	Get Into It, Woman
3	3	UN VERANO SIN TI	2022	9.71	Ojitos lindos, Neverita, Me Fui De Vacaciones
4	4	EL MADRILEÑO	2021	6.21	Cuando Olvidaré, Los Tontos
5	5	ORIGIN OF SYMMETRY	2001	14.53	Bliss, New Born, Megalomania
6	6	< º_º >	2022	6.59	Lone Digger, Wonderland
7	7	RED MOON	2022	6.38	Egotistic, Sleep In The Car
8	8	DEATH OF A BACHELOR	2016	9.1	Emperors New Clothes, Golden Days, Victorious
(7 rows)					

(Imagen 50. Resultado de la consulta)

#### - Grupos musicales y sus miembros

```

SELECT id_autor, nombre_autor, discografia, string_agg(nombre_integrante, ', ')
FROM autores
INNER JOIN grupo USING (id_autor)
GROUP BY id_autor
ORDER BY id_autor ASC;

```

id_autor	nombre_autor	discografia	string_agg
5	MUSE	Warner Records	Matt Bellamy, Chris Wolstenholme, Dominic Howard
6	Caravan Palace	Le Plan Recordings	Zoe Colletti, Arnaud Vial, Hugues Payen, Camille Chapeltière, Charles Delaporte, Antoine Toustou, Paul-Marie Barbier, Victor Ralmondeau
7	MAMAKOO	RBW Entertainment	Moonbyut, Solar, Wheein, Hwasa

(Imagen 51. Resultado de la consulta)



## 8. Implementación de la API REST

Se ha implementado mediante Flask una API REST para permitir realizar operaciones CRUD sobre ULL Music Database.

En el fichero [app.py](#), se definirán las funciones de inserción, eliminación, modificación y consulta de datos para las tablas que conforman la base de datos. Por otro lado, se definen en la carpeta [templates](#), los ficheros html necesarios para que el usuario interactúe y realice las operaciones CRUD.

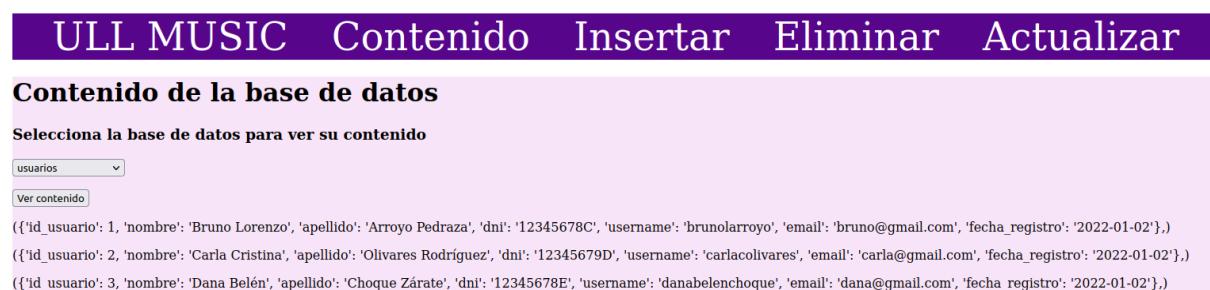
Desde el HOME podemos ver una pequeña descripción, y los componentes del grupo de trabajo. Para acceder a este apartado, que sería la raíz de la API '/', se puede acceder también pulsando en ULL MUSIC.



(Imagen 52. Raíz de la API REST)

- Consultar

Para consultar la base de datos se puede acceder desde la pestaña de Contenido. Inicialmente solo veremos un desplegable con las distintas tablas que podemos consultar. Posteriormente, al seleccionar una, y pulsar en ver contenido, se nos mostrará el contenido de la tabla.



(Imagen 53. Visualización de la tabla usuarios a través de la API)



En algunos casos, como listas de canciones o álbumes se nos mostrará información adicional. En el caso de la lista de canciones, además de la información de la propia tabla, se nos muestra los usuarios que pueden acceder a ella, y las canciones que la conforman. Destacar que solo se visualizan las listas que contienen al menos una canción.

The screenshot shows a web browser window with the URL 192.168.1.134:8080/index/. The title bar says "ULL MUSIC Contenido Insertar Eliminar Actualizar". Below it, a purple header bar says "Contenido de la base de datos". A dropdown menu shows "listas de canciones" and a "Ver contenido" button. The main content area displays a JSON-like list of song lists:

```
{'id_lista': 1, 'nombre_lista': 'Clásicos del rock', 'descripcion': 'Una lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos', 'duracion': 23.63, 'usuarios': 'Bruno Lorenzo, Carla Cristina', 'canciones': 'Bliss, New Born, Megalomania, Emperors New Clothes, Golden Days, Victorious'}, {'id_lista': 2, 'nombre_lista': 'Hits del pop', 'descripcion': 'Una lista de reproducción con algunas de las canciones más populares del pop actual', 'duracion': 4.7, 'usuarios': 'Dana Belén, Bruno Lorenzo', 'canciones': 'Get Into It, Woman'}, {"id_lista": 3, "nombre_lista": "Lo mejor del hip hop", "descripcion": "Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos", "duracion": 14.99, "usuarios": "Carla Cristina", "canciones": "Que te vaya mal, Los Gatos, Gratine, Cuando Olvidare, Los Tontos"}, {"id_lista": 4, "nombre_lista": "Éxitos del K-Pop", "descripcion": "Una lista de reproducción con algunas de las mejores canciones de K-pop", "duracion": 6.38, "usuarios": "Dana Belén", "canciones": "Egotistic, Sleep In The Car"}, {"id_lista": 5, "nombre_lista": "Electrónica para bailar", "descripcion": "Una lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar", "duracion": 6.59, "usuarios": "Bruno Lorenzo", "canciones": "Lone Digger, Wonderland"}, {"id_lista": 6, "nombre_lista": "Lo más nuevo del reggaetón", "descripcion": "Una lista de reproducción con las canciones más recientes y populares del reggaetón", "duracion": 9.71, "usuarios": "Carla Cristina", "canciones": "Ojitos lindos, Neverita, Me Fui De Vacaciones"}, {"id_lista": 7, "nombre_lista": "Indie rock", "descripcion": "Una lista de reproducción con algunos de los grandes éxitos del indie rock", "duracion": 3.39, "usuarios": "Dana Belén", "canciones": "Que te vaya mal"}}
```

(Imagen 54. Visualización de listas de canciones a través de la API)

Por otro lado, se ha dividido autores para mostrar los autores junto con la información de la tabla grupo, y por otro lado autores con artista

The screenshot shows a web browser window with the URL 192.168.1.134:8080/index/. The title bar says "ULL MUSIC Contenido Insertar Eliminar Actualizar". Below it, a purple header bar says "Contenido de la base de datos". A dropdown menu shows "grupos musicales" and a "Ver contenido" button. The main content area displays a JSON-like list of authors grouped by musical group:

```
{'id_autor': 1, 'nombre_autor': 'Motherflowers', 'discografia': None, 'integrantes': 'Irepelusa, Vetzalone, Frank Lucas, Tayko'}, {"id_autor": 5, "nombre_autor": "MUSE", "discografia": "Warner Records", "integrantes": "Matt Bellamy, Chris Wolstenholme, Dominic Howard"}, {"id_autor": 6, "nombre_autor": "Caravan Palace", "discografia": "Le Plan Recordings", "integrantes": "Zoé Colotis, Arnaud Vial, Hugues Payen, Camille Chapelière, Charles Delaporte, Antoine Toustou, Paul-Marie Barbier, Victor Raimondeau"}, {"id_autor": 7, "nombre_autor": "MAMAMOO", "discografia": "RBW Entertainment", "integrantes": "Moonhyul, Solar, Wheein, Hwasa"}}
```

(Imagen 55. Visualización de autores que son un grupo musical a través de la API)



### - Insertar

Para insertar elementos en la base de datos lo podemos hacer desde la pestaña Insertar. Desde aquí podemos insertar cualquier elemento en la base de datos, ya sea de forma directa o indirectamente.

The screenshot shows the 'Insertar' (Insert) tab selected in the top navigation bar. Below it, three sub-forms are displayed:

- Agregar un usuario**: Fields for nombre, apellido, dni, username, email, and fecha\_registro. A 'Anadir un usuario' button is at the bottom.
- Agrega las canciones que escucha cierto usuario**: Fields for ID del usuario and ID de la canción, with an 'Añadir canciones escuchadas' button.
- Agregar una lista de canciones**: Fields for nombre de la lista, descripción de la lista, and two lists of IDs separated by commas (ID de las canciones que componen la lista and ID de los usuarios que pueden acceder a la lista).

(Imagen 56. Apartado de inserta de la API)

Por ejemplo, podemos insertar un usuario

id_usuario	nombre	apellido	dni	username	email	fecha_registro
1	Bruno Lorenzo	Arroyo Pedraza	12345678C	brunolarroyo	bruno@gmail.com	2022-01-02
2	Carla Cristina	Olivares Rodríguez	12345679D	carlacolivares	carla@gmail.com	2022-01-02
3	Dana Belén	Choque Zárate	12345678E	danaabelenchoque	dana@gmail.com	2022-01-02
4	Miguel Herradores	Tanaisu	12345678Z	miguelisu	miguel@gmail.com	2022-01-02
5	Micaela Belén	Pedraza Zuminich	12345678V	zuminica	micaela@gmail.com	2022-01-02

(Imagen 57. Tabla usuarios antes de la inserción)

The screenshot shows the 'Agregar un usuario' (Add user) form. It contains the following fields:

- nombre: Aragon Diaz
- apellido: Lucky Caballero
- username: araly
- email: araly@gmail.com
- dni: 12345673P

A 'Añadir un usuario' button is located at the bottom of the form.

(Imagen 58. Inserción de un usuario desde la API)



usuarios
Ver contenido
({"id_usuario": 1, "nombre": "Bruno Lorenzo", "apellido": "Arroyo Pedraza", "dni": "12345678C", "username": "brunolarroyo", "email": "bruno@gmail.com", "fecha_registro": "2022-01-02"},)
({"id_usuario": 2, "nombre": "Carla Cristina", "apellido": "Olivares Rodríguez", "dni": "12345679D", "username": "carlacolivares", "email": "carla@gmail.com", "fecha_registro": "2022-01-02"},)
({"id_usuario": 3, "nombre": "Dana Belén", "apellido": "Choque Zárate", "dni": "12345678E", "username": "danabelenchoque", "email": "dana@gmail.com", "fecha_registro": "2022-01-02"},)
({"id_usuario": 4, "nombre": "Miguel Herradores", "apellido": "Tanaisu", "dni": "12345678Z", "username": "miguelisu", "email": "miguel@gmail.com", "fecha_registro": "2022-01-02"},)
({"id_usuario": 5, "nombre": "Micaela Belén", "apellido": "Pedraza Zuminich", "dni": "12345678V", "username": "zuminica", "email": "micaela@gmail.com", "fecha_registro": "2022-01-02"},)
({"id_usuario": 6, "nombre": "Aragon Diaz", "apellido": "Lucky Caballero", "dni": "12345673P", "username": "araly", "email": "araly@gmail.com", "fecha_registro": "2023-01-12"},)

(Imagen 59. Tabla usuarios después de la inserción)

Pero, si insertamos una lista de canciones, se nos pedirá los elementos necesarios para también hacer las inserciones en las subtablas generadas por las relaciones. Esto pasa con varias inserciones, como álbumes o autores, también para llenar las tablas generadas por la herencia.

En este caso se observa, que al realizar la inserción y también a los disparadores vistos en el apartado anterior, la nueva lista calcula su duración, esto no sería posible si no se realizan las inserciones en las subtablas correspondientes, así como conseguir los nombres de los usuarios y las canciones.

listas de canciones
Ver contenido
({"id_lista": 1, "nombre_lista": "Clásicos del rock", "descripcion": "Una lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos", "duracion": 23.63, "usuarios": "Bruno Lorenzo, Carla Cristina", "canciones": "Bliss, New Born, Megalomania, Emperors New Clothes, Golden Days, Victorious"},)
({"id_lista": 2, "nombre_lista": "Hits del pop", "descripcion": "Una lista de reproducción con algunas de las canciones más populares del pop actual", "duracion": 4.7, "usuarios": "Dana Belén, Bruno Lorenzo", "canciones": "Get Into It, Woman"},)
({"id_lista": 3, "nombre_lista": "Lo mejor del hip hop", "descripcion": "Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos", "duracion": 14.99, "usuarios": "Carla Cristina", "canciones": "Que te vaya mal, Los Gatos, Gratiné, Cuando Olvidare, Los Tontos"},)
({"id_lista": 4, "nombre_lista": "Éxitos del K-Pop", "descripcion": "Una lista de reproducción con algunas de las mejores canciones de K-pop", "duracion": 6.38, "usuarios": "Dana Belén", "canciones": "Egotistic, Sleep In The Car"},)
({"id_lista": 5, "nombre_lista": "Electrónica para bailar", "descripcion": "Una lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar", "duracion": 6.59, "usuarios": "Bruno Lorenzo", "canciones": "Lone Digger, Wonderland"},)
({"id_lista": 6, "nombre_lista": "Lo más nuevo del reggaetón", "descripcion": "Una lista de reproducción con las canciones más recientes y populares del reggaetón", "duracion": 9.71, "usuarios": "Carla Cristina", "canciones": "Ojitos lindos, Neverita, Me Fui De Vacaciones"},)

(Imagen 60. Visualización de la lista de canciones antes de la inserción)

**Agregar una lista de canciones**

lista de prueba
esto es una lista de prueba
1,2,3,4,5,6,7
6
<input type="button" value="Añadir una lista de canciones"/>

(Imagen 61. Inserción de una nueva lista de canciones)

listas de canciones
Ver contenido
({"id_lista": 1, "nombre_lista": "Clásicos del rock", "descripcion": "Una lista de reproducción con algunos de los grandes éxitos del rock de todos los tiempos", "duracion": 23.63, "usuarios": "Bruno Lorenzo, Carla Cristina", "canciones": "Bliss, New Born, Megalomania, Emperors New Clothes, Golden Days, Victorious"},)
({"id_lista": 2, "nombre_lista": "Hits del pop", "descripcion": "Una lista de reproducción con algunas de las canciones más populares del pop actual", "duracion": 4.7, "usuarios": "Dana Belén, Bruno Lorenzo", "canciones": "Get Into It, Woman"},)
({"id_lista": 3, "nombre_lista": "Lo mejor del hip hop", "descripcion": "Una lista de reproducción con algunos de los mejores éxitos del hip hop de todos los tiempos", "duracion": 14.99, "usuarios": "Carla Cristina", "canciones": "Que te vaya mal, Los Gatos, Gratiné, Cuando Olvidare, Los Tontos"},)
({"id_lista": 4, "nombre_lista": "Éxitos del K-Pop", "descripcion": "Una lista de reproducción con algunas de las mejores canciones de K-pop", "duracion": 6.38, "usuarios": "Dana Belén", "canciones": "Egotistic, Sleep In The Car"},)
({"id_lista": 5, "nombre_lista": "Electrónica para bailar", "descripcion": "Una lista de reproducción con algunos de los éxitos más populares de la música electrónica para bailar", "duracion": 6.59, "usuarios": "Bruno Lorenzo", "canciones": "Lone Digger, Wonderland"},)
({"id_lista": 6, "nombre_lista": "Lo más nuevo del reggaetón", "descripcion": "Una lista de reproducción con las canciones más recientes y populares del reggaetón", "duracion": 9.71, "usuarios": "Carla Cristina", "canciones": "Ojitos lindos, Neverita, Me Fui De Vacaciones"},)
({"id_lista": 8, "nombre_lista": "lista de prueba", "descripcion": "esta es una lista de prueba", "duracion": 20.19, "usuarios": "Aragon Diaz", "canciones": "Que te vaya mal, Los Gatos, Gratiné, Get Into It, Woman, Ojitos lindos, Neverita"},)

(Imagen 62. Visualización de la lista de canciones después de la inserción)



- Eliminar

Para eliminar elementos en la base de datos lo podemos hacer desde la pestaña Eliminar. Desde aquí podemos eliminar cualquier elemento en la base de datos. Podriamos elegir si queremos eliminar un usuario en concreto, o un álbum de música, o sólo una canción de este álbum, las eliminaciones se hacen buscando por ID.

ULL MUSIC Contenido Insertar Eliminar Actualizar

Elimine el elemento que deseé en la base de datos de ULL Music mediante el ID

**Eliminar un usuario**  
ID del usuario   
Eliminar usuario

**Eliminar las canciones que ha escuchado un usuario**  
ID del usuario  ID de la canción   
Eliminar canciones escuchadas

**Eliminar una lista de canciones**  
ID de la lista   
Eliminar una lista de canciones

**Eliminar una canción de una lista de canciones**  
ID de la lista  ID de la canción   
Eliminar una canción de una lista de canciones

(Imagen 63. Apartado de eliminar de la API)

Para mostrarlo, eliminamos un integrante de autor con grupo musical, y posteriormente, ese autor.

grupos musicales ▾

Ver contenido

```
({"id_autor": 1, "nombre_autor": "Motherflowers", "discografia": None, "integrantes": "Irepelusa, Veztalone, Frank Lucas, Tayko"},)  
({"id_autor": 5, "nombre_autor": "MUSE", "discografia": "Warner Records", "integrantes": "Matt Bellamy, Chris Wolstenholme, Dominic Howard"},)  
({"id_autor": 6, "nombre_autor": "Caravan Palace", "discografia": "Le Plan Recordings", "integrantes": "Zoé Colotis, Arnaud Vial, Hugues Payen, Camille Raimondeau"},)  
({"id_autor": 7, "nombre_autor": "MAMAMOO", "discografia": "RBW Entertainment", "integrantes": "Moonbyul, Solar, Wheein, Hwasa"},)
```

(Imagen . Visualización de los autores antes de la eliminación)

Eliminar un integrante de un autor

1  Irepelusa  
Eliminar integrante de un autor musical

(Imagen 64. Eliminación de un integrante de un grupo)



Observamos que efectivamente se eliminó Irepelusa.

```
usuarios
Ver contenido
([{"id_autor": 1, "nombre_autor": "Motherflowers", "discografia": None, "integrantes": "Veztalone, Frank Lucas, Tayko"},),  
([{"id_autor": 5, "nombre_autor": "MUSE", "discografia": "Warner Records", "integrantes": "Matt Bellamy, Chris Wolstenholme, Dominic Howard"},),  
([{"id_autor": 6, "nombre_autor": "Caravan Palace", "discografia": "Le Plan Recordings", "integrantes": "Zoé Colotis, Arnaud Vial, Hugues Payen, Camille Raimondeau"},),  
([{"id_autor": 7, "nombre_autor": "MAMAMOO", "discografia": "RBW Entertainment", "integrantes": "Moonbyul, Solar, Wheein, Hwasa"},])
```

(Imagen 65. Visualización de los autores después de la eliminación)

Ahora eliminaremos el autor Motherflowers

#### Eliminar un autor musical

(Imagen 66. Eliminación de un autor musical)

```
grupos musicales
Ver contenido
([{"id_autor": 5, "nombre_autor": "MUSE", "discografia": "Warner Records", "integrantes": "Matt Bellamy, Chris Wolstenholme, Dominic Howard"},),  
([{"id_autor": 6, "nombre_autor": "Caravan Palace", "discografia": "Le Plan Recordings", "integrantes": "Zoé Colotis, Arnaud Vial, Hugues Payen, Camille Chapelière, Char Raimondeau"},),  
([{"id_autor": 7, "nombre_autor": "MAMAMOO", "discografia": "RBW Entertainment", "integrantes": "Moonbyul, Solar, Wheein, Hwasa"},])
```

(Imagen 67. Visualización de los autores después de la eliminación)



- Actualizar

Para actualizar elementos en la base de datos lo podemos hacer desde la pestaña Actualizar. Desde aquí podemos actualizar cualquier elemento en la base de datos. Podemos elegir que registró modificar de cualquier tabla, siempre que se indique el ID.

Actualizar el elemento que deseé en la base de datos de ULL Music

**Actualizar un usuario**

ID del usuario	nombre	apellidos	nombre de usuario
dis		correo electrónico	
<input type="button" value="Actualizar un usuario"/>			

**Actualizar una lista de canciones**

ID de la lista	nombre de la lista
descripción de la lista	
<input type="button" value="Actualizar una lista de canciones"/>	

**Actualizar un comentario**

ID del comentario	ID del usuario	ID de la canción
texto del comentario		
<input type="button" value="Actualizar un comentario"/>		

((Imagen 68. Apartado de actualizar de la API)

En los siguiente ejemplo podemos cambiar un género musical de nombre.

<input type="button" value="generos"/>
<input type="button" value="Ver contenido"/>
({'id_genero': 1, 'nombre_genero': 'Rock'},)
({'id_genero': 2, 'nombre_genero': 'Pop'},)
({'id_genero': 3, 'nombre_genero': 'Jazz'},)
({'id_genero': 4, 'nombre_genero': 'Metal'},)
({'id_genero': 5, 'nombre_genero': 'Clásica'},)
({'id_genero': 6, 'nombre_genero': 'Reggae'},)
({'id_genero': 7, 'nombre_genero': 'Salsa'},)
({'id_genero': 8, 'nombre_genero': 'Folk'},)
({'id_genero': 9, 'nombre_genero': 'Blues'},)
({'id_genero': 10, 'nombre_genero': 'Rap'},)

((Imagen 69. Visualización de los géneros musicales antes de modificar)



**Actualizar un género musical**

1	POP-ROCK-ESPAÑOL
<input type="button" value="Actualizar un género musical"/>	

(Imagen 70. Modificación de un género musical)

generos
<input type="button" value="Ver contenido"/>
({'id_genero': 1, 'nombre_genero': 'POP-ROCK-ESPAÑOL'},)
({'id_genero': 2, 'nombre_genero': 'Pop'},)
({'id_genero': 3, 'nombre_genero': 'Jazz'},)
({'id_genero': 4, 'nombre_genero': 'Metal'},)
({'id_genero': 5, 'nombre_genero': 'Clásica'},)
({'id_genero': 6, 'nombre_genero': 'Reggae'},)
({'id_genero': 7, 'nombre_genero': 'Salsa'},)
({'id_genero': 8, 'nombre_genero': 'Folk'},)
({'id_genero': 9, 'nombre_genero': 'Blues'},)
({'id_genero': 10, 'nombre_genero': 'Rap'},)

(Imagen 71. Visualización de los géneros musicales después de modificar)

## 9. Conclusiones

Después de haber realizado el trabajo, nos ha permitido profundizar más en lo realizado en las sesiones prácticas, sobre todo en las últimas de Disparadores y API REST. Además de servirnos como método para juntar todo lo aprendido, ya que hemos tenido que aplicar tanto el desarrollo y diseño conceptual de la propia base de datos, como los elementos técnicos para poder implementarla.

Por otro lado, sirve para darse cuenta del tiempo que conlleva realizar estos proyectos, y de las muchas variables y cosas a tener en cuenta para que sea robusta ante los fallos.

Además, en el propio proyecto, y aún más cuando se desarrolla la API, mientras vas trabajando te vas dando cuenta de las posibles mejoras, y que siempre puede ser aún mejor.