



UNIVERSIDAD DE CÓRDOBA  
ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

GRADO EN INGENIERÍA INFORMÁTICA  
MENCIÓN: COMPUTACIÓN  
CUARTO CURSO. PRIMER CUATRIMESTRE

BASES DE DATOS AVANZADAS

## Trabajo Fin de Prácticas

*Ventura Lucena Martínez*  
i72lumav@uco.es

Año académico 2021-2022  
Córdoba, 5 de enero de 2022

# Índice

Lista de Figuras	II
<b>1. Introducción</b>	<b>1</b>
<b>2. Antecedentes</b>	<b>2</b>
<b>3. Diseño de una base de datos relacional</b>	<b>3</b>
3.1. Descripción del problema, requisitos y reglas de negocio . . . . .	3
3.2. Modelo conceptual de la base de datos . . . . .	5
3.2.1. Comentarios extra sobre algunos tipos de entidad . . . . .	6
3.2.2. Comentarios extra sobre algunos tipos de interrelación . . . . .	6
3.3. Modelo relacional de la base de datos . . . . .	7
3.4. <i>Script</i> de creación de la base de datos . . . . .	8
3.5. Prueba y validación del diseño . . . . .	10
3.6. Uso de disparadores . . . . .	11
3.6.1. Disparador de auditoría . . . . .	11
3.6.2. Disparador de seguridad . . . . .	12
3.6.3. Disparador de restricción de dominio . . . . .	13
3.6.4. Disparador de restricción de integridad . . . . .	15
3.6.5. Disparador sobre alguna tabla cuya condición se satisfaga en función de la extensión de otra tabla . . . . .	16
<b>4. Bases de Datos objeto-relacional</b>	<b>16</b>
<b>5. Uso de <i>Middleware</i> de bases de datos</b>	<b>20</b>
5.1. Descripción del proyecto <i>Java</i> . . . . .	21
5.2. Pruebas sobre la base de datos . . . . .	21
5.2.1. Prueba de inserción . . . . .	22
5.2.2. Prueba de modificación . . . . .	23
5.2.3. Prueba de eliminación . . . . .	24
5.2.4. Prueba de consulta . . . . .	25
<b>A. Comentarios extra</b>	<b>27</b>

## Índice de figuras

1.	Estadísticas <i>Spotify</i> [7]. . . . .	2
2.	Modelo conceptual de la base de datos. . . . .	5
3.	Modelo relacional de la base de datos. . . . .	7
4.	Disparador de auditoría (1). . . . .	12
5.	Disparador de auditoría (2). . . . .	12
6.	Tablas de la base de dato en <i>phpMyAdmin</i> . . . . .	21
7.	Menú del programa. . . . .	22
8.	Operaciones sobre tablas. . . . .	22
9.	Operación de inserción (1). . . . .	23
10.	Operación de inserción (2). . . . .	23
11.	Operación de modificación (1). . . . .	24
12.	Operación de modificación (2). . . . .	24
13.	Operación de eliminación (1). . . . .	25
14.	Operación de eliminación (2). . . . .	25
15.	Operación de consulta. . . . .	26

# 1. Introducción

El presente informe explica el desarrollo de una base de datos basada en un problema del mundo real, en la que se aplican las técnicas vistas en la asignatura de *Bases de Datos Avanzadas*, de la Universidad de Córdoba. Concretamente, se verá una base de datos genérica a una aplicación de reproducción de música, tal y como nos ofrecen algunas empresas muy conocidas como *Spotify*, con la aplicación con el mismo nombre [1]; *Apple* con *Apple Music* [2] o Amazon con *Amazon Music* [3], pero con una funcionalidad añadida relacionada con la tecnología *blockchain* [4] y los *tokens* no fungibles (comúnmente conocidos como *NFTs* [5]), aunque sin indagar mucho en el tema.

En primer lugar, se presenta el problema acompañado de una serie de supuestos que sientan las bases del desarrollo, a la vez que permiten definir las restricciones del mismo. A continuación, se desarrolla un modelo conceptual y relacional de la base de datos, complementado con un *script* consistente en la creación e inserción de las tablas que satisfacen dichos modelos.

En segundo lugar, se encuentran los disparadores o *triggers*, que son procedimientos que controlan la integridad de los datos: disparadores de auditoría, de seguridad, de restricciones de dominio, de referencia y condicionales.

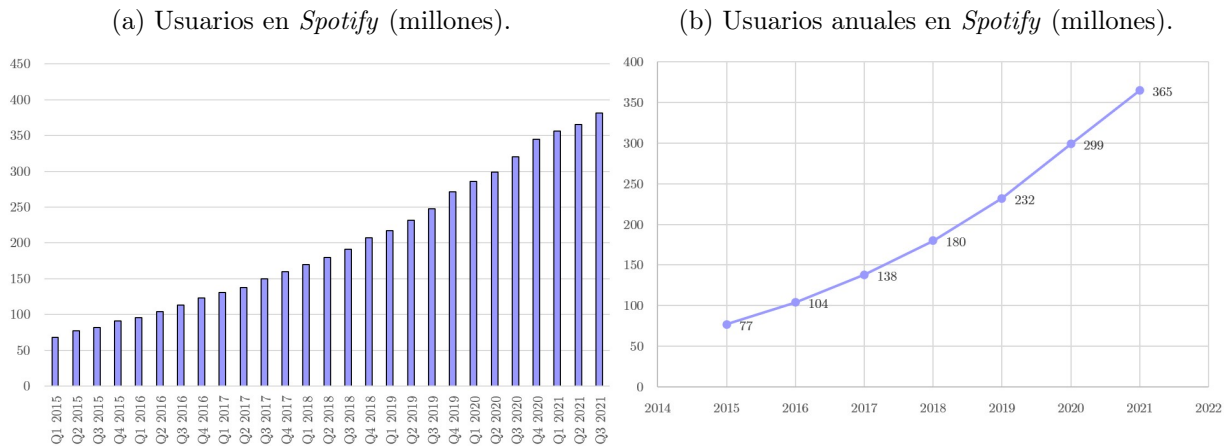
Acto seguido se muestra la implementación de tablas objeto y tablas anidadas, junto con el tipo de dato *varray*.

Por último, se hace uso de *middleware*, de tal forma que se pueda acceder a la base de datos desde entornos externos, pudiendo realizar operaciones sobre la misma. Para ello, se verá una pequeña implementación en el lenguaje *Java*.

## 2. Antecedentes

A lo largo de los años, el formato musical ha sido muy cambiante: cintas de radio *cassette*, cd's, formatos de reproducción como MP3 o el *streaming*<sup>1</sup> de contenido. En la industria musical, tal y como se ha comentado anteriormente, *Spotify* es una de las empresas más influyentes y destacables, cuyo crecimiento en los últimos años es cuanto menos remarcable:

Figura 1: Estadísticas *Spotify* [7].



Fuente: *BussinessofApps*.

Por otro lado, hay artistas que desean recompensar a sus seguidores de alguna forma: es aquí donde se abre paso el uso de los *NFTs*. Aunque inicialmente nacieron con el objetivo de verificar la unicidad de un objeto en el marco digital, su uso se ha ido y se está extendiendo con utilidades complementarias en diversos campos: mercado del arte, videojuegos, mundos virtuales, cine o música son los más conocidos.

<sup>1</sup>“El *streaming* es un tipo de tecnología multimedia que envía contenidos de vídeo y audio a su dispositivo conectado a Internet. Esto le permite acceder a contenidos (TV, películas, música, pódcast) en cualquier momento que lo desee, en un PC o un móvil, sin someterse a los horarios del proveedor”. [6]

### 3. Diseño de una base de datos relacional

#### 3.1. Descripción del problema, requisitos y reglas de negocio

Numerosos artistas han lanzado álbumes en formato *NFT*, a los que se les podría (si es que todavía no se ha implementado) añadir la funcionalidad de acceso exclusivo a cierto contenido, ya sea el álbum al completo o ciertos temas musicales del mismo. Esto se verifica en la *blockchain* del *NFT*, pero una empresa, de manera paralela, podría guardar los datos referentes a los usuarios y sus *NFTs* asociados para tener un acceso más rápido y centralizado a dicha información.

En este sentido, se creará una base de datos que almacene los datos de los usuarios de una aplicación de música, sus listas de reproducción creadas y sus *NFTs* asociados; y los artistas disponibles en la aplicación, con sus álbumes disponibles en el mercado musical y sus estilos musicales asociados.

Se considerarán los siguientes supuestos:

##### 1. Usuarios:

- a) Pueden disponer de ninguna o muchas listas de reproducción, mientras que una lista de reproducción es creada por uno, y sólo un usuario.
- b) Pueden tener en propiedad ninguno o numerosos *NFTs*.
- c) Los usuarios se identifican por su email, teniendo además una contraseña.

##### 2. Playlist:

- a) En una lista de reproducción se pueden añadir desde ninguna a numerosas canciones. Además, cada canción puede pertenecer a muchas listas de reproducción o a ninguna.
- b) Las listas de reproducción vienen identificadas por un título y por el usuario creador. Por otro lado, también se especifica una descripción y la duración de la misma, junto con el número de canciones que contiene.

##### 3. NFT:

- a) Puede estar asociado a un, y sólo un álbum de música, mientras que un álbum puede tener asociados desde ningún *NFT* a muchos.
- b) Viene identificado por la dirección de la *blockchain* y su identificador de *token*. Además, será necesario conocer la *blockchain* a la que pertenecen y el estándar del mismo.

##### 4. Álbum:

- a) Un álbum puede pertenecer a uno o más artistas, y este puede tener tanto una única canción como numerosas canciones, dependiendo del tipo del álbum. Los álbumes<sup>2</sup> se dividen en:
  - 1) **Single:** contienen como mínimo una canción y como máximo tres. Duración máxima de diez minutos.
  - 2) **EP:** contienen como mínimo cuatro canciones y como máximo seis. Duración máxima de treinta minutos.
  - 3) **LP:** contienen como mínimo seis canciones. Duración mínima de treinta minutos.
- b) Los álbumes vienen identificados por un código o identificador de álbum, además de verse caracterizados por la fecha de estreno, el título, la duración, el número de canciones y una descripción.

## 5. Canción:

- a) Una canción puede pertenecer a un, y sólo un álbum.
- b) Vienen identificadas por un código identificativo. Entre otras características, se destacan el título, la duración y el número de reproducciones.

## 6. Artista:

- a) Un artista puede tener desde ningún álbum a numerosos álbumes. Además, el artista tiene asociado como mínimo un estilo musical.
- b) Vienen identificados por un código identificativo, además de tener también nombre, descripción y posición en el ranking de la plataforma.

## 7. Estilos musicales:

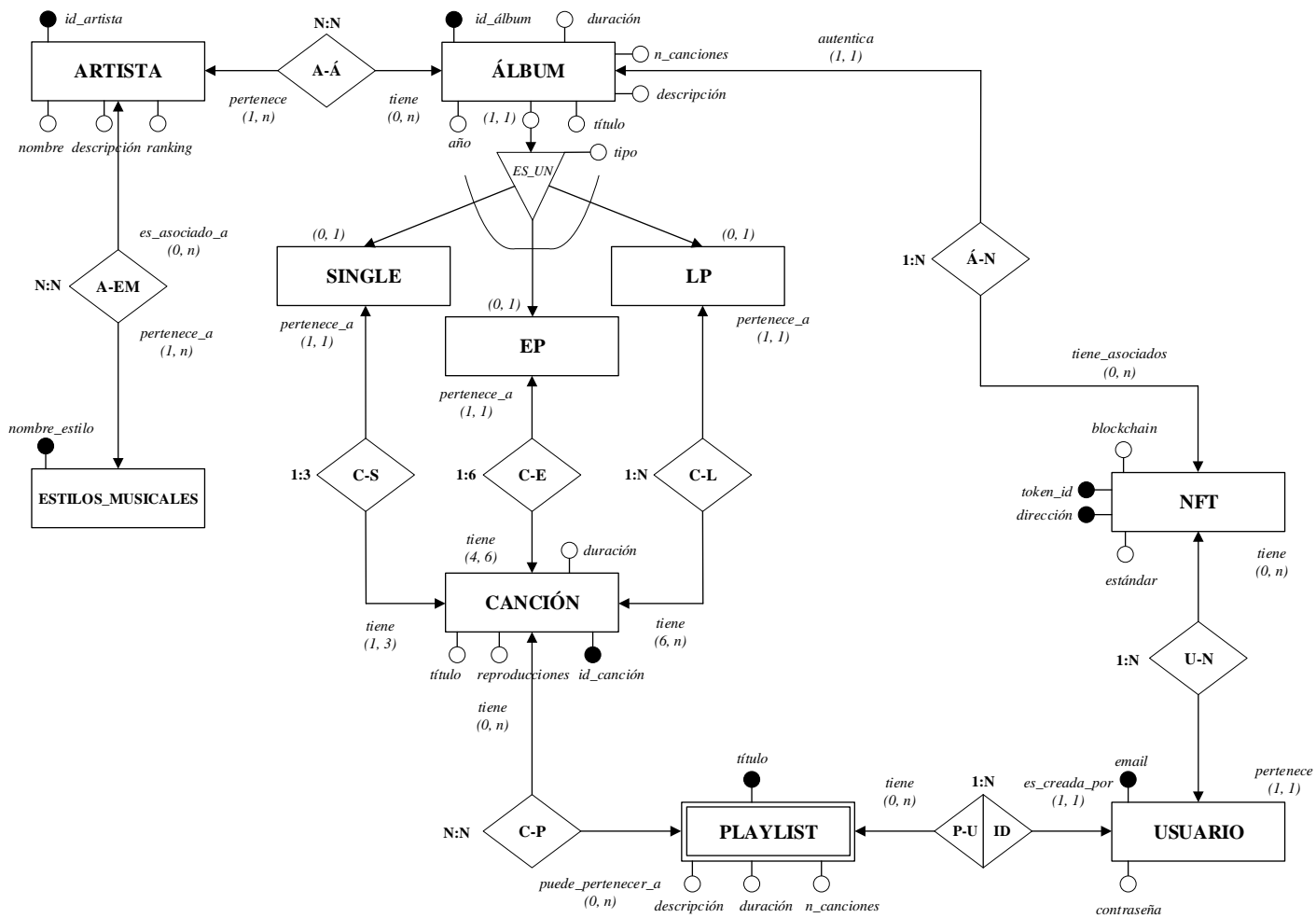
- a) Los estilos musicales puede no estar asociado a ningún artista, pero también puede estar asociado a muchos.
- b) Se identifican por el nombre del estilo, por ejemplo, *blues*.

---

<sup>2</sup>Dependiendo del tipo de álbum, estos se dividen en sencillos o *singles*, reproducciones extendidas o *Extended Play* (EP) y reproducciones de larga duración o *Long Play* (LP). Para ver qué condiciones debe cumplir cada uno de manera detallada, puede consultar las siguientes referencias: [8, 9, 10, 11]

### 3.2. Modelo conceptual de la base de datos

Figura 2: Modelo conceptual de la base de datos.





Algunos aspectos a tener en cuenta en el modelo conceptual de cara al modelo relacional, respecto a los tipos de entidad y a los tipos de interrelación son los siguientes:

### 3.2.1. Comentarios extra sobre algunos tipos de entidad

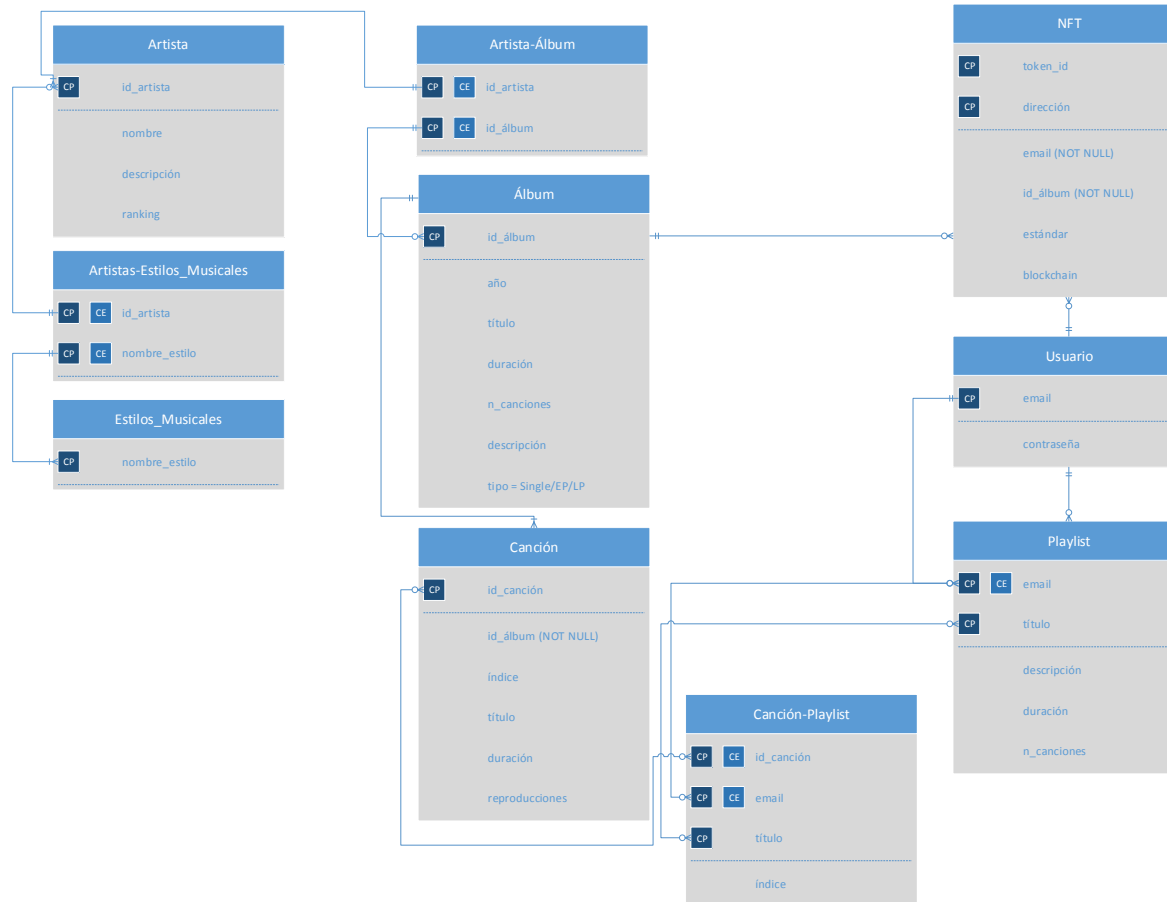
- **Tipo de entidad *Álbum*:** pertenece a una relación jerárquica total y exclusiva en la que interviene como supertipo de entidad. Los subtipos de entidades a los que afecta son *Single*, *EP* y *LP*, es decir, un álbum puede ser del tipo sencillo, reproducción extendida o reproducción de larga duración.
- **Tipo de entidad *Playlist*:** mantiene una debilidad por identificación respecto a usuario, es decir, es necesario un usuario para identificar una lista de reproducción creada. De esta manera se pueden crear listas de reproducción con el mismo nombre si, y sólo si, el usuario que las crea es distinto en cada lista.

### 3.2.2. Comentarios extra sobre algunos tipos de interrelación

- Se crearán nuevas tablas para los tipos de interrelaciones con cardinalidades N:N, cuya clave principal será la combinación de los tipos de entidad que interfieren en la interrelación.
- Se aplicará una eliminación de los subtipos de entidad en la interrelación jerárquica existente en el modelo:
  - Los atributos pertenecientes a los subtipos de entidad pasan a pertenecer al supertipo de entidad en caso de que existan.
  - El atributo *tipo* del supertipo de entidad solo podrá tomar uno de los siguientes valores: *Single*, *EP* o *LP*.

### 3.3. Modelo relacional de la base de datos

Figura 3: Modelo relacional de la base de datos.



### 3.4. *Script* de creación de la base de datos

```
drop table usuarios cascade constraints;
drop table playlist cascade constraints;
drop table nfts cascade constraints;
drop table albumes cascade constraints;
drop table artistas cascade constraints;
drop table estilos_musicales cascade constraints;
drop table canciones cascade constraints;

drop table artistas_em cascade constraints;
drop table artistas_albumes cascade constraints;
drop table canciones_playlist cascade constraints;

create table usuarios(
    email varchar2(32) primary key,
    contrasena varchar2(32) not null
);

create table playlist(
    titulo varchar2(32),
    email varchar2(32),
    n_canciones number(38) not null,
    duracion varchar2(32) not null,
    primary key (titulo, email) disable,
    constraint fk_email foreign key(email) references usuarios(email)
);

create table albumes(
    id_album number(16) primary key,
    ano number(16) not null,
    titulo varchar2(32) not null,
    duracion varchar2(32) not null,
    n_canciones number(16) not null,
    descripcion varchar2(2048),
    tipo varchar2(32) not null,
    constraint ck_tipo check(tipo in ('Single', 'EP', 'LP'))
);

create table nfts(
    token_id number(16),
    direccion varchar2(64),
    email varchar2(32) not null,
    id_album number(16) not null,
    estandar varchar2(32) not null,
    blockchain varchar2(32) not null,
    primary key (token_id, direccion) disable,
```

```

        constraint fk_nfts_id_album foreign key(id_album) references
        ↪ albums(id_album)
    );

create table canciones(
    id_cancion number(16) primary key,
    id_album number(16) not null,
    indice number(16) not null,
    titulo varchar2(32) not null,
    duracion varchar2(32) not null,
    reproducciones number(32),
    constraint fk_canciones_id_album foreign key(id_album) references
    ↪ albums(id_album)
);

create table artistas(
    id_artista number(16) primary key,
    nombre varchar2(32) not null,
    descripcion varchar2(2048),
    ranking number(16) not null
);

create table estilos_musicales(
    nombre_estilo varchar2(32) primary key
);

create table artistas_em(
    id_artista number(16),
    nombre_estilo varchar2(32),
    primary key (id_artista, nombre_estilo) disable,
    constraint fk_artistas_em_id_artista foreign key(id_artista) references
    ↪ artistas(id_artista),
    constraint fk_artistas_em_nombre_estilo foreign key(nombre_estilo)
    ↪ references estilos_musicales(nombre_estilo)
);

create table artistas_albumes(
    id_artista number(16),
    id_album number(16),
    primary key (id_artista, id_album) disable,
    constraint fk_artistas_albumes_id_artista foreign key(id_artista)
    ↪ references artistas(id_artista),
    constraint fk_artistas_albumes_id_album foreign key(id_album) references
    ↪ albums(id_album)
);

create table canciones_playlist(
    id_cancion number(16),

```

```

        email varchar2(32),
        titulo varchar2(32),
        indice number(16) not null,
        primary key (id_cancion, email, titulo) disable,
        constraint fk_canciones_p_id_cancion foreign key(id_cancion) references
            ↪ canciones(id_cancion),
        constraint fk_canciones_p_email foreign key(email) references
            ↪ usuarios(email)
    );

```

### 3.5. Prueba y validación del diseño

A continuación, se muestran algunas consultas realizadas en la base de datos:

```

select * from usuarios;
select * from playlist;
select * from nfts;
select * from albumes;
select * from artistas;
select * from estilos_musicales;
select * from canciones;
select * from artistas_em;
select * from artistas_albumes;
select * from canciones_playlist;

-- Selección de todas las características de las tablas nfts y álbumes.
select *
from nfts, albumes
where nfts.id_album = albumes.id_album;

-- Selección de los temas musicales y sus álbumes.
select canciones.titulo "Tema musical", albumes.titulo "Álbum"
from canciones, albumes
where canciones.id_album = albumes.id_album;

-- Selección de todos los estilos de música que abarca un artista.
select artistas.nombre "Artista", artistas_em.nombre_estilo "Estilo musical"
from artistas, artistas_em
where artistas.id_artista = artistas_em.id_artista;

-- Selección de los artistas, el título del álbum y el estándar de la cadena de
↪ bloques a la que pertenece el NFT del álbum.
select unique(artistas.nombre) "Artista", albumes.titulo "Álbum", nfts.estandar
↪ "Estándar del NFT"
from artistas, artistas_albumes, albumes, nfts
where artistas.id_artista = artistas_albumes.id_artista and

```

```

artistas_albumes.id_album = albumes.id_album and
albumes.id_album = nfts.id_album;

-- Selección de los temas musicales, de sus álbumes correspondientes y el nombre
-- de la playlist a la que pertenecen de todas las playlists existentes.
select albumes.titulo "Álbum", canciones.titulo "Tema musical",
       canciones_playlist.titulo "Título playlist"
from albumes, canciones, canciones_playlist
where canciones.id_album = albumes.id_album and
canciones.id_cancion = canciones_playlist.id_cancion
order by canciones.id_cancion;

-- Selección de las direcciones de los NFTs de los usuarios.
select nfts.direccion "Dirección NFT", usuarios.email "Usuario"
from nfts, usuarios
where nfts.email = usuarios.email;

```

## 3.6. Uso de disparadores

Se presentará un ejemplo para cada uno de los disparadores o *triggers* solicitados.

### 3.6.1. Disparador de auditoría

Informa de las modificaciones de uno o varios atributos de una tabla. En este caso se ha creado el disparador *auditoria\_nft*, para tener en todo momento información de modificación sobre la tabla *nfts*.

```

-- Trigger de auditoría.
create or replace trigger auditoria_nft
after delete or update or insert on nfts
for each row
declare

begin
    if inserting then
        insert into audit_table
        values ('Valores insertados: [TOKEN_ID = ' ||
               ↳ :new.token_id || ']' [DIRECCION = ' || :new.direccion
               ↳ || ']' [EMAIL = ' || :new.email || ']' [ID_ALBUM = ' ||
               ↳ :new.id_album || ']' [ESTANDAR = ' || :new.estandar ||
               ↳ ']' [BLOCKCHAIN = ' || :new.blockchain || ']', 'nfts');
    end if;

    if deleting then
        insert into audit_table

```

```

values ('Valores insertados: [TOKEN_ID = ' ||
↪ :old.token_id || ']' [DIRECCION = ' || :old.direccion
↪ || ']' [EMAIL = ' || :old.email || ']' [ID_ALBUM = '
↪ || :old.id_album || ']' [ESTANDAR = ' ||
↪ :old.estandar || ']' [BLOCKCHAIN = ' ||
↪ :old.blockchain || ']', 'nfts');

end if;

end;

```

Figura 4: Disparador de auditoría (1).

TOKEN_ID	DIRECCION	EMAIL	ID_ALBUM	ESTANDAR	BLOCKCHAIN
1	1 8A875218EF3A3FFCABE9B26FAAC51C1F60A31052DCE1332C57B09E4009B0EB5	admin@bbddaa.com	7 ERC-721	ethereum	
2	2 9CBB7192AB86163AC7E9735188AF7E4D9E822793DD959BA7FE42E85CEF89103	admin@bbddaa.com	7 ERC-721	ethereum	
3	3 BD4327CD0C8531785658EBCF87664D51CD76BB2159AA27FDD059057EAC2EBD6E	admin@bbddaa.com	7 ERC-721	ethereum	
4	4 B64CA1B9E3210F8273036BC87450F78871B202C4DAF9A561AF0614401DBA66E2	admin@bbddaa.com	7 ERC-721	ethereum	
5	5 CE4BB5E0E112826E4F9E21B396F60049D6E217B75DA08A6AEE06054040C7C952	admin@bbddaa.com	7 ERC-721	ethereum	
6	1 38708F733FC00E4EF5166C57F4B39E57FCB023DAF74E3F002DEA87ED6E8B3BAC	prauvegagriffei-4787@bbddaa.com	13 metaplex	solana	
7	2 45CA9A55A36DFF64E945FB251735C73D641690AA5F8CA9FE764F0A178FD4DB75	prauvegagriffei-4787@bbddaa.com	13 metaplex	solana	
8	3 5D387DF7BF87BE9BED9259020E1A7C5BA31558569E2C50ED285FE9BDB556F766	prauvegagriffei-4787@bbddaa.com	13 metaplex	solana	
9	4 8A4EB5F952D08AD796017FBB217203F57C9E0ADC6636A76DBA2F01FF6C67CA0	prauvegagriffei-4787@bbddaa.com	13 metaplex	solana	
10	5 492FEB201BF2196C4EBE8E0C223931386805E14FACDB9EB9F9DF488C1763A0A5	prauvegagriffei-4787@bbddaa.com	13 metaplex	solana	
11	1 09583F737139EC577D156B9ABA0451FA1C3AF23A3D7DE53E5C6C3003890904DA	gruleixatreija-8928@bbddaa.com	15 BEP-721	binance smart chain	
12	2 090A9D59B51633E466D3783361484424205BB31983DC0850D48F6C60530380E1	gruleixatreija-8928@bbddaa.com	15 BEP-721	binance smart chain	
13	3 6B3FA89B18250E3C08071A662949153B72A5879220708E718262EEBD1E812F8B	gruleixatreija-8928@bbddaa.com	15 BEP-721	binance smart chain	
14	4 8B11EFE3ECD5B8849E0F782E0331EC9AE68325C2CBCE646D6A7138F33667493	gruleixatreija-8928@bbddaa.com	15 BEP-721	binance smart chain	
15	5 1AE8C01064590B38139F53F5EC40E45C34980FEEEDB54B5520EE0451FB326CA4	gruleixatreija-8928@bbddaa.com	15 BEP-721	binance smart chain	
16	1 0E46ACA4DBC5A030E188F997E988BAC29F8A27A37BDFD92931F4C4376BB9CC52	admin@bbddaa.com	8 CIP-25	cardano	

Figura 5: Disparador de auditoría (2).

DATOS	TABLA
1 Valores insertados: [TOKEN_ID = 1] [DIRECCION = 0E46ACA4DBC5A030E188F997E988BAC29F8A27A37BDFD92931F4C4376BB9CC52] [EMAIL = admin@bbddaa.com] [ID_ALBUM = 8] [ESTANDAR = CIP-25] [BLOCKCHAIN = cardano] nfts	nfts

### 3.6.2. Disparador de seguridad

Impide realizar actualizaciones de la base de datos en base a algún criterio establecido. Dado que los *NFTs* tienen direcciones únicas, no puede darse que exista un *NFT* con la misma dirección que otro. Esto, como se ha dicho anteriormente, se verifica en la *blockchain*, pero también se debe verificar en la base de datos:

```

-- Trigger de seguridad.
create or replace trigger seguridad_nft
before update or insert on nfts
for each row
declare
    cursor c is (select nfts.direccion from nfts);
    flag NUMBER := 0;

```

```

begin
    for row in c loop
        if :new.direccion = row.direccion then
            flag := 1;
        end if;
    end loop;

    if flag = 0 then
        insert into nfts values (:new.token_id, :new.direccion, :new.email,
        ↪ :new.id_album, :new.estandar, :new.blockchain);
    else
        Raise_application_error(-20001, 'La dirección del NFT ya existe.');
```

Al introducir los siguientes valores en la tabla *NFTs* intentando (malintencionadamente) modificar la dirección de un *NFT* ya existente, se observa como el *trigger* impide que dicho valor se añada, debido a que ya está en uso por otro usuario:

```

insert into nfts values (1,
↪ '0E46ACA4DBC5A030E188F997E988BAC29F8A27A37BDFD92931F4C4376BB9CC52',
↪ 'admin@bbddaa.com', 8, 'CIP-25', 'cardano');
```

La salida obtenida es la siguiente:

```

Error que empieza en la línea: 1 del comando :
insert into nfts values (1,
↪ '0E46ACA4DBC5A030E188F997E988BAC29F8A27A37BDFD92931F4C4376BB9CC52',
↪ 'admin@bbddaa.com', 8, 'CIP-25', 'cardano')
Informe de error -
Error SQL: ORA-20001: La dirección del NFT ya existe.
ORA-06512: en "I72LUMAV.SEGURIDAD_NFT", línea 17
ORA-04088: error durante la ejecución del disparador 'I72LUMAV.SEGURIDAD_NFT'
```

### 3.6.3. Disparador de restricción de dominio

Restrige un dominio existente en la base de datos. Nótese que en la figura 2 existe una interrelación jerárquica en la que el supertipo de entidad puede tomar los valores *Single*, *EP* o *LP* dadas unas características de un álbum. Para ello, se ha implementado la siguiente funcionalidad que hace se satisfagan los criterios necesarios para una inserción de dicho tipo:

```

-- Trigger de dominio.
create or replace trigger dominio_albumes
```



```

before update or insert on albums
for each row
begin
    if :new.tipo = 'Single' and :new.n_canciones > 3 then
        Raise_application_error(-20001, 'Si el álbum es un "Single",
        ↪ la cantidad máxima de canciones es [3].');
        rollback;
    end if;

    if :new.tipo = 'EP' and (:new.n_canciones > 6 or :new.n_canciones < 4)
    ↪ then
        Raise_application_error(-20001, 'Si el álbum es un "EP", la
        ↪ cantidad mínima de canciones es [4] y la máxima es [6].');
        rollback;
    end if;

    if :new.tipo = 'LP' and :new.n_canciones < 6 then
        Raise_application_error(-20001, 'Si el álbum es un "LP", la
        ↪ cantidad mínima de canciones es [6].');
        rollback;
    end if;
end;

```

Al introducir un álbum del tipo *LP* como otro tipo:

```

insert into albums values (17, 1989, '...But Seriously', '59 minutos', 12,
↪ '...But Seriously es el nombre del cuarto album de estudio del cantante,
↪ compositor y baterista britanico Phil Collins. El album fue publicado el 7
↪ de noviembre de 1989 a traves de la discografica Virgin Records y Atlantic
↪ Records.', 'EP');

```

El disparador no lo permite, lanzando el correspondiente error:

Error que empieza en la línea: 1 del comando :

```

insert into albums values (17, 1989, '...But Seriously', '59 minutos', 12,
↪ '...But Seriously es el nombre del cuarto album de estudio del cantante,
↪ compositor y baterista britanico Phil Collins. El album fue publicado el 7
↪ de noviembre de 1989 a traves de la discografica Virgin Records y Atlantic
↪ Records.', 'EP')

```

Informe de error -

```

Error SQL: ORA-20001: Si el álbum es un "EP", la cantidad mínima de canciones es
↪ [4] y la máxima es [6].

```

ORA-06512: en "I72LUMAV.DOMINIO\_ALBUMES", línea 7

ORA-04088: error durante la ejecución del disparador 'I72LUMAV.DOMINIO\_ALBUMES'

### 3.6.4. Disparador de restricción de integridad

Sustituye una restricción de integridad de referencia existente en la base de datos. En este caso, el disparador comprueba que antes de una inserción en la tabla *Artistas\_Albumes*, exista la referencia a su respectivo artista en la tabla *Artistas* y la referencia a su respectivo álbum en la tabla *Álbumes*:

```
-- Trigger de integridad.
create or replace trigger i_artistas_albumes
before insert or update on artistas_albumes
for each row
declare
    cursor c1 is (select id_artista from artistas);
    cursor c2 is (select id_album from albumes);
    flag_c1 number := 0;
    flag_c2 number := 0;
begin
    -- Comprobar que el dato introducido no es null para una clave primaria:
    if :new.id_artista is null then
        Raise_application_error(-20001, 'ERROR DE INTEGRIDAD: Es
        ↪ necesario el identificador del artista.');
```

```
    end if;

    if :new.id_album is null then
        Raise_application_error(-20001, 'ERROR DE INTEGRIDAD: Es
        ↪ necesario el identificador del álbum.');
```

```
    end if;

    -- Comprobar si existe el dato introducido en la base de datos:
    for row in c1 loop
        if :new.id_artista = row.id_artista then
            flag_c1 := 1;
        end if;
    end loop;

    for row in c2 loop
        if :new.id_album = row.id_album then
            flag_c2 := 1;
        end if;
    end loop;

    -- Si existe, no deja introducirlo:
    if flag_c1 = 0 then
        Raise_application_error(-20001, 'ERROR DE INTEGRIDAD: El artista
        ↪ no existe en la base de datos.');
```

```
        rollback;
    end if;
```

```

    if flag_c2 = 0 then
        Raise_application_error(-20001, 'ERROR DE INTEGRIDAD: El álbum
        ↪ no existe en la base de datos.');
```

```

        rollback;
    end if;
end;
```

Al introducir unos datos que todavía no están definidos en dichas tablas:

```
insert into artistas_albumes values (8, 17);
```

Se obtiene la siguiente salida:

```

Error que empieza en la línea: 1 del comando :
insert into artistas_albumes values (8, 17)
Informe de error -
Error SQL: ORA-20001: ERROR DE INTEGRIDAD: El artista no existe en la base de
↪ datos.
ORA-06512: en "I72LUMAV.I_ARTISTAS_ALBUMES", línea 31
ORA-04088: error durante la ejecución del disparador
↪ 'I72LUMAV.I_ARTISTAS_ALBUMES'
```

### 3.6.5. Disparador sobre alguna tabla cuya condición se satisfaga en función de la extensión de otra tabla

La condición de este disparador se ha visto también satisfecha por el último implementado en la sección 3.6.4, al no permitir la inserción si previamente otras tablas no cumplen con las condiciones necesarias.

## 4. Bases de Datos objeto-relacional

El objetivo es el uso de objetos con *Oracle* mediante la definición y manipulación de tablas de objetos y de tablas anidadas.

Se ha modificado la tabla *Álbumes*, de tal manera que ahora interactúe con objetos. Además, se le han añadido los campos *Reparto*, que hace referencia a una tabla donde se almacenan los datos de los participantes del álbum<sup>3</sup>; e *Intrumentos* de tipo *varray*, donde se almacena una lista de los instrumentos que aparecen.

```

/
create type reparto_t as object(
```

---

<sup>3</sup>Por simplicidad, se ha añadido únicamente el campo *nombre*.

```

    nombre varchar2(1024),

    member function getnombre return varchar2,
    member procedure display_reparto (self in out nocopy reparto_t)
);
/
-- Creo el tipo.
create type album_reparto_tab as table of reparto_t;
/
create type instrumentos_t as varray(10) of varchar2(64);
/
create table albumes(
    id_album number(16) primary key,
    ano number(16) not null,
    titulo varchar2(32) not null,
    duracion varchar2(32) not null,
    n_canciones number(16) not null,
    descripcion varchar2(2048),
    tipo varchar2(32) not null,
    reparto album_reparto_tab,
    instrumentos instrumentos_t,
    constraint ck_tipo check(tipo in ('Single', 'EP', 'LP'))
)
nested table reparto store as reparto_album;

```

Los datos de la tabla han sido modificados acorde a los cambios realizados en su estructura. De nuevo, por simplicidad se han modificado tres filas, dejando el resto con el valor de *null*:

```

insert into i72lumav.albumes (id_album, titulo, duracion, ano, n_canciones,
↪ tipo, descripcion, reparto, instrumentos) values (1, 'Dire Straits', '41
↪ minutos', 1978, 9, 'LP', 'Dire Straits es el album homonimo de la banda
↪ britanica Dire Straits, lanzado en 1978. Marca el debut de la banda en el
↪ mundo musical y uno de los mayores exitos del grupo con el sencillo -Sultans
↪ of Swing-.\nElaborado a partir de una serie de maquetas que el grupo habia
↪ conseguido grabar de su propio bolsillo, incorporaba practicamente todo el
↪ repertorio de la banda. Excluidos del album quedaron -Move It Away-, -Real
↪ Girl-, -Me and My Friends- y una version del tema de Chuck Berry -Nadine-.
↪ Durante la promocion del album, la banda interpretaria la cancion -What s
↪ The Matter, Baby?-, escrita conjuntamente por Mark y David Knopfler, y
↪ posteriormente publicada en el album de 1995 Live at the BBC.\nLa imagen de
↪ la portada esta extraida de una pintura de Chuck Loyola.\nDire Straits fue
↪ remasterizado y reeditado junto al resto del catalogo musical de Dire
↪ Straits el 19 de septiembre de 2000.?', album_reparto_tab(reparto_t('Mark
↪ Knopfler, David Knopfler, John Illsley y Pick Withers')),
↪ instrumentos_t('Guitarra', 'Bajo', 'Bateria'));

```

```

insert into i72lumav.albumes (id_album, titulo, duracion, ano, n_canciones,
→ tipo, descripcion, reparto, instrumentos) values (4, 'Journeyman', '56
→ minutos', 1989, 12, 'LP', 'Journeyman es el undecimo album de estudio del
→ musico britanico Eric Clapton, publicado por la compania discografica
→ Reprise Records en 1989.\nEl album fue anunciado como un retorno a la forma
→ de Clapton, que habia luchado durante mediados de la decada de 1980 con su
→ adiccion al alcohol. Parte del album tiene un sonido electronico influido
→ por la escena del rock de la decada, aunque tambien incluye canciones blues
→ como -Before You Accuse Me-, -Running on Faith- y -Hard Times-. Clapton
→ obtuvo un notable exito con -Bad Love-, que gano el Grammy a la mejor
→ interpretacion vocal de rock masculina y alcanzo el puesto uno en la lista
→ Mainstream Rock Tracks. Otro sencillo, -Pretending-, tambien llego al puesto
→ numero uno el ano anterior, manteniendose en lo mas alto de la lista durante
→ cinco semanas.\nAunque Journeyman obtuvo solo un exito comercial moderado,
→ alcanzando el puesto dieciseis en la lista estadounidense Billboard 200, se
→ convirtio en su primer disco de estudio en alcanzar el estatus de doble
→ platino al vender mas de dos millones de copias en los Estados Unidos. Fue
→ tambien mencionado por el propio musico como uno de sus trabajos
→ favoritos.?', album_reparto_tab(reparto_t('Eric Clapton, Nathan East, Phil
→ Collins, Robert Cray, Phil Palmer, Alan Clark, Greg Phillinganes, George
→ Harrison, Daryl Hall y Tessa Niles')), instrumentos_t('Guitarra', 'Bajo',
→ 'Bateria', 'Saxofon', 'Trompeta', 'Piano'));

insert into i72lumav.albumes (id_album, titulo, duracion, ano, n_canciones,
→ tipo, descripcion, reparto, instrumentos) values (8, 'On Every Street', '1
→ hora', 1991, 12, 'LP', 'On Every Street es el sexto y ultimo album de la
→ banda britanica Dire Straits, lanzado en 1991. Tras la gira llevada a cabo
→ como promocion del anterior album de estudio, Brothers In Arms, el grupo
→ llevo a cabo la misma formula en la elaboracion de su nuevo disco: un
→ sexteto basico, junto a una larga lista de musicos de estudio que apoyarian
→ en la grabacion. A estas alturas, Dire Straits, lejos de la banda
→ cohesionada que fue en los anos 70, se habia transformado en una marca a
→ traves de la cual Mark Knopfler, rodeado de una serie de musicos que mas o
→ menos colaboraban de forma asidua, publicaba sus canciones. Mark Knopfler se
→ encontraba cansado del estilo que la gente esperaba de Dire Straits tras su
→ exito en los 80 y en este album trato de experimentar con otros estilos como
→ el country o el blues. El album fue recibido con division de opiniones y
→ unas ventas medianas, si bien hasta la fecha se contabilizan en torno a los
→ 8 millones. Esto empujo a Knopfler a lanzar una carrera en solitario al
→ margen de la banda para buscar una mayor libertad artistica.',
→ album_reparto_tab(reparto_t('Mark Knopfler, John Illsley, Alan Clark, Guy
→ Fletcher, Danny Cummings, Paul Franklin, Vince Gill, Phil Palmer, Jeff
→ Porcaro y Chris White')), instrumentos_t('Guitarra', 'Bajo', 'Bateria',
→ 'Teclado', 'Pedal Steel Guitar', 'Saxofon'));

```

Para ver algunos datos referentes a los objetos, se han implementado algunas funcionalidades para ver los instrumentos de cada álbum y el reparto:

```

-- Funcionalidad para ver los instrumentos de los álbumes:
set serveroutput on;

declare
    instrumento instrumentos_t;
    cursor c is (select titulo, instrumentos from albumes where instrumentos is
        ↪ not null);

begin
    for i in c loop
        dbms_output.put_line(i.titulo);
        dbms_output.put_line('-----');
        instrumento := i.instrumentos;
        for j in instrumento.first..instrumento.last loop
            dbms_output.put_line(instrumento(j));
        end loop;
        dbms_output.put_line(' ');
    end loop;
end;

-- Método para ver el reparto:
create type reparto_t as object(
    nombre varchar2(1024),

    member function getnombre return varchar2,
    member procedure display_reparto (self in out nocopy reparto_t)
);

create type body reparto_t as
    member function getnombre return varchar2 is
        begin
            return nombre;
        end;
end;

```

Procedimiento PL/SQL terminado correctamente.

Dire Straits

-----

Guitarra

Bajo

Bateria

Journeyman

-----

Guitarra

Bajo

Bateria

Saxofon  
Trompeta  
Piano

On Every Street

-----

Guitarra  
Bajo  
Bateria  
Teclado  
Pedal Steel Guitar  
Saxofon

## 5. Uso de *Middleware* de bases de datos

En esta última sección, se ha realizado un acceso a la base de datos desde un entorno externo para la manipulación de los datos. Concretamente, se ha realizado un programa de prueba en *Java* que permite acceder, insertar, consultar, modificar y eliminar elementos de la base de datos<sup>4</sup>.

Antes de describir cómo se ha implementado el programa, es importante mencionar que, debido a que no se ha conseguido acceder a la base de datos de *Oracle* de la Universidad de Córdoba, se han realizado estas operaciones sobre la base de datos de *phpMyAdmin* de la Universidad de Córdoba. **Para ello, es obligatorio estar conectado a la VPN de la UCO.** Esto ha derivado en las siguientes modificaciones<sup>5</sup> de los *scripts* de generación de tablas:

1. Se ha modificado los tipos de variable *varchar2* por *varchar*.
2. Se han modificado los tipos de variable *number* por *integer*.

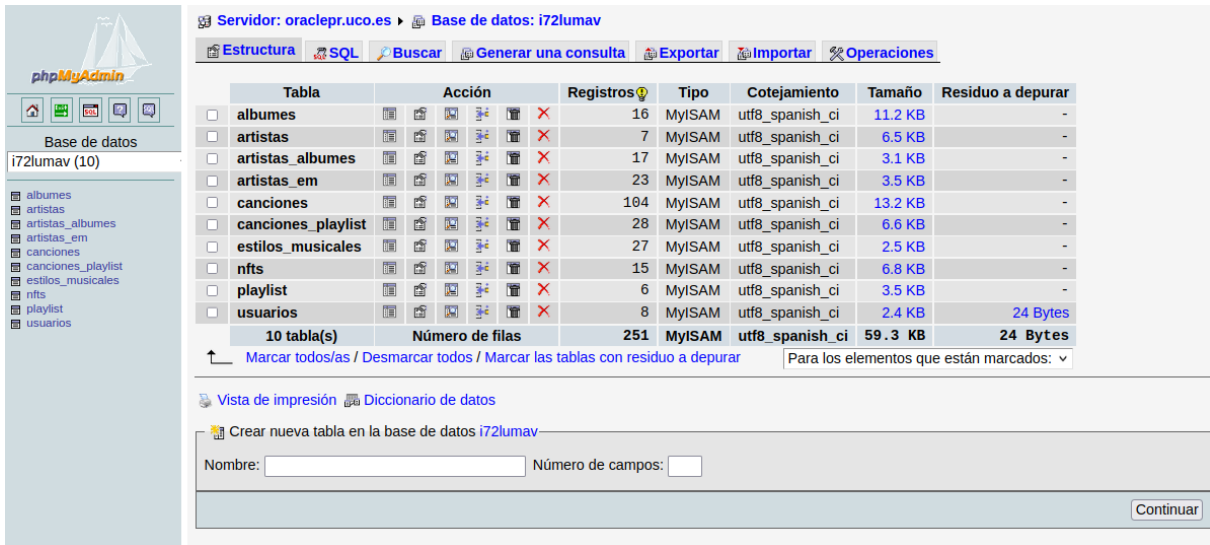
Una vez realizadas las modificaciones sintácticas para introducir comandos *SQL* en *phpMyAdmin*, se han creado las tablas del modelo:

---

<sup>4</sup>Para simplificar el proceso, se ha realizado una prueba con la tabla *Usuarios*, dado que las funcionalidades son similares en el resto de tablas.

<sup>5</sup>Pueden consultarse los cambios en el archivo *table\_generation\_phpmyadmin.sql*.

Figura 6: Tablas de la base de dato en *phpMyAdmin*.



The screenshot shows the phpMyAdmin interface for a database named 'i72lumav'. The left sidebar lists the database and its tables. The main area displays a table with columns: Tabla, Acción, Registros, Tipo, Cotejamiento, Tamaño, and Residuo a depurar. The tables listed are: albums, artistas, artistas\_albumes, artistas\_em, canciones, canciones\_playlist, estilos\_musicales, nfts, playlist, and usuarios. The 'usuarios' table has 8 records and a size of 2.4 KB. The total for all tables is 251 records and 59.3 KB.

Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
albums		16	MyISAM	utf8_spanish_ci	11.2 KB	-
artistas		7	MyISAM	utf8_spanish_ci	6.5 KB	-
artistas_albumes		17	MyISAM	utf8_spanish_ci	3.1 KB	-
artistas_em		23	MyISAM	utf8_spanish_ci	3.5 KB	-
canciones		104	MyISAM	utf8_spanish_ci	13.2 KB	-
canciones_playlist		28	MyISAM	utf8_spanish_ci	6.6 KB	-
estilos_musicales		27	MyISAM	utf8_spanish_ci	2.5 KB	-
nfts		15	MyISAM	utf8_spanish_ci	6.8 KB	-
playlist		6	MyISAM	utf8_spanish_ci	3.5 KB	-
usuarios		8	MyISAM	utf8_spanish_ci	2.4 KB	24 Bytes
10 tabla(s)	Número de filas	251	MyISAM	utf8_spanish_ci	59.3 KB	24 Bytes

## 5.1. Descripción del proyecto *Java*

El programa se divide en las siguientes secciones<sup>6</sup>:

- ***App.java***: programa principal que permite realizar la operaciones anteriormente comentadas sobre la tabla *Usuarios*.
- ***conexion\_base\_datos.java***: permite crear la conexión y desconexión entre el programa *Java* y la base de datos mencionada. Para que funcione esta conexión, será necesario preinstalar *mysql-connector* de *Java* [12].
- ***usuario\_Bean.java***: clase que representa un usuario de la base de datos.
- ***usuario\_DAO.java***: clase que permite realizar el intercambio de información entre el objeto *usuario* y el usuario de la base de datos.

## 5.2. Pruebas sobre la base de datos

Una vez ejecutado el programa, se da opción al usuario a elegir sobre qué tabla desea operar. Dado que la prueba es sobre la tabla de usuarios, esta es la única operativa:

<sup>6</sup>Puede consultarse el código en la sección *src/java*.



Figura 7: Menú del programa.

```
ventura@ventura:~/Ingenieria/Cuarto/Advanced_Data_Bases/src/java/advanced_data_bases$ /usr/bin/env /opt/jdk/jdk1.8.0_271/bin/java -cp /tmp/cp_61ckg659r9h3a47gfeil854zr.jar App

Middleware de prueba para la ejecución de sentencias SQL desde Java.
-----

Menú:
1. Operaciones sobre usuarios.
2. Resto de operaciones (No implementado: Salida del programa).
```

Cuando se selecciona la tabla sobre la que operar, se le ofrece al usuario qué operación desea realizar:

Figura 8: Operaciones sobre tablas.

```
ventura@ventura:~/Ingenieria/Cuarto/Advanced_Data_Bases/src/java/advanced_data_bases$ /usr/bin/env /opt/jdk/jdk1.8.0_271/bin/java -cp /tmp/cp_61ckg659r9h3a47gfeil854zr.jar App

Middleware de prueba para la ejecución de sentencias SQL desde Java.
-----

Menú:
1. Operaciones sobre usuarios.
2. Resto de operaciones (No implementado: Salida del programa).

1

1. Insertar usuario.
2. Modificar usuario.
3. Borrar usuario.
4. Ver usuario.
[]
```

### 5.2.1. Prueba de inserción

Para la prueba de inserción se ha utilizado la siguiente sentencia:

```
String INSERT = "insert into usuarios (email, contrasena) values (?, ?);";
```

Al introducir los datos en los campos requeridos, se ejecuta la instrucción sobre la base de datos:

Figura 9: Operación de inserción (1).

```
Menú:
1. Operaciones sobre usuarios.
2. Resto de operaciones (No implementado: Salida del programa).

1

1. Insertar usuario.
2. Modificar usuario.
3. Borrar usuario.
4. Ver usuario.
1

Email de usuario:
test@uco.es

Contraseña de usuario:
1234
```

Figura 10: Operación de inserción (2).

Base de datos: i72lumav (10)

Mostrando registros 0 - 8 (9 total, La consulta tardó 0.0001 seg)

consulta SQL:

```
SELECT *
FROM 'usuarios'
LIMIT 0, 30
```

[ Editar ] [ Explicar el SQL ] [ Crear código PHP ] [ Actualizar ]

Mostrar: 30 filas empezando de 0

en modo horizontal y repetir los encabezados cada 100 celdas

Organizar según la clave: Ninguna

	email	contrasena
<input type="checkbox"/>	admin@bbddaa.com	admin1
<input type="checkbox"/>	lixepasuha-1174@bbddaa.com	JVSSthSeLs\$
<input type="checkbox"/>	brixatroudeppa-2336@bbddaa.com	me2P\$4vHglV\$
<input type="checkbox"/>	ceiffiwafreza-6533@bbddaa.com	oQeArNw@0Lfe
<input type="checkbox"/>	prauvegagriffei-4787@bbddaa.com	3RoUusqEm2*u
<input type="checkbox"/>	goibriliquousoi-6568@bbddaa.com	^VUYmznbs6fz
<input type="checkbox"/>	higuquauyeyou-7236@bbddaa.com	G*Vv1Pz0Ap%4
<input type="checkbox"/>	gruleixatreija-8928@bbddaa.com	hA*xDcyJVMca
<input type="checkbox"/>	test@uco.es	1234

Marcar todos/as / Desmarcar todos Para los elementos que están marcados:

Mostrar: 30 filas empezando de 0

en modo horizontal y repetir los encabezados cada 100 celdas

Insertar nueva fila Vista de impresión Previsualización para imprimir (documento completo) Exportar

### 5.2.2. Prueba de modificación

Para la prueba de modificación se ha utilizado la siguiente sentencia:

```
String UPDATE = "update usuarios set email = ?, contrasena = ? where email =  
↪ ?;";
```

Al introducir los datos en los campos requeridos, se ejecuta la instrucción sobre la base de datos:

Figura 11: Operación de modificación (1).

```
2. Resto de operaciones (No implementado: Salida del programa).

1

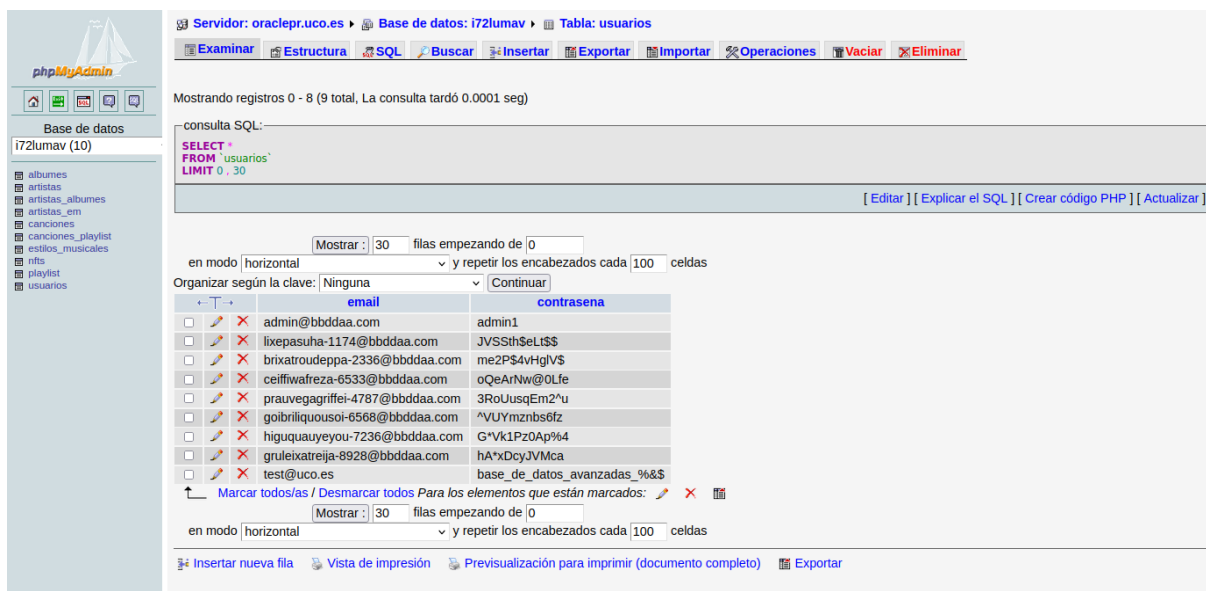
1. Insertar usuario.
2. Modificar usuario.
3. Borrar usuario.
4. Ver usuario.
2

Email de usuario a modificar:
test@uco.es

Conexión establecida con éxito.

Nueva contraseña:
base de datos avanzadas %&$
```

Figura 12: Operación de modificación (2).



The screenshot shows the phpMyAdmin interface for the 'i72lumav' database. The 'usuarios' table is selected, and the SQL query 'SELECT \* FROM usuarios LIMIT 0, 30' is executed. The results are displayed in a table with columns 'email' and 'contrasena'.

	email	contrasena
<input type="checkbox"/>	admin@bbddaa.com	admin1
<input type="checkbox"/>	lixepasuha-1174@bbddaa.com	JVSSth\$eLt\$
<input type="checkbox"/>	brixatroudeppa-2336@bbddaa.com	me2P\$4vHgIV\$
<input type="checkbox"/>	ceiffiwafreza-6533@bbddaa.com	oQeArNw@0Lfe
<input type="checkbox"/>	prauvegagriffei-4787@bbddaa.com	3RoUusqEm2*u
<input type="checkbox"/>	goibriliquousoi-6568@bbddaa.com	^VUYmznbs6fz
<input type="checkbox"/>	higuquauyeyou-7236@bbddaa.com	G*Vx1Pz0Ap%4
<input type="checkbox"/>	gruleixatrejia-8928@bbddaa.com	hA*xDcyJVMca
<input type="checkbox"/>	test@uco.es	base_de_datos_avanzadas_%&\$

### 5.2.3. Prueba de eliminación

Para la prueba de eliminación se ha utilizado la siguiente sentencia:

```
String UPDATE = "delete from usuarios where email = ?;";
```

Al introducir los datos en los campos requeridos, se ejecuta la instrucción sobre la base de datos:

Figura 13: Operación de eliminación (1).

```

1. Insertar usuario.
2. Modificar usuario.
3. Borrar usuario.
4. Ver usuario.
3

Email de usuario a eliminar:
test@uco.es

Conexión establecida con éxito.
¿Está seguro de que desea eliminar el usuario test@uco.es? [y/n]:
y

Desconexión establecida con éxito.

Eliminación realizada con éxito.

```

Figura 14: Operación de eliminación (2).

The screenshot shows the phpMyAdmin interface for the database 'i72lumav'. The left sidebar lists the database and its tables. The main area displays a table structure view with columns: Tabla, Acción, Registros, Tipo, Cotejamiento, Tamaño, and Residuo a depurar. The tables listed are: albums, artistas, artistas\_albumes, artistas\_em, canciones, canciones\_playlist, estilos\_musicales, nfts, playlist, and usuarios. The 'usuarios' table is highlighted, showing it has 8 records, is MyISAM type, and has a size of 2.4 KB. Below the table list, there are links for 'Vista de impresión' and 'Diccionario de datos'. At the bottom, there is a form to 'Crear nueva tabla en la base de datos i72lumav' with fields for 'Nombre:' and 'Número de campos:'.

Tabla	Acción	Registros	Tipo	Cotejamiento	Tamaño	Residuo a depurar
albums		16	MyISAM	utf8_spanish_ci	11.2 KB	-
artistas		7	MyISAM	utf8_spanish_ci	6.5 KB	-
artistas_albumes		17	MyISAM	utf8_spanish_ci	3.1 KB	-
artistas_em		23	MyISAM	utf8_spanish_ci	3.5 KB	-
canciones		104	MyISAM	utf8_spanish_ci	13.2 KB	-
canciones_playlist		28	MyISAM	utf8_spanish_ci	6.6 KB	-
estilos_musicales		27	MyISAM	utf8_spanish_ci	2.5 KB	-
nfts		15	MyISAM	utf8_spanish_ci	6.8 KB	-
playlist		6	MyISAM	utf8_spanish_ci	3.5 KB	-
usuarios		8	MyISAM	utf8_spanish_ci	2.4 KB	24 Bytes
<b>10 tabla(s)</b>	<b>Número de filas</b>	<b>251</b>	<b>MyISAM</b>	<b>utf8_spanish_ci</b>	<b>59.3 KB</b>	<b>24 Bytes</b>

[↑](#) [Marcar todos/as](#) / [Desmarcar todos](#) / [Marcar las tablas con residuo a depurar](#)

[Vista de impresión](#) [Diccionario de datos](#)

Crear nueva tabla en la base de datos i72lumav

Nombre:  Número de campos:

Continuar

#### 5.2.4. Prueba de consulta

Para la prueba de consulta se ha utilizado la siguiente sentencia:

```
String SELECT = "select * from usuarios where usuarios.email = ?";
```

Al introducir los datos en los campos requeridos, se ejecuta la instrucción sobre la base de datos:

Figura 15: Operación de consulta.

```
1. Insertar usuario.  
2. Modificar usuario.  
3. Borrar usuario.  
4. Ver usuario.  
4  
  
Email de usuario a buscar:  
test@uco.es  
  
Conexión establecida con éxito.  
  
Email: test@uco.es  
Contraseña: base_datos_avanzadas_%&$  
  
Desconexión establecida con éxito.
```

## A. Comentarios extra

- La metodología seguida en la tabla *Usuarios* es extrapolable al resto de tablas de la base de datos.
- Sería interesante implementar una interfaz gráfica de usuario en *Java* con las librerías pertinentes para facilitar al usuario las operaciones sobre la base de datos desde el *middleware*.
- Se ha encontrado cierta complejidad con las tablas anidadas en la base de datos objeto-relacional, concretamente en el acceso a la información almacenada en tablas anidadas.
- Se ha utilizado la base de datos de *phpMyAdmin* de la UCO en el puerto 3306 en lugar de la base de datos de *Oracle* de la UCO en el puerto 1521, debido a que no se conseguía establecer la conexión.

## Referencias

- [1] Spotify España. *Spotify*. Accessed 28 December 2021. URL: <https://www.spotify.com/es/>.
- [2] Apple Inc. España. *Apple Music*. Accessed 28 December 2021. URL: <https://www.apple.com/es/apple-music/>.
- [3] Amazon Music España. *Amazon Music*. Accessed 28 December 2021. URL: <https://music.amazon.es/>.
- [4] Wikipedia contributors. *Blockchain* — *Wikipedia, The Free Encyclopedia*. <https://en.wikipedia.org/w/index.php?title=Blockchain&oldid=1061603596>. [Online; accessed 28-December-2021]. 2021.
- [5] Wikipedia contributors. *Non-fungible token* — *Wikipedia, The Free Encyclopedia*. [https://en.wikipedia.org/w/index.php?title=Non-fungible\\_token&oldid=1062306657](https://en.wikipedia.org/w/index.php?title=Non-fungible_token&oldid=1062306657). [Online; accessed 28-December-2021]. 2021.
- [6] Avast. *¿Qué es el streaming y cómo funciona?* Accessed 29 December 2021. URL: <https://www.avast.com/es-es/c-what-is-streaming>.
- [7] Mansoor Iqbal. “Spotify Revenue and Usage Statistics (2021)”. En: *BusinessofApps* (2021). Accessed 29 December 2021. URL: <https://www.businessofapps.com/data/spotify-statistics/>.
- [8] Wikipedia. *Extended play* — *Wikipedia, La enciclopedia libre*. [Internet; descargado 28-diciembre-2021]. 2021. URL: [https://es.wikipedia.org/w/index.php?title=Extended\\_play&oldid=137467749](https://es.wikipedia.org/w/index.php?title=Extended_play&oldid=137467749).
- [9] *¿Qué es un single o sencillo?* 2020. URL: <https://latinwmg.com/que-es-un-single-o-sencillo/>.
- [10] *¿Qué es un EP?* 2021. URL: <https://musicodiy.cdbaby.com/que-es-un-ep/>.
- [11] *¿Cuál es la diferencia entre un single, un EP y un álbum?* URL: <https://support.landr.com/hc/es/articles/115009568227--Cu%C3%A1l-es-la-diferencia-entre-un-single-un-EP-y-un-%C3%A1lbum->.
- [12] Oracle. *MySQL Community Downloads*. Accessed 5 January 2022. URL: <https://dev.mysql.com/downloads/connector/j/>.