

## “CollectMedia”



<https://github.com/franciscobertolo/proyecto-base-datos>

**Nome Alumno/a: Francisco Bértolo Prado**

***Nome completo e apelidos***

**Curso: 1º DAM**

**Materia: *Bases de Datos – Proyecto Final 24/25***

## Contido

1. Introducción	1
2. Descripción del Problema / Requisitos	2
3. Modelo Conceptual	2
4. Modelo Relacional	2
5. Proceso de Normalización	2
6. Script de Creación de la Base de Datos	2
7. Carga de Datos Inicial	2
8. Funciones y Procedimientos Almacenados	2
9. Triggers	2
10. Consultas SQL	2
11. Casos de Prueba y Simulación	2
12. Resultados y Verificación	2
13. Capturas de Pantalla (opcional)	2
14. Conclusiones y Mejoras Futuras	2
15. Enlace al Repositorio en GitHub	3

## 1. Introducción

*"Soy el fundador de CollectMedia, una aplicación que permite a los usuarios organizar y gestionar toda su colección de entretenimiento digital y física en un solo lugar. Nuestra app ayuda a los amantes del entretenimiento a catalogar sus videojuegos, películas, series, libros y música, manteniendo un registro detallado de su colección personal."*

## 2. Descripción del Problema / Requisitos

Problema:

Los coleccionistas de entretenimiento tienen dificultad para organizar y gestionar sus colecciones multimedia (físicas y digitales) de manera centralizada.

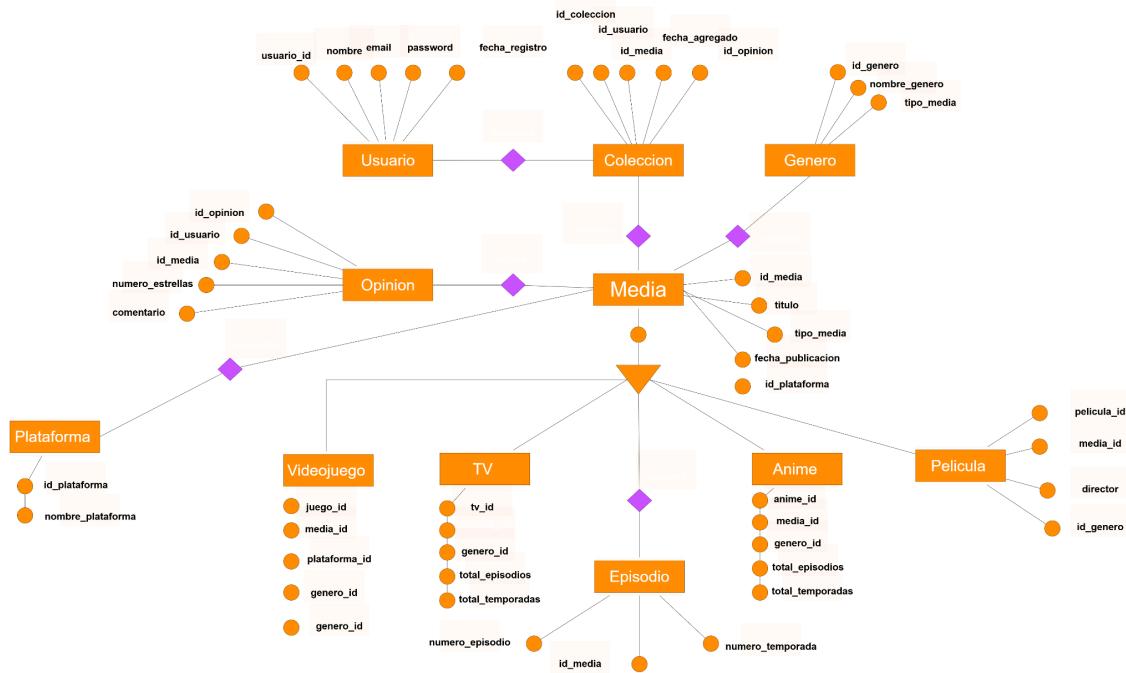
No existe una herramienta integral que permita catalogar diferentes tipos de medios (videojuegos, películas, series, libros, música) en un solo lugar.

Los usuarios necesitan mantener un registro detallado de sus colecciones personales.

Requisitos:

- Gestión de usuarios
- Registro e inicio de sesión
- Perfiles personalizables
- Gestión de colecciones
- Catalogar múltiples tipos de media (videojuegos, películas, series, anime, etc.)
- Funcionalidades sociales
- Seguimiento de episodios (series/anime)
- Registro de fechas de publicación

### 3. Modelo Conceptual



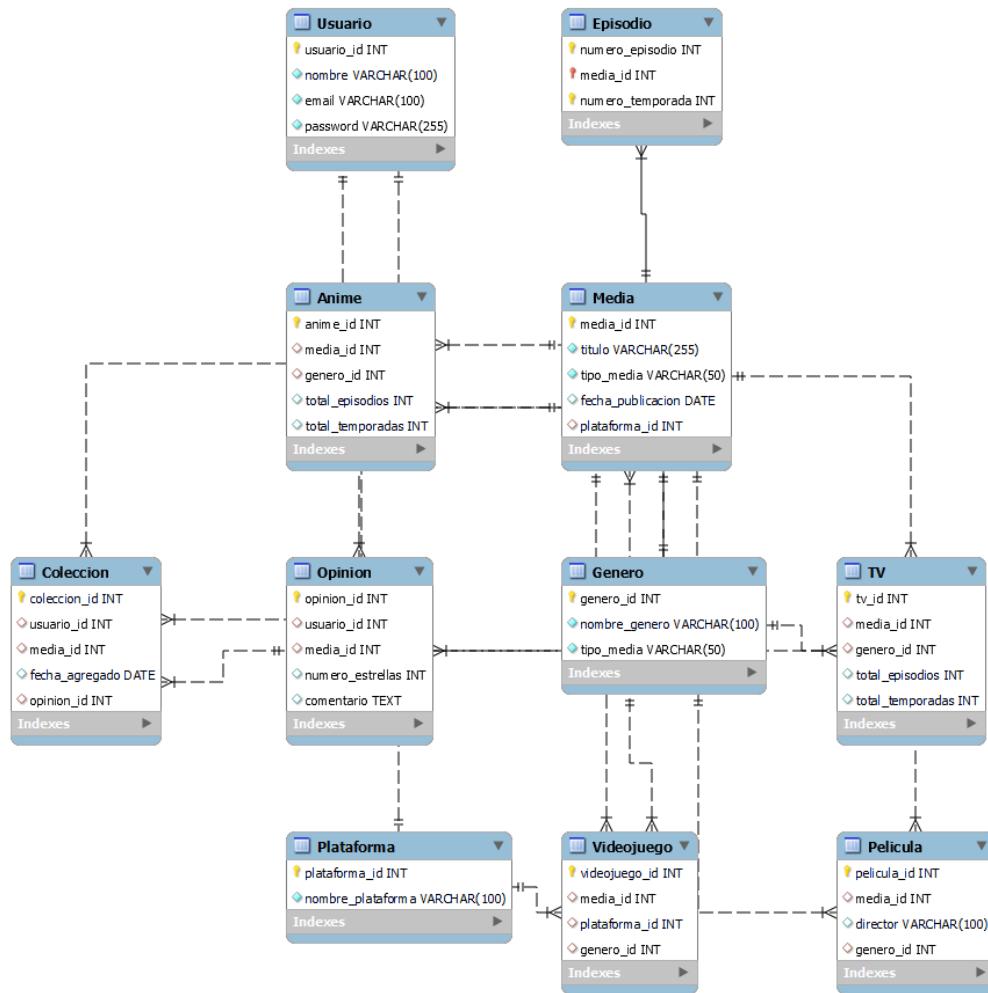
Un usuario puede tener muchos elementos en su colección, pero cada elemento en la colección pertenece a un solo usuario.

Cada elemento en la colección de un usuario se relaciona con un solo elemento de media.

Cada elemento de media puede estar en múltiples colecciones.

Cada media se relaciona con un solo registro en su tabla de especialización correspondiente.

## 4. Modelo Relacional



## 5. Proceso de Normalización

**Media:** Se relaciona con Película (1:1), Se relaciona con Anime (1:1), Se relaciona con TV (1:1), Se relaciona con Videojuego (1:1), Se relaciona con Género (N:M), Se relaciona con Colección (N:M).

**Usuario:** Se relaciona con Colección (1:N), Se relaciona con Opinion (1:N)

**TV y Anime:** Se relacionan con Episodio (1:N)

**Plataforma:** Se relaciona con Videojuego (N:M) Se relaciona con Media (N:M)

**Opinion:** Se relaciona con Media (1:N) Se relaciona con Usuario (1:N)

**Colección:** Se relaciona con Usuario (1:N) Se relaciona con Media (N:M).

Para normalizar el modelo, se ha revisado que todas las tablas tengan una clave primaria, después se han eliminado las dependencias parciales y transitivas, a esto se llegó colocando los campos "género" y "episodio" en una tabla separada y utilizar sus ids en las otras tablas.

## 6. Script de Creación de la Base de Datos

```
-- Crear Base Datos
DROP DATABASE IF EXISTS CollectMedia;
CREATE DATABASE IF NOT EXISTS CollectMedia;
USE CollectMedia;

-- Tabla Usuario
CREATE TABLE Usuario (
    usuario_id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

El resto del Script está colgado en la carpeta raíz del directorio de github.

## 7. Carga de Datos Inicial

```
225 •    -- ;;Inserts!!
226      -- Plataformas
227      INSERT INTO Plataforma (nombre_plataforma) VALUES
228          ('Netflix'),
229          ('PlayStation 5'),
230          ('Steam'),
231          ('HBO Max'),
232          ('Crunchyroll');
233
234      -- Géneros
235 •    INSERT INTO Genero (nombre_genero, tipo_media) VALUES
236          ('Acción', 'videojuego'),
237          ('Drama', 'tv'),
238          ('Aventura', 'anime'),
239          ('Ciencia Ficción', 'pelicula'),
240          ('RPG', 'videojuego');
241
242      -- Media
243      -- Videojuegos
244 •    INSERT INTO Media (titulo, tipo_media, fecha_publicacion, plataforma_id) VALUES
245          ('The Last of Us', 'videojuego', '2013-06-14', 2),
246          ('God of War', 'videojuego', '2018-04-20', 2),
247          ('Elden Ring', 'videojuego', '2022-02-25', 3);
```

El resto del Script está colgado en la carpeta raíz del directorio de github.

## 8. Funciones y Procedimientos Almacenados

```

112      -- !!!Funciones!!!
113      -- a) Calcular el promedio de calificaciones para un medio:
114
115      DELIMITER //
116  •  CREATE FUNCTION calcular_promedio_calificaciones(media_id_param INT)
117      RETURNS DECIMAL(3,2)
118      DETERMINISTIC
119      BEGIN
120          DECLARE avg_rating DECIMAL(3,2);
121          SELECT AVG(numero_estrellas) INTO avg_rating
122          FROM Opinion
123          WHERE media_id = media_id_param;
124          RETURN avg_rating;
125      END;
126      //
127      DELIMITER ;

```

El resto del Script está colgado en la carpeta raíz del directorio de github.

## 9. Triggers

```

-- !!!Triggers!!!
-- a) Verificar que la fecha de publicación no sea futura al insertar un nuevo medio:

DELIMITER //
CREATE TRIGGER check_fecha_publicacion
BEFORE INSERT ON Media
FOR EACH ROW
BEGIN
    IF NEW.fecha_publicacion > CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'La fecha de publicación no puede ser futura';
    END IF;
END;
//
DELIMITER ;

```

El resto del Script está colgado en la carpeta raíz del directorio de github.

## 10. Consultas SQL

```

288  -- 1. Mostrar todos
289 •   SELECT m.titulo, m.tipo_media, p.nombre_plataforma
290   FROM Media m
291   JOIN Plataforma p ON m.plataforma_id = p.plataforma_id
292   ORDER BY m.tipo_media, m.titulo;
293

```

The screenshot shows a database query results grid with three columns: 'titulo', 'tipo\_media', and 'nombre\_plataforma'. The data includes various media titles like 'Attack on Titan', 'Death Note', 'One Piece', etc., categorized by type ('anime', 'pelicula', 'tv', 'videojuego') and associated with platforms like 'Crunchyroll', 'Netflix', 'HBO Max', or 'Steam'.

	titulo	tipo_media	nombre_plataforma
▶	Attack on Titan	anime	Crunchyroll
	Death Note	anime	Crunchyroll
	One Piece	anime	Crunchyroll
	Inception	pelicula	Netflix
	Interstellar	pelicula	Netflix
	The Matrix	pelicula	Netflix
	Breaking Bad	tv	Netflix
	Game of Thrones	tv	HBO Max
	The Wire	tv	HBO Max
	Elden Ring	videojuego	Steam
	God of War	videojuego	PlayStation 5
	The Last of Us	videojuego	PlayStation 5

```

300  -- 3. Contar cuántos medios hay de cada tipo
301 •   SELECT tipo_media, COUNT(*) as total
302   FROM Media
303   GROUP BY tipo_media
304   ORDER BY total DESC;
305
306  -- 4. Mostrar todos los videojuegos con su género y platafor

```

The screenshot shows a database query results grid with two columns: 'tipo\_media' and 'total'. It displays four categories: 'videojuego', 'tv', 'anime', and 'pelicula', each having a count of 3.

	tipo_media	total
▶	videojuego	3
	tv	3
	anime	3
	pelicula	3

El resto de las pruebas está dentro del script colgado en la carpeta raíz del directorio de github.

## 11. Casos de Prueba y Simulación

Prueba de creación de tablas.

Action Output			Message
#	Time	Action	
✓ 9	15:54:04	CREATE TABLE Coleccion ( colección_id INT PRIMARY KEY AUTO_INCREMENT, usuario_id INT, media_id INT, fecha_agregado DATE)	0 row(s) affected
✓ 10	15:54:04	CREATE TABLE Videojuego ( videojuego_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, plataforma_id INT, genero_id INT)	0 row(s) affected
✓ 11	15:54:04	CREATE TABLE TV ( tv_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, genero_id INT, total_episodios INT, total_temporadas INT)	0 row(s) affected
✓ 12	15:54:04	CREATE TABLE Anime ( anime_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, genero_id INT, total_episodios INT, total_temporadas INT)	0 row(s) affected
✓ 13	15:54:05	CREATE TABLE Pelicula ( pelicula_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, director VARCHAR(100), genero_id INT)	0 row(s) affected
✓ 14	15:54:05	CREATE TABLE Episodio ( numero_episodio INT, media_id INT, numero_temporada INT, PRIMARY KEY (numero_episodio, media_id))	0 row(s) affected

Prueba de inserción de datos.

#	Time	Action	Message
✓ 19	16:08:34	INSERT INTO Media (título, tipo_media, fecha_publicación, plataforma_id) VALUES ('Breaking Bad', 'tv', '2008-01-20', 1), ('Game of Thrones', 'tv', '2011-04-17', 2), ('Attack on Titan', 'anime', '2013-04-07', 5), ('Death Note', 'anime', '2006-07-01', 6), ('Inception', 'película', '2010-07-16', 1), ('The Matrix', 'película', '1999-03-31', 9), ('The Last of Us', 'videojuego', '2013-06-14', 4), ('Lana y Lilly Wachowski', 'película', '2012-05-18', 12), ('Christopher Nolan', 'director', '2010-07-16', 10), ('Christopher Nolan', 'director', '2012-05-18', 11)	0 row(s) affected
✓ 20	16:08:34	INSERT INTO TV (media_id, genero_id, total_episodios, total_temporadas) VALUES (4, 2, 62, 5), (5, 2, 73, 8), (6, 2, 60, 5)	0 row(s) affected
✓ 21	16:08:34	INSERT INTO Media (título, tipo_media, fecha_publicación, plataforma_id) VALUES ('Attack on Titan', 'anime', '2013-04-07', 5), ('Death Note', 'anime', '2006-07-01', 6), ('Inception', 'película', '2010-07-16', 1), ('The Matrix', 'película', '1999-03-31', 9), ('The Last of Us', 'videojuego', '2013-06-14', 4), ('Lana y Lilly Wachowski', 'película', '2012-05-18', 12), ('Christopher Nolan', 'director', '2010-07-16', 10), ('Christopher Nolan', 'director', '2012-05-18', 11)	0 row(s) affected
✓ 22	16:08:34	INSERT INTO Anime (media_id, genero_id, total_episodios, total_temporadas) VALUES (7, 3, 75, 4), (8, 3, 37, 1), (9, 3, 1000, 20)	0 row(s) affected
✓ 23	16:08:34	INSERT INTO Media (título, tipo_media, fecha_publicación, plataforma_id) VALUES ('Inception', 'película', '2010-07-16', 1), ('The Matrix', 'película', '1999-03-31', 9), ('The Last of Us', 'videojuego', '2013-06-14', 4), ('Lana y Lilly Wachowski', 'película', '2012-05-18', 12), ('Christopher Nolan', 'director', '2010-07-16', 10), ('Christopher Nolan', 'director', '2012-05-18', 11)	0 row(s) affected
✓ 24	16:08:34	INSERT INTO Pelicula (media_id, director, genero_id) VALUES (10, 'Christopher Nolan', 4), (11, 'Lana y Lilly Wachowski', 4), (12, 'Christopher Nolan', 9)	0 row(s) affected

Prueba de creación de procedures, functions y triggers.

✓ 26	16:09:19	CREATE FUNCTION total_medios_usuario(usuario_id_param INT) RETURNS INT DETERMINISTIC BEGIN DECLARE total INT; SELECT ...	0 row(s) affected
✓ 27	16:09:19	CREATE PROCEDURE agregar_a_colección( IN p_usuario_id INT, IN p_media_id INT, IN p_fecha_agregado DATE ) BEGIN INSERT ...	0 row(s) affected
✓ 28	16:09:19	CREATE PROCEDURE actualizar_media( IN p_media_id INT, IN p_título VARCHAR(255), IN p_fecha_publicación DATE, IN p_platafor... ) BEGIN UPDATE ...	0 row(s) affected
✓ 29	16:09:19	CREATE TRIGGER check_fecha_publicación BEFORE INSERT ON Media FOR EACH ROW BEGIN IF NEW.fecha_publicación > CURDATE... END IF;	0 row(s) affected
✓ 30	16:09:19	CREATE TRIGGER actualizar_fecha_colección AFTER INSERT ON Opinion FOR EACH ROW BEGIN UPDATE Colección SET fecha_agr... END IF;	0 row(s) affected
✓ 31	16:09:19	CREATE TRIGGER check_numero_estrellas BEFORE INSERT ON Opinion FOR EACH ROW BEGIN IF NEW.numero_estrellas < 1 OR NEW... END IF;	0 row(s) affected

## 12. Resultados y Verificación

328	-- función
329	• <b>SELECT título, calcular_promedio_calificaciones(1) as promedio</b>
330	FROM Media
331	WHERE media_id = 1;
332	

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
titulo	promedio			
The Last of Us	4.50			

Se puede verificar que los triggers, funciones y procedimientos comprueban los datos correctamente.

El resto de las pruebas se encuentran dentro del script antes mencionado.

## 13. Capturas de Pantalla (opcional)

```

1   -- Crear Base Datos
2 •  DROP DATABASE IF EXISTS CollectMedia;
3 •  CREATE DATABASE IF NOT EXISTS CollectMedia;
4 •  USE CollectMedia;
5
6   -- Tabla Usuario
7 •  CREATE TABLE Usuario (
8     usuario_id INT PRIMARY KEY AUTO_INCREMENT,
9     nombre VARCHAR(100) NOT NULL,
10    email VARCHAR(100) NOT NULL UNIQUE,
11    password VARCHAR(255) NOT NULL
12 );
13
14   -- Tabla Plataforma
15 •  CREATE TABLE Plataforma (
16     plataforma_id INT PRIMARY KEY AUTO_INCREMENT,
17     nombre_plataforma VARCHAR(100) NOT NULL
18 );
19
20   -- Tabla Genero
21 •  CREATE TABLE Genero (
22     genero_id INT PRIMARY KEY AUTO_INCREMENT,
23     nombre_genero VARCHAR(100) NOT NULL,
24     tipo_media VARCHAR(50) NOT NULL
25 );
26
27   -- Tabla Media
28 •  CREATE TABLE Media (
29     media_id INT PRIMARY KEY AUTO_INCREMENT,
30     titulo VARCHAR(255) NOT NULL,

```

Output :

Action Output	#	Time	Action	Message
5 13:03:32	5	13:03:32	CREATE TABLE Opinion ( opinion_id INT PRIMARY KEY AUTO_INCREMENT, usuario_id INT, media_id INT, numero_estrellas INT CH...	0 row(s) affected
6 13:03:32	6	13:03:32	CREATE TABLE Coleccion ( colección_id INT PRIMARY KEY AUTO_INCREMENT, usuario_id INT, media_id INT, fecha_agregado D...	0 row(s) affected
7 13:03:32	7	13:03:32	CREATE TABLE Videojuego ( videojuego_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, plataforma_id INT, genero_id INT...	0 row(s) affected
8 13:03:32	8	13:03:32	CREATE TABLE TV ( tv_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, genero_id INT, total_episodios INT, total_tempo...	0 row(s) affected
9 13:03:32	9	13:03:32	CREATE TABLE Anime ( anime_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, genero_id INT, total_episodios INT, total...	0 row(s) affected
10 13:03:33	10	13:03:33	CREATE TABLE Pelicula ( película_id INT PRIMARY KEY AUTO_INCREMENT, media_id INT, director VARCHAR(100), genero_id INT...	0 row(s) affected

## 14. Conclusiones y Mejoras Futuras

Se ha creado una base de datos robusta y genérica que se puede ocupar para diferentes proyectos. Para mejoras a futuro, se propondrá agregar campos de imagen y utilizar acceso a APIs como The Movie Database de donde se podrán recoger colecciones de artículos directamente en la base de datos con sus respectivas imágenes.

## 15. Enlace al Repositorio en GitHub

<https://github.com/franciscobertolo/proyecto-base-datos>