Music Library Manager

Aplicación web CRUD para la **gestión de biblioteca musical** con **datos relacionados**.

- **Frontend:** Angular
- **Backend:** Node + Express (API REST)
- **Base de datos:** MongoDB
- **Sin autenticación**

1. Requisitos del Proyecto

Estructura de carpetas

`/API`

`/WEB`

Puertos obligatorios

Servicio	Puerto
Angular	4300
API Node + Express	3400
MongoDB	27017

Conexión MongoDB (OBLIGATORIA)

`mongodb://usemongo:secretoarab@localhost:27017/music_library_db`

- Usuario: `mongoadmin`
- Password: `secret`
- No se aceptan variables de entorno alternativas
- No se aceptan otros nombres de BD

2. Base de Datos (MongoDB)

Colecciones obligatorias

albums

```
{
  _id: ObjectId,
  title: String,         // required
  artist: String,        // required
  year: Number,
  genre: String,
  coverUrl: String
}
```

songs

```
{
  _id: ObjectId,
  title: String,         // required
  duration: Number,     // segundos
  rating: Number,       // 0-5
  albumId: ObjectId,    // required (referencia a albums)
  listened: Boolean
}
```

Relación obligatoria:
Cada canción **pertenece a un álbum** mediante `albumId`.

3. Backend (Node + Express)

Endpoints REST

Álbumes

```
GET    /api/albums

GET    /api/albums/:id

POST   /api/albums

PUT    /api/albums/:id

DELETE /api/albums/:id
```

Canciones

```
GET    /api/songs

GET    /api/songs/:id

POST   /api/songs

PUT    /api/songs/:id

DELETE /api/songs/:id
```

Endpoint AVANZADO (OBLIGATORIO)

```
GET /api/albums/:id/songs
```

Devuelve **un álbum con todas sus canciones asociadas**.

Ejemplo de respuesta:

```
{
  "_id": "...",
  "title": "Hybrid Theory",
  "artist": "Linkin Park",
  "songs": [
    { "title": "In the End", "duration": 216 },
    { "title": "Crawling", "duration": 209 }
  ]
}
```

Requisitos técnicos backend

- Uso obligatorio de **MongoDB/Mongoose**

- Manejo de errores:
    - 400 → validación
    - 404 → recurso inexistente
    - 500 → error servidor
- No se permite lógica en rutas directamente (usar controladores)

4. Frontend (Angular)

Componentes Standalone (OBLIGATORIOS)

Componentes mínimos

- `AlbumListComponent`
- `AlbumDetailComponent`
- `SongTableComponent`
- `SongFormComponent`
- `NavbarComponent`
- `AppComponent`

Routing obligatorio

Mínimo 4 rutas:

- `/albums`
- `/albums/new`
- `/albums/:id`
- `/albums/:id/songs`

5. Funcionalidades Angular (OBLIGATORIAS)

Conceptos evaluados

- Componentes Standalone
- Servicios HTTP
- Reactive Forms
- `@for`, `@if`

- Manejo de eventos
- Comunicación entre componentes
- Navegación con parámetros

Funcionalidades funcionales

Vista de Álbumes

- Mostrar álbumes en **tabla** (NO cards) y el número de canciones.
- Filtro por:
  - Artista
  - Género
- Ordenación por año

Vista de canciones por álbum

- Tabla de canciones asociadas al álbum
- Columnas:
  - Título
  - Duración
  - Rating (estrellas)
  - Escuchada (checkbox)
- Cambiar `listened` desde la tabla

Formularios

- Crear y editar:
  - Álbum
  - Canción (seleccionando álbum con `<select>`)

Comunicación entre componentes

- `SongTableComponent` recibe las canciones vía `@Input`
- Emite eventos `@Output` cuando:
  - Se marca como escuchada
  - Se elimina una canción

6. CRUD completo

El frontend debe soportar:

Entidad	Create	Read	Update	Delete
Album	✅	✅	✅	✅
Song	✅	✅	✅	✅

Restricción adicional:
No se puede eliminar un álbum si tiene canciones asociadas (debe gestionarse el error).

7. Entregables

Archivo ZIP con:

`/API`

`/WEB`

- Eliminar `node_modules`
- Eliminar `.angular`