Explicacion Base de Datos NoSql PlayList

🧱 1. Tipo de Base de Datos: NoSQL (MongoDB)

- Elegimos MongoDB porque permite guardar **documentos flexibles en formato JSON**.
- Esto hace que podamos anidar listas y referencias fácilmente sin necesidad de relaciones estrictas como en bases SQL.

🧍 2. Colección `usuarios`

- Cada usuario tiene sus **propios datos personales**: `id_usr`, `nickname`, `password`, `foto`.
- También tiene **múltiples listas**, así que las guardamos como un **array anidado** dentro del usuario.

🔷 ¿Por qué se anidan las listas en el usuario?

> Porque las listas pertenecen solo a ese usuario. Anidarlas mejora el acceso rápido sin tener que hacer otra búsqueda.

📂 3. Estructura de `listas` (dentro de `usuarios`)

Cada lista tiene:

- `id_list`, `nombre`, `descrip`, `genaro`
- Un array de IDs de canciones (`canciones: [ "can1", "can2" ]`)

🔷 ¿Por qué solo guardamos los IDs de canciones?

> Porque una canción puede pertenecer a muchas listas. En vez de duplicarla, solo guardamos su ID.

🎵 4. **Colección** `canciones`

- Aquí se guarda cada canción con:
  - `id_can`, `nom_can`, `duracion`
  - Un array de IDs de artistas (`artistas: ["art1", "art2"]`)

🔷 ¿Por qué aquí no se anidan los artistas?

> Porque los artistas también pueden participar en muchas canciones. Guardar solo sus IDs evita duplicación.

🎤 5. **Colección** `artistas`

- Cada artista tiene:
  - `id_art`, `nom1`, `ap1`, `nom2`, `ap2`

🔷 Se guarda aparte porque puede aparecer en muchas canciones, y así se reutiliza fácilmente.

🔁 6. **Relaciones pensadas**

Entidad	Tipo de Relación	Forma de representar
Usuario → Listas	1 a muchos	Listas anidadas en el usuario
Lista → Canciones	Muchos a muchos	Guardamos IDs de canciones
Canción → Artistas	Muchos a muchos	Guardamos IDs de artistas