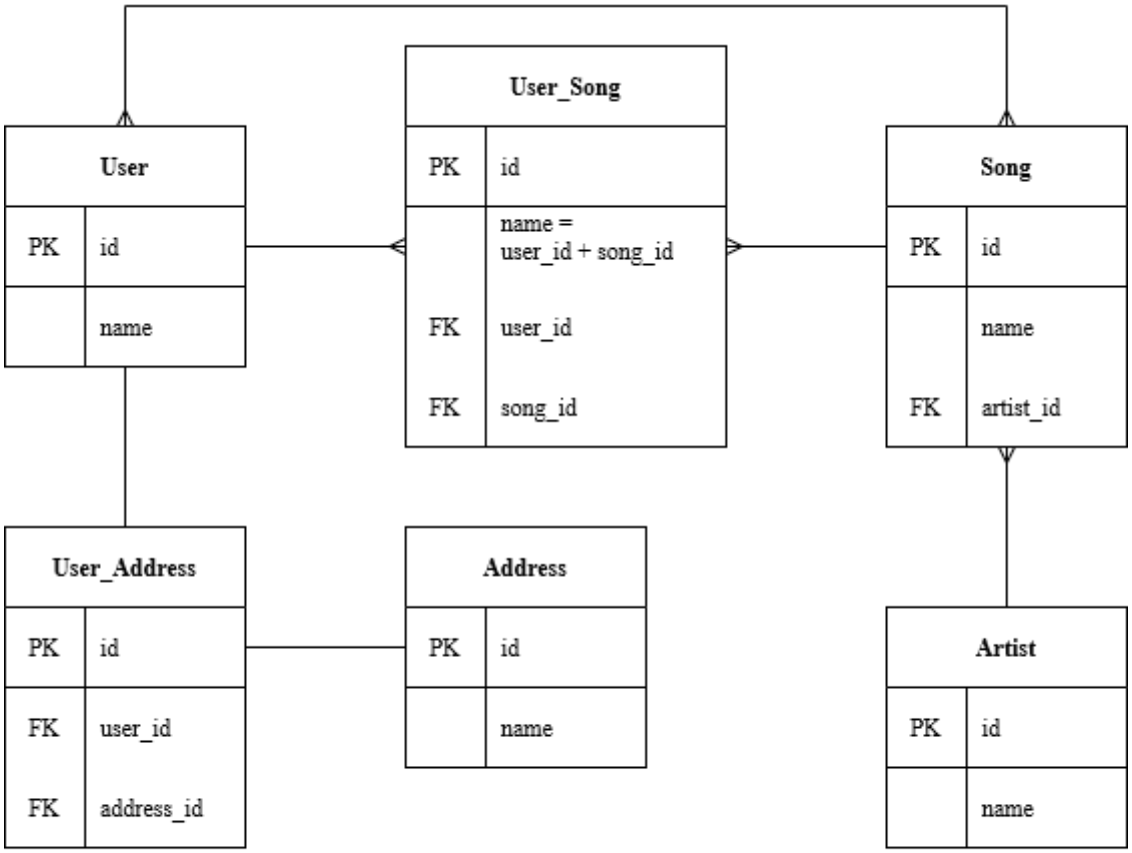
# MusicApp

Badea Bogdan-Andrei

Gr. 406




**User:**

Entitatea conține câmpurile id și name.

Este legată de clasa Song atât printr-o relație many-to-many direct, cât și prin clasa de legătură UserSong printr-o relație one-to-many. Este legată și de clasa Address printr-o relație one-to-one prin intermediul tabelei de legătura UserAddress.

Clasei User îi corespund două DTO-uri:

- UserDto, care pe lângă câmpurile id și name conține și câmpul address, care reprezintă numele adresei;
- UserMemberDto, care conține id, name și o listă de SongDto-uri, corespunzătoare melodiilor preferate de utilizator.

Metodele din controler (cum pot fi apelate din Postman):

- POST http://localhost:8099/users: Primește un body de forma { „name” : „user1”, „address” : „address1” }. Metoda va crea un obiect în clasa User cu numele „user1”, un obiect in clasa Address cu numele „address1”, cât și un obiect cu id-urile celor două în clasa de legătură User_Address;
- GET http://localhost:8099/users: Întoarce toate instanțele din clasa User sub forma de UserDto;
- GET http://localhost:8099/users/{name}: Întoarce utilizatorul cu numele dat în url sub forma de UserDto;
- GET http://localhost:8099/users/member/{name}: Întoarce utilizatorul cu numele dat în url sub forma de UserMemberDto;
- DELETE http://localhost:8099/users/{name}: Șterge utilizatorul cu numele dat în url;
- PUT http://localhost:8099/users: Primește doi parametrii, „name1” și „name2”. Metoda va schimba numele utilizatorului name1 în valoarea parametrului name2.

**Address:**

Entitatea conține câmpurile id și name.

Este legată de clasa User printr-o relație one-to-one, prin intermediul tabelei de legătură UserAddress.

Clasei Address îi corespunde DTO-ul AddressDto cu aceleași câmpuri.

Metode din controler:

- DELETE http://localhost:8099/addresses/{id}: Șterge adresa cu id-ul dat în url;

- PUT http://localhost:8099/addresses: Primește doi parametrii, „id” și „name”. Metoda va schimba numele adresei cu id-ul dat în valoarea parametrului name.

**Song:**

Entitatea conține câmpurile id, name și artistId.

Este legată de clasa User atât direct printr-o relație many-to-many, cât și prin intermediul clasei de legătură User_Song, printr-o relație one-to-many. Clasa Song este legată și de clasa Artist printr-o relație many-to-one.

Clasei Address îi corespund două DTO-uri:

- SongDto, ce conține câmpurile id, name și artistId, cât și câmpul artistName;
- SongMemberDto, ce conține câmpurile id și name, plus o listă de UserDto-uri, corespunzătoare utilizatorilor ce au ales melodia.

Metode din controler:

- POST http://localhost:8099/songs: Primește un body de forma { „name” : „song1”, „artistId” : 1 }. Metoda creează o melodie cu numele „song1”, care aparține artistului cu id-ul 1. Trebuie mai întâi creat artistul, apoi melodia;
- GET http://localhost:8099/songs: Întoarce toate melodiile din clasa Song sub formă de SongDto;
- GET http://localhost:8099/songs/{name}: Întoarce melodia cu numele dat în url sub formă de SongDto;
- GET http://localhost:8099/songs/member/song1: Întoarce melodia cu numele dat în url sub formă de SongMemberDto;
- GET http://localhost:8099/songs/filterByArtistName: Primește un parametru „artistName” și întoarce toate melodiile corespunzătoare artistului;
- DELETE http://localhost:8099/songs/{name}: Șterge melodia cu numele dat în url;
- PUT http://localhost:8099/songs: Primește doi parametrii, „name1” și „name2”. Înlocuiește numele melodiei cu numele corespunzător valorii name1 cu valoare parametrului name2.

**UserSong:**

Entitatea conține câmpurile id, name, userId și songId. Name este format din concatenarea userId și songId cu simbolul underscore între ele. Acesta trebuie să fie unic și are rolul de a împiedica ca un utilizator să aleagă de mai multe ori aceeași melodie.

Este legată de clasele User și Song prin cate o relație many-to-one.

Clasei UserSong îi corespunde DTO-ul UseSongDto, care conține câmpurile id, name, userId, songId, userName și songName.

Metode din controler:

- POST http://localhost:8099/users_songs: Primește un body de forma { „userId” : 1, „songId” : 1 } și creează o instanță de legătură între utilizatorul cu id-ul 1 și melodia cu id-ul 1;
- GET http://localhost:8099/users_songs/{id}: Întoarce obiectul cu id-ul din url;
- GET http://localhost:8099/users_songs/filterByUserName: Primește ca parametru „userName” și întoarce toate obiectele din clasa UserSong cu id-ul utilizatorului dat;
- GET http://localhost:8099/users_songs/filterBySongName: Primește ca parametru „songName” și întoarce toate obiectele din clasa UserSong cu id-ul melodiei date;
- DELETE http://localhost:8099/users_songs/{id}: Șterge obiectul UserSong cu id-ul din url.

**Artist:**

Entitatea conține câmpurile id și name.

Este legată de clasa Song printr-o relație one-to-many.

Clasei Artist îi corespund două DTO-uri:

- ArtistDto, cu aceleași câmpuri;
- ArtistMemberDto, care conține id, name și o listă de SongDto-uri, corespunzătoare melodiilor compuse de artist.

Metode controller:

- POST http://localhost:8099/artists: Primește un body de forma { „name” : „artist1” } și creează un artist cu numele dat;
- GET http://localhost:8099/artists: Întoarce toți artiștii sub forma de ArtistDto;
- GET http://localhost:8099/artists/{name}: Întoarce artistul cu numele din url sub formă de ArtistDto;
- GET http://localhost:8099/artists/member/{name}: Întoarce artistul cu numele din url sub formă de ArtistMemberDto;
- DELETE http://localhost:8099/artists/{name}: Șterge artistul cu numele dat din url:
- PUT http://localhost:8099/artists: Primește doi parametrii, „name1” și „name2”. Metoda înlocuiește numele artistului corespunzător „name1” cu valoarea parametrului „name2”.