DOCUMENTAȚIA PROIECTULUI

Sound Sphere

STUDENȚI

Bacalu George

Buta Cătălin

**Cuprins**

1. Prezentarea proiectului, ce isi propune, ce probleme rezolva

2. Tehnologiile folosite

3. Baza de date: diagrama bazei de date + scurta prezentare a tabelelor si a relatiilor dintre ele

4. Prezentarea API-ului: screenshot la Swagger pentru a putea vedea toate endpoint-ruile + scurta descriere a CRUD-urilor

5. Prezentare despre cum poate fi utilizata aplicatia: tipuri de utilizatori, ce vede fiecare, autentificare etc

6. Concluzii si contributii (cum v-ati impartit task-urile, ce ati invatat in urma acestui proiect etc)

7. Link GIT catre codul proiectului

# Prezentarea proiectului, ce isi propune, ce probleme rezolva

**SoundSphere** este o aplicatie de music streaming conceputa sa ofere utilizatorilor o experienta placuta si simplificata in descoperirea si ascultarea de muzica. Acest lucru este realizat printr-un set minimal de functionalitati care include:

* Inregistrare si autentificare;
* Crearea playlist-urilor;
* Sectiuni de feedback si notificari etc.

Scopul SoundSphere este sa solutioneze probleme comune intalnite de utilizatorii altor servicii de music streaming, precum complexitatea excesiva a interfetei utilizator si dificultatea in gestionarea colectiilor de muzica.

Probleme rezolvate:

* Proces simplu de inregistrare si autentificare in aplicatie cu posibilitatea de resetare a parolei;
* Functionalitati de paginare, sortare si filtrare care permit utilizatorilor sa-si gaseasca rapid melodiile, albumele si artistii preferati;
* Contorizarea numarului de urmaritori pentru artisti, precum si numarul de ascultari pentru melodii pentru a facilita generarea de recomandari muzicale potrivite in pagina principala;
* Generarea de statistici privind procentajul melodiilor pe genuri muzicale și al feedback-urilor pe categorii;
* Crearea si gestionarea eficienta a playlist-urilor, adaugand sau eliminand melodii cu usurinta;
* Implicarea utilizatorilor in lumea aplicatiei prin sectiuni de feedback si notificari;
* Introducerea unei sectiuni prin care utilizatorii isi pot alege tema aplicatiei si daca vor sa primeasca notificari si pe email;

# Tehnologiile folosite

**Limbaje de programare:** C#

**Platforme de dezvoltare:** .NET

**Frameworks:** ASP.NET Core

**Frameworks de testare:** xUnit

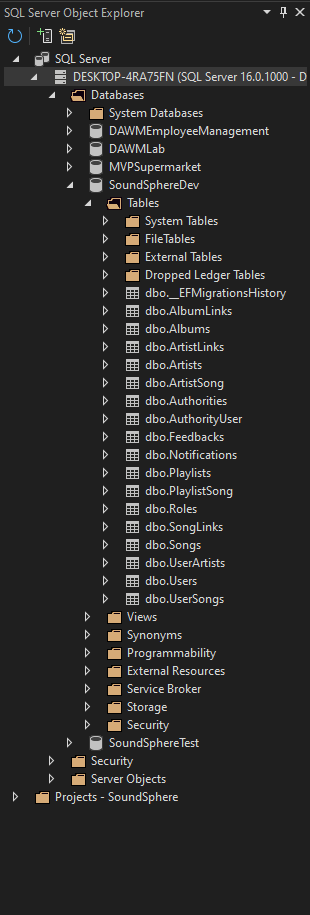
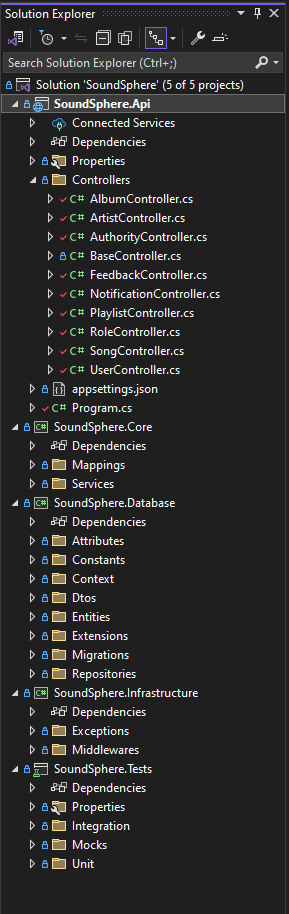
**Baze de date:** SQLServer

**Sistem de versionare:** Git

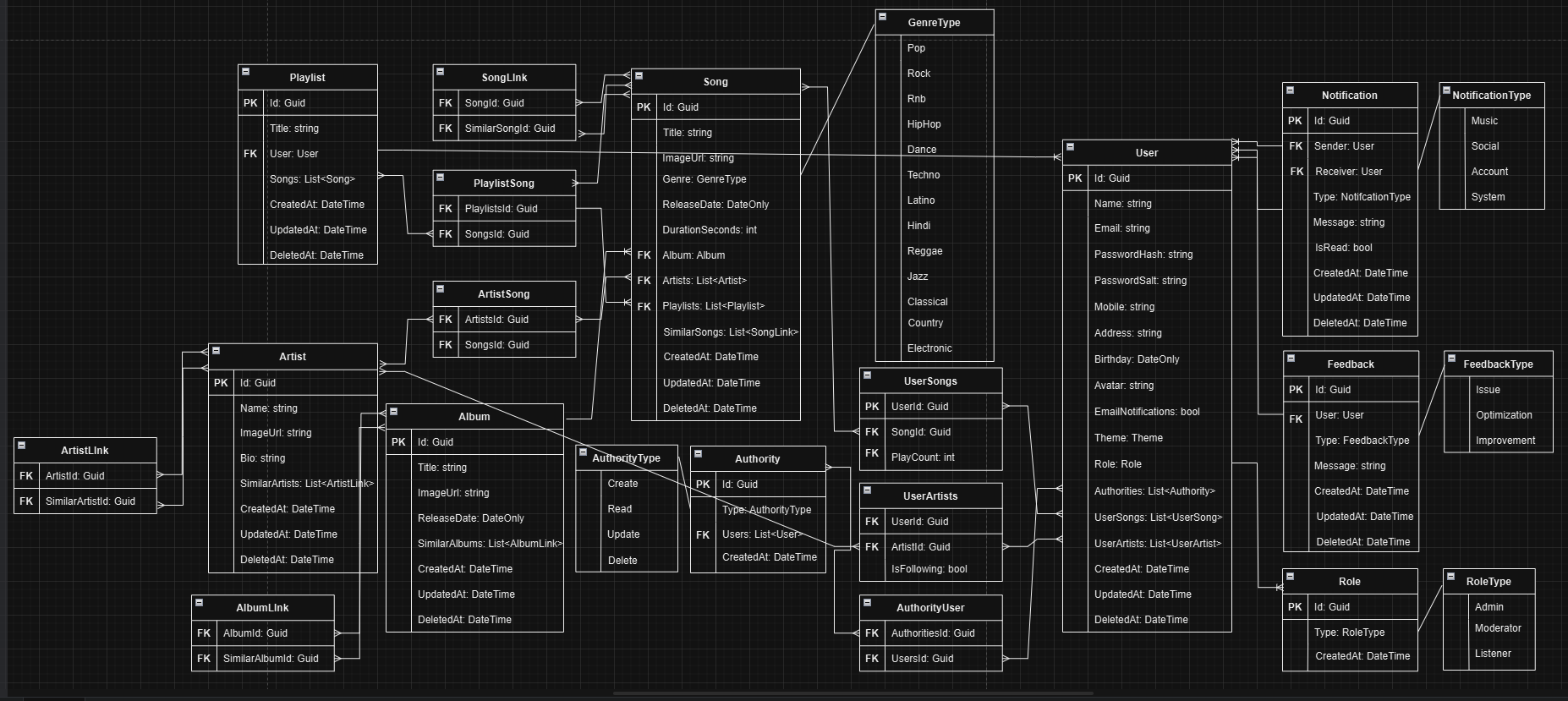
**Web Server:** Kestrel

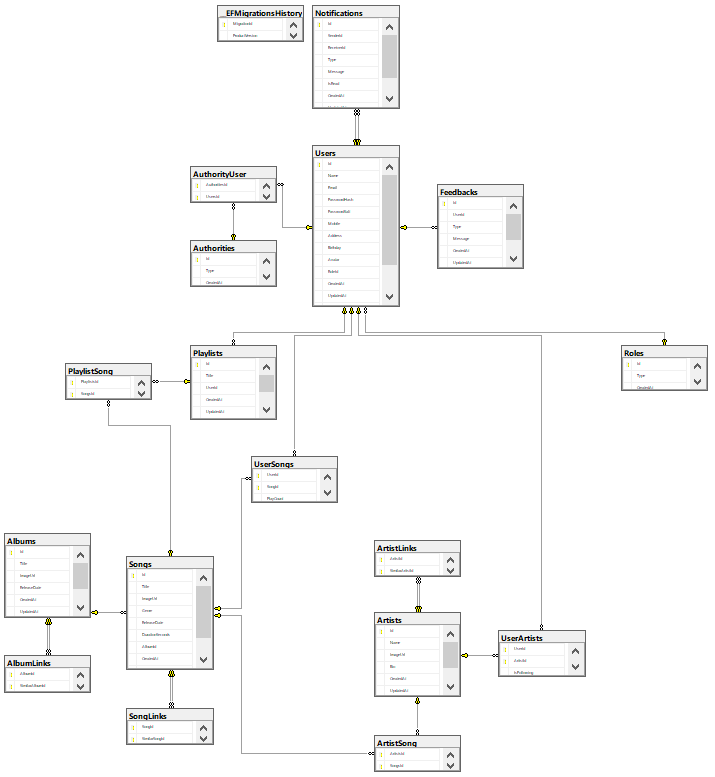
**Unelte pentru urmărirea defectelor:** Jira

**Metodologii de dezvoltare software:** Scrum

# Baza de date

****



Entitatile principale ale aplicatiei:

* User - atribute: Id, Name, Email, PasswordHash, PasswordSalt etc.
* Rol - atribute: Id, Type, CreatedAt
* Autoritate - atribute: Id, Type, CreatedAt
* Feedback - atribute: Id, User (Sender), Type, Message etc.
* Notificare - atribute: Id, Sender, Receiver, Type, Message etc.
* Melodie - atribute: Id, Title, ImageUrl, Genre, ReleaseDate etc.
* Artist - atribute: Id, Name, ImageUrl, Bio etc.
* Album - atribute: Id, Title, ImageUrl, ReleaseDate etc.
* Playlist - atribute: Id, Title, User (Owner) etc.

Relatii intre tabele:

1. Relatii reflexive (auto-referinta)

In pagina de detalii pentru o melodie, album sau artist, avem o sectiune care recomanda melodii, albume sau artisti similari celui selectat (relatie One to Many). Ca sa persistam pentru o inregistrare cu unul din aceste tipuri, o lista de inregistrari de tip similar am creat urmatoarele 3 tabele:

* AlbumLink - atribute: AlbumId, SimilarAlbumId (ambele coloane reprezinta chei straine care referentiaza acelasi tabel Albums)
* ArtistLink - atribute: ArtistId, SimilarArtistId (analog)
* SongLink - atribute: SongId, SimilarSongId (analog)

1. Relatii Many to Many intre entitati diferite (rezulta in creerea de tabele de legatura)
2. Urmatoarele 3 tabele de legatura au fost generate de Entity Framework Core pe baza proprietatilor de navigare de tip colectie adaugate in clasele de entitate aflate in relatie.

* ArtistSong - atribute: ArtistsId, SongsId (ambele coloane reprezinta chei straine; prima referentiaza tabelul Artist, iar a doua tabelul Song)
* AuthorityUser - atribute: AuhoritiesId, UsersId (analog)
* PlaylistSong - atribute: PlaylistsId, SongsId (analog)

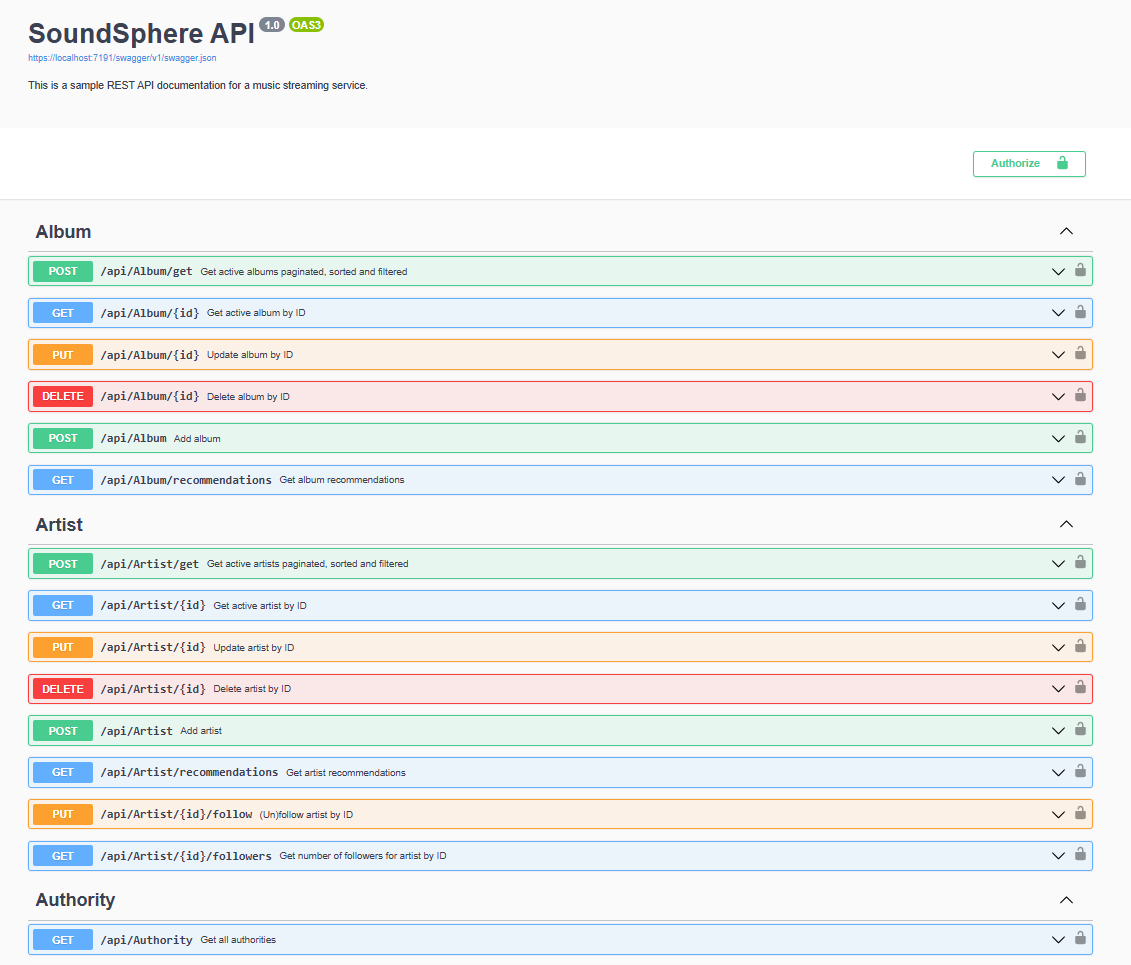
1. Urmatoarele 2 tabele de legatura au fost generate de Entity Framework Core pe baza unor clase de model destinate acestora, definite de noi. Motivul pentru care am folosit o strategie diferita este ca am dorit sa adaugam cate o proprietate in plus, pe langa cele 2 chei straine care referentiaza fiecare tabel aflat in relatie.

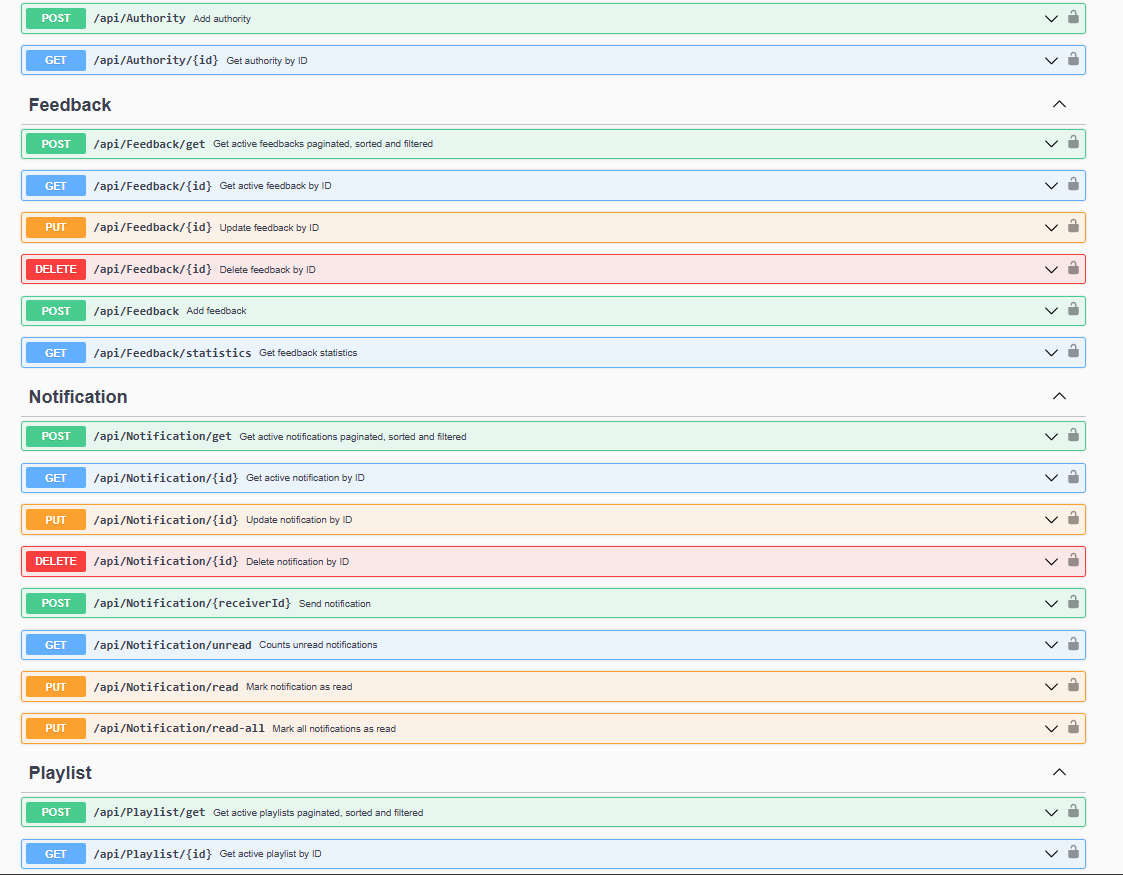
In cazul de mai jos dorim sa stim pentru fiecare utilizator pe care artisti ii urmareste si pe care nu, precum si de cate ori a ascultat fiecare dintre melodiile existente in aplicatie. Pentru acest lucru, fiecare utilizator trebuie sa aiba propriul sau snapshot al artistilor urmariti si al melodiilor ascultate.

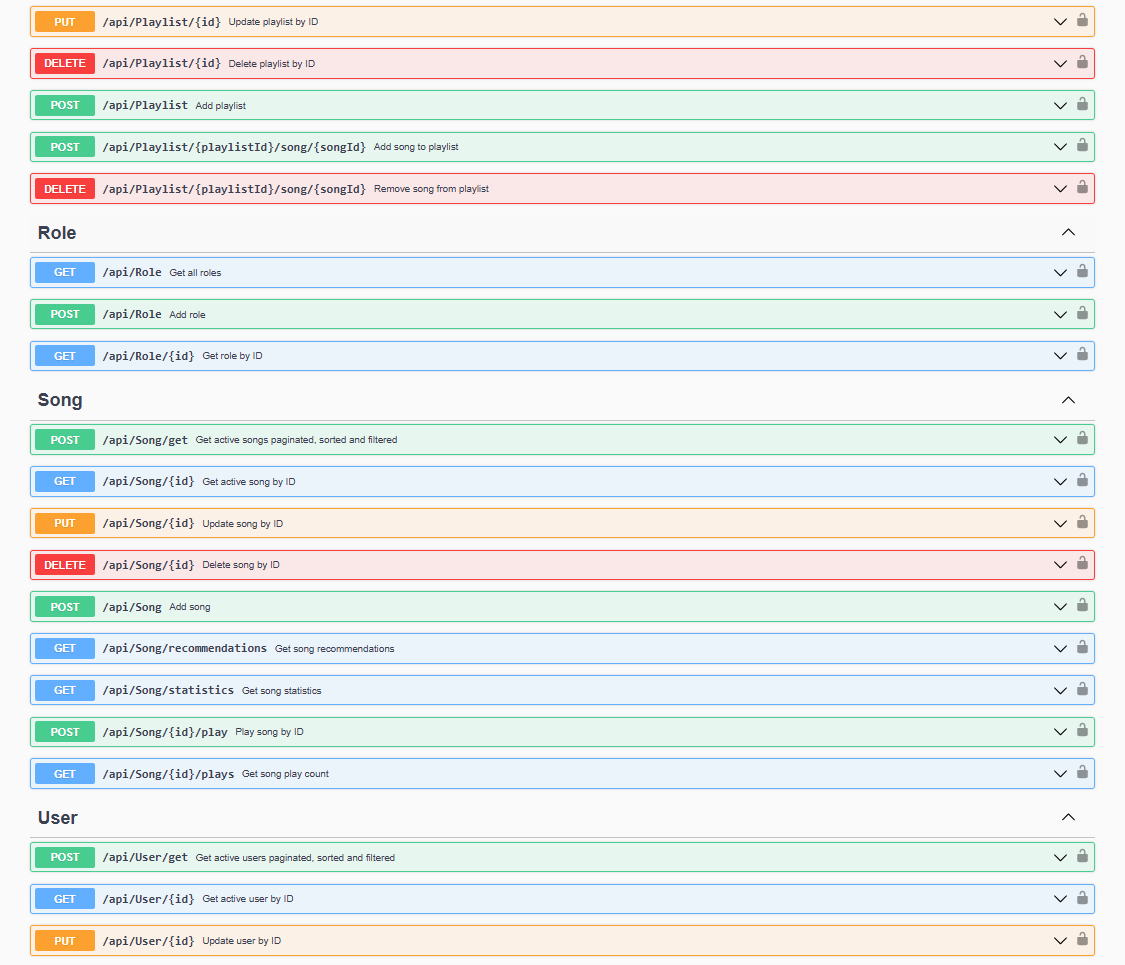
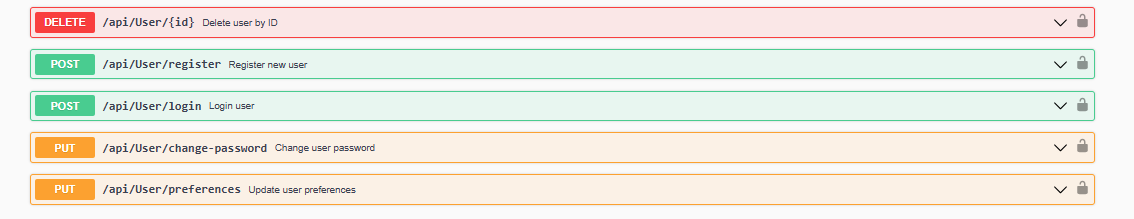
* UserArtists - atribute: UserId, ArtistId, IsFollowing
* UserSong - atribuite: UserId, SongId, PlayCount

# Prezentarea API-ului

**Documentatia REST API-ului in Swagger:**





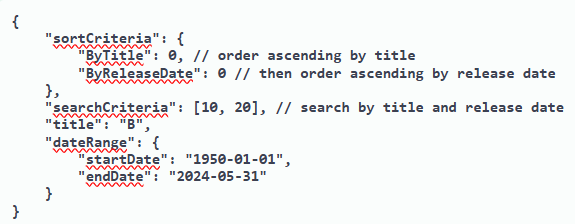
 

**Descrierea functionalitatilor:**

Logica operatiilor de CRUD utilizate este similara pentru majoritatea entitatilor. Pentru exemplificare prezentam functionalitatile corespunzatoare entitatii Album:

* POST /api/Album/get

Realizeaza o selectie a tuturor albumelor din baza de date care se incadreaza intr-un set de criterii de filtrare, iar acestea sunt ordonate dupa un set de criterii de sortare. In final albumele rezultate sunt paginate. Motivul pentru care aceasta este o cerere HTTP POST in loc de GET, este pentru ca informatiile despre paginare, sortare si filtrare sunt transmise ca request body, iar in cazul GET, nu se pot trimite date pe cerere. Toate criteriile sunt optionale. Exemplu:



Mai sus avem un exemplu de request body prin intermediul caruia ne propunem sa filtram doar acele albume al caror titlu contine litera “B” si au fost lansate intre datele 01/01/1950 si 31/05/2024. Albumele filtrate, vor fi sortate ascendent dupa titlu si dupa data lansarii. Rezultatele vor fi paginate, cate 10 pe pagina si raspunsul va contine doar prima pagina (page = 0, size = 10 sunt optiunile implicite).

* GET /api/Album/{id}

Selecteaza din tabelul Albums acea inregistrare al carei ID coincide cu ID-ul care apare in path-ul/route-ul cererii ca valoare pentru {id}, cu conditia ca acea resursa sa nu fie marcata ca stearsa (DeletedAt != null)

* POST /api/Album

Salveaza/Adauga un nou album in baza de date. Informatiile despre acest album sunt trimise in cerere pe request body.

* PUT /api/Album/{id}

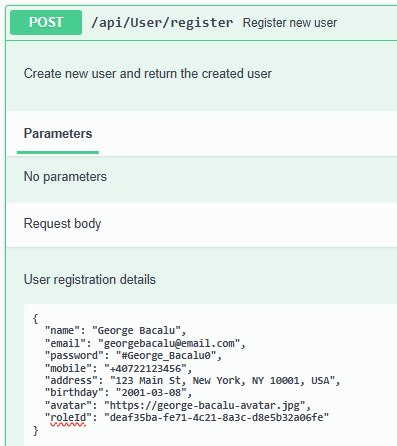
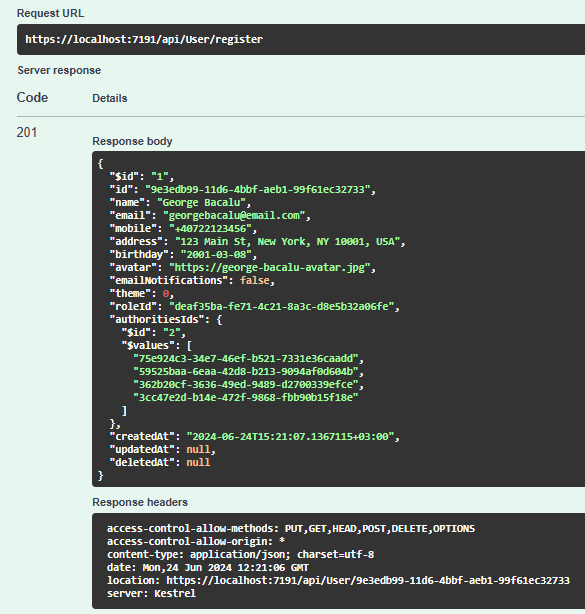
Modifica detaliile albumului al carui ID coincide cu ID-ul gasit in path-ul cererii in locul placeholder-ului {id}, cu conditia ca acea resursa sa existe si sa nu fie marcata ca stearsa. Toate detaliile noului album (inclusiv cele nemodificate) sunt trimise in cerere pe request body.

* DELETE /api/Album/{id}

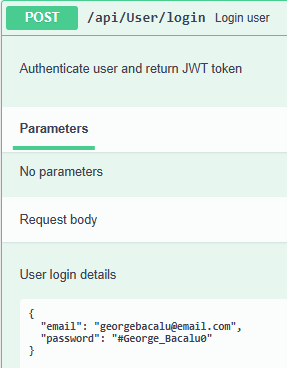
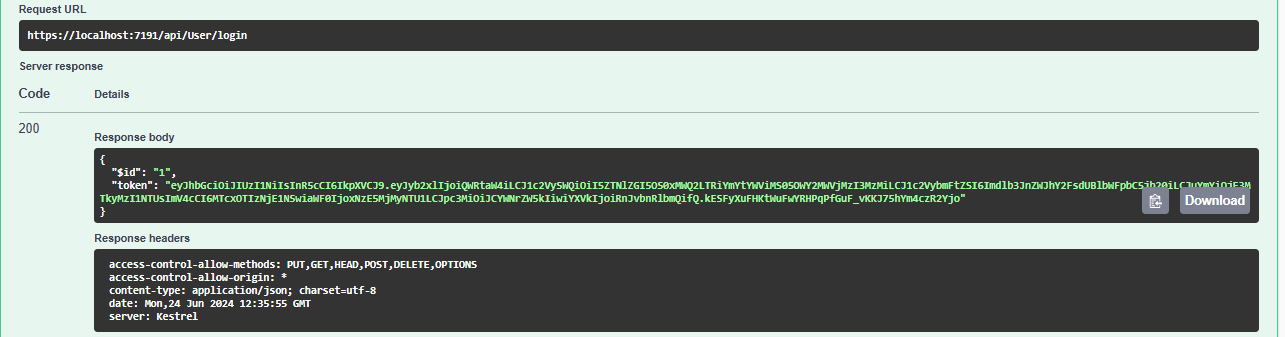
Realizam o stergere soft a albumului cu ID-ul transmis in path. Acesta nu o sa fie eliminat din baza de date, insa o sa fie marcat ca sters (coloana DeletedAt va avea ca si valoare data cand elementul a fost “sters”). Prin soft delete eliminam item-ului din frontend, insa il pastram in baza de date pentru a nu afecta statistici si metrici inregistrate pe diverse perioade de timp.

# Utilizarea aplicatiei

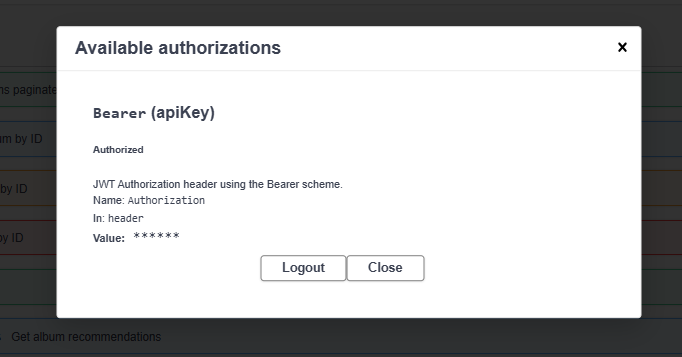
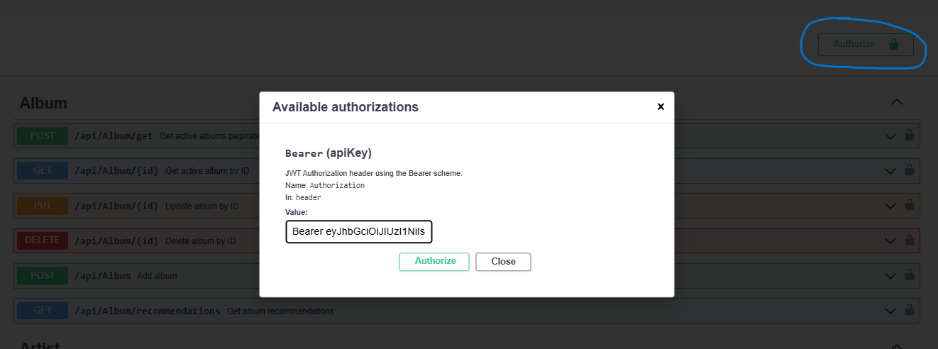
1. Restore asupra backup-ului “SoundSphere.Api/Scripts/SoundSphereBackup.sql” din repo-ul proiectului pentru popularea rapida a bazei de date. Cand pornim aplicatia trebuie sa se deschida in browser pagina cu documentatia REST API-ului in Swagger. Daca nu se intampla automat, atunci se poate accesa la <https://localhost:7191/swagger/index.html>.
2. Pentru a înregistra un utilizator nou, trimitem o cerere POST la /api/User/register cu detaliile utilizatorului în format JSON. Nivelul de acces al utilizatorului variază în funcție de rolul ales. În exemplul de mai jos, înregistrăm un utilizator cu rolul de "Admin".

1. Dupa inregistrare ne autentificam pentru a obtine JSON Web Token necesar pentru a autoriza restul cererilor in aplicatie.

1. Prefixam tokenul obtinut cu “Bearer “ si punem stringul obtinut pe headerul Authorization al tuturor cererilor urmatoare.



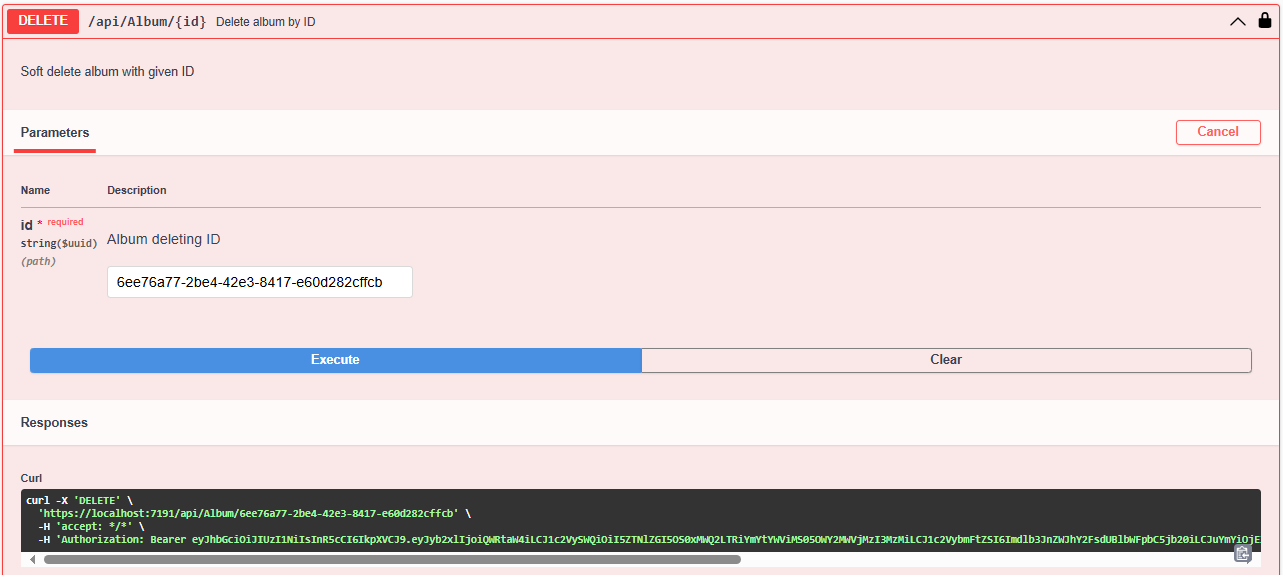
1. La nivel general ne-am propus distribuirea accesul userilor intr-o modalitate simpla:

* Listener are autorizarea de a citi date
* Moderatorul are autorizarea de a citi, crea si modifica date
* Adminul are autorizarea de a citi, crea, modifica si sterge date.

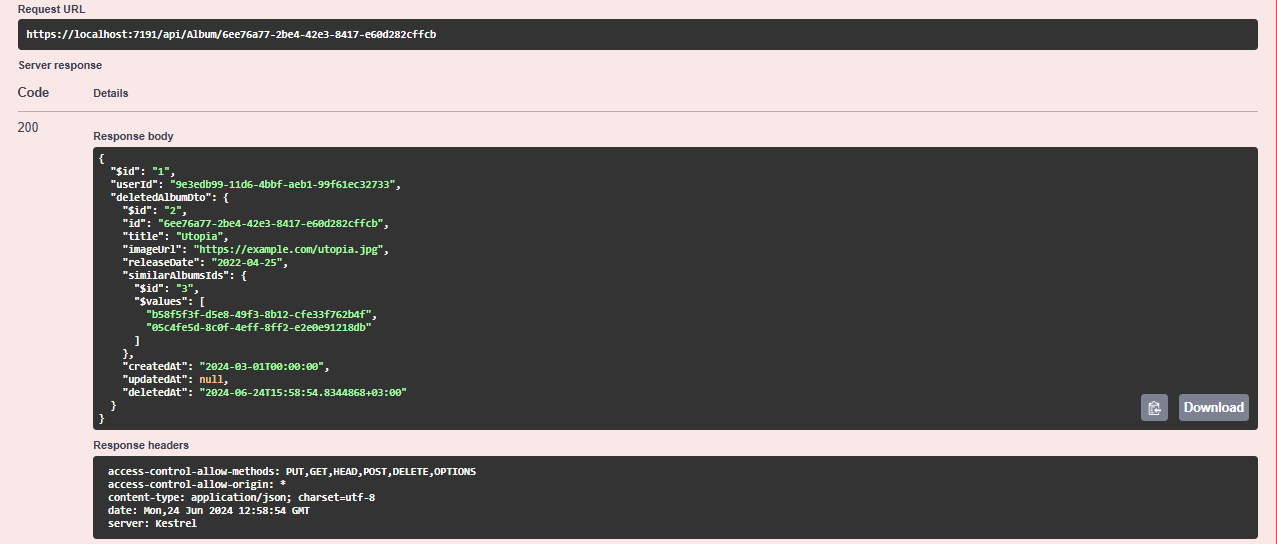
Desigur, exista si exceptii de la aceasta regula. De exemplu, distributia nivelurilor de acces de mai sus se aplica in cazul operatiilor de CRUD pe albume, artisti si melodii, dar nu se aplica la feedback, notificari, playlist-uri si useri. De exemplu, orice user trebuie sa aiba posibilitatea de a-si gestiona playlist-urile proprii indiferent de rolul sau. Restrictiile de securitate pe care le-am aplicat in cazul acestor entitati din urma sunt:

* sa nu permitem unui user sa editeze sau sa stearga datele altuia
* sa nu poata crea date noi in numele altui user
* sa nu poata accesa datele altuia (de exemplu un user trebuie sa poate vedea doar propriile notificari si playlist-uri).

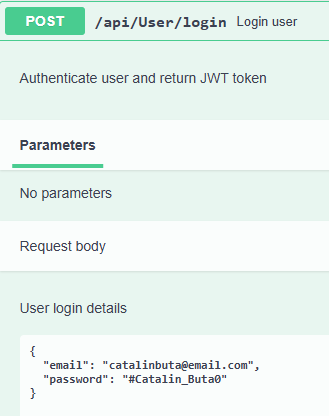
In continuare exemplificam stergerea unui album de catre Admin si dupa de catre Moderator.



Putem observa ca valoarea pentru headerul Authorization contine token-ul obtinut la autentificarea adminului prefixat de “Bearer “. De aceea primim raspuns 200 OK in acest caz.

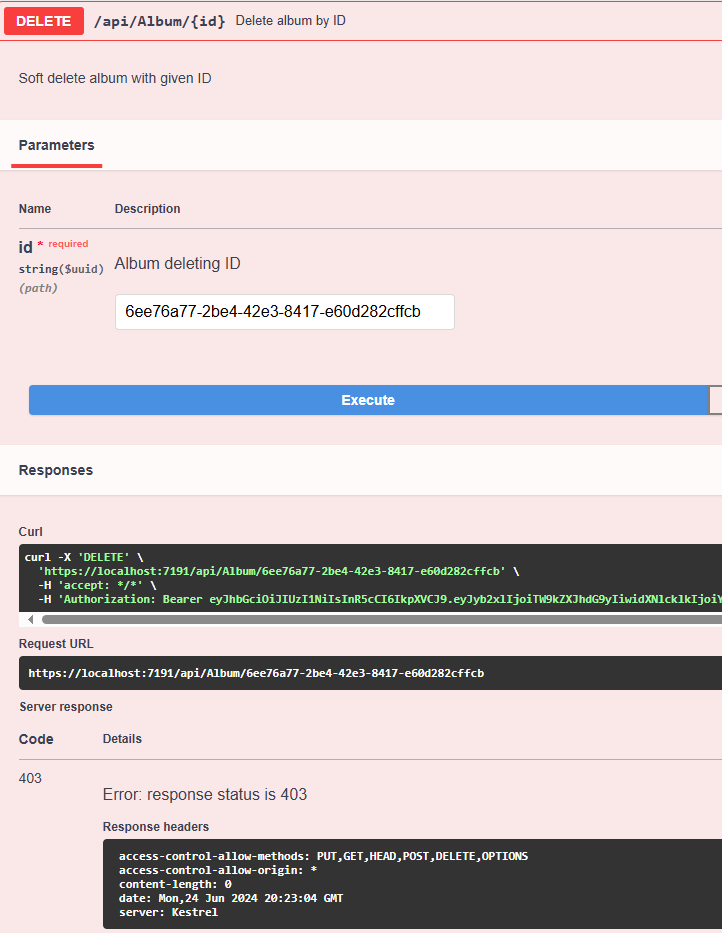


Inregistram si ne autentificam cu un cont de moderator.





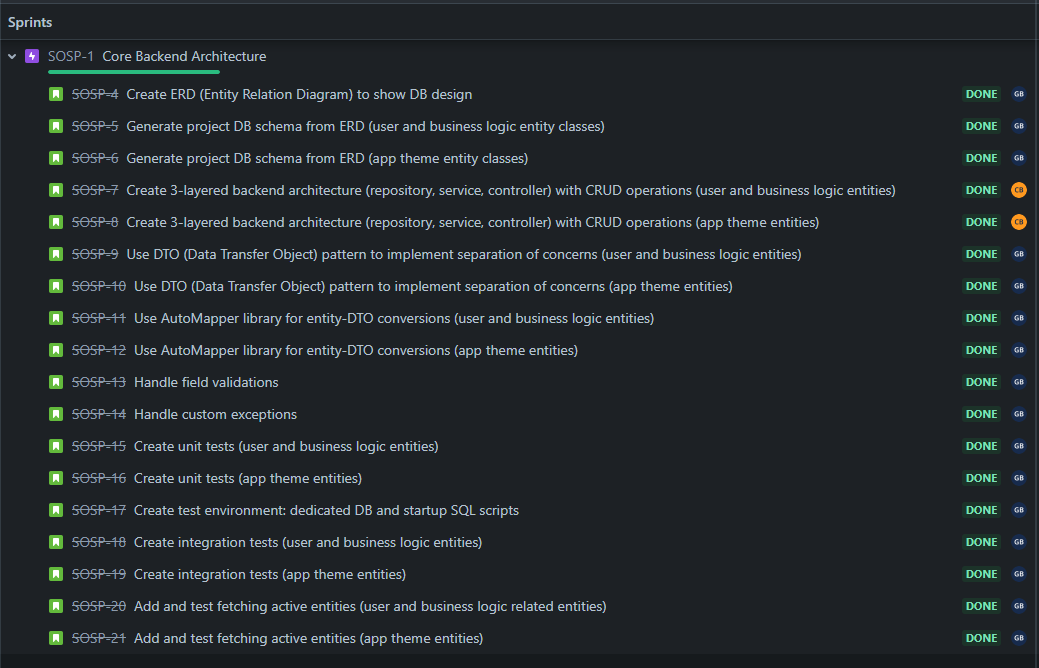
Moderatorul nu este autorizat sa stearga albume in aplicatie si, de aceea, atunci cand incearca sa faca acest lucru raspunsul primit o sa fie 403 Forbidden.

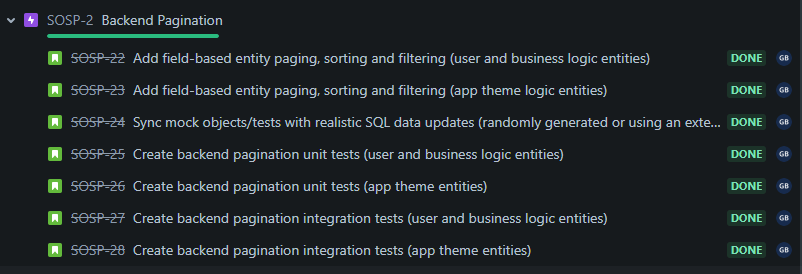


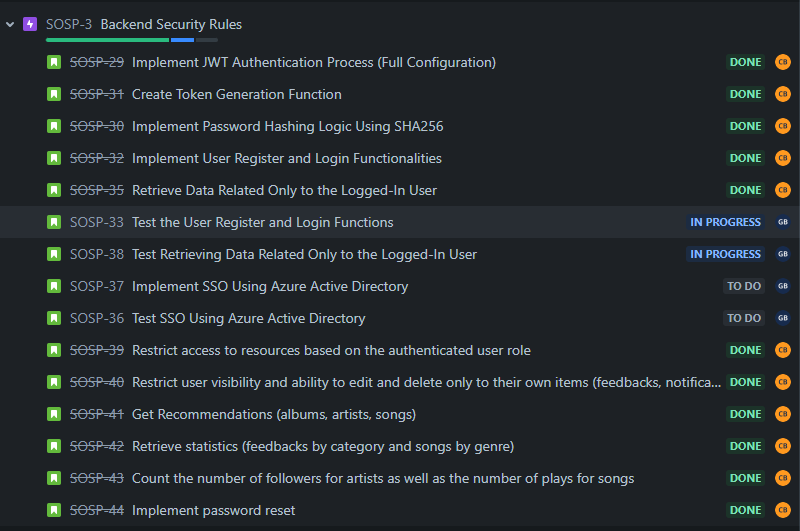
# Concluzii si contributii

**SoundSphere** a fost si continua sa fie o oportunitate excelenta de dezvoltare a aptitudinilor tehnice in backend, dar in acelasi timp, conceperarea si dezvoltarea acestei aplicatii ne-a invatat si despre importanta planificarii in avans a caracteristicilor aplicatiei si a sarcinilor pe care aveam sa le dezvoltam. De asemenea, faptul ca am invatat cu timpul sa utilizam Git corespunzator a facilitat mult colaborarea pe proiect.

Pentru planificarea, estimarea si impartirea sarcinilor am folosit platforma Jira si metodologia de dezvoltare Scrum. Contributiile se pot vedea in screenshot-urile urmatoare (GB – George Bacalu, CB – Catalin Buta).







# Link Git

https://github.com/GeorgeBacalu/SoundSphere