API Endpoint Documentation [Version 1]

Emilio Moya

Table of Contents

1. AuthController

1.1) Register User

- **Request Type:** POST
- **Endpoint:** "/api/auth/signup"
- **Required Parameters:**
  - {
    "username": String,
    "email": String,
    "password": String,
    "roles": Set<String>
    }
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns an *AuthResponse* containing an instance of the newly created *User* and a message indicating the user was successfully registered.
  - 400 (bad request): Returns an *AuthResponse* indicating which aspect of the request was invalid.

1.2) User Login

- **Request Type:** POST
- **Endpoint:** "/api/auth/signin"
- **Required Parameters:**
  - {
    "username": String,
    "password": String
    }
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a *JwtRepsonse* that includes the user's JWT token and general information.

- 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully logged in (*currently bugged: isn't returning JwtResponse*)

2.AlbumController

2.1) Get All Albums

- **Request Type:** GET
- **Endpoint:** "/api/albums"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of all albums in our database.

2.2) Get Album By Id

- **Request Type:** GET
- **Endpoint:** "/api/albums/{albumId}"
- **Required Parameters:** "albumId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns an *AlbumResponse* containing the *Album* that was requested and a message saying the album was successfully found.
  - 404 (not found): Returns an *AlbumResponse* containing a message saying the album was not found.

2.3) Get Albums By Title

- **Request Type:** GET
- **Endpoint:** "/api/albums/title/{title}"
- **Required Parameters:** "title": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**

- 200 (success): Returns a list of all albums in our database that have the name specified in the endpoint URL

2.4) Get Discovery Albums

- **Request Type:** GET
- **Endpoint:** "/api/albums/discovery"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of 21 random albums from our database

2.5) Get Newly Released Albums

- **Request Type:** GET
- **Endpoint:** "/api/albums/newReleases"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of the 21 newest albums in our database, based on their 'releaseDate'.

2.6) Create Album

- **Request Type:** POST
- **Endpoint:** "/api/albums/create"
- **Required Parameters:**
  - {
    "title": String,
    "artist": String,
    "releaseDate": String,
    "genres": List<String>
    }
- **Required Authorization(s):** 'Admin' (JWT Bearer Token)
- **Response:**

- 200 (success): Returns an *AlbumResponse* containing the newly created *Album* and a message indicating the album was successfully created.
- 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to create the album (*currently bugged: isn't returning JwtResponse*)

3. UserController

3.1) Get All Users

- **Request Type:** GET
- **Endpoint:** "/api/users"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of all *User* objects in the database

3.2) Get User By Id

- **Request Type:** GET
- **Endpoint:** "/api/users/{userId}"
- **Required Parameters:** "userId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a *UserResponse* which contains the *User* associated with 'userId' and a message saying the user was successfully found.
  - 404 (not found): Returns a *UserResponse* which contains a message saying the user was not found.

3.3) Get User By Username

- **Request Type:** GET

- **Endpoint:** “/api/users/username/{username}”
- **Required Parameters:** “username”: String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a *UserResponse* which contains the *User* associated with ‘username’ and a message saying the user was successfully found.
    - 404 (not found): Returns a *UserResponse* which contains a message saying the user was not found.

3.4) Follow User

- **Request Type:** PUT
- **Endpoint:** “/api/users/follow”
- **Required Parameters:**

```
{
    “followerId”: String,
    “followeeId”: String,
}
```

- **Required Authorization(s):** ‘User’ or ‘Admin’ (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *UserResponse* containing the following *User,* the followed *User*, and a message indicating the user was successfully followed.
    - 400 (bad request): Returns a *UserResponse* containing a message indicating the *User* being followed was already followed
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to follow the user (*currently bugged: isn’t returning JwtResponse*)
    - 404 (not found): Returns a *UserResponse* containing a message indicating the following *User* or followed *User* was not found

3.5) Unfollow User

- **Request Type:** PUT
- **Endpoint:** “/api/users/unfollow”

- **Required Parameters:**

```
{
    “followerId”: String,
    “followeeId”: String,
}
```

- **Required Authorization(s):** ‘User’ or ‘Admin’ (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *UserResponse* containing the unfollowing *User,* the unfollowed *User*, and a message indicating the user was successfully unfollowed.
    - 400 (bad request): Returns a *UserResponse* containing a message indicating the *User* being unfollowed was already unfollowed
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to unfollow the user (*currently bugged: isn’t returning JwtResponse*)
    - 404 (not found): Returns a *UserResponse* containing a message indicating the unfollowing *User* or unfollowed *User* was not found

3.6) Get Users Being Followed

- **Request Type:** GET
- **Endpoint:** “/api/users/following/{userId}”
- **Required Parameters:** “userId”: String - specified in the endpoint URL
- **Required Authorization(s):** ‘User’ or ‘Admin’ (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *UserResponse* which contains the *User* who is making the request, a list of *Users* who are being followed, and a message indicating that the request was successful
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to get the users being followed (*currently bugged: isn’t returning JwtResponse*)
    - 404 (not found): Returns a *UserResponse* which contains a message indicating that the *User* making the request could not be found

3.7) Get User Feed

- **Request Type:** GET
- **Endpoint:** "/api/users/feed/{userId}"
- **Required Parameters:** "userId": String - specified in the endpoint URL
- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a list of *ReviewResponses* that contains *Reviews* created by *Users* whom the requesting *User* is following, sorted by most recent 'creationDate' within the *Reviews*
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to get their feed (*currently bugged: isn't returning JwtResponse*)

4. ReviewController

4.1) Get All Reviews

- **Request Type:** GET
- **Endpoint:** "/api/reviews"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of *ReviewResponses* which are created from each *Review* in the database

4.2) Get Reviews By Music ID

- **Request Type:** GET
- **Endpoint:** "/api/reviews/{musicId}"
- **Required Parameters:** "musicId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of *ReviewResponses* which are created from each *Review* in the database that is associated with a song or album (as specified by the 'musicId')

4.3) Get Reviews By User ID

- **Request Type:** GET
- **Endpoint:** "/api/reviews/users/{userId}"
- **Required Parameters:** "userId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of *ReviewResponses* which are created from each *Review* in the database that is associated with a *User* (as specified by the 'userId')

4.4) Create Review

- **Request Type:** POST
- **Endpoint:** "/api/reviews/create"
- **Required Parameters:**
    - {
      "body": String,
      "rating": int,
      "userId": String,
      "musicId": String
      }
- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *ReviewResponse* which contains the created *Review,* the *User* and *Album*/*Song* associated with the review, and a message indicating the request was successful.
    - 400 (bad request): Returns a *ReviewResponse* containing a message indicating the review was already associated with an album or song.
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to create the review (*currently bugged: isn't returning JwtResponse*)
    - 404 (not found): Returns a *ReviewResponse* containing a message indicating the *Album*, *Song* or *User* associated with the new *Review* was not found.

4.5) Edit Review

- **Request Type:** PUT
- **Endpoint:** "/api/reviews/edit"
- **Required Parameters:**

```
{
    "body": String,
    "rating": int,
    "reviewId": String,
    "musicId": String
}
```

- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *ReviewResponse* which contains the updated *Review,* the *User* and *Album*/*Song* associated with the review, and a message indicating the request was successful.
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to edit the *Review* (*currently bugged: isn't returning JwtResponse*)
    - 404 (bad request): Returns a *ReviewResponse* containing a message indicating the *Review* was not successfully updated.

4.6) Delete Review

- **Request Type:** DELETE
- **Endpoint:** "/api/reviews/delete/{reviewId}"
- **Required Parameters:** "reviewId" : String - specified in the endpoint URL
- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *ReviewResponse* which contains the deleted *Review,* the *User* and *Album*/*Song* associated with the review, and a message indicating the request was successful.
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to delete the *Review* (*currently bugged: isn't returning JwtResponse*)

- 404 (not found): Returns a *ReviewResponse* containing a message indicating the *Review* was not successfully deleted.

5.ImageController

5.1) Upload Album Image

- **Request Type:** POST
- **Endpoint:** "/api/images/uploadAlbum"
- **Required Parameters:**

```
{
    "file": MultipartFile,
    "id": String
}
```

- **Required Authorization(s):** 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns an *ImageUploadResponse* containing a file path string for the newly created image and a creation date.
  - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to upload an image (*currently bugged: isn't returning JwtResponse*)

5.2) Upload Song Image

- **Request Type:** POST
- **Endpoint:** "/api/images/uploadSong"
- **Required Parameters:**

```
{
    "file": MultipartFile,
    "id": String
}
```

- **Required Authorization(s):** 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns an *ImageUploadResponse* containing a file path string for the newly created image and a creation date.

- 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to upload an image (*currently bugged: isn't returning JwtResponse*)

5.3) Upload Profile Image

- **Request Type:** POST
- **Endpoint:** "/api/images/uploadProfile"
- **Required Parameters:**

```
{
    "file": MultipartFile,
    "id": String
}
```

- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns an *ImageUploadResponse* containing a file path string for the newly created image and a creation date.
  - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to upload an image (*currently bugged: isn't returning JwtResponse*)

5.4) Upload Artist Image

- **Request Type:** POST
- **Endpoint:** "/api/images/uploadArtist"
- **Required Parameters:**

```
{
    "file": MultipartFile,
    "id": String
}
```

- **Required Authorization(s):** 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns an *ImageUploadResponse* containing a file path string for the newly created image and a creation date.

- 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to upload an image (*currently bugged: isn't returning JwtResponse*)

6. SongController

6.1) Get All Songs

- **Request Type:** GET
- **Endpoint:** "/api/songs"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of all *Songs* in the database

6.2) Get Song By ID

- **Request Type:** GET
- **Endpoint:** "/api/songs/{songId}"
- **Required Parameters:** "songId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a *SongResponse* containing the *Song* that was requested and a message saying the song was successfully found.
  - 404 (not found): Returns a *SongResponse* containing a message saying the song was not found.

6.3) Get Songs By Title

- **Request Type:** GET
- **Endpoint:** "/api/songs/title/{title}"
- **Required Parameters:** "title": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**

- 200 (success): Returns a list of all *Songs* based on their ‘title’

6.4) Create Song

- **Request Type:** POST
- **Endpoint:** “/api/songs/create”
- **Required Parameters:**
  - {
    “title”: String,
    “releaseDate”: String
    }
- **Required Authorization(s):** ‘Admin’ (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns a *SongResponse* containing the newly created *Song* and a message indicating the song was successfully created.
  - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to create the song (*currently bugged: isn’t returning JwtResponse*)

7. ArtistController

7.1) Get All Artists

- **Request Type:** GET
- **Endpoint:** “/api/artists”
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
  - 200 (success): Returns a list of all *Artists* in the database

7.2) Get Artist By ID

- **Request Type:** GET
- **Endpoint:** "/api/artists/{artistId}"
- **Required Parameters:** "artistId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns an *ArtistResponse* containing the *Artist* that was requested and a message saying the *Artist* was successfully found.
    - 404 (not found): Returns an *ArtistResponse* containing a message saying the *Artist* was not found.

7.3) Get Artists By Name

- **Request Type:** GET
- **Endpoint:** "/api/artists/username/{name}"
- **Required Parameters:** "name": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of all *Artists* based on their 'name'

7.4) Create Artist

- **Request Type:** POST
- **Endpoint:** "/api/artists/create"
- **Required Parameters:**
    - {
      "name": String,
      "genres": List<String>
      }
- **Required Authorization(s):** 'Admin' (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns an *ArtistResponse* containing the newly created *Artist* and a message indicating the artist was successfully created.
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to create the artist (*currently bugged: isn't returning JwtResponse*)

8. PlaylistController

8.1) Get All Playlists

- **Request Type:** GET
- **Endpoint:** "/api/playlists"
- **Required Parameters:** None
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of all *Playlists* in the database

8.2) Get Playlists By Owner Id

- **Request Type:** GET
- **Endpoint:** "/api/playlists/user/{userId}"
- **Required Parameters:** "userId": String - specified in the endpoint URL
- **Required Authorization(s):** None
- **Response:**
    - 200 (success): Returns a list of all *Playlists* associated with a *User*, based on a 'userId'.

8.3) Create Playlist

- **Request Type:** POST
- **Endpoint:** "/api/playlists/create"
- **Required Parameters:** PlaylistRequest(2 options)
    - {
      "name": String,
      "description": String,
      "creatorId": String
      }
    - {
      "name": String,

"creatorId": String
}

- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns a *PlaylistResponse* indicating the *Playlist* was successfully created.
  - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to create the *Playlist* (*currently bugged: isn't returning JwtResponse*)
  - 404 (not found): Returns a *PlaylistResponse* containing a message indicating the *Playlist* was not successfully created.

8.4) Add Song To Playlist

- **Request Type:** PUT
- **Endpoint:** "/api/playlists/add"
- **Required Parameters:** PlaylistRequest
  - {
    "songId": String,
    "playlistId": String
    }
- **Required Authorization(s):** 'User' or 'Admin' (JWT Bearer Token)
- **Response:**
  - 200 (success): Returns a *PlaylistResponse* indicating the *Playlist* was successfully updated.
  - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to update the *Playlist* (*currently bugged: isn't returning JwtResponse*)
  - 404 (not found): Returns a *PlaylistResponse* containing a message indicating the *Playlist* was not successfully updated.

8.5) Remove Song From Playlist

- **Request Type:** PUT
- **Endpoint:** "/api/playlists/remove"
- **Required Parameters:** PlaylistRequest

    - {
      “songId”: String,
      “playlistId”: String
      }
- **Required Authorization(s):** ‘User’ or ‘Admin’ (JWT Bearer Token)
- **Response:**
    - 200 (success): Returns a *PlaylistResponse* indicating the *Playlist* was successfully updated.
    - 401 (unauthorized): Returns a *JwtResponse* indicating that the user was not granted authorization and was not successfully able to update the *Playlist* (*currently bugged: isn’t returning JwtResponse*)
    - 404 (not found): Returns a *PlaylistResponse* containing a message indicating the *Playlist* was not successfully updated.