1. Intro

My Music API is a SpringBoot/Java application that consumes Deezer open API and lets you create your music playlists with your desired tracks and edit them.

This documentation is designed to developers, devops, potential clients and enthusiasts.

2. Syntax Notations

1. Search patterns:

1. Search tracks from deezer library:

This search request is used to get the music ids, which can be used when adding tracks to a playlist.

<scheme>://<url>/<path>?q=<product>

scheme = used protocol: "https".

url = "deezerdevs-deezer.p.rapidapi.com".

path = "search".

product = music title or artist name e.g. "beat it", "michael jackson", "boys don't cry", "50 cent".

2. Search users by id:

This search request is used to find and get users info by inputing user id.

scheme>://<server>:<port>/<path>/<userId>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "users".

userId = the user registered id on repository.

3. Search playlist by user:

This search request is used to find and get all user playlists by inputing user id.

<scheme>://<server>:<port>/<path>/ <userId>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists/user".

userId = the user registered id on repository.

4. Search playlist by id:

This search request is used to find and get a playlists by inputing its id.

<scheme>://<server>:<port>/<path>/<playlistId>/<product>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists ".

playlistId = the playlist registered id on repository.

product = “tracks”.

2. Create Patterns:

1. Create new user:

This create request is used to create new user.

<scheme>://<server>:<port>/<path>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "users".

2. Create new playlist:

This create request is used to create new playlist.

<scheme>://<server>:<port>/<path>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists".

3. Update Patterns:

1. Updating existing playlist:

This update request is used to edit playlists info.

<scheme>://<server>:<port>/<path>/<playlistId>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists".

playlistId = the playlist id from repository.

4. Delete Patterns:

1. Deleting existing user:

This delete request is used to erase a user from repository.

<scheme>://<server>:<port>/<path>/<userId>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "users".

userId = the user id from repository.

2. Deleting existing playlist:

This delete request is used to erase a user from repository.

<scheme>://<server>:<port>/<path>/<playlistId>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists".

playlistId = the playlist id from repository.

3. Deleting track from playlist:

This delete request is used to remove tracks from playlists.

<scheme>://<server>:<port>/<path>/<playlistId>/<product>

scheme = used protocol: "http".

server = "localhost".

port = "8081".

path = "playlists/tracks".

playlistId = the playlist id from repository.

product = “tracks”

5. Response codes

1. Search Patterns:

When using GET requests, the expected successful status code is 200, while the expected fail status should be 400, 403 or 404, depending on error.

2. Create Patterns:

When using create requests, the expected successful status code is 201, while the expected fail status should be 400, 403 or 404, depending on error.

2. Updating Patterns:

When using PUT requests, the expected successful status code is 200, while the expected fail status should be 400, 403 or 404, depending on error.

4. Delete Patterns:

When using DELETE requests, the expected successful status code is 200, while the expected fail status should be 400, 403 or 404, depending on error.

3. Resources

1. Object 1: User

1. User parameters: Long id, List<Playlist> playlists, String userName, String email, String password, List<Profiles> profiles.

2. User operations:

1. Create new user.

1. By sending a POST request to the endpoint “http://localhost:8081/users”, clients can create a new user, inserting some information through the request body, in JSON format. The request body must have the follow key/value variables e.g.: INSERIR AQUI OBSERVAÇÕES SOBRE A VALIDAÇÃO DESSES CAMPOS?

{

    "name": "Pedro",

    "email": "qa@test.com",

    "password": "12345678"

}

The expected success response body for this example is the new user data:

{

    "id": 3,

    "name": "Pedro",

    "email": "qa@test.com"

}

2. Get user by id.

1. By sending a GET request to the endpoint “http://localhost:8081/users/<userId>”, clients can search a user by inserting the user id in the endpoint. This request does not need a request body. The expected success response body for this example is the new user data:

{

    "id": 3,

    "name": "Pedro",

    "email": "qa@test.com"

}

3. Login.

1. After created, users can log into system by sending a POST request to the endpoint “http://localhost:8081/login”. Clients can retrieve the authentication token by inserting email and password through the request body, in JSON format, e.g.: INSERIR AQUI OBSERVAÇÕES SOBRE A VALIDAÇÃO DESSES CAMPOS?

{

    "email": "qa@test.com",

    "password": "12345678"

}

The expected success response body for this example is the token and its type:

{

    "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJNeSBtdXNpYyBhcGkiLCJzdWIiOiIzIiwiaWF0IjoxNjQ0MjYzMjA3LCJleHAiOjE2NDQyNjUwMDd9.tJSvzuQhgd4Jpx1GZTtY-xiSUMyfnS5e47qsBpsrjDA",

    "type": "Bearer"

}

3. Delete user by id.

1. By sending a DELETE request to the endpoint “http://localhost:8081/users/<userId>”, clients can remove a user from the database repository by inserting the user id in the endpoint. This request does not need a request body. The expected success response body for this request is:

“User <userId> deleted.”

2. Object 2: Playlist

1. Playlist parameters: Long id, String title, String description, User user, List<String> trackIds.

2. Playlist operations:

1. Create new playlist.

1. By sending a POST request to the endpoint “http://localhost:8081/playlists”, clients can create a new playlist, inserting some information through the request body, in JSON format. The request body must have the follow key/value variables e.g.: INSERIR AQUI OBSERVAÇÕES SOBRE A VALIDAÇÃO DESSES CAMPOS?

{

    "title": "title",

    "description": "my first playlist",

    "userId" : "1"

}

The expected success response body for this example is a dto of the playlist created:

{

    "id": 8,

    "title": "title",

    "description": "my first playlist",

    "userId": 5,

    "tracks": []

}

2. Get user playlists.

1. By sending a GET request to the endpoint “http://localhost:8081/playlists/user/<userId>”, clients can get all the playlists of a user inserting their id in the endpoint. This request does not need a request body. The expected success response body of this request is the list of the user’s playlist, e.g.:

[

    {

        "id": 8,

        "title": "title",

        "description": "my first playlist",

        "userId": 5,

        "tracks": []

    },

    {

        "id": 9,

        "title": "title 2",

        "description": "my second playlist",

        "userId": 5,

        "tracks": []

    }

]

3. Edit playlist title and description.

1. By sending a PUT request to the endpoint “http://localhost:8081/playlists/<playlistId>”, clients can update a existing playlist, inserting some information through the request body, in JSON format. The request body must have the follow key/value variables e.g.: INSERIR AQUI OBSERVAÇÕES SOBRE A VALIDAÇÃO DESSES CAMPOS?

{

    "title": "another title",

    "description": "my playlist",

    "userId" : "5"

}

The expected success response body for this example is a dto of the playlist updated:

{

    "id": 8,

    "title": "another title",

    "description": "my playlist",

    "userId": 5,

    "tracks": []

}

4. Delete playlist.

1. By sending a DELETE request to the endpoint “http://localhost:8081/playlists/<playlistId>”, clients can delete an existing playlist from the database repository, inserting the playlistId in the endpoint. This request does not need a request body. The expected success response body for this example is:

“Playlist <playlistId> deleted.”

3. Object 3: Tracks

1. Tracks parameters: String id, String title, String duration, ArtistDto artist, AlbumDto album.

2. Tracks operations:

1. Search tracks in the external API (Deezer).

1. By sending a DELETE request to the endpoint “https://deezerdevs-deezer.p.rapidapi.com/search?q=<keyword>”, clients can look for music, albums, artists in the deezer repository, inserting the keyword in the endpoint. This request does not need a request body. The expected response body for this example is a list of tracks that have that keyword, by relevance order. Clients will need the track’s “id” value from the response in order to add that music to a playlist. The follow response is received when <keyword> = peaches.

{

    "data": [

        {

            "id": 1637412872,

            "readable": **true**,

            "title": "Peaches",

            "title\_short": "Peaches",

            "title\_version": "",

            "link": "https://www.deezer.com/track/1637412872",

            "duration": 198,

            "rank": 279052,

            "explicit\_lyrics": **true**,

            "explicit\_content\_lyrics": 1,

            "explicit\_content\_cover": 0,

            "preview": "https://cdns-preview-3.dzcdn.net/stream/c-39ac9995ca433ba9ccdf3721b87152f1-8.mp3",

            "md5\_image": "626c80c646832b8240a126d9597accbb",

            "artist": {

                "id": 288166,

                "name": "Justin Bieber",

                "link": "https://www.deezer.com/artist/288166",

                "picture": "https://api.deezer.com/artist/288166/image",

                "picture\_small": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/56x56-000000-80-0-0.jpg",

                "picture\_medium": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/250x250-000000-80-0-0.jpg",

                "picture\_big": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/500x500-000000-80-0-0.jpg",

                "picture\_xl": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/1000x1000-000000-80-0-0.jpg",

                "tracklist": "https://api.deezer.com/artist/288166/top?limit=50",

                "type": "artist"

            },

            "album": {

                "id": 291212762,

                "title": "AFTER Ski Pt.1",

                "cover": "https://api.deezer.com/album/291212762/image",

                "cover\_small": "https://e-cdns-images.dzcdn.net/images/cover/626c80c646832b8240a126d9597accbb/56x56-000000-80-0-0.jpg",

                "cover\_medium": "https://e-cdns-images.dzcdn.net/images/cover/626c80c646832b8240a126d9597accbb/250x250-000000-80-0-0.jpg",

                "cover\_big": "https://e-cdns-images.dzcdn.net/images/cover/626c80c646832b8240a126d9597accbb/500x500-000000-80-0-0.jpg",

                "cover\_xl": "https://e-cdns-images.dzcdn.net/images/cover/626c80c646832b8240a126d9597accbb/1000x1000-000000-80-0-0.jpg",

                "md5\_image": "626c80c646832b8240a126d9597accbb",

                "tracklist": "https://api.deezer.com/album/291212762/tracks",

                "type": "album"

            },

            "type": "track"

        },

        {

            "id": 1280165222,

            "readable": **true**,

            "title": "Peaches",

            "title\_short": "Peaches",

            "title\_version": "",

            "link": "https://www.deezer.com/track/1280165222",

            "duration": 198,

            "rank": 992645,

            "explicit\_lyrics": **true**,

            "explicit\_content\_lyrics": 1,

            "explicit\_content\_cover": 1,

            "preview": "https://cdns-preview-1.dzcdn.net/stream/c-179bee1bfc65440d7df06d2246209699-4.mp3",

            "md5\_image": "87468622c8e7ac9dce7b541be136aa4c",

            "artist": {

                "id": 288166,

                "name": "Justin Bieber",

                "link": "https://www.deezer.com/artist/288166",

                "picture": "https://api.deezer.com/artist/288166/image",

                "picture\_small": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/56x56-000000-80-0-0.jpg",

                "picture\_medium": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/250x250-000000-80-0-0.jpg",

                "picture\_big": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/500x500-000000-80-0-0.jpg",

                "picture\_xl": "https://e-cdns-images.dzcdn.net/images/artist/22dd86b628a03d8dad3c7dfb33320a91/1000x1000-000000-80-0-0.jpg",

                "tracklist": "https://api.deezer.com/artist/288166/top?limit=50",

                "type": "artist"

            },

            "album": {

                "id": 215962322,

                "title": "Justice",

                "cover": "https://api.deezer.com/album/215962322/image",

                "cover\_small": "https://e-cdns-images.dzcdn.net/images/cover/87468622c8e7ac9dce7b541be136aa4c/56x56-000000-80-0-0.jpg",

                "cover\_medium": "https://e-cdns-images.dzcdn.net/images/cover/87468622c8e7ac9dce7b541be136aa4c/250x250-000000-80-0-0.jpg",

                "cover\_big": "https://e-cdns-images.dzcdn.net/images/cover/87468622c8e7ac9dce7b541be136aa4c/500x500-000000-80-0-0.jpg",

                "cover\_xl": "https://e-cdns-images.dzcdn.net/images/cover/87468622c8e7ac9dce7b541be136aa4c/1000x1000-000000-80-0-0.jpg",

                "md5\_image": "87468622c8e7ac9dce7b541be136aa4c",

                "tracklist": "https://api.deezer.com/album/215962322/tracks",

                "type": "album"

            },

            "type": "track"

        },

… and so it goes.

2. Add tracks to existing playlist.

1. By sending a POST request to the endpoint “http://localhost:8081/playlists/<playlistId>/tracks”, clients can add tracks to an existing playlist, inserting the playlist id in the endpoint and the deezer’s track id through the request body, in JSON format, e.g.:

{

    "id": "1280165222"

}

The expected success response body for this example is a track object:

{

    "id": "1280165222",

    "title": "Peaches",

    "duration": "198",

    "artist": {

        "name": "Justin Bieber"

    },

    "album": {

        "title": "Justice"

    }

}

3. Get playlist tracks.

1. By sending a GET request to the endpoint “http://localhost:8081/playlists/<playlistId>/tracks”, clients can get all the tracks added to a playlist, inserting the playlist id in the endpoint. This request does not need a request body. The expected success response body of this request is the playlist object with a list of the playlist’s tracks, e.g.:

{

    "id": 8,

    "title": "another title",

    "description": "my playlist",

    "userId": 5,

    "tracks": [

        {

            "id": "1280165222",

            "title": "Peaches",

            "duration": "198",

            "artist": {

                "name": "Justin Bieber"

            },

            "album": {

                "title": "Justice"

            }

        },

        {

            "id": "4763165",

            "title": "Beat It",

            "duration": "257",

            "artist": {

                "name": "Michael Jackson"

            },

            "album": {

                "title": "Michael Jackson's This Is It"

            }

        }

    ]

}

4. Delete tracks from playlist.

1. By sending a DELETE request to the endpoint “http://localhost:8081/playlists/<playlistId>/tracks”, clients can remove a track from the playlist by inserting the playlist id in the endpoint and the track id in the response body, in JSON format, e.g.:

{

    "id": "4763165"

}

The expected success response body for this request is, e.g.:

“Track 4763165 deleted from playlist 8.”

4. Representação de Erros

1. Parâmetros das Respostas de Erros

5. Mudanças entre Versões

This documentation refers to the base version of this API (1.0.1).