Web Engineering - API Specification

E. Waterink (s3417611), T. de Vries (s2367610), I. Oralin (s4171446)

September 2019
v1.0

1 Overview

This document specifies an API for the Million Song Dataset. It covers everything there is to know, from obtaining artist data to adding a song. Examples are provided for each set of API calls. Example responses for example requests used in this document are not always the real ones.

Please note that we are considering moving our documentation over to Swagger, so our documentation style may change in the future.

2 Split data into resources

1. List of artists, subsetted by (a combination of) name and genre
2. List of songs, subsetted by (a combination of) id, artist and genre. Provides all available information for each song.
3. Descriptive statistics of songs by a specific artist, optionally filtered by year.
4. Frequency information for the number of songs per genre, optionally above a certain popularity threshold.
5. Frequency information for the key of songs in the dataset, optionally subsetted by (a combination of) genre and popularity threshold.

3 Resources

3.1 Artists

3.1.1 Obtaining filtered data

It is possible to get all artists in the dataset, optionally filtered by artist name and/or genre.

Example requests:

```
# Get all artists
GET /artists

# Filtered by artist name and/or genre
GET /artists?name=
GET /artists?genre=
GET /artists?name=&genre=
GET /artists?genre=hip-hop
```

Example response:

```
{
   "artists" :
   [
    {
     "name" : "Casual",
     "genre" : "hip hop",
     "familiarity" : 0.58123
     "hotttness" : 0.49123
     [other_attributes]
    },
     {
     "name" : "Redman",
     "genre" : "hip hop",
     "familiarity" : 0.53456
     "hotttness" : 0.23134
     [other_attributes]
    }
   ]
}
```

3.1.2 Obtaining sorted data

It is possible to get an ordered collection of artists ranked by their popularity (hotttness index) from more to less popular, with the possibility to subset this order, e.g. the top 50 artists.

Example request:

```
# Get artists sorted by their popularity(hotness)
GET /artists?sort=hotness
```

Example response:

```
{
    "artists" :
    [
        {
            "name" : "Seventh Day Slumber",
            "genre" : "christian rock",
            "familiarity" : 0.92321
            "hotttness" : 0.99999
            // [other_attributes]
        },
        {
            "name" : "Winston Reedy",
            "genre" : "lovers rock",
            "familiarity" : 0.96753
            "hotttness" : 0.99998
            // [other_attributes]
        }
    ]
}
```

Errors:
400: BAD REQUEST - Invalid sort parameter

3.2 Songs

3.2.1 Obtaining data

```
GET /songs
```

Example response:

```
[
    {
        "id": "string",
        "familiarity": 0,
        "hotness": 0,
        "latitude": 0,
        "longitude": 0,
        "location": 0,
        "name": "string",
        "similar": 0,
        "terms": "string",
        "terms_freq": 0
    }
]
```

3.2.2 Adding songs

```
POST /songs
```

Accepted representation is a JSON or CSV file
Errors:

3.2.3 Modifying songs

```
PUT /songs/{songId}
```

Accepted representation is a JSON or CSV file
Errors:
404: Song not found

3.2.4 Deleting songs

`DELETE /songs/{songId}`

Errors:
404: Song not found

3.3 Statistics

3.3.1 Obtaining data

It is possible to get statistical data (mean, median, standard deviation) about artist popularity. To get this data for one artist, use the artist id you obtained from the /artists endpoint.

Example request:

`GET /stats?artist=AR10USD1187B99F3F1`

Example response:

```
{
    "title": "Statistical data for Tweeterfriendly Music",
    "description": "descriptive statistics (mean, median, standard
        deviation) for
    the popularity of the songs for a particular artist with an optional
        filter by year"
    "mean": 0.605507136,
    "median": 0.605507136,
    "standard deviation": 0
}
```

It is also possible to get the popularity of an artist in a specific year. **Example request:**

`GET /stats?artist=AR10USD1187B99F3F1&year=0`

Example response:

```
{
    "title": "Statistical data for Tweeterfriendly Music",
    "description": "descriptive statistics (mean, median, standard
        deviation) for
    the popularity of the songs for a particular artist with an optional
        filter by year"
    "mean": 0.605507136,
    "median": 0.605507136,
    "standard deviation": 0
}
```

Errors:
400: BAD REQUEST - Invalid artist
400: BAD REQUEST - Invalid year

3.4 Genres

Returns a frequency table of genres. Only shows genres above a given threshold. Frequency values below the threshold are summed and returned as ’other’.

Example request:

```
# Get genres frequency of occurrence
GET /genres?threshold=20
```

Example response:

```
{
  "hip hop" : 120
  "rock" : 230
  "jazz" : 109
  . . . . .
  "other" : 98
}
```

Errors:
400: BAD REQUEST - Invalid threshold

3.5 Keys

It is possible to get the distribution of keys used in the songs, with an option to subset by genre and/or the popularity of songs.

Example request:

```
GET /keys?genre=hip%20hop&threshold=0.5
```

Example response:

```
{
   "title": "Key distribution data",
   "description": "Distribution data for keys of songs in the Million
       Songs Dataset,
   optionally subsetted by genre and/or popularity"
   "genre": hip hop,
   "threshold": 0.5,
   "keys": [ "0": 0,
       "1": 5,
       "2": 2,
       "3": 8,
       "4": 16,
       "5": 35,
       "6": 46,
       "7": 18,
       "8": 11,
       "9": 7,
       "10": 2,
       "11": 0 ]
}
```

Errors:
400: BAD REQUEST - Invalid Threshold

4 Pagination

Any request with more than 50 results will be paginated with 50 results maximum per page.

Example request:

`GET /artists?page=2`

Example response:

```
{
    . . . . . .
    "page-current": 2
    "page-next": "/artists?page=3"
    "page-previous": "/artists?page=1"
    . . . . . .
}
```

Errors:
400: BAD REQUEST - Invalid Year
400: BAD REQUEST - No such page

5 Content Negotiation

The API serves both JSON and CSV representations of resources, with JSON as the default. To receive CSV, attach a Accept-Encoding header specifying you wish to receive CSV.

Example request:

```
    GET /artists
    . . . . .
    Accept-Encoding: csv
    . . . . .
```