# Web Engineering
# API Design

Group 13
Maria-Sophia Stefan (s3413896) & Anda-Amelia Palamariuc (s3443817)

October 3, 2019

## Introduction

This is the first version of the API Design document, which outlines the API endpoints that fit the initial requested functionalities. In addition to this, the second endpoint supports deleting a song and updating the popularity index of a song. As far as the third functionality is concerned, we have decided to split it in two different endpoints (3, respectively 4).

Throughout this document, we will use **:parameter** notation to denote path parameters. Moreover, each endpoint supports both JSON and CSV representations of the resources which is available by the query parameter `?content-typ=csv`. Thus, if not specified, the default representation is JSON.

## API Endpoints

1.
**URI:** `HTTPS GET .../artists/?name='artist.name'&terms='artist.terms'`
*Example URIs:*

1. Retrieve all artists: `HTTPS GET .../artists`
2. Retrieve all artists filtered by name:
`HTTPS GET .../artists/?name=Van%20Halen`
3. Retrieve all artists filtered by genre (terms):
`HTTPS GET .../artists/?terms=indie%20rock`
4. Retrieve all artists filtered by name and genre (terms):
`HTTPS GET .../artists/?name=Savage%20Garden&terms=easy%20listening`

**Description:** This request will retrieve a representation of all the artists in the data set if no query parameters (i.e. `name` and `terms`) are specified, as per example 1. The artists in the data set can optionally be filtered by `name` and/or `terms`, the latter query parameter corresponding to the genre, as in examples 2-4.
**Status codes:**

- **Success:** 200
- **Bad request:** 400
- **Resource not available:** 500

**Responses:**

– **Status code 200 (for example 4 above):**

*CSV*

```
"familiarity","hotttnesss","id","latitude","location","longitude","name","similar
","terms","terms_freq"
"0.728209707","0.538070295","ARBGWMW1187B9AEA3E","0","0","0","Savage
Garden","0","easy listening","0.896826245"
```

*JSON*

```
{
    "familiarity": 0.728209707,
    "hotttnesss": 0.538070295,
    "id": "ARBGWMW1187B9AEA3E",
    "latitude": 0,
    "location": 0,
    "longitude": 0,
    "name": "Savage Garden",
    "similar": 0,
    "terms": "easy listening",
    "terms_freq": 0.896826245
}
```

– **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

– **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```

2.

**URI:** `HTTPS GET .../songs/song/:id`

*Example URI:* `HTTPS GET .../songs/song/SOEHWGF12A6D4F8B2B`

**Description:** This request will retrieve all the available information of a specific song identified by its data set id, as illustrated by the path parameter `:id`.

**Status codes:**

- **Success:** 200
- **Bad request:** 400
- **Resource not available:** 500

**Responses:**

- **Status code 200 (for the above example):**

*CSV*

```
"artist_mbtags","artist_mbtags_count","bars_confidence","bars_start",
"beats_confidence","beats_start","duration","end_of_fade_in","hotttnesss","id","key",
"key_confidence","loudness","mode","mode_confidence","start_of_fade_out",
"tatums_confidence","tatums_start","tempo","time_signature",
"time_signature_confidence","title","year"
  "0","1","0.497","0.71063","1","0.10911","217.36444","0.224","0.73738531",
"SOEHWGF12A6D4F8B2B",
    "10","0.805","−5.152","0","0.688","212.161","0.538","0.10911","100.023","4","0.937","0","0"
```

*JSON*

```
{
    "artist_mbtags": 0,
    "artist_mbtags_count": 1,
    "bars_confidence": 0.497,
    "bars_start": 0.71063,
    "beats_confidence": 1,
    "beats_start": 0.10911,
    "duration": 217.36444,
    "end_of_fade_in": 0.224,
    "hotttnesss": 0.73738531,
    "id": "SOEHWGF12A6D4F8B2B",
    "key": 10,
    "key_confidence": 0.805,
    "loudness": −5.152,
    "mode": 0,
    "mode_confidence": 0.688,
    "start_of_fade_out": 212.161,
    "tatums_confidence": 0.538,
    "tatums_start": 0.10911,
    "tempo": 100.023,
    "time_signature": 4,
    "time_signature_confidence": 0.937,
    "title": 0,
    "year": 0
}
```

- **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

- **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{ "response": "Resource not available"}
```

**URI:** `HTTPS DELETE .../songs/song/:id`
*Example URI:* `HTTPS DELETE .../songs/song/SOIAZJW12AB01853F1`
**Description:** This call will remove the song resource identified by specified path parameter `:id` from its parent (i.e. songs).
**Status codes:**

- **Success:** 200

**Responses:**

- **Status code 200 (for the above example):**

*CSV*

```
"response"
"Song with id SOIAZJW12AB01853F1 successfully removed"
```

*JSON*

```
{ "response": "Song with id SOIAZJW12AB01853F1 successfully removed"}
```

**URI:** `HTTPS PUT .../songs/song/:id/?hotttnesss=`
*Example URI:* `HTTPS POST .../songs/song/SOUDSGM12AC9618304/?hotttnesss=0.5786`
**Description:** This call will update the hotttnesss (popularity) index for the song resource identified by the path parameter `:id`. If the query parameter (`hotttnesss`) is left NULL, then the update cannot be performed.
**Status codes:**

- **Success:** 200

**Responses:**

- **Status code 200 (for the above example):**

*CSV*

```
"response"
"Update successful for song with id SOUDSGM12AC9618304"
```

*JSON*

```
{ "response": "Update successful for song with id
SOUDSGM12AC9618304" }
```

3.

**URI:** `HTTPS GET ...songs/artists/:artists.id/?year='song.year'`
*Example URI:*

1. Retrieve all songs by a specific artist
   `HTTPS GET .../songs/artists/ARXPPEY1187FB51DF4`
2. Retrieve all songs by a specific artist and in a specific year:
   `HTTPS GET .../song/artists/ARXPPEY1187FB51DF4/?year=1995`

**Description:** This request will retrieve a representation of all songs in the data set belonging to a specific artist, identified by the path parameter `artist.id` (example 1). The songs of the specific artist can be optionally filtered by the query parameter `year` (example 2).
**Status codes:**

- **Success:** 200
- **Bad request:** 400
- **Resource not available:** 500

**Responses:**

- **Status code 200 (for example 2):**

*CSV*

```
"artist_mbtags","artist_mbtags_count","bars_confidence","bars_start","
    beats_confidence","beats_start","duration","end_of_fade_in","hotttnesss","id","key
    ","key_confidence","loudness","mode","mode_confidence","start_of_fade_out","
    tatums_confidence","tatums_start","tempo","time_signature","
    time_signature_confidence","title","year"

 "0","4","0.024","0.77199","0.878","0.29091","249.99138","0.131","−1","
    SOGRKKB12A8C14595A","11",
  "0.383","−2.617","1","0.169","243.879","0.107","0.04614","128.702","4","0.495","0","1995"

 "0","4","0.13","1.7206","0.738","0.62842","272.71791","0.189","−1","
    SOYIUPV12AF72AC56B","10",
  "0.596","−6.779","0","0.545","268.771","0.335","0.22139","111.132","4","1","0","1995"
```

*JSON*

```
[{
    "artist_mbtags": 0,
    "artist_mbtags_count": 4,
    "bars_confidence": 0.024,
    "bars_start": 0.77199,
    "beats_confidence": 0.878,
    "beats_start": 0.29091,
    "duration": 249.99138,
    "end_of_fade_in": 0.131,
    "hotttnesss": −1,
    "id": "SOGRKKB12A8C14595A",
    "key": 11,
    "key_confidence": 0.383,
    "loudness": −2.617,
    "mode": 1,
    "mode_confidence": 0.169,
    "start_of_fade_out": 243.879,
    "tatums_confidence": 0.107,
    "tatums_start": 0.04614,
    "tempo": 128.702,
    "time_signature": 4,
    "time_signature_confidence": 0.495,
    "title": 0,
    "year": 1995
},

{
    "artist_mbtags": 0,
    "artist_mbtags_count": 4,
    "bars_confidence": 0.13,
    "bars_start": 1.7206,
    "beats_confidence": 0.738,
    "beats_start": 0.62842,
    "duration": 272.71791,
    "end_of_fade_in": 0.189,
    "hotttnesss": −1,
    "id": "SOYIUPV12AF72AC56B",
    "key": 10,
    "key_confidence": 0.596,
    "loudness": −6.779,
    "mode": 0,
    "mode_confidence": 0.545,
    "start_of_fade_out": 268.771,
    "tatums_confidence": 0.335,
    "tatums_start": 0.22139,
    "tempo": 111.132,
    "time_signature": 4,
    "time_signature_confidence": 1,
    "title": 0,
    "year": 1995
}]
```

- **Status code 400:**

  *CSV*

  ```
  "response"
  "Bad request"
  ```

  *JSON*

  ```
  {"response": "Bad request"}
  ```

- **Status code 500:**

  *CSV*

  ```
  "response"
  "Resource not available"
  ```

  *JSON*

  ```
  { "response": "Resource not available"}
  ```

4. **URI:** `HTTPS GET .../songs/year/:year`
   *Example URI:* `HTTPS GET .../songs/year/1954`
   **Description:** This request will retrieve all the available information of the songs in the data set that were released in a specific year (denoted by the path parameter `:year`).
   **Status codes:**

   - **Success:** 200
   - **Bad request:** 400
   - **Resource not available:** 500

   **Responses:**

   - **Status code 200 (for the above example):**

     *CSV*

     ```
     "artist_mbtags","artist_mbtags_count","bars_confidence","bars_start","
      beats_confidence","beats_start","duration","end_of_fade_in","hotttnesss","id","key
      ","key_confidence","loudness","mode","mode_confidence","start_of_fade_out","
      tatums_confidence","tatums_start","tempo","time_signature","
      time_signature_confidence","title","year"
        "0","0","0.181","0.96974","0.961","0.28679","190.64118","0.223","0.427446572","
      SOEWMAK12A8C13FD07
      ","11","0.253","−9.302","1","0.228","185.202","1","0.28679","98.018","3","1","0","1954"
     ```

     *JSON*

     ```
     {
         "artist_mbtags": 0,
         "artist_mbtags_count": 0,
         "bars_confidence": 0.181,
     ```

```
        "bars_start": 0.96974,
        "beats_confidence": 0.961,
        "beats_start": 0.28679,
        "duration": 190.64118,
        "end_of_fade_in": 0.223,
        "hotttnesss": 0.427446572,
        "id": "SOEWMAK12A8C13FD07",
        "key": 11,
        "key_confidence": 0.253,
        "loudness": −9.302,
        "mode": 1,
        "mode_confidence": 0.228,
        "start_of_fade_out": 185.202,
        "tatums_confidence": 1,
        "tatums_start": 0.28679,
        "tempo": 98.018,
        "time_signature": 3,
        "time_signature_confidence": 1,
        "title": 0,
        "year": 1954
    }
```

- **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

- **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```

5.
**URI:** `HTTPS GET .../artists/songs/terms/:terms`
*Example URI:* `HTTPS GET .../artists/songs/terms/batucada`
**Description:** This request will retrieve all the available information of the songs in the dataset that belong to an artist in a specific genre (denoted by the path parameter `:terms`). Note that, the backend will fetch all the artists belonging to the specified genre (terms) and then return all the songs of such artists.

**Status codes:**

- **Success:** 200
- **Bad request:** 400
- **Resource not available:** 500

**Responses:**

- **Status code 200 (for the above example):**

*CSV*

```
        "artist_mbtags","artist_mbtags_count","bars_confidence","bars_start","
beats_confidence","beats_start","duration","end_of_fade_in","hotttnesss","id","key
","key_confidence","loudness","mode","mode_confidence","start_of_fade_out","
tatums_confidence","tatums_start","tempo","time_signature","
time_signature_confidence","title","year"

  "0","0","0.265","1.32015","0","0.61203","188.89098","0.351","0.215080319","
SOHDIVC12A6D4F8DA7
","4","0.367","−8.239","0","0.372","174.481","0.859","0.14007","77.41","3","1","0","0"
```

*JSON*

```
{
    "artist_mbtags": 0,
    "artist_mbtags_count": 0,
    "bars_confidence": 0.265,
    "bars_start": 1.32015,
    "beats_confidence": 0,
    "beats_start": 0.61203,
    "duration": 188.89098,
    "end_of_fade_in": 0.351,
    "hotttnesss": 0.215080319,
    "id": "SOHDIVC12A6D4F8DA7",
    "key": 4,
    "key_confidence": 0.367,
    "loudness": −8.239,
    "mode": 0,
    "mode_confidence": 0.372,
    "start_of_fade_out": 174.481,
    "tatums_confidence": 0.859,
    "tatums_start": 0.14007,
    "tempo": 77.41,
    "time_signature": 3,
    "time_signature_confidence": 1,
    "title": 0,
    "year": 0
}
```

– **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

– **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```

6.

**URI:** `HTTPS GET .../artists/hotttnesssdesc/?maxlimit='subsetlimit'`
*Example URI:*

1. Retrieve an ordering of the artists ranked by their popularity (in a descending order):
   `HTTPS GET .../artists/hotttnesssdesc'`
2. Retrieve a subset (with a maximum limit) of the descending ordering of the artists ranked by popularity:
   `HTTPS GET .../artists/hotttnesssdesc/?maxlimit=2`

**Description:** This request will retrieve all the artists ranked by their popularity in a descending order (from most to least popular), as per example 1. Optionally, this ordering can be subsetted given a maximum limit (query parameter `maxlimit`), as illustrated by example 2.
**Status codes:**

– **Success:** 200

– **Bad request:** 400

– **Resource not available:** 500

**Responses:**

– **Status code 200 (for example 2)**:

*CSV*

```
" familiarity "," hotttnesss "," id "," latitude "," location "," longitude "," name","similar ","
    terms","terms_freq"
"0.877213746","1.082502557","ARRH63Y1187FB47783","0","0","0","Kanye West","0","hip
    hop","1"
"0.845601887","1.005941966","ARTDQRC1187FB4EFD4","0","0","0","Black Eyed Peas
    ","0","hip hop","1"
```

*JSON*

```
[{
      " familiarity ": 0.877213746,
       "hotttnesss": 1.082502557,
       "id": "ARRH63Y1187FB47783",
       " latitude ": 0,
       " location ": 0,
       "longitude": 0,
       "name": "Kanye West",
       " similar ": 0,
       "terms": "hip hop",
       "terms_freq": 1
     },
     {
        " familiarity ": 0.845601887,
        "hotttnesss": 1.005941966,
        "id": "ARTDQRC1187FB4EFD4",
        " latitude ": 0,
        " location ": 0,
        "longitude": 0,
        "name": "Black Eyed Peas",
        " similar ": 0,
        "terms": "hip hop",
        "terms_freq": 1,
}]
```

– **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

– **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```

7.
**URI:** `HTTPS GET .../songs/hotttnesssdesc/year/:year/?maxlimit='subsetlimit'`
*Example URI:*

1. Retrieve an ordering of the songs in a specific year ranked by their popularity (in a descending order):
   `HTTPS GET .../songs/hotttnesssdesc/1998`
2. Retrieve a subset (with a maximum limit) of the descending ordering of the songs in a specific year:
   `HTTPS GET .../songs/hotttnesssdesc/2010/?maxlimit=2`

**Description:** This request will retrieve all the songs in a specific year (as denoted by the path parameter `:year`) ranked by their popularity in a descending order (from most to least popular), as per example 1. Optionally, this ordering can be subsetted given a maximum limit (query parameter `maxlimit`), as illustrated in example 2.
**Status codes:**

- **Success:** 200
- **Bad request:** 400
- **Resource not available:** 500

**Responses:**

- **Status code 200 (for example 2):**

*CSV*

```
"artist_mbtags","artist_mbtags_count","bars_confidence","bars_start",
"beats_confidence","beats_start","duration","end_of_fade_in","hotttnesss","id","key",
"key_confidence","loudness","mode","mode_confidence","start_of_fade_out",
"tatums_confidence","tatums_start","tempo","time_signature",
"time_signature_confidence","title","year"

  "0","1","0.022","1.1388","0","0.59288","269.63546","5.283","1",
"SOULTKQ12AB018A183","10","0.403",
  "−5.388","1","0.411","258.461","0.68","0.31134","104.038","4","0.742","0","2010"

  "0","2","0.055","0.91722","1","0.35764","196.20526","0.334","0.918534291","
    SOGCDYR12AC961854A
    ","2","1","−7.045","1","0.706","179.664","1","0.07644","107.042","4","1","0","2010"
```

*JSON*

```
      [    {
              "artist_mbtags": 0,
              "artist_mbtags_count": 1,
              "bars_confidence": 0.022,
              "bars_start": 1.1388,
              "beats_confidence": 0,
              "beats_start": 0.59288,
              "duration": 269.63546,
              "end_of_fade_in": 5.283,
              "hotttnesss": 1,
              "id": "SOULTKQ12AB018A183",
              "key": 10,
        "key_confidence": 0.403,
        "loudness": −5.388,
        "mode": 1,
        "mode_confidence": 0.411,
        "start_of_fade_out": 258.461,
        "tatums_confidence": 0.68,
        "tatums_start": 0.31134,
        "tempo": 104.038,
        "time_signature": 4,
        "time_signature_confidence": 0.742,
        "title": 0,
        "year": 2010
    },

    {
        "artist_mbtags": 0,
        "artist_mbtags_count": 2,
        "bars_confidence": 0.055,
        "bars_start": 0.91722,
        "beats_confidence": 1,
        "beats_start": 0.35764,
        "duration": 196.20526,
        "end_of_fade_in": 0.334,
        "hotttnesss": 0.918534291,
        "id": "SOGCDYR12AC961854A",
        "key": 2,
        "key_confidence": 1,
        "loudness": −7.045,
        "mode": 1,
        "mode_confidence": 0.706,
        "start_of_fade_out": 179.664,
        "tatums_confidence": 1,
        "tatums_start": 0.07644,
        "tempo": 107.042,
        "time_signature": 4,
        "time_signature_confidence": 1,
        "title": 0,
        "year": 2010
    }   ]
```

– **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

– **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```

8.

**URI:** `HTTPS GET .../artists/artist/songs/statistics/:id/?year='song.year'`
*Example URI:*

1. Retrieve statistics for the songs of a specific artist:
   `HTTPS GET .../artists/artist/songs/statistics/ARXPPEY1187FB51DF4`
2. Retrieve statistics for the songs of a specific artist in a specific year:
   `HTTPS GET .../artists/artist/songs/statistics/ARXPPEY1187FB51DF4/?year=1962`

**Description:** This request will retrieve descriptive statistics (mean, median, standard deviation) for the songs of a given artist, identified by the path parameter `:id` (example 1) with an optional filter by year (query parameter `year`) (example 2).
**Status codes:**

– **Success:** 200
– **Bad request:** 400
– **Resource not available:** 500

**Responses:**

– **Status code 200 (for example 1)**:

*CSV*

```
"mean","median","standard deviation"
"−0.65036704033333","−1","0.65485286018423"
```

*JSON*

```
{
    "mean": −0.65036704033333,
    "median": −1,
    "standard deviation": 0.65485286018423
}
```

– **Status code 400:**

*CSV*

```
"response"
"Bad request"
```

*JSON*

```
{"response": "Bad request"}
```

– **Status code 500:**

*CSV*

```
"response"
"Resource not available"
```

*JSON*

```
{"response": "Resource not available"}
```