Web Engineering
API Design

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

October 24, 2019

Introduction

This is the final version of the API Design document, which outlines the API endpoints that fit the initial requested functionalities. In addition to this, the third endpoint supports deleting updating and creating a song, besides fetching a song based on the id. As far as the third functionality is concerned, we have decided to split it in two different endpoints (4, respectively 5). Moreover, functionalities 1 and 5 share the same endpoint, namely 2. Along with the the endpoint that also supports CUD, we have introduced endpoint 8, which is particularly useful when creating a new song entry in the database. Moreover, we created endpoint 9 which will fetch all the songs in the database, which we later used for rendering purposes in the front-end part of the application.

By using a third party API (i.e. **pixabay** [1]) our application delivers advanced features, that is the ability to visualize the picture of an artist.

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 `?contentType=csv`. Thus, if not specified, the default representation is JSON.

Last but not least, the API we designed lies on the third maturity level, characterized by having hypermedia controls and stateless interactions on top of the lower levels features such as: uniform interface, self-describing messages and resource identification.

API Endpoints

1. **URI:** `HTTP POST .../populateDB`
   Example URI: `HTTP POST .../populateDB`
   Description: This request will create all the initial resources by creating and populating the 3 tables: artists, releases, songs.
   Status codes:
   - **Success:** 200
   - **Internal server error:** 500

[1] `https://pixabay.com/api/docs/`

Responses:

- **Status code 200:**

CSV

```
"response"
"Populating was successful!"
```

JSON

```
{"response": "Populating was successful!"}
```

- **Status code 500:**

CSV

```
"response"
"Request unfulfilled by server"
```

JSON

```
{"response": "Request unfulfilled by server"}
```

2.

URI: `HTTP GET .../artists`

2.1 This is the implementation for functionality nr. **1** from the Project Description.

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

1. Retrieve all artists: `HTTP GET .../artists`
2. Retrieve all artists filtered by name:
   `HTTP GET .../artists/?name=Van%20Halen`
3. Retrieve all artists filtered by genre (terms):
   `HTTP GET .../artists/?terms=indie%20rock`
4. Retrieve all artists filtered by name and genre (terms):
   `HTTP 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:** 404

Responses:

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

CSV

```
data/ID,data/FAMILIARITY,data/HOTTTNESSS,data/LATITUDE,data/
    LOCATION,data/LONGTITUDE,data/NAME,data/SIMILAR,data/TERMS,
    data/FREQ,links/rel,links/href,
"ARBGWMW1187B9AEA3E",0.728209707,0.538070295,0,0,0,"Savage Garden",0,"
    easy listening",0.896826245,"self","artists/?name=Savage Garden","artistSongs
    ","songs/artists/ARBGWMW1187B9AEA3E","artistStatistics","artists/
    statistics/ARBGWMW1187B9AEA3E","artistReleases","releases/artist/
    ARBGWMW1187B9AEA3E",
```

JSON


```
[
   {
      "data": {
         "ID": "ARBGWMW1187B9AEA3E",
         "FAMILIARITY": 0.728209707,
         "HOTTTNESSS": 0.538070295,
         "LATITUDE": 0,
         "LOCATION": 0,
         "LONGTITUDE": 0,
         "NAME": "Savage Garden",
         "SIMILAR": 0,
         "TERMS": "easy listening",
         "FREQ": 0.896826245
      },
      "links": [
         {
            "rel": "self",
            "href": "artists/?name=Savage Garden"
         },
         {
            "rel": "artistSongs",
            "href": "songs/artists/ARBGWMW1187B9AEA3E"
         },
         {
            "rel": "artistStatistics",
            "href": "artists/statistics/ARBGWMW1187B9AEA3E"
         },
         {
            "rel": "artistReleases",
            "href": "releases/artist/ARBGWMW1187B9AEA3E"
         }
      ]
   }
]
```

– **Status code 400:**

CSV

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

JSON

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

– **Status code 404:**

CSV

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

JSON

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

2.2 This is the implementation for functionality nr. **5** from the Project Description.

URI: `HTTP GET .../artists/?rankBy=hotttnesss&order=desc&offset='start'&limit='end'`

Example URI:

1. Retrieve an ordering of the artists ranked by their popularity (in a descending order):
   `HTTP GET .../artists/?rankBy=hotttnesss&order=desc&offset=start`
2. Retrieve a subset (with an upper limit) of the descending ordering of the artists ranked by popularity:
   `HTTP GET .../artists/?rankBy=hotttnesss&order=desc&offset=start&limit=2`
3. Retrieve a subset (with a lower and upper limit) of the descending ordering of the artists ranked by popularity:
   `HTTP GET .../artists/?rankBy=hotttnesss&order=desc&offset=20&limit=50`

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 `limit`), as illustrated by example 2. Also, the ordering can be subsetted given a minimum (query parameter `offset`) and a maximum limit (query parameter `limit`), as shown in example 3.

Status codes:

– **Success:** 200
– **Bad request:** 400
– **Resource not available:** 404

Responses:

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

CSV

```
data/ID,data/FAMILIARITY,data/HOTTTNESSS,data/LATITUDE,data/
    LOCATION,data/LONGTITUDE,data/NAME,data/SIMILAR,data/TERMS,
    data/FREQ links/rel,links/href , "ARRH63Y1187FB47783
    ",0.877213746,1.082502557,0,0,0,"Kanye West",0,"hip hop",1 , "self","artists/?
    name=Kanye West" , "artistSongs","songs/artists/ARRH63Y1187FB47783" , "
    artistStatistics","artists/statistics/ARRH63Y1187FB47783" , "artistReleases","
    releases/artist/ARRH63Y1187FB47783" , "ARF8HTQ1187B9AE693
    ",0.90284076,1.021255588,0,0,0,"Daft Punk",0,"techno",1 , "self","artists/?name
    =Daft Punk" , "artistSongs","songs/artists/ARF8HTQ1187B9AE693" , "
    artistStatistics","artists/statistics/ARF8HTQ1187B9AE693" , "artistReleases","
    releases/artist/ARF8HTQ1187B9AE693" ,
```

JSON


```
[
   {
      "data": {
         "ID": "ARRH63Y1187FB47783",
         "FAMILIARITY": 0.877213746,
         "HOTTTNESSS": 1.082502557,
         "LATITUDE": 0,
         "LOCATION": 0,
         "LONGTITUDE": 0,
         "NAME": "Kanye West",
         "SIMILAR": 0,
         "TERMS": "hip hop",
         "FREQ": 1
      },
      "links": [
         {
            "rel": "self",
            "href": "artists/?name=Kanye West"
         },
         {
            "rel": "artistSongs",
            "href": "songs/artists/ARRH63Y1187FB47783"
         },
         {
            "rel": "artistStatistics",
            "href": "artists/statistics/ARRH63Y1187FB47783"
         },
         {
            "rel": "artistReleases",
            "href": "releases/artist/ARRH63Y1187FB47783"
         }
      ]
   },
   {
      "data": {
         "ID": "ARF8HTQ1187B9AE693",
```

```
            "FAMILIARITY": 0.90284076,
            "HOTTTNESSS": 1.021255588,
            "LATITUDE": 0,
            "LOCATION": 0,
            "LONGTITUDE": 0,
            "NAME": "Daft Punk",
            "SIMILAR": 0,
            "TERMS": "techno",
            "FREQ": 1
        },
        "links": [
            {
                "rel": "self",
                "href": "artists/?name=Daft Punk"
            },
            {
                "rel": "artistSongs",
                "href": "songs/artists/ARF8HTQ1187B9AE693"
            },
            {
                "rel": "artistStatistics",
                "href": "artists/statistics/ARF8HTQ1187B9AE693"
            },
            {
                "rel": "artistReleases",
                "href": "releases/artist/ARF8HTQ1187B9AE693"
            }
        ]
    }
]
```


– **Status code 400:**

CSV


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


JSON


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


– **Status code 404:**

CSV


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


JSON


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

3.

URI: `HTTP GET .../songs/:id`
Example URI: `HTTP GET .../songs/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:** 404

Responses:

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

CSV

```
data/ID,data/RELEASE_ID,data/HOTTTNESSS,data/TITLE,data/YEAR,data/
    ARTIST_MBTAGS,data/ARTIST_MBTAGS_COUNT,data/
    BARS_CONFIDENCE,data/BARS_START,data/BEATS_CONFIDENCE,data/
    BEATS_START,data/DURATION,data/END_OF_FADE_IN,data/KEY,data/
    LOUDNESS,data/KEY_CONFIDENCE,data/MODE,data/
    MODE_CONFIDENCE,data/START_OF_FADE_OUT,data/
    TATUMS_CONFIDENCE,data/TATUMS_START,data/TEMPO,data/
    TIME_SIGNATURE,data/TIME_SIGNATURE_CONFIDENCE links/rel,links/
    href ,
"SOEHWGF12A6D4F8B2B",302176,0.73738531,"Hips Don't Lie (featuring Wyclef Jean)
    ",0,0,1,0.497,0.71063,1,0.10911,217.36444,0.224,10,-5.152,0.805,0,0.688,212.161,
0.538,0.10911,100.023,4,0.937 , " self " , "songs/SOEHWGF12A6D4F8B2B"
```

JSON

```
[
   {
      "data": {
         "ID": "SOEHWGF12A6D4F8B2B",
         "RELEASE_ID": 302176,
         "HOTTTNESSS": 0.73738531,
         "TITLE": "Hips Don't Lie (featuring Wyclef Jean)",
         "YEAR": 0,
         "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,
         "KEY": 10,
         "LOUDNESS": -5.152,
         "KEY_CONFIDENCE": 0.805,
         "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
        },
        "links": {
            "rel": "self",
            "href": "songs/SOEHWGF12A6D4F8B2B"
        }
    }
]
```


– **Status code 400:**

CSV


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


JSON


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


– **Status code 404:**

CSV


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


JSON


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


URI: `HTTP DELETE .../songs/:id`
Example URI: `HTTP DELETE .../songs/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

– **Resource not available:** 400

Responses:

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

CSV


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

JSON

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

– **Status code 400:**

CSV

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

JSON

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

URI: `HTTP PUT .../songs/:id`
Example URI: `HTTP PUT .../songs/SOUDSGM12AC9618304`
Description: This call will update the song resource identified by the path parameter `:id` based on the request body. If the request body is NULL, then the update cannot be performed.
Status codes:

– **Success:** 200

– **Resource not available:** 400

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" }
```

– **Status code 400:**

CSV

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

JSON

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

URI: `HTTP POST .../songs/:id`
Example URI: `HTTP POST .../songs/QOPOMUO12AA72LA7D3`
Description: This call will create a new song resource, which can be identified by the path parameter `:id` based on the request body.
Status codes:

- **Success:** 200
- **Resource not available:** 400

Responses:

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

CSV

```
"response"
"Song with id QOPOMUO12AA72LA7D3 was created successfully"
```

JSON

```
{ "response": "Song with id QOPOMUO12AA72LA7D3 was created successfully" }
```

- **Status code 400:**

CSV

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

JSON

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

4.

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

1. Retrieve all songs by a specific artist
   `HTTP GET .../songs/artists/ARXPPEY1187FB51DF4`
2. Retrieve all songs by a specific artist and in a specific year:
   `HTTP GET .../songs/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:** 404

Responses:

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

CSV

```
data/ID,data/RELEASE_ID,data/HOTTTNESSS,data/TITLE,data/YEAR,data/
    ARTIST_MBTAGS,data/ARTIST_MBTAGS_COUNT,data/
    BARS_CONFIDENCE,data/BARS_START,data/BEATS_CONFIDENCE,data/
    BEATS_START,data/DURATION,data/END_OF_FADE_IN,data/KEY,data/
    LOUDNESS,data/KEY_CONFIDENCE,data/MODE,data/
    MODE_CONFIDENCE,data/START_OF_FADE_OUT,data/
    TATUMS_CONFIDENCE,data/TATUMS_START,data/TEMPO,data/
    TIME_SIGNATURE,data/TIME_SIGNATURE_CONFIDENCE links/rel,links/
    href , "SOGRKKB12A8C14595A",316040,−1,"HISTORY
    ",1995,0,4,0.024,0.77199,0.878,0.29091,249.99138,0.131,11,−2.617,0.383,1,0.169,
243.879,0.107,0.04614,128.702,4,0.495  , " self " , "songs/SOGRKKB12A8C14595A" "
    SOYIUPV12AF72AC56B",280410,−1,"Tabloid Junkie
    ",1995,0,4,0.13,1.7206,0.738,0.62842,272.71791,0.189,10,−6.779,0.596,0,
0.545,268.771,0.335,0.22139,111.132,4,1  , " self " , "songs/SOYIUPV12AF72AC56B"
```

JSON


```
[
  {
    "data": {
      "ID": "SOGRKKB12A8C14595A",
      "RELEASE_ID": 316040,
      "HOTTTNESSS": −1,
      "TITLE": "HISTORY",
      "YEAR": 1995,
      "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,
      "KEY": 11,
      "LOUDNESS": −2.617,
      "KEY_CONFIDENCE": 0.383,
      "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
    },
    " links ": {
      " rel ": " self ",
      " href ": "songs/SOGRKKB12A8C14595A"
    }
```

```
    },
    {
        "data": {
            "ID": "SOYIUPV12AF72AC56B",
            "RELEASE_ID": 280410,
            "HOTTTNESSS": −1,
            "TITLE": "Tabloid Junkie",
            "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,
            "KEY": 10,
            "LOUDNESS": −6.779,
            "KEY_CONFIDENCE": 0.596,
            "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
        },
        "links": {
            "rel": "self",
            "href": "songs/SOYIUPV12AF72AC56B"
        }
    }
]
```


– **Status code 400:**

CSV


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


JSON


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


– **Status code 404:**

CSV


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

JSON

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

5.

URI: `HTTP GET .../songs/year/:year`

5.1 **URI:** `HTTP GET .../songs/year/:year`

Example URI: `HTTP 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:** 404

Responses:

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

CSV

```
data/ID,data/NAME,data/YEAR links/rel,links/href ,
"SOEWMAK12A8C13FD07","I'm in Korea",1954 , "self" , "songs/
    SOEWMAK12A8C13FD07"
```

JSON

```
[
   {
      "data": {
         "ID": "SOEWMAK12A8C13FD07",
         "NAME": "I'm in Korea",
         "YEAR": 1954
      },
      "links": {
         "rel": "self",
         "href": "songs/SOEWMAK12A8C13FD07"
      }
   }
]
```

- **Status code 400:**

CSV

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

JSON

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

– **Status code 404:**

CSV

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

JSON

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

5.2 **URI:** `HTTP GET .../songs/year/:year/?rankBy=hotttnesss&order=desc&offset='start'&limit='end'`

Example URI:

1. Retrieve an ordering of the songs in a specific year ranked by their popularity (in a descending order):
   `HTTP GET .../songs/year/1998/?rankBy=hotttnesss&order=desc`
2. Retrieve a subset (with a maximum limit) of the descending ordering of the songs in a specific year:
   `HTTP GET .../songs/year/2010/?rankBy=hotttnesss&order=desc&limit=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:** 404

Responses:

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

CSV

```
data/ID,data/NAME,data/YEAR links/rel,links/href ,
"SOULTKQ12AB018A183","Nothin' On You [feat. Bruno Mars] (Album Version)
    ",2010 , "self" , "songs/SOULTKQ12AB018A183" "SOGCDYR12AC961854A","
    You And Your Heart",2010 , "self" , "songs/SOGCDYR12AC961854A"
```

JSON

```
[
  {
    "data": {
      "ID": "SOULTKQ12AB018A183",
      "NAME": "Nothin' On You [feat. Bruno Mars] (Album Version)",
      "YEAR": 2010
    },
    "links": {
      "rel": "self",
      "href": "songs/SOULTKQ12AB018A183"
```

```
        }
    },
    {
        "data": {
            "ID": "SOGCDYR12AC961854A",
            "NAME": "You And Your Heart",
            "YEAR": 2010
        },
        "links": {
            "rel": "self",
            "href": "songs/SOGCDYR12AC961854A"
        }
    }
]
```


- **Status code 400:**

CSV


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


JSON


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


- **Status code 404:**

CSV


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


JSON


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


6. **URI:** `HTTP GET .../songs/artists/terms/:terms`
Example URI: `HTTP GET .../songs/artists/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:** 404

Responses:

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

CSV

```
data/ID,data/RELEASE_ID,data/HOTTTNESSS,data/TITLE,data/YEAR,data/
    ARTIST_MBTAGS,data/ARTIST_MBTAGS_COUNT,data/
    BARS_CONFIDENCE,data/BARS_START,data/BEATS_CONFIDENCE,data/
    BEATS_START,data/DURATION,data/END_OF_FADE_IN,data/KEY,data/
    LOUDNESS,data/KEY_CONFIDENCE,data/MODE,data/
    MODE_CONFIDENCE,data/START_OF_FADE_OUT,data/
    TATUMS_CONFIDENCE,data/TATUMS_START,data/TEMPO,data/
    TIME_SIGNATURE,data/TIME_SIGNATURE_CONFIDENCE links/rel,links/
    href ,
"SOHDIVC12A6D4F8DA7",29874,0.215080319,"Baile Da Pesada
    ",0,0,0,0.265,1.32015,0,0.61203,188.89098,0.351,4,-8.239,0.367,0,0.372,174.481,0.859,

0.14007,77.41,3,1  ,  " self "  ,  "songs/SOHDIVC12A6D4F8DA7"
```

JSON


```
[
   {
      "data": {
         "ID": "SOHDIVC12A6D4F8DA7",
         "RELEASE_ID": 29874,
         "HOTTTNESSS": 0.215080319,
         "TITLE": "Baile Da Pesada",
         "YEAR": 0,
         "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,
         "KEY": 4,
         "LOUDNESS": -8.239,
         "KEY_CONFIDENCE": 0.367,
         "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
      },
      " links ": {
         " rel ": " self ",
         " href ": "songs/SOHDIVC12A6D4F8DA7"
      }
   }
]
```

- **Status code 400:**

CSV

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

JSON

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

- **Status code 404:**

CSV

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

JSON

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

7. **URI:** `HTTP GET .../artists/statistics/:id/?year='song.year'`
Example URI:

   1. Retrieve statistics for the songs of a specific artist:
      `HTTP GET .../artists/statistics/ARXPPEY1187FB51DF4`
   2. Retrieve statistics for the songs of a specific artist in a specific year:
      `HTTP GET .../artists/statistics/ARXPPEY1187FB51DF4/?year=1995`

   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:** 404

   Responses:

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

   CSV

```
data/MEAN,data/MEDIAN,data/STD , links/rel,links/href ,
−1,−1,0 , "" , ""
```

JSON


```
[
   {
      "data": {
         "MEAN": -1,
         "MEDIAN": -1,
         "STD": 0
      },
      "links": {
         "rel": "",
         "href": ""
      }
   }
]
```


– **Status code 400:**

CSV


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


JSON


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


– **Status code 404:**

CSV


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


JSON


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


8. **URI:** `HTTP GET .../releases/artist/:id`
Example URI: `HTTP GET .../releases/artist/ARD7TVE1187B99BFB1`
Description: This request will retrieve the IDs of the albums (releases) of a given artist, identified by the path parameter `:id`.
Status codes:

   – **Success:** 200
   – **Bad request:** 400
   – **Resource not available:** 404

Responses:

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

CSV

```
data/ID , links/rel , links/href ,
249138 , "" , "" 249472 , "" , "" 300848 , "" , "" 368596 , "" , "" 399933 , "" , ""
```

JSON

```
[
   {
      "data": {
         "ID": 249138
      },
      "links": ""
   },
   {
      "data": {
         "ID": 249472
      },
      "links": ""
   },
   {
      "data": {
         "ID": 300848
      },
      "links": ""
   },
   {
      "data": {
         "ID": 368596
      },
      "links": ""
   },
   {
      "data": {
         "ID": 399933
      },
      "links": ""
   }
]
```

- **Status code 400:**

CSV

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

JSON

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

– **Status code 404:**

CSV

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

JSON

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

9.

URI: `HTTP GET .../songs`
Example URI: `HTTP GET .../songs`
Description: This request will retrieve useful information (name, hottness, year, id) about all songs in the database.
Status codes:

– **Success:** 200

– **Bad request:** 400

– **Resource not available:** 404

Responses:

– **Status code 200 (note that we only show the first two elements of the response for readability reasons)**:

CSV

```
data/NAME,data/HOTTTNESSS,data/YEAR,data/ID links/rel,links/href,
"Soul Deep",3,1969,"SOCIWDW12A8C13D406" , "self" , "songs/
    SOCIWDW12A8C13D406"
"Amor De Cabaret",-1,0,"SOXVLOJ12AB0189215" , "self" , "songs/
    SOXVLOJ12AB0189215"
```

JSON

```
[
  {
    "data": {
      "NAME": "Soul Deep",
      "HOTTTNESSS": 3,
      "YEAR": 1969,
      "ID": "SOCIWDW12A8C13D406"
    },
    "links": {
      "rel": "self",
      "href": "songs/SOCIWDW12A8C13D406"
    }
  },
  {
    "data": {
      "NAME": "Amor De Cabaret",
      "HOTTTNESSS": -1,
```

```
            "YEAR": 0,
            "ID": "SOXVLOJ12AB0189215"
        },
        "links": {
            "rel": "self",
            "href": "songs/SOXVLOJ12AB0189215"
        }
    }
]
```


– **Status code 400:**

CSV


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


JSON


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


– **Status code 404:**

CSV


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


JSON


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