Music Discovery



Lyon Berends

Klas: DHI1V.Sa

Table of Contents

[Introduction 3](#_Toc171537995)

[Class diagram 4](#_Toc171537996)

[GET requests 5](#_Toc171537997)

[POST requests 8](#_Toc171537998)

[PUT requests 10](#_Toc171537999)

[DELETE requests 11](#_Toc171538000)

[Sequence diagrams 12](#_Toc171538001)

[Test report 13](#_Toc171538002)

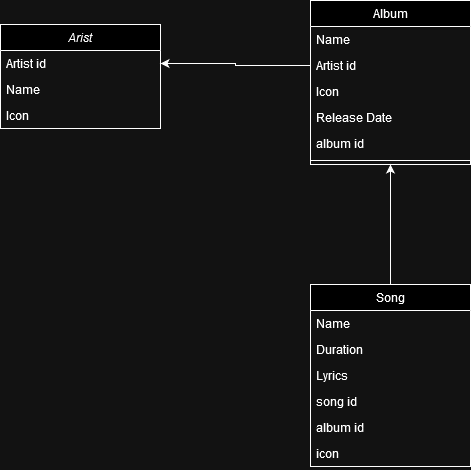
[Usecase diagram 14](#_Toc171538003)

[Wireframes 15](#_Toc171538004)

[Explanation 19](#_Toc171538005)

Introduction  
This website will offer a nice music discovery of much which I recommend people listen to, this website will contain albums, songs and artists. You can add new albums, songs and artists using the live website and they will be presented on a webpage.

# Class diagram



Alle classes hebben een unieke ID.

De albums hebben een reference naar artists in de vorm van een artist id

De songs hebben een reference naar albums in de vorm van een album id

# GET requests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/artist/{id}** | | | |
| Gets one artist, based on given id | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Id \* | path | Id of artist to find |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends the artist class as JSON of 1 artist with a status code, it will be structured as: {  "artist\_id": id,  "name": name  "icon": icon  "status": 200  } | |
|  | | 400 | Sends JSON with 400 error code if the id path parameter contains a character which is not a number or empty.  The JSON will be expressed as: {‘status’: 400} | |
|  | | 404 | Sends JSON with 404 error code if the id cannot be found  The JSON will be expressed as: {‘status’: 404} | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/artists** | | | |
| Gets a list of artists | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Get\* | query | A query parameter which is required. This parameter has 2 options, recommended and all. With recommended you get 8 results, with all you get all the possible results back. |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends a list of artists as JSON with status code This list will be structured as: {  ‘artists’: [{artist}, {artist}],  ‘status’: 200  } | |
| 400 | Sends JSON with 400 error code if the query get is not all, recommended or empty. The JSON will be expressed as: {‘status’: 400} | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/album/{id}** | | | |
| Gets one album, based on given id | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
|  | | Id \* | path | Id of album to find |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends the album class as JSON of 1 album with a status code, it will be structured as: {  "album\_id": id,  "name": name  "release\_date": release-date  "icon": icon  "artist\_id": artist-id,  "status": 200  } | |
|  | | 400 | Sends JSON with 400 error code if the id path parameter contains a character which is not a number or empty.  The JSON will be expressed as: {‘status’: 400} | |
|  | | 404 | Sends JSON with 404 error code if the id cannot be found  The JSON will be expressed as: {‘status’: 404} | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/albums** | | | |
| Gets a list of albums | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Get\* | query | A query parameter which is required. This parameter has 2 options, recommended and all. With recommended you get 8 results, with all you get all the possible results back. |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends a list of albums as JSON with status code  The JSON will be structured as  {  ‘albums: [{album}, {album}],  ‘status’: 200  } | |
| 400 | Sends JSON with 400 error code if the query get is not all, recommended or empty. The JSON will be expressed as: {‘status’: 400} | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/song/{id}** | | | |
| Gets one songs, based on given id | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Id \* | path | Id of song to find |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends the song class as JSON of 1 song with a status code, it will be structured as: {  "song\_id": id,  "name": name  "duration": duration  "lyrics": lyrics,  "icon": icon  "album\_id": album-id,  "status": 200  } | |
| 400 | Sends JSON with 400 error code if the id path parameter contains a character which is not a number or empty.  The JSON will be expressed as: {‘status’: 400} | |
|  | | 404 | Sends JSON with 404 error code if the id cannot be found  The JSON will be expressed as: {‘status’: 404} | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **GET** | **/api/songs** | | | |
| Gets a list of songs | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Get\* | query | A query parameter which is required. This parameter has 2 options, recommended and all. With recommended you get 8 results, with all you get all the possible results back. |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Sends a list of songs as JSON with status code  The JSON will be structured as  {  ‘songs: [{song}, {song}],  ‘status’: 200  } | |
| 400 | Sends JSON with 400 error code if the query get is not all, recommended or empty. The JSON will be expressed as: {‘status’: 400} | |

# POST requests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **/api/artists** | | | |
| Adds an artist with the given name, albums and songs will be empty, id is automatically generated | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Name \* | body | The name of the artists to be created. Has to be in JSON with a JSON content header |
|  | | Icon | Body | The icon of the artist |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 201 | Created successfully, will send a response in JSON. | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 500 | If something unexpected goes wrong with executing this request it will send a JSON or plain text response with the error code. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **/api/albums** | | | |
| Adds an album with the given name, only JSON will be accepted | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Name \* | body | The album name |
| * *required* | | Artist \* | body | The artist id which has to be linked to the album |
|  | | Release date | body | The release date of the album |
|  | | Icon \* | body | The icon of the album |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 201 | Created successfully, will send a response in JSON. | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 500 | If something unexpected goes wrong with executing this request it will send a JSON or plain text response with the error code. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **POST** | **/api/songs** | | | |
| Adds a song with the given name, album id, icon, duration and lyrics | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Name \* | body | Name of the songs |
| * *required* | | Duration \* | body | Duration of the song in seconds |
| * *required* | | Album ID\* | body | The id of the album this song is in |
|  | | Icon | body | The icon of the song |
|  | | Lyrics | body | Lyrics of the song, not required, can also be added later. |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 201 | Created successfully, will send a response in JSON. | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 500 | If something unexpected goes wrong with executing this request it will send a JSON or plain text response with the error code. | |

# PUT requests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PUT** | **/api/artists/{id}** | | | |
| Modifies an artist based on the given ID with a JSON body | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Name \* | body | The new name of the artist |
| Icon \* | body | The new icon of the artist |
|  | | Id \* | Path | The artist id of the artist to modify |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Artist successfully modified, returns the new artist JSON in the body. | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 404 | Artist not found, cannot modify a non existing artist. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PUT** | **/api/albums/{id}** | | | |
| Modifies an album based on the given ID with a JSON body | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Name \* | body | The new name of the album |
| Release date\* | body | The new release date of the album |
| Icon \* | body | The new icon of the album |
| Artist id \* | body | The new artist id of the album |
|  | | Id \* | path | The album id of the album to modify |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Album successfully modified | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 404 | Artist not found, cannot modify a non existing artist. | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PUT** | **/api/songs/{id}** | | | |
| Modifies an song based on the given ID with a JSON body | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| *\*required* | | Name \* | body | The new name of the song |
| Duration \* | body | The new duration of the song |
| Lyrics \* | body | The new lyrics of the song |
| Icon \* | body | The new icon of the song |
| Album id \* | body | The new album id of the song |
|  | | Id \* | Path | The song id of the song to modify |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Album successfully modified, returns the new album JSON in the body. | |
| 400 | Some of the required fields are empty or the JSON format is wrong. | |
| 404 | Artist not found, cannot modify a non existing artist. | |

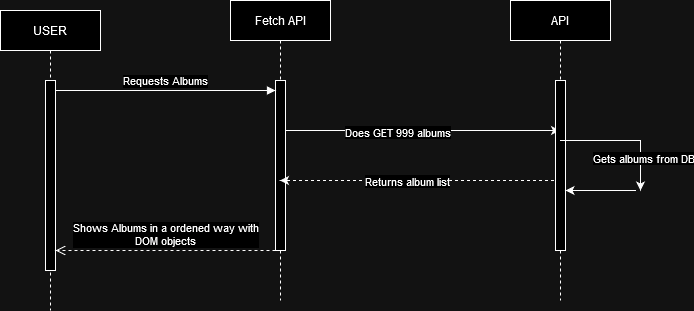
# DELETE requests

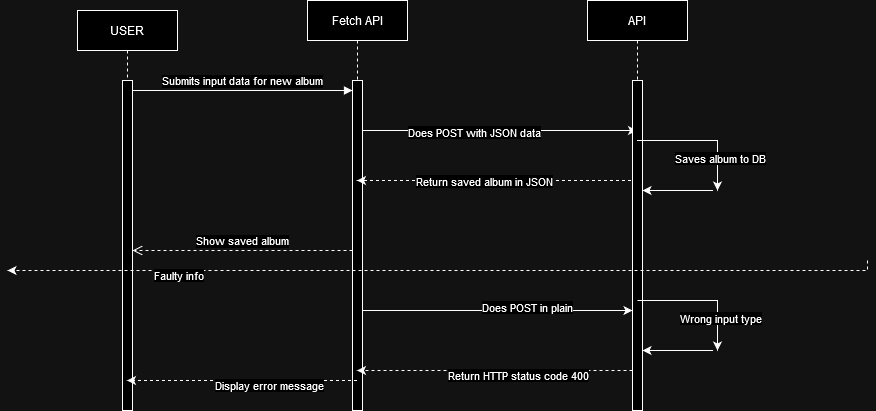
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **/api/artists/{id}** | | | |
| Deletes one artist, based on given id | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Id \* | path | Id of artist to delete |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Artist successfully deleted | |
| 404 | Artist not found | |
|  | | 500 | Something unexpected went wrong on the server side | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **/api/albums/{id}** | | | |
| Deletes one album, based on given id | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Id \* | path | Id of album to delete |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Album successfully deleted | |
| 404 | Album not found | |
|  | | 500 | Something unexpected went wrong on the server side | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DELETE** | **/api/songs/{id}** | | | |
| Deletes one song, based on given id | | | | |
|  | | | | |
| **Parameters:** | | **Name** | **Type** | **Description** |
| * *required* | | Id \* | path | Id of song to delete |
| **Responses:** | | **Code** | **Description / example if successful** | |
|  | | 200 | Song successfully deleted | |
| 404 | Song not found | |
|  | | 500 | Something unexpected went wrong on the server side | |

# Sequence diagrams





Test report  
To make sure the website work in the way it was supposed to I tested every function on the website myself, I walked through all the processes which you are able to use on the website like editing, posting and deleting data. I made changes to the site based on the results of these tests, for example: the class diagrams have been changed based on the results of my tests, the case here was that an object couldn’t have an array of foreign keys.

# Usecase diagram

Afbeelding met tekst, diagram, schermopname, lijn

Automatisch gegenereerde beschrijving

# Wireframes

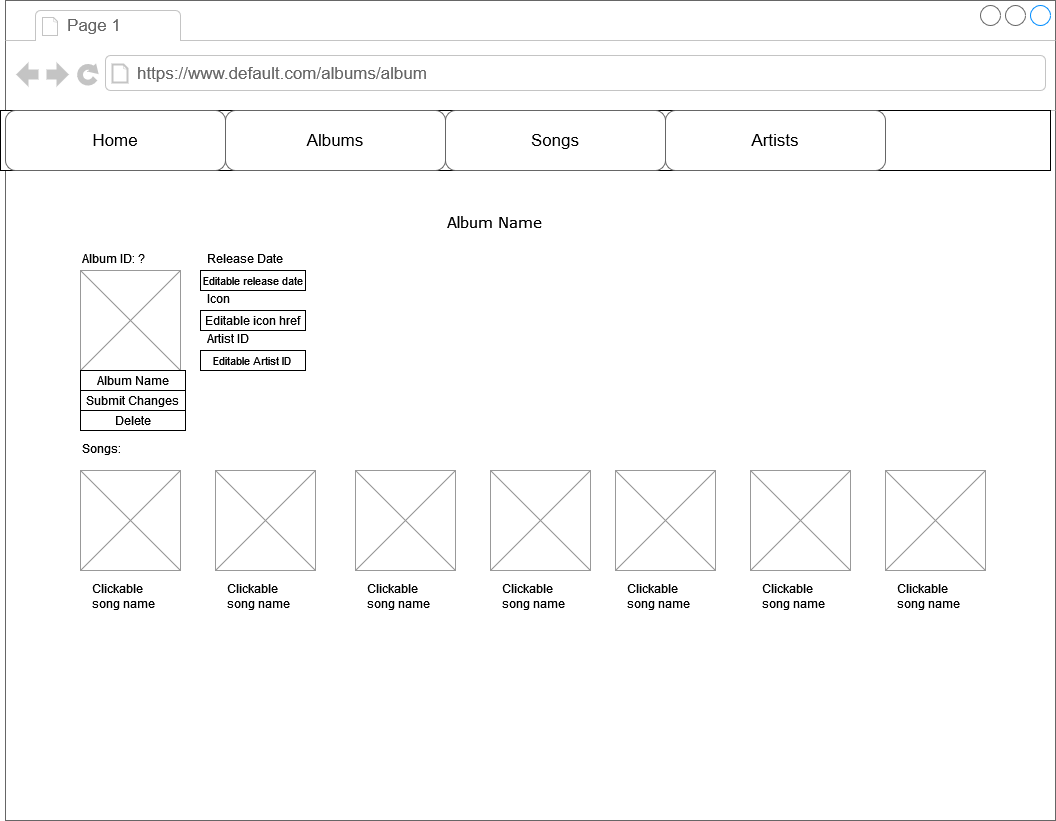
Index:

## 

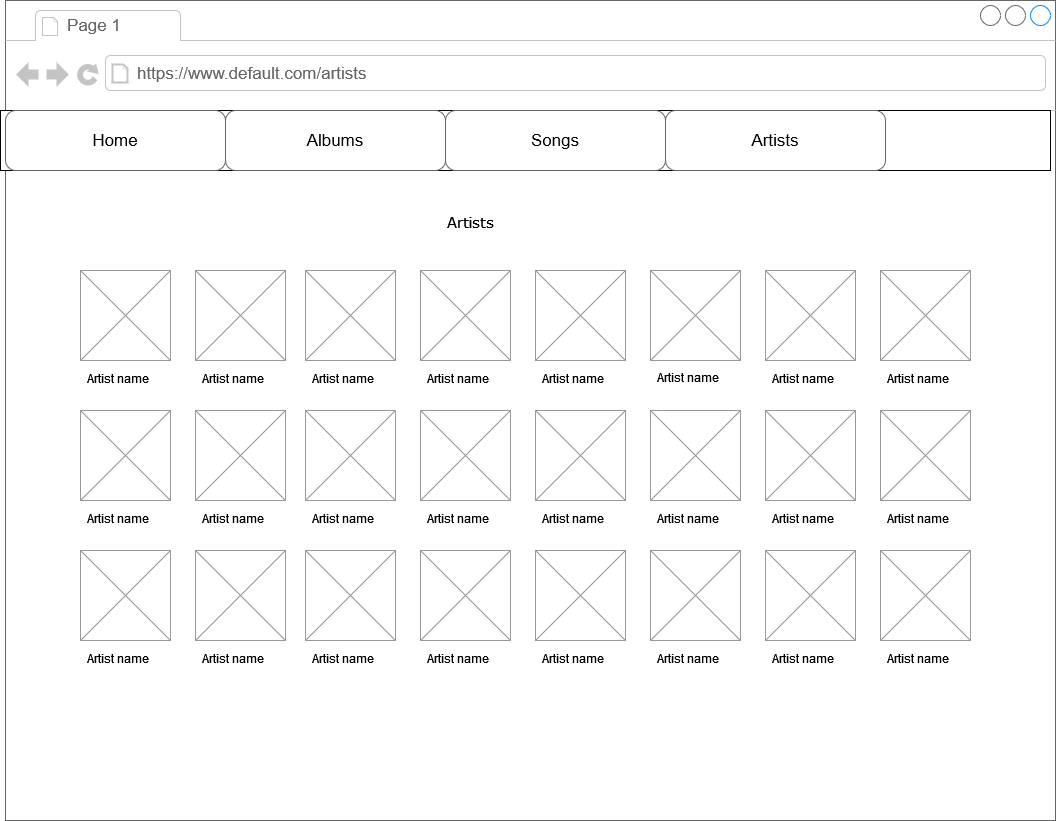
Albums:



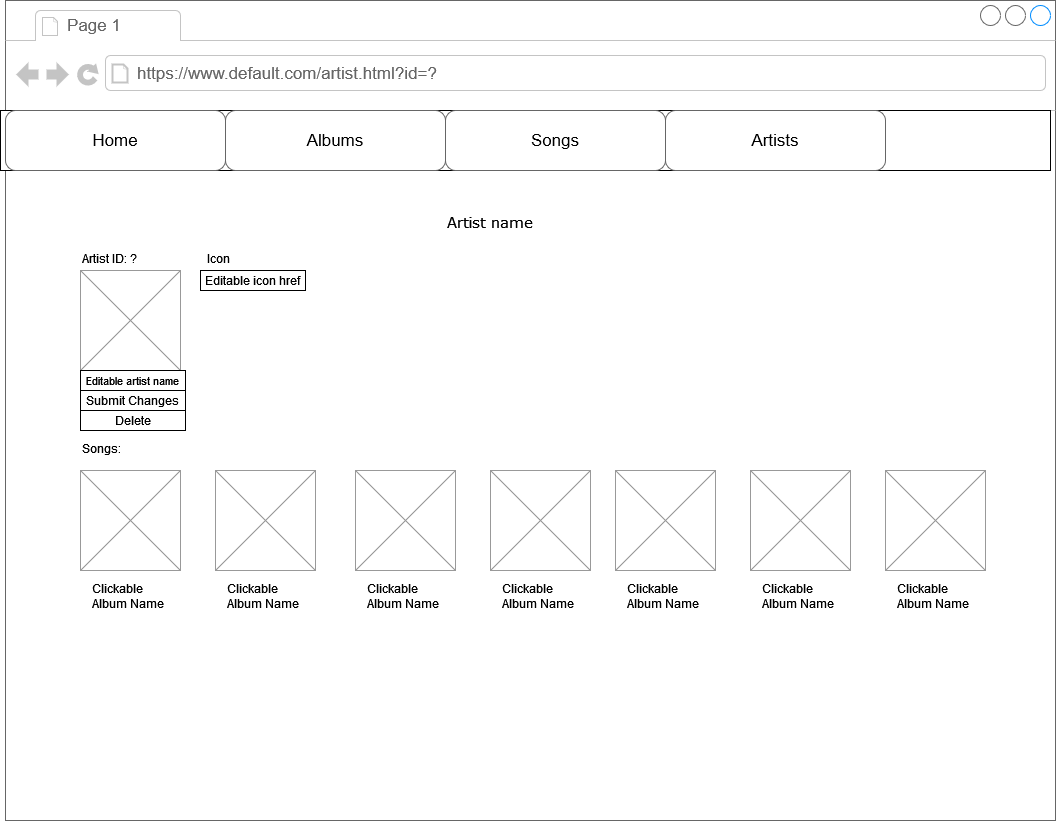
Album:



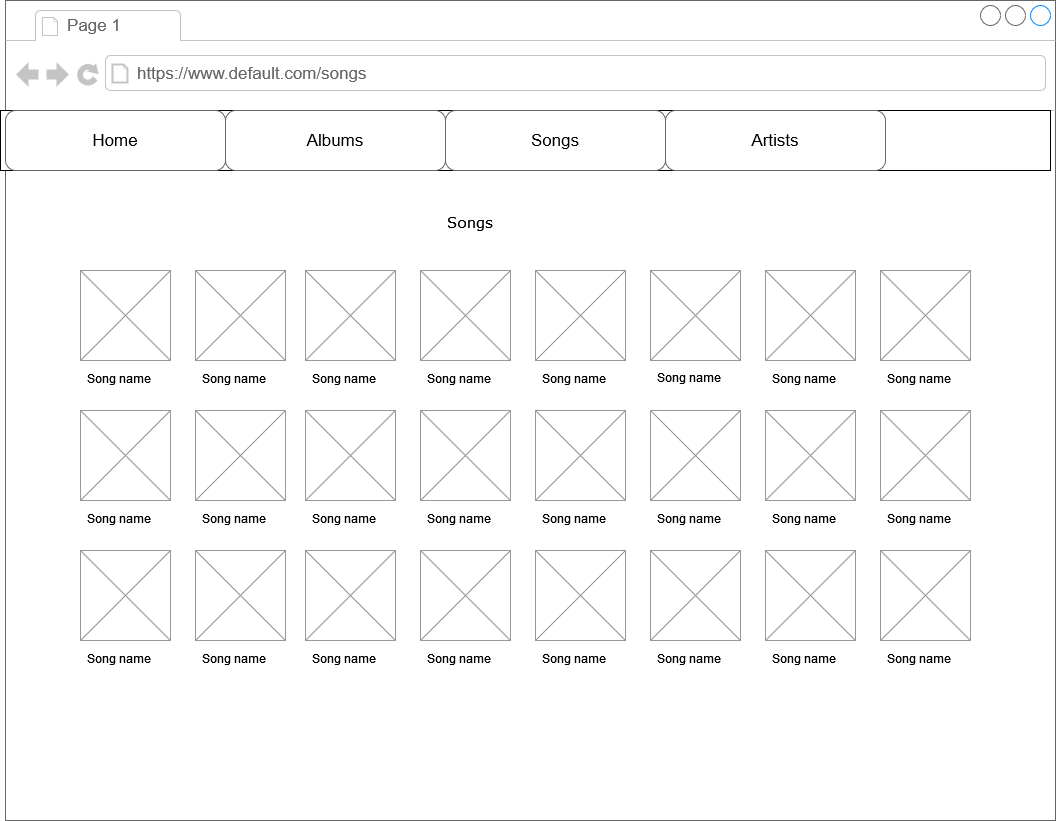
Artists:



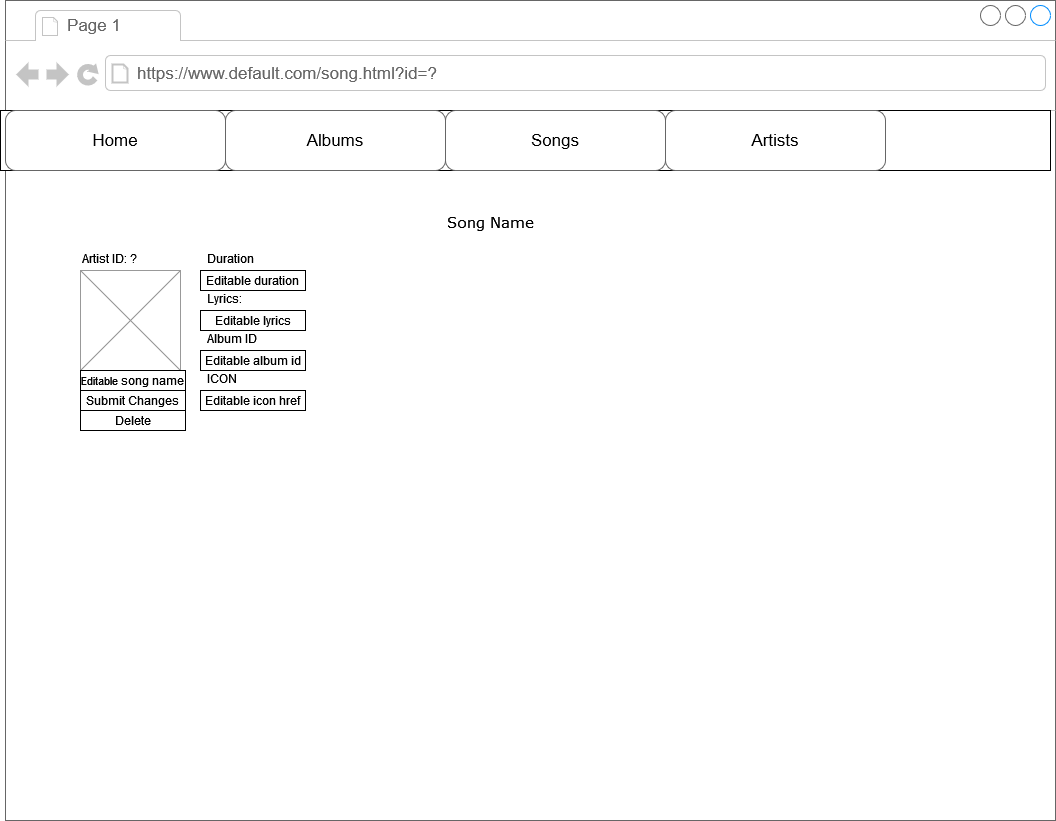
Artist:



Songs:



Song:



## Explanation

For the song, album, artist wireframes there are some editable fields, these field their values can be edited and if you press submit these changes will be send to the backend through the rest api.  
Every image at songs, artists, albums and index has a clickable link under it which directs you to the page of that entity where you can edit/delete it.