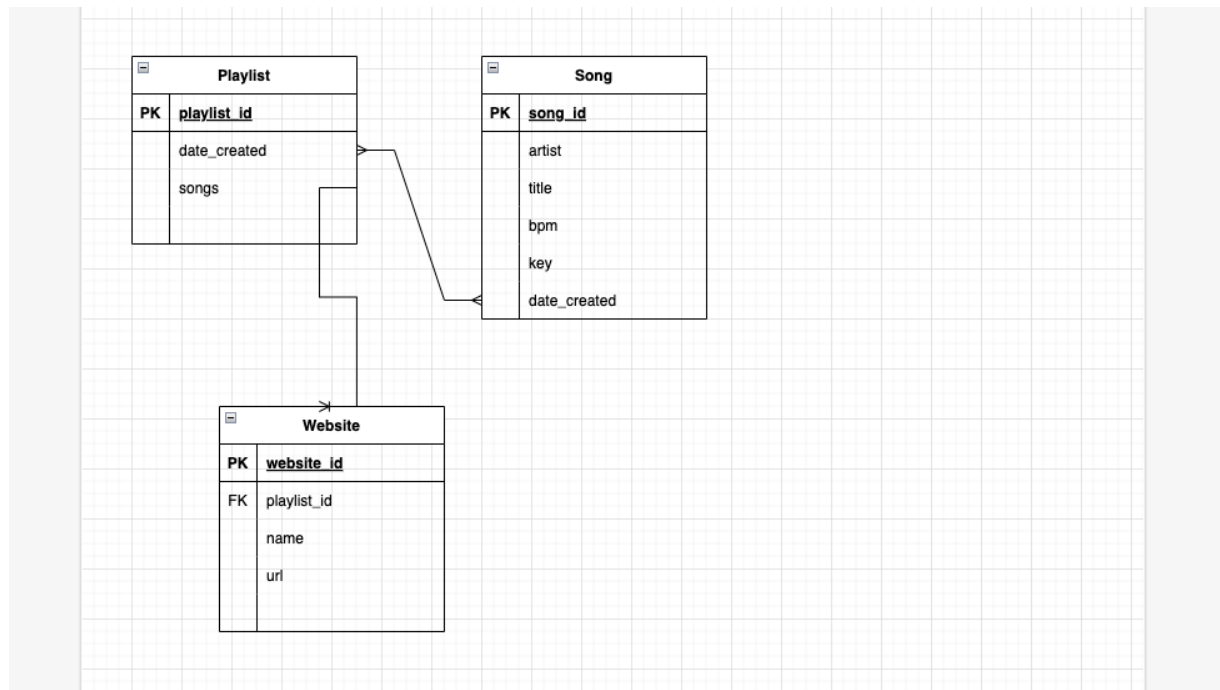


Week 6 Final Project

GitHub URL: https://github.com/Rowan-Bear/Final_Project



Playlist CRUD Operations

GET

▼

http://localhost:8080/playlists

Send

▼

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings ●

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

Beautifuly

1

Body

Cookies

Headers (5)

Test Results

🌐

Status: 200 OK

Time: 51 ms

Size: 166 B

Save Response ▼

Pretty

Raw

Preview

Visualize

JSON ▼

🔄

📄

🔍

1

PUT

http://localhost:8080/websites/4

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1  {
2    "name": "MyWebsite2.0",
3    "url": "www.mywebsite_2.0.com",
4    "id": 4,
5    "playlist": []
6  }
```

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 46 ms

Size: 238 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "name": "MyWebsite2.0",
3    "url": "www.mywebsite_2.0.com",
4    "id": 4,
5    "playlist": []
6  }
```

GET

http://localhost:8080/websites

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1  []
```

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 38 ms

Size: 166 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  []
```

POST

http://localhost:8080/websites

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

6

1

2

3

4

5

6

Body

Cookies

Headers (5)

Test Results

Status: 201 Created

Time: 45 ms

Size: 238 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

1

2

3

4

5

6

```
{
  "name": "MyWebsite",
  "url": "www.mywebsite.com",
  "id": "1"
}
```

```
{
  "name": "MyWebsite",
  "url": "www.mywebsite.com",
  "id": 4,
  "playlist": null
}
```

Website CRUD Operations

DELETE



http://localhost:8080/websites/4

Send



Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings ●

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▾

Beautify

1

Body

Cookies

Headers (5)

Test Results



Status: 200 OK

Time: 16 ms

Size: 203 B

Save Response ▾

Pretty

Raw

Preview

Visualize

Text ▾



1 Successfully deleted website with id: 4

GET

http://localhost:8080/websites/4

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 42 ms

Size: 231 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

```
{
  "name": "MyWebsite",
  "url": "www.mywebsite.com",
  "id": 4,
  "playlist": []
}
```

Song CRUD Operations:

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method `PUT`, URL `http://localhost:8080/songs/2`, and a `Send` button.
- Request Tabs:** Params, Authorization, Headers (8), **Body** (selected), Pre-request Script, Tests, Settings.
- Body Format:** Radio buttons for none, form-data, x-www-form-urlencoded, **raw** (selected), binary, GraphQL, and a `JSON` dropdown.
- Request Body:** A JSON object with the following structure:

```
1 {
2   "id": 2,
3   "artist": "MJ",
4   "title": "title",
5   "bpm": 118,
6   "key": "C#Minor",
7   "playlist": []
8 }
```
- Response Bar:** Status: `200 OK`, Time: `44 ms`, Size: `242 B`, and a `Save Response` button.
- Response Tabs:** **Body** (selected), Cookies, Headers (5), Test Results.
- Response Format:** Buttons for Pretty, Raw, Preview, Visualize, and a `JSON` dropdown.
- Response Body:** The same JSON object as the request body, formatted with syntax highlighting:

```
1 {
2   "id": 2,
3   "artist": "MJ",
4   "title": "title",
5   "bpm": 118,
6   "key": "C#Minor",
7   "playlist": []
8 }
```


POST

http://localhost:8080/songs/

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1
2
3
4
5
6
7
{
  "id": "2",
  "artist": "Michael Jackson",
  "title": "title",
  "bpm": "118",
  "key": "C#Minor"
}
```

Body

Cookies

Headers (5)

Test Results

Status: 201 Created

Time: 53 ms

Size: 262 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1
2
3
4
5
6
7
8
{
  "id": 2,
  "artist": "Michael Jackson",
  "title": "title",
  "bpm": 118,
  "key": "C#Minor",
  "playlist": null
}
```

DELETE

http://localhost:8080/songs/2

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1
2
3
4
5
6
7
8
{
  "id": 2,
  "artist": "MJ",
  "title": "title",
  "bpm": 118,
  "key": "C#Minor",
  "playlist": []
}
```

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 37 ms

Size: 200 B

Save Response

Pretty

Raw

Preview

Visualize

Text

```
1 Successfully deleted song with id: 2
```

GET

http://localhost:8080/songs/2

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 17 ms

Size: 255 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

"id": 2,

"artist": "Michael Jackson",

"title": "title",

"bpm": 118,

"key": "C#Minor",

"playlist": []

GET

http://localhost:8080/songs/

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 67 ms

Size: 166 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

[]

API Documentation

URL: Songs

Allowed requests: GET, POST, PUT, DELETE

Input required in Response Body:

GET(read a song object) - id in path variable for a single song

POST(create and object of song) - {}

PUT(update a song object) - request body must be the same as original object created, other than changes

DELETE(deletes a song object) - id

Appropriate HTTP status's and error messages: HttpStatus.CREATED, HttpStatus.BAD_REQUEST, HttpStatus.OK, HttpStatus.NOT_FOUND, "Unable to retrieve customer by id", "Unable to update song.", "Unable to delete customer."

URL: Websites

Allowed Requests: GET, POST, PUT, DELETE

Input required in Response Body:

GET(read a website object) - id of website in path variable for a single website

POST(create a website object) - {}

PUT(update a website object) - request body must be the same as original object created, other than changes

DELETE(deletes a website object) - id

Appropriate HTTP status's and error messages: HttpStatus.CREATED, HttpStatus.BAD_REQUEST, HttpStatus.OK, HttpStatus.NOT_FOUND, "Unable to retrieve website by id", "Unable to update website.", "Unable to delete customer."

URL: Playlists

Allowed Requests: GET

Input required in response body:

GET(shows all playlists) - {} to see all playlists

Appropriate HTTP status's and error messages: HttpStatus.OK,
HttpStatus.NOT_FOUND

