

Betche Nathan Stephane

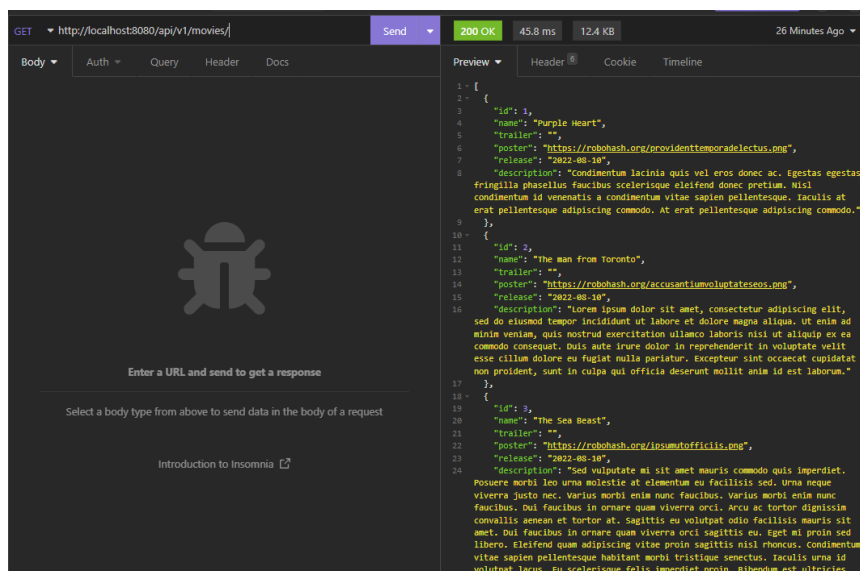
Link to the repo: <https://github.com/nathanstephane/MovieApp>

Movie App Documentation.

The Movie App runs on 3 different REST API.

A REST API about movies, using Spring Boot.

Data from this API is consumed at the following URL: localhost:8080/api/v1/movies



Getting all movies.






Users and rating API (microservice) run respectively on the different URL:

<http://localhost:3001/users> & <http://localhost:3001/ratings>

Databases:

Postgresql: users and movies data are stored in a postgresql database.

A table about the movies watched by which user is also present in the database.

| | WHERE | | ORDER BY | | | | |
|---|--|---|----------|---|---|--|---|
| |  id ÷ |  createdAt | ÷ |  updatedAt | ÷ |  userId ÷ |  movieId ÷ |
| 1 | 1 | 2022-08-16 16:27:52.219000 +00:00 | | 2022-08-16 16:27:52.219000 +00:00 | | 1 | 5 |
| 2 | 2 | 2022-08-16 16:56:51.576000 +00:00 | | 2022-08-16 16:56:51.576000 +00:00 | | 1 | 16 |
| 3 | 3 | 2022-08-16 16:57:00.692000 +00:00 | | 2022-08-16 16:57:00.692000 +00:00 | | 1 | 9 |

Rating: ratings about watched movies are stored in a MongoDB Database

test.ratings

DOCUMENTS INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET

ADD DATA VIEW ()

Displaying documents 1 - 3 of 3 REFRESH

| |
|--|
| <pre> _id: ObjectId('62fbb4c55f144e019c34e0a6') movie: 5 rating: 2 createdAt: 2022-08-16T15:16:21.405+00:00 updatedAt: 2022-08-16T15:16:21.405+00:00 __v: 0 </pre> |
| <pre> _id: ObjectId('62fbb4cd5f144e019c34e0a9') movie: 5 rating: 3 createdAt: 2022-08-16T15:16:29.596+00:00 updatedAt: 2022-08-16T15:16:29.596+00:00 __v: 0 </pre> |
| <pre> _id: ObjectId('62fbcc7f07aeb4dcf8bf60a') movie: 9 rating: 2 createdAt: 2022-08-16T16:57:35.818+00:00 updatedAt: 2022-08-16T16:57:35.818+00:00 __v: 0 </pre> |

API endpoint

Users Controller.

The following endpoints are used for the users API.

POST: <http://localhost:3001/users/register>

POST: <http://localhost:3001/users/login>

Movies Controller.

The following endpoints are used for the movies API.

GET: <http://localhost:8080/api/v1/movies> : retrieves a list of all the movies

PUT : [http://localhost:8080/api/v1/movies/\\${id}](http://localhost:8080/api/v1/movies/${id}) : updates a particular movie

DELETE: [http://localhost:8080/api/v1/movies/\\${id}](http://localhost:8080/api/v1/movies/${id}) : Deletes a particular movie

POST: <http://localhost:8080/api/v1/movies> : creates a movie

Rating Controller:

The following endpoints is used for the rating API:

POST: [http://localhost:8080/rating/\\${id}](http://localhost:8080/rating/${id}) : updates the rating for a specific id

AXIOS:

In order for the client app (frontend) to consume the API resources, the package axios is used by making requests to the different endpoints mentioned above.

Project Structure.

Inside the main folder are 3 different folder:

1 folder for the Backend in JAVA containing the datamodel and the REST services associated with movies

1 folder containing the Backend Javascript and nodejs containing the datamodel about users and rating and the REST services associated with these. It also contains the business logic regarding the movies(such as watch and rate a movie).

1 folder containing the frontend application.

Technology used:

JAVA Backed: Spring boot, Hibernate(JPA) , Postgresql.

JS Backend: Mongodb, Express, NodeJS,

Below is a list of the dependencies:

```
"axios": "^0.27.2",
"concurrently": "^7.3.0",
"cors": "^2.8.5",
"dotenv": "^16.0.1",
"express": "^4.18.1",
"fs": "^0.0.1-security",
"http-errors": "^2.0.0",
"mongoose": "^6.5.2",
"mysql2": "^2.3.3",
"nodemon": "^2.0.19",
"path": "^0.12.7",
"pg": "^8.7.3",
"sequelize": "^6.21.3"
```

JS Frontend: React, axios, Bootstrap

How to run:

1) Navigate to the Backend folder and run the following command: *npm install* to install the dependencies.

Then, *npm run start* (This will start the microservices for the users and the rating. It will also create the databases (using sequelize))

2) Navigate to the Frontend folder and run the following command: *npm install* (dependencies)

Then *npm run start* (This will start the front end application)

3) Navigate the BackendJava then make sure the correct libraries are loaded through the pom.xml file using the maven command (mvn clean install) then start the application from MovieApplication.java file It is the spring starter file).

4) Navigate to <http://localhost:3000> from the browser and voila!