

API Documentation: Movies API with JWT Authentication and Docker Compose.

Overview

The Movies API provides CRUD functionality for managing movies and allows users to authenticate using **JWT Bearer Tokens**. The API is containerized using **Docker** and **Docker Compose** to streamline deployment and ensure that the application runs in a consistent environment.

Technologies Used

- .NET Core 8.0
- Entity Framework Core (EF Core) Database-First
- SQL Server
- JWT Bearer Authentication
- Docker and Docker Compose

Prerequisites

To run this project locally, you will need:

- **Docker** and **Docker Compose**.
- **Postman** or another API testing tool (for testing API endpoints)

Getting Started

A. Docker Compose Setup

The docker-compose.yml file orchestrates the API and the SQL Server database.

```

1   networks:
2     moviesapi:
3
4
5
6   services:
7     moviesapidb:
8       container_name: demo-movies-api-db
9       image: mcr.microsoft.com/mssql/server:2022-latest
10      ports:
11        - 8002:1433
12      environment:
13        - ACCEPT_EULA=Y
14        - MSSQL_SA_PASSWORD=Password@1234
15      volumes:
16        - db_data:/var/opt/mssql
17      networks:
18        - moviesapi
19    moviesapi:
20      container_name: demo-movies-api
21      image: jrcjancy/moviesapi
22      build:
23        context: .
24        dockerfile: Movies API/Dockerfile
25      ports:
26        - 8001:8080
27      depends_on:
28        - moviesapidb
29      environment:
30        - DB_HOST=moviesapidb
31        - DB_NAME=Movies
32        - MSSQL_SA_PASSWORD=Password@1234
33      networks:
34        - moviesapi
35    volumes:
36      db_data:

```

B. Running the Application

Docker

1. To run the project in docker:
 - Search for **docker-compose.yml** file.
 - Run the command **docker-compose up -d**. from the directory where the docker-compose file is located.

API Endpoints

A. Authentication <

POST /api/users/login

- **Description:** Authenticate a user and return a JWT token.
- **Request Body:**

```
1  {
2    "email": "Jane@gmail.com",
3    "password": "qwerty"
4  }
```

- **Response:**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": {
5      "email": "Jane@gmail.com",
6      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bm90dG8iOiJpYXN0IiwiaWF0IjE4NDI4OTAzLCJpYXQiOiE3Mjg0MjgzMDN9.9BiP0Q11-IncT5L3VqXLWVhw2aK7ubEiBBZMoakffWE"
7    }
8  }
```

B. Movies

GET /api/movies

- **Description:** Retrieve a list of movies
- **Response**

```
{
  "success": 1,
  "message": null,
  "data": [
    {
      "id": 1,
      "name": "Alien: Romulus",
      "idGenre": 1,
      "genre": "Horror",
      "idAgeRating": 4,
      "rating": "R",
      "releaseDate": "2024-08-16"
    },
    {
      "id": 2,
      "name": "Dune: Part 2",
      "idGenre": 3,
      "genre": "Sci-fi",
      "idAgeRating": 3,
      "rating": "PG-13",
      "releaseDate": "2024-03-01"
    }
  ]
}
```

POST /api/movies

- **Description:** Add a movie to list of movies
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
1  {
2    "Name": "The Batman",
3    "IdGenre": 4,
4    "IdAgeRating": 3,
5    "ReleaseDate": "2022-03-04"
6  }
7
```

- **Response**

```
1  {
2    "success": 1,
3    "message": "Item added",
4    "data": null
5  }
```

PUT/api/movies

- **Description:** Updates a movie from list of movies
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
1  {
2    "id": 4,
3    "name": "Aliens: the return",
4    "idGenre": 1,
5    "idAgeRating": 4,
6    "releaseDate": "1986-07-14"
7  }
```

- **Response**

```
1  {
2    "success": 1,
3    "message": "Item updated",
4    "data": null
5  }
```

DELETE/api/movies/id

- **Description:** Deletes a movie from the list of movies
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Response**

```
1  {
2    "success": 1,
3    "message": "Item deleted",
4    "data": null
5  }
```

C. Genres

GET /api/genres

- **Description:** Retrieve a list of genres
- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": [
5      {
6        "id": 1,
7        "genre1": "Horror",
8        "movies": []
9      },
10     {
11       "id": 3,
12       "genre1": "Sci-fi",
13       "movies": []
14     },
15   ]
16 }
```

POST /api/genres

- **Description:** Add a genre to list of genres
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
1  {
2    "genre": "Western"
3  }
```

- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": null
5  }
```

PUT/api/genres

- **Description:** Updates a genre from the list of genres
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
{
  "id": 8,
  "genre": "Drama"
}
```

- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": null
5  }
```

DELETE/api/genres/id

- **Description:** Deletes a genre from the list of genres
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": null
5  }
```

D. Movies Rating

GET /api/moviesRating

- **Description:** Retrieve a list of movie ratings.
- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": [
5      {
6        "id": 1,
7        "rating": "G",
8        "movies": []
9      },
10     {
11       "id": 2,
12       "rating": "PG",
13       "movies": []
14     },
15   ]
16 }
```

POST /api/moviesRating

- **Description:** Add a movie rating to a list of movie ratings.
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
1  {
2    "rating": "R"
3  }
```

- **Response**

```
1  {
2    "success": 1,
3    "message": "All ok",
4    "data": null
5  }
```


PUT/api/moviesRating

- **Description:** Updates a movie rating to a list of movie ratings.
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Request body**

```
1  {  
2    "id": 6,  
3    "rating": "RE"  
4  }  
5  }
```

- **Response**

```
1  {  
2    "success": 1,  
3    "message": "All ok",  
4    "data": null  
5  }
```

DELETE/api/moviesRating/id

- **Description:** Deletes a movie rating to a list of movie ratings.
- **Headers:** Authorization: Bearer <JWT_TOKEN>
- **Response**

```
1  {  
2    "success": 1,  
3    "message": "All ok",  
4    "data": null  
5  }
```

Testing with Postman

To test the API:

1. Use Postman to send a POST request to `/api/users/login` to retrieve a JWT token.
2. Use the token in the Authorization header for subsequent requests to secured endpoints (e.g `POST/api/movies`).

Example Authorization header:

[illegible]