Kotlin-based software development

Stancz Levente

RI8WCW

Movies Backend API

Short specification

The software is made with Ktor. This application is using a public API to send data about movies to the user in which the user can:

- Search for movies by title.
- Filter by:
    - The year of release
    - Type (Movie, Series)

The user can also save movies as favorite and can also remove items from this list. The favorite movies are saved in a database. The application also saves the users information. This means that the user first has to register through the api and then log in to make the requests.

Public API (OMDb api)

The application uses a public API. The API is OMDb api which is a RESTful web service to obtain movie information. The API is by Brian Fritz, where all content licenced under CC BY-NC 4.0.

Database (MongoDB)

The application connects to a MongoDB database to store users data and favorite movies.

Prerequisites

Before starting the application, the user must have MongoDB account which ensures the use of a MongoDB database. The user also needs an API key from the OMBD website. Without these prerequisites the application does not work.

Creating MongoDB database connection

The user must register (or login) to the MongoDB website here. After successfully registering and logging into the account, in the projects menu click on “New Project”. Give a name to the project and click on “Create Project”. After creating a project, in the project menu click on “Database” under “Deployment”. Click on “Build a Database”. Now select the free version and leave everything on default. In the “Security QuickStart” select “Username and Password” and create a database user (do not forget the password as it will be necessary later on). Select “My Local Environment” and “Add My Current IP Address”. After finishing the security setup click on “Overview” in the project menu. Now click “Connect” on the cluster that just been created. In the popup windows click on “Drivers”. Select “Kotlin” as a driver and copy the connection string into the application source code under “main/resources/application.conf” inside the “dbConString” variable. Replace the <password> with the password that was add when creatint a user for the database. Inside the “dbName” variable write the database name that you want, it will be automatically created. The changes made by the application can be found under “Deployment -> Database -> Browse Collections”.

Getting an API KEY from OMDb

To get an api key from OMDb go to the website. On the website click “Api key” on the top navigation menu. Select the free version and fill in the form. The api key should be in your email inbox in less than an hour. After getting the api key from the website put the key inside the application source code under “main/resources/application.conf” inside the “OMDb_api” “key” variable.

Starting the application

To start the application run the source code in any IDE. The application can be started inside the “Application.kt” with the “fun main()” function.

Testing the application

To test the application, I recommend using Postman which helps sending requests to the application. Once postman is downloaded create an account or continue with the lightweight API client. In the application create a collection, after that requests to the application can be created in the collection. Keep in mind that the application only accepts JSON data in the POST requests body. (JSON body can be added in each request at “Body -> raw -> (Dropdown)JSON”).

Here are some recommended requests for the application:

- POST – CreateUser
  URL: localhost:8080/users/create
  BODY:
  ```
  {
          “username”: “username”,
          “email”: “email@gmail.com”,
          “password”: “password”
  }
  ```
  Returns: Created users id in headers (if successful)

- GET – GetUserById
  URL: localhost:8080/users/{userId}
  Returns: User’s data with the given id (if user exists)

- PUT - UpdateUserById
  URL: localhost:8080/users/update/{userId}
  BODY:
  ```
  {
          “username”: “new_username”,
          “email”: “new_email@gmail.com”,
          “password”: “new_password”
  }
  ```

- DELETE – DeleteUserById
  URL: localhost:8080/users/delete/{userId}

- GET – GetFavoritesByUserId
  URL: localhost:8080/favorites/byUser/{userId}
  Returns:
  “User does not have favorites or do not exist” (if user didn’t add any favourites yet, or the user with the id do not exist)
  Favorites collection (List) with OMDb moive id-s

- PUT – AddToFavorites
  URL: localhost:8080/favorites/add/{userId}/{movieId}

- DELETE – RemoveFromFavorites
  URL: localhost:8080/favorites/remove/{userId}/{movieId}

- GET – SearchById
  URL: localhost:8080/search/byId/{moveiId}
  Returns: JSON about the movie with the given id from OMDb

- GET – Search
  URL: localhost:8080/search/{moveiName}/{yearOfRelease}/{type}
  Parameters:
  movieName: required parameter – the name of the movie - case sensitive
  yearOfRelease: optional parameter – the year the move released
  type: optional parameter – accepted types: movie / series / episode
  Returns: Information about the search results in the OMDb api