Lab 3: APIs with Express

In this lab, you will develop a simple back-end for your FilmLibrary using the Express framework. The back-end will consist of a set of APIs to support CRUD operations (Create, Read, Update, Delete) for films. The data will continue to be stored in a SQLite database.

1. API design

Design a set of APIs to support the main features of a web-based FilmLibrary. Data *passed to* or *received from* the API should be in **JSON format**. The APIs should allow a web application to:

- **Retrieve** the list of all the available films.
- **Retrieve** a list of all the films that fulfill each of the following filters:
  - All the favorite films.
  - All the best films (i.e., those rated 5 out of 5).
  - All the films seen in the last month.
  - All the unseen films (i.e., the films without a specified “watchDate”).
- **Retrieve** a specific film, i.e., given its “id”.
- **Create** a new film, by providing all its information (as per the previous labs) – except the “id” that will be automatically assigned by the back-end.
- **Update** an existing film, by providing its information, i.e., all the properties except the “id”.
- **Update the rating** of a specific film.
- **Mark** an existing film as favorite/unfavorite.
- **Delete** an existing film.

List the designed APIs, together with a short description of the parameters and the exchanged entities, in a README file. Be sure to identify which are the collections and elements you are representing, as seen in class. You might want to follow this structure for reporting each API:

```
[HTTP Method] [URL, optionally with parameter(s)]
      [One-line about what this API is doing]
      [Sample request, with body (if any)]
      [Sample response, with body (if any)]
      [Error response(s), if any]
```

2. API implementation

Implement the designed HTTP APIs with Express. Films are stored persistently in the same SQLite database provided for Lab 2. When creating a new film, assign its user to an already existing user (e.g., user with id=1).

Remember to add proper validations to the implemented APIs. For instance, you should check that IDs and ratings are integer values, and you should check that the date format is valid.

3. API testing

Test the realized API with the **REST Client extension** for Visual Studio Code. To this end, you will have to write an API.http file according to the following syntax:

```
[HTTP Method] [URL, optionally with parameter(s)] HTTP/1.1
content-type: application/json (if needed)

[Sample request, with body (if any), in JSON format]
###
```

Notes:

1. The database file "**`films.db`**" is included in the repository available on GitHub: *https://github.com/polito-webapp1/lab-2024/blob/main/lab03-express/films.db*
2. As a suggestion, create a back-up copy of the database before testing your APIs.
3. Visual Studio Code **REST Client** extension is available at the following link: https://marketplace.visualstudio.com/items?itemName=humao.rest-client