Web Applications [2024/2025]

Lab 2: Database integration

In this lab, you will implement the integration between your JavaScript application and a local database.

1. Retrieve data from the database

Modify the program developed in the last lab to integrate it with a local database. To that end, the first step is to download the database provided with this lab called "`films.db`" (see hints below). It contains a collection of films with the fields described in the first lab (note: if you did not complete the lab, you can download the proposed solution and start to work from that one).

Specifically, you should add the following asynchronous methods to the **FilmLibrary** to retrieve data from the database (you are free to choose the name of the methods, but we strongly suggest assigning descriptive ones):

- Get **all** the films stored in the database and return (a Promise that resolves to) an array of **Film** objects.
- Get all the **favorite** films stored in the database and return (a Promise that resolves to) an array of **Film** objects.
- Get all the films **watched today** stored in the database and return (a Promise that resolves to) an array of **Film** objects.
- Get, through a parametric query, the films stored in the database whose **watch date** is earlier than a given date received as a parameter. Return (a Promise that resolves to) an array of **Film** objects.
- Get, through a parametric query, the films in the database whose **rating** is greater than or equal to a given number received as a parameter. Return (a Promise that resolves to) an array of **Film** objects.
- Get, through a parametric query, the films in the database whose **title** contains a given string received as a parameter. Return (a Promise that resolves to) an array of **Film** objects.

Invoke the methods you have just implemented to check if they are working correctly and print the result.

2. Modify the data stored in the database

In this exercise, you will add a set of methods to the **FilmLibrary** object to manipulate the data stored in the database.
Note: before implementing this exercise, create a copy of the local database file since, if correctly implemented, the following methods permanently modify the content of the database.

Specifically, you should implement the following functionalities and invoke them to check if they are working correctly:

- **Store** a new film into the database. Once completed, print a confirmation/failure message.
- **Delete** a film from the database (using its ID as a reference). Once completed, print a confirmation/failure message.

- **Delete** the watch date of all the films stored in the database. Once completed, print a confirmation/failure message.

Hints:

1. The file "**`films.db`**" is included in the repository available on GitHub:
   https://github.com/polito-WA-2025/labs-code/tree/main/lab02-database
2. If you prefer, you can use the available Lab 1 solution as starting point:
   https://github.com/polito-WA-2025/labs-code/tree/main/lab01-node
3. As you saw in the lectures, you can use the SQLite database by means of the following module: **sqlite3** (https://www.npmjs.com/package/sqlite3) – the basic library. NOTE: In some systems, you may need to install some native libraries and/or tools. Check for potential error messages during npm install
4. To browse the content of the database, you can use one of the two following approaches:
   a. Download the Visual Studio Code *SQLite extension* (you can search for it in VSCode extension hub or browsing the following link):
      https://marketplace.visualstudio.com/items?itemName=alexcvzz.vscode-sqlite
      NOTE: In Linux, you may need to install the `sqlite3` native software package of your Linux distribution to make it work.
   b. Download the application *DB Browser for SQLite*:
      https://sqlitebrowser.org/dl/