Enterprise programming

Critical analysis

Work and Techniques

Contents

[Introduction: 3](#_Toc29580503)

[Back-end: 4](#_Toc29580504)

[Database creation: 4](#_Toc29580505)

[Data access using HTTP Web Service Calls - CRUD 4](#_Toc29580506)

[Create 5](#_Toc29580507)

[Retrieve: 6](#_Toc29580508)

[Update: 7](#_Toc29580509)

[Delete: 7](#_Toc29580510)

[Variety of formats used 7](#_Toc29580511)

[Front-end 9](#_Toc29580512)

[Functionality 9](#_Toc29580513)

[Search with name 11](#_Toc29580514)

[Search with attribute 12](#_Toc29580515)

[Delete film 12](#_Toc29580516)

[Insert film 13](#_Toc29580517)

[Update film 14](#_Toc29580518)

[Format retrieval 14](#_Toc29580519)

# Introduction:

A web service based around films was created to display the range of skills and techniques that, studying a Software Engineering degree provided. This service is easily accessible in different data formats, such as Text, JSON and XML, all of which are heavily used in the coding industry. The Film Web Service also has functionality of gathering and manipulating data using CRUD. Coding technologies have been implemented throughout this assignment, to enhance functionality and efficiency of the service created, which will be discussed in detail within this analysis.

# Back-end:

## Database creation:

As this web service requires data to be retrieved and manipulated, a permanent storage solution was essential in ensuring that all the requirements of the web service was met. A database was created on MySQLWorkBench that contained a film table. This table needed to contain important information about the data that would be inserted later on. Figure 1 shows the relevant data that would need to be stored for the web service to achieve the functionality it needs. After researching about Primary keys, it was evident that a filmID was required, but not only that, it had to be unique. Ensuring that was the case, filmID was made to be use the AUTO\_INCREMENT feature.

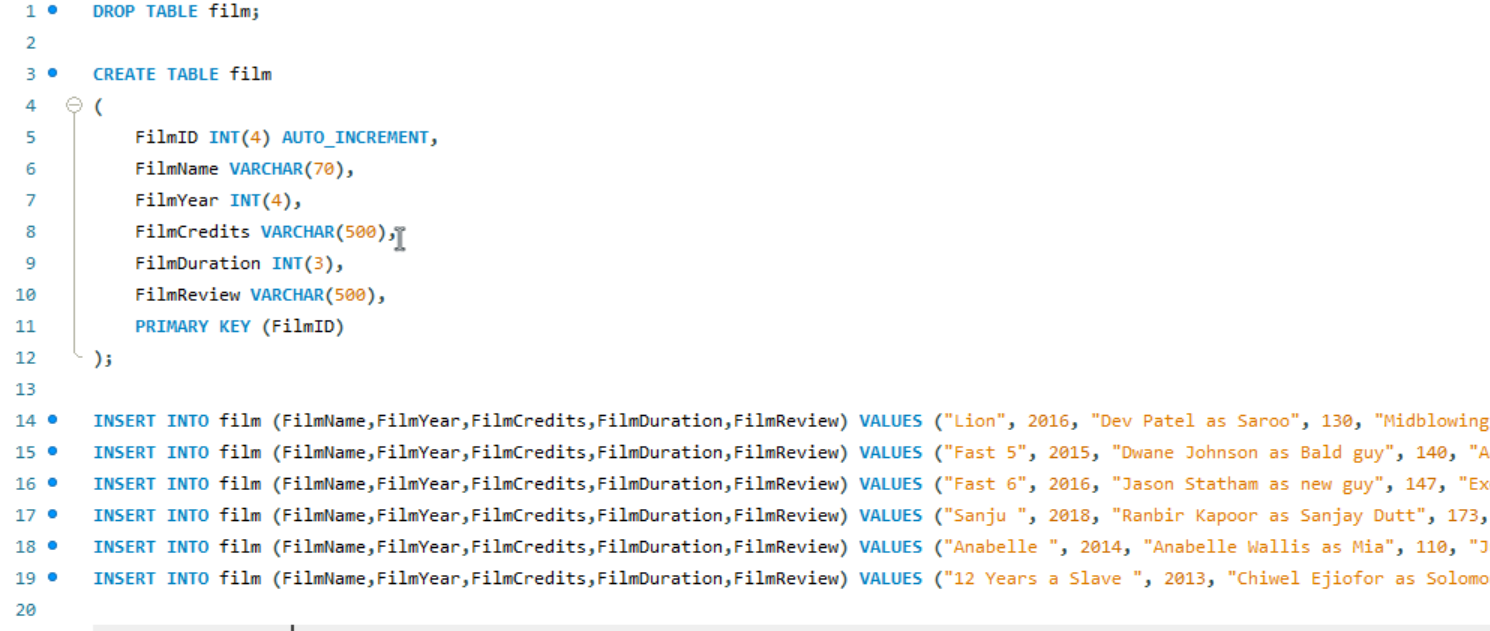


Figure 1: creation and population film table

## Data access using HTTP Web Service Calls - CRUD

As data has been permanently stored in the database, to access and manipulate the data, a data class is required. Film.java was created using MVC pattern is adhered to, located in the model package. This film class specifies how to construct a film object using a constructor with properties of the film, identical to those in the database. Getters and setters and a toString were generated to increase functionality, allowing the ability to retrieve specific data, set specific data and display in a suitable format.

Once this Film java class is created, a FilmDAO is required. This file is where the connection between the permanent storage database and the client side code, will be established. As well as this, CRUD methods are implemented here, as well as some extra functions offering added functionality. Figure 2 shows the connection to the database, with figure 3 showing the CRUD Create method implemented

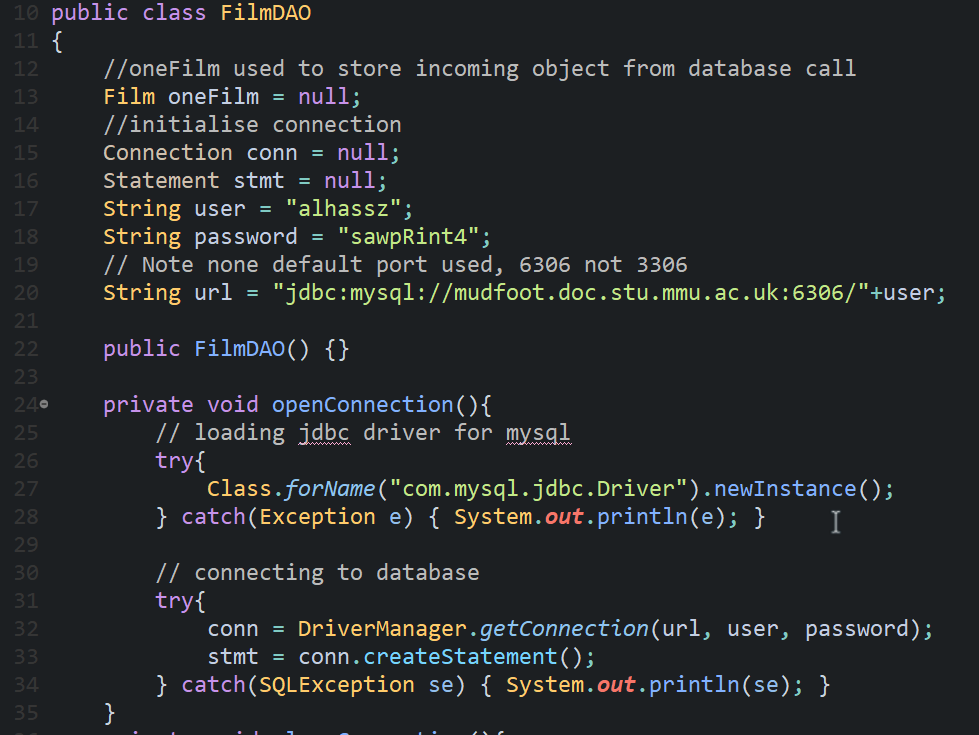


Figure 2: database connection

### Create

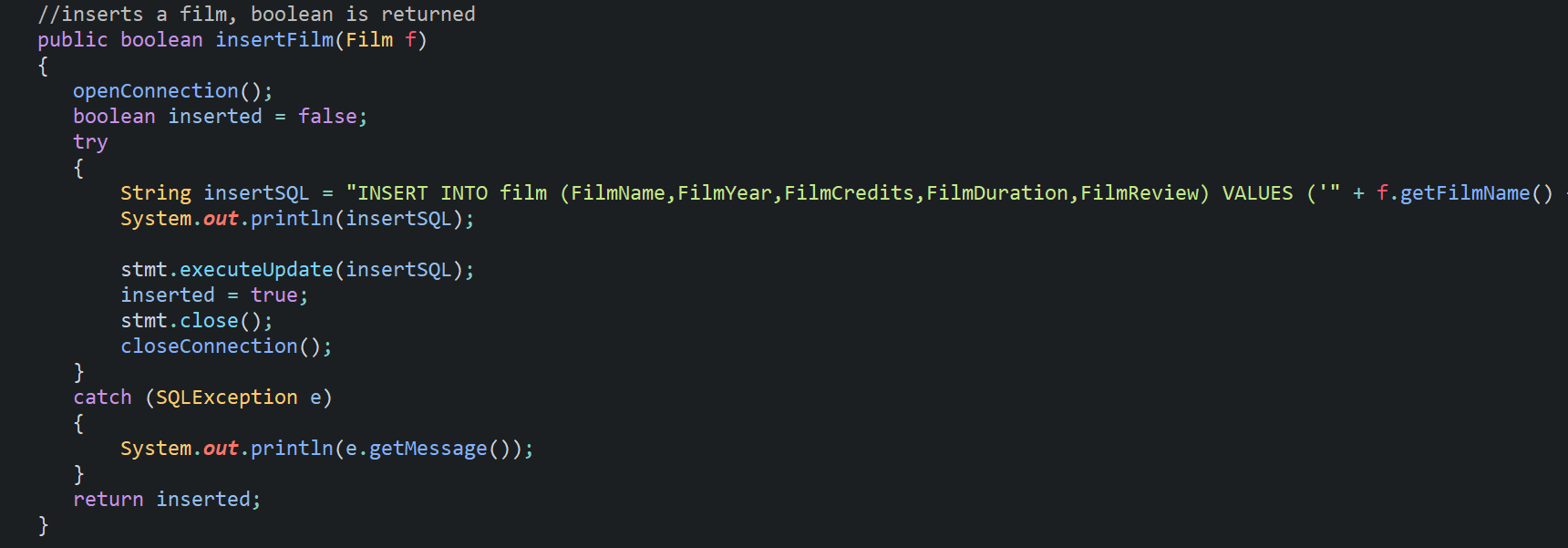
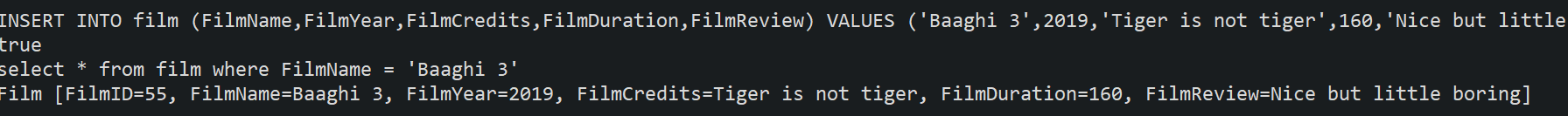


Figure 3: Method to insert a film

To make use of this method, a servlet was used that implemented doGet to insert a film into the database. The insert servlet took film information through the URL to insert the film. A demonstration of this is shown below(Figure 4) where the film has been inserted and displayed in json format, using the doGet method

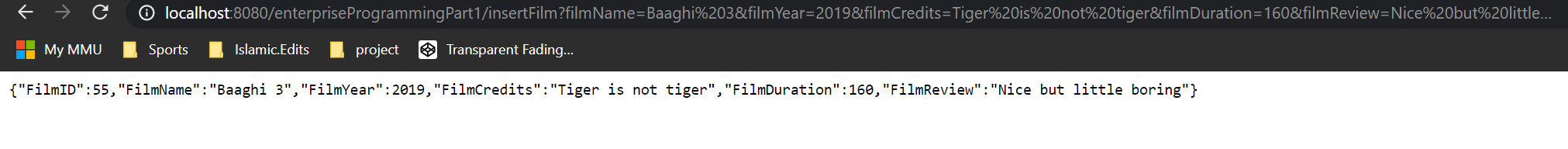


Figure 4: insertion of film Bhaaghi 3

Figure 5: confirmation of insertion in Eclipse console

### Retrieve:

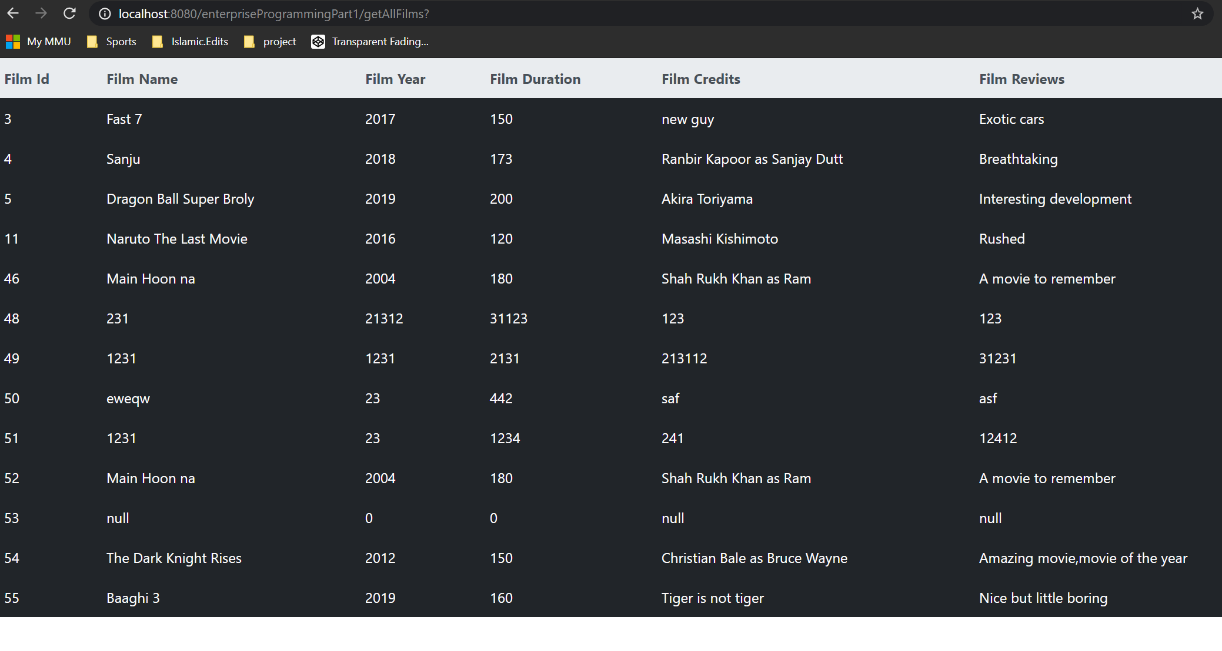


Figure 6: retrieve all films

This web service allows for two types of retrieve: retrieve all films and retrieve specific film. Retrieve all films uses the mainController servlet to run getAllFilms method. This uses an array list to place all of the data being received from the database through the DAO, and then displays in a table shown above in text format(Figure 6).

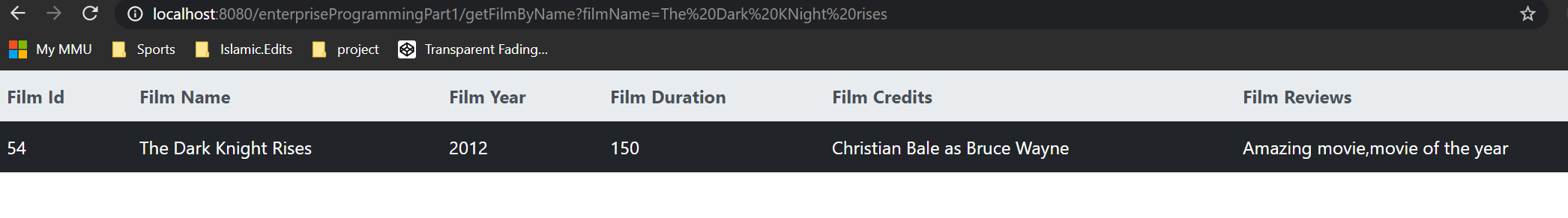
To retrieve a single film, the user needs to specify the film name they would like to search for. Using the getFimByName servlet, a user specified name is provided and a table is displayed showing the relevant film (Figure 7)

Figure 7: Retrieve specific film with name

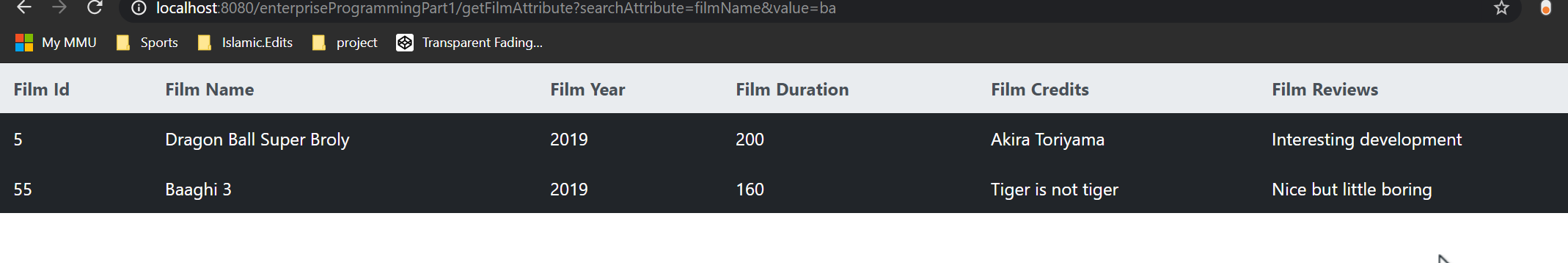
To retrieve a film using a desired attribute, The getFilmAttribute servlet is used, the user specifies the attribute they would like and the value. And this function is using the LIKE operator, this method allows the user to partially enter a film name and the web service will still manage to find relevant films.

Figure 8: Retrieve film using attribute

### Update:

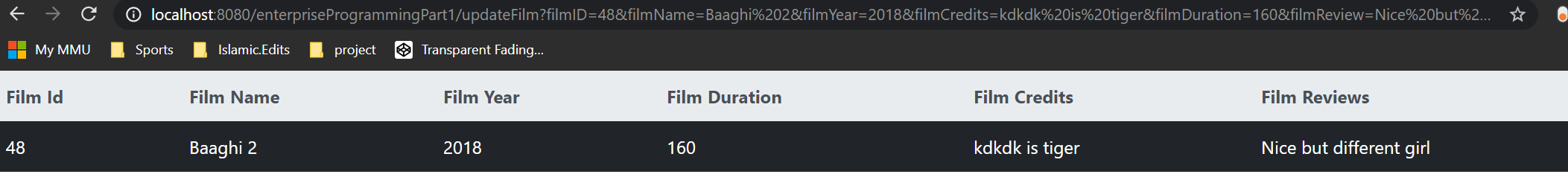
To update a film, the update film servlet is used. This servlet requires the ID of the film to be updated as well as the updated values of the film.

Figure 9: Update film with id of 48

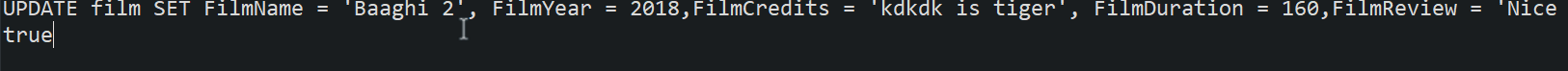


Figure 10: Confirmation of update in eclipse

### Delete:

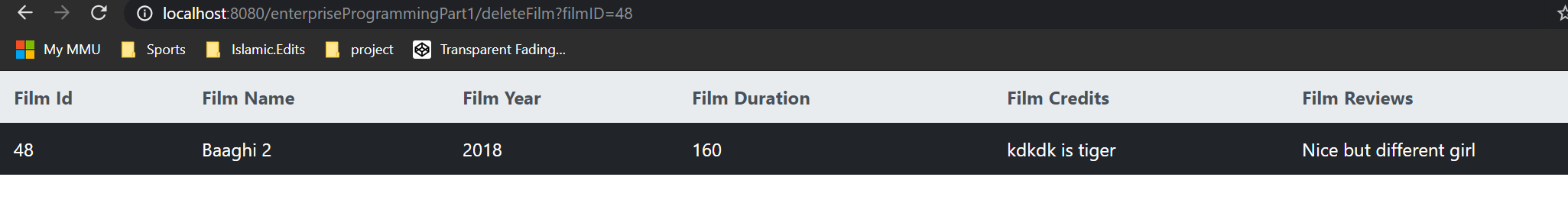
The deleteFilm servlet requires only the ID of the film to acknowledge which film the user wishes to delete.

Figure 11: Deletion of film

## Variety of formats used

JSON,XML, and text are all used to ensure that the web service system allows for a range of functionality in terms of data transaction. JSON’s popularity has risen greatly over the years and it is important for web services to be futureproof. Implementing json format somewhat aids in this as the diagram below goes to show that the usage of json is increasing.

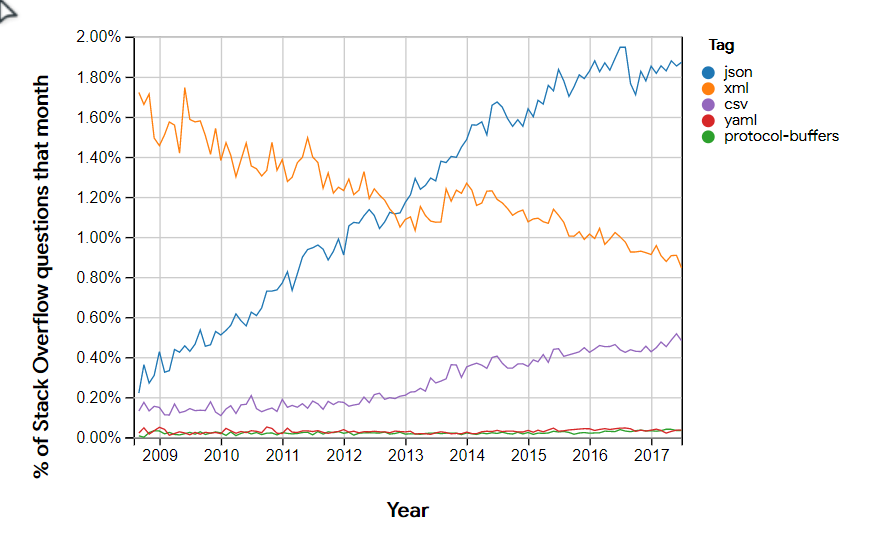
[](https://twobithistory.org/2017/09/21/the-rise-and-rise-of-json.html)

Figure 12: popularity of json

Below, are 3 figures, each displaying a film searched in 3 different data formats.

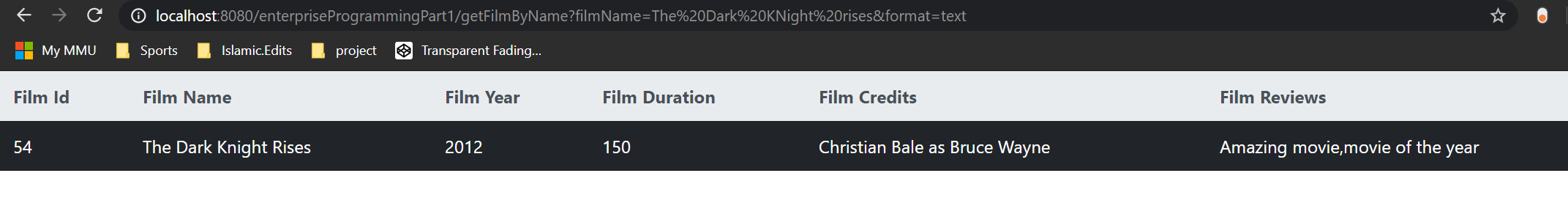


Figure 13: text format

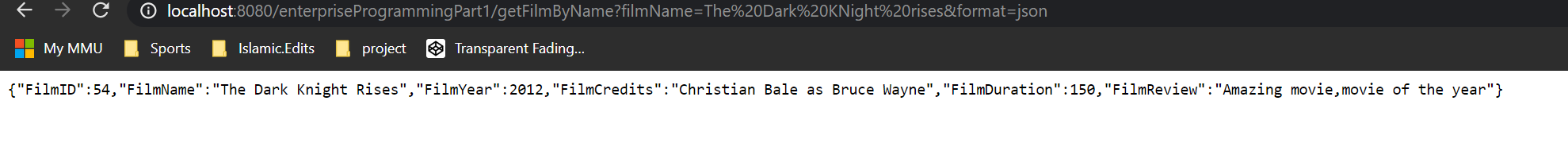


Figure 14: json format

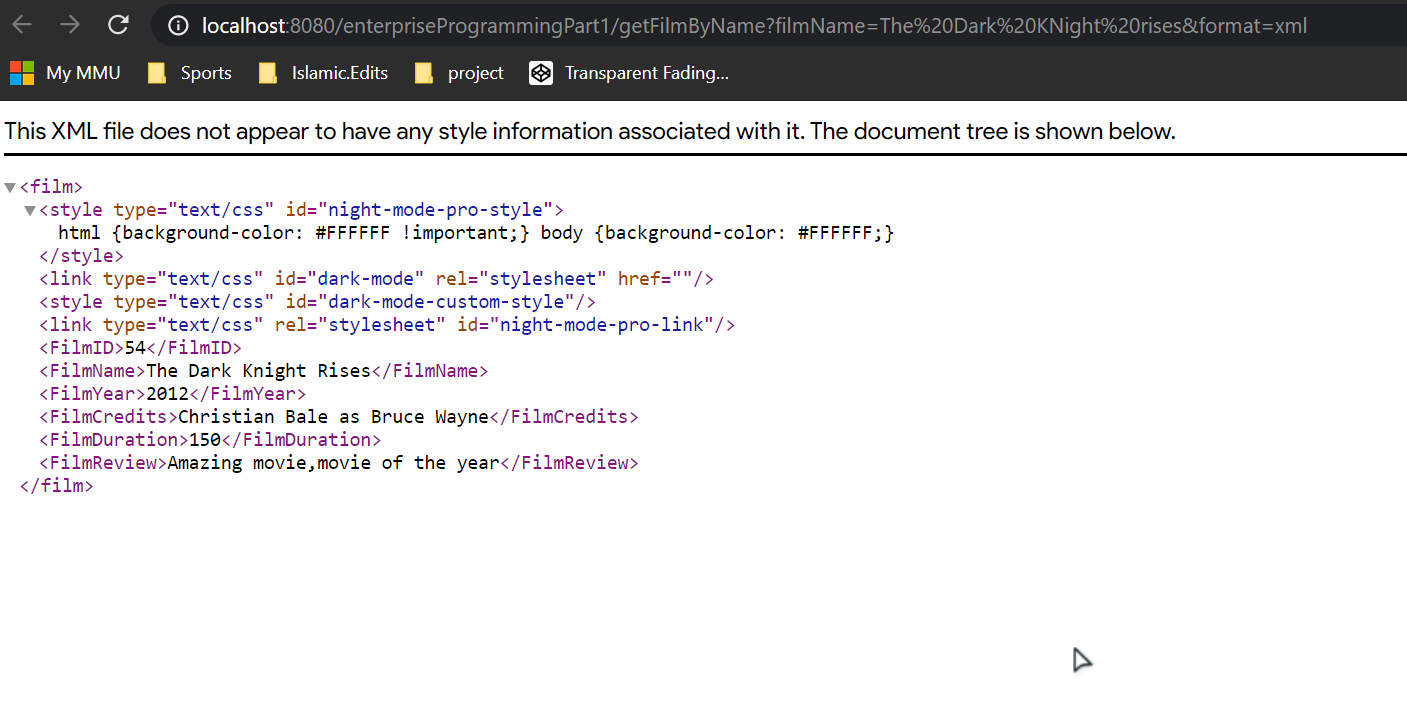


Figure 15: xml format

# Front-end

“It takes 0.05 seconds for an average user to form an opinion of a B2B company by looking at their website.”( <https://www.rishabhsoft.com/blog/why-is-front-end-development-important-for-business>). Front end evidently has a huge impact on the overall opinion that a user will have on a web system.

## Functionality

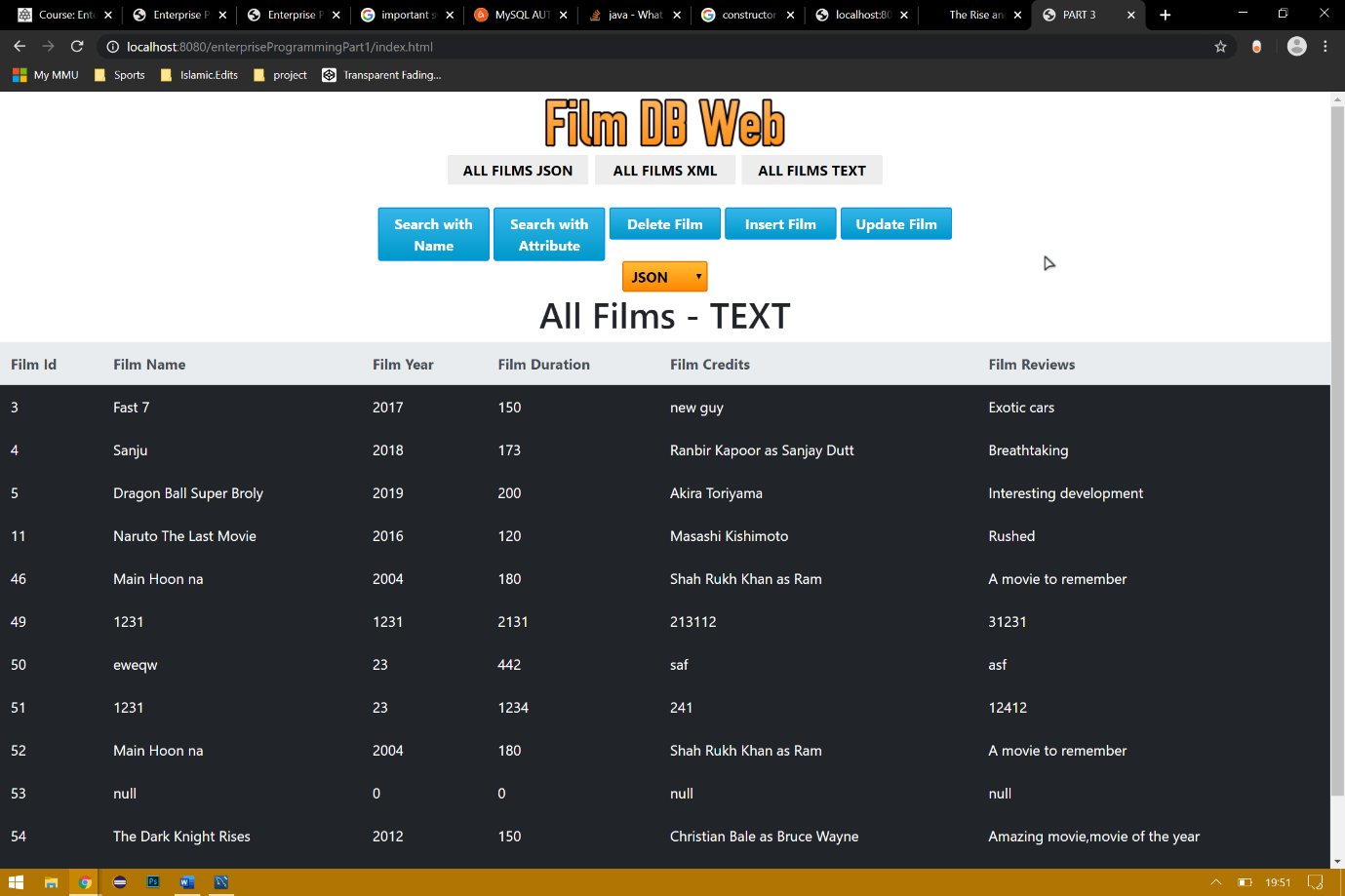


Figure 16: front end main page

Above is the main page of the front end. The Front-end delivers all the functionality of the back-end but also with a more pleasing and easy to use aesthetic. This has known to increase the user satisfaction in using web services.

Figure 16: displays a table in which all films are shown, by default, in text format. To change the format of the all films table, the 3 grey buttons change the format of data retrieved from the server. figure 17 shows all films json whilst figure 18 shows all films xml.

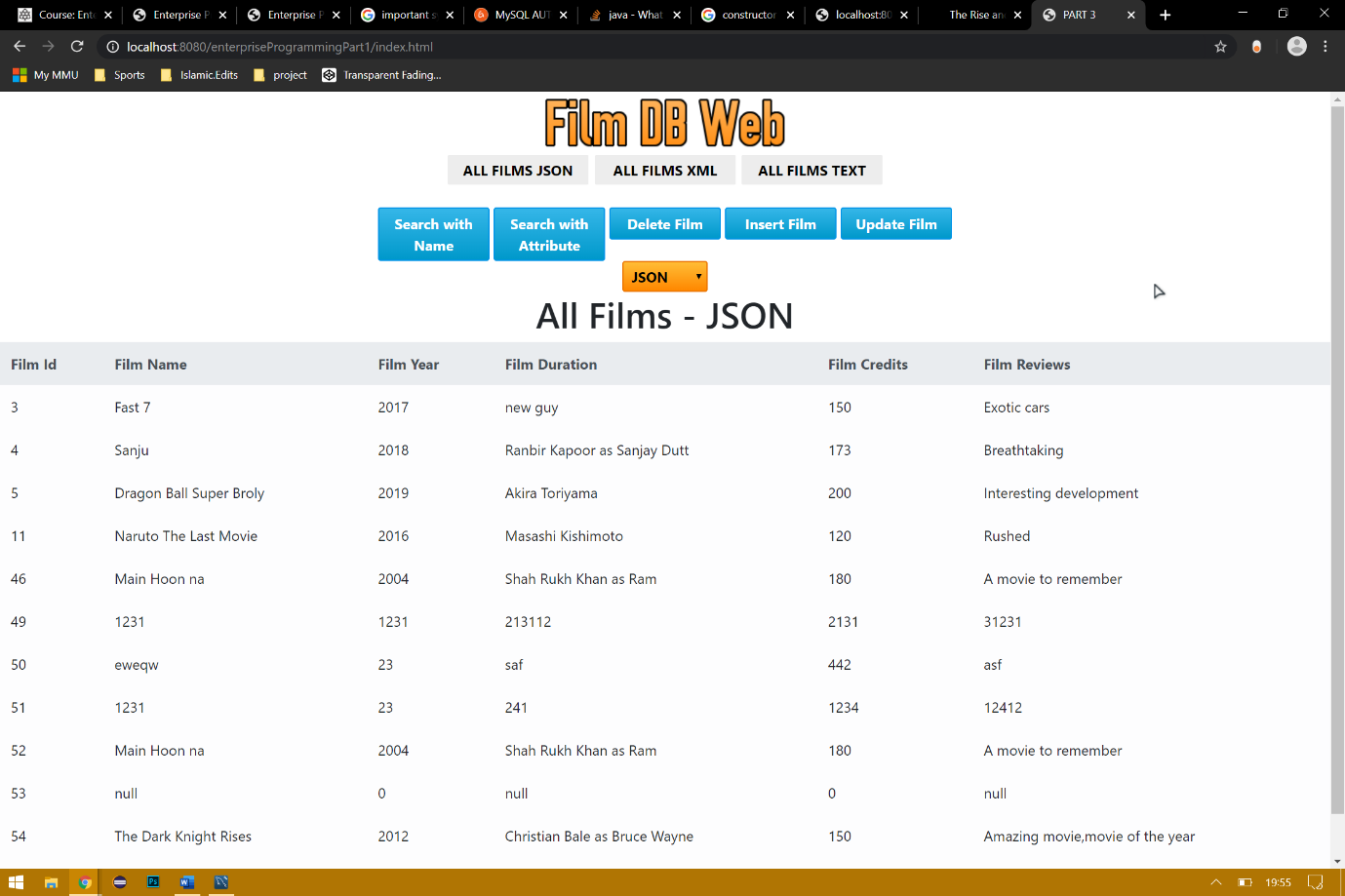


Figure 17: all films json

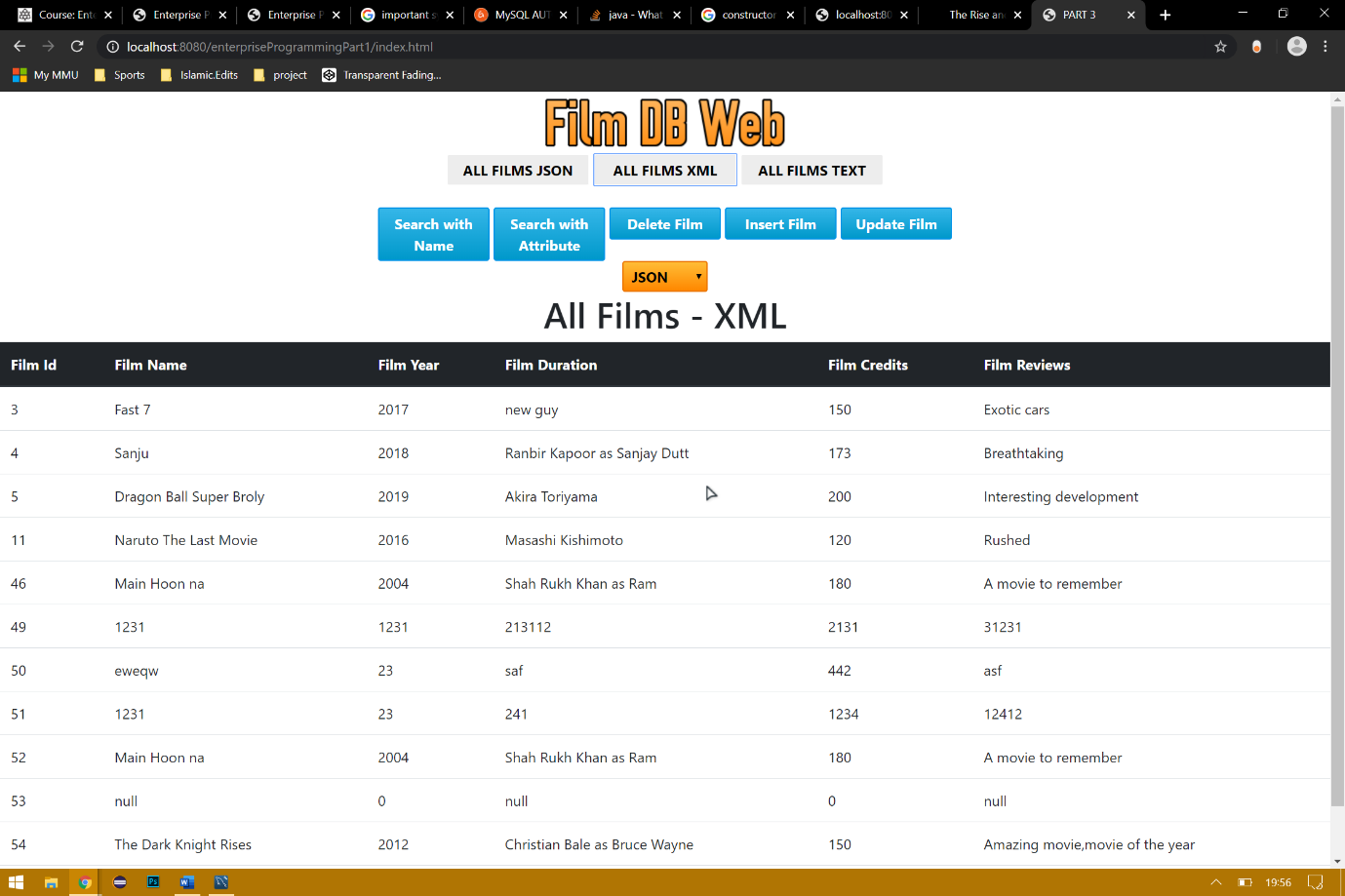


Figure 18: all films xml

### Search with name

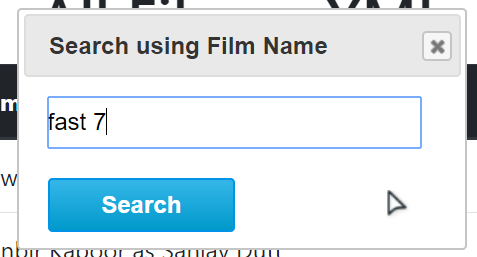
Each film can be searched for individually, using the search with name button. Clicking the button opens a dialog used from jquery ui library. This provides more interactivity between service and user.

Figure 19: dialog box

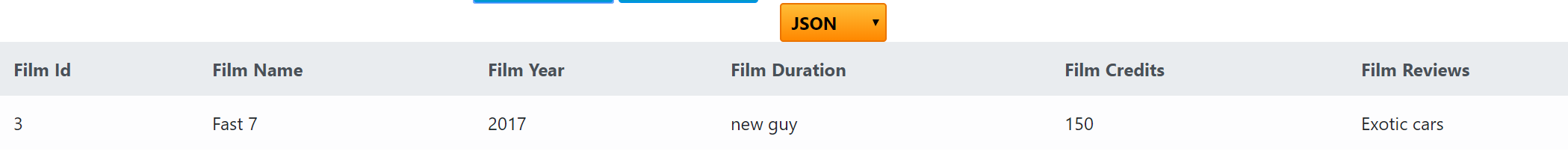


Figure 20: searched film in json



Figure 21: function used

This function uses JavaScript properties as a more efficient way to store data in an organised manner, this property is used in ajax to direct to the getFilmByName servlet. Dependent on the format, the appropriate function is then called to display the data in the table

### Search with attribute

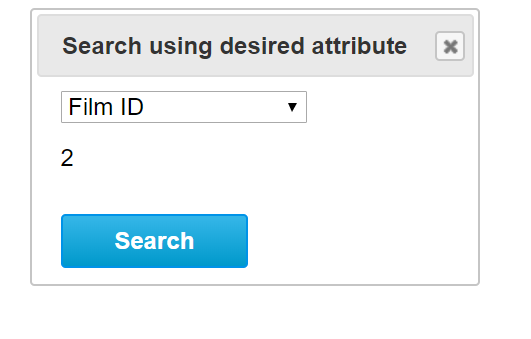


Figure 22: dialog

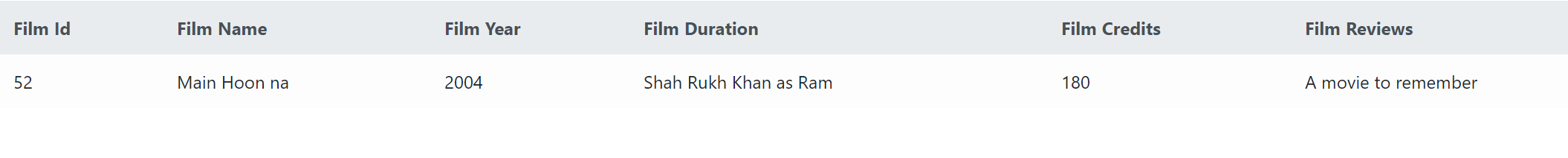
Figure 23: dialog box, filmID attribute chosen

Figure 24: film with id that has 2 in it

### Delete film

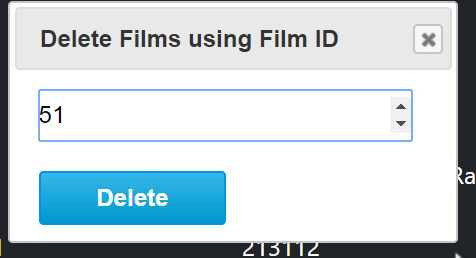


Figure 25: dialog



Figure 26: film id of 24 deleted

### Insert film

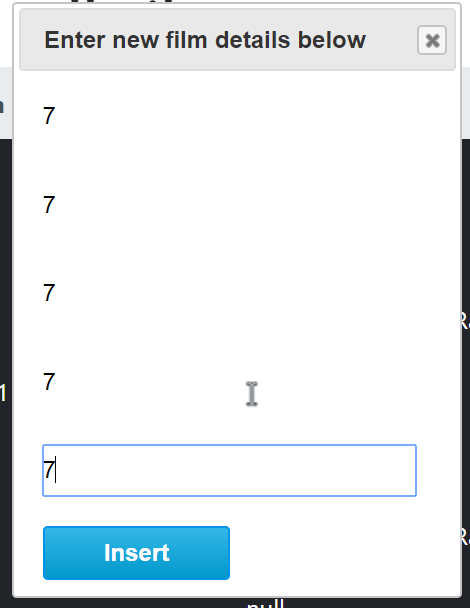


Figure 27: dialog

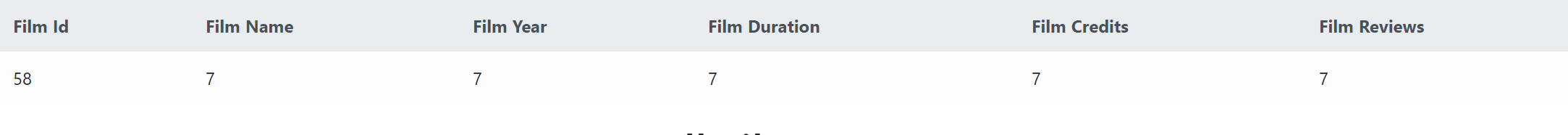


Figure 28: inserted

### Update film

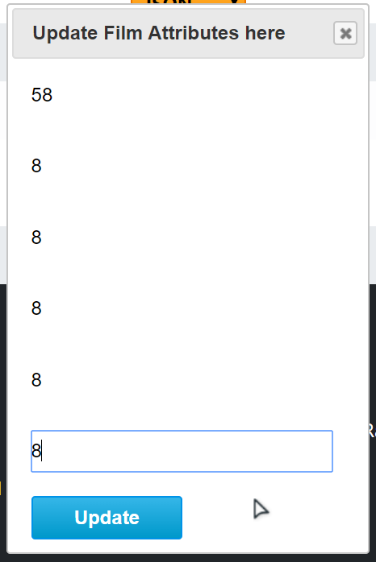


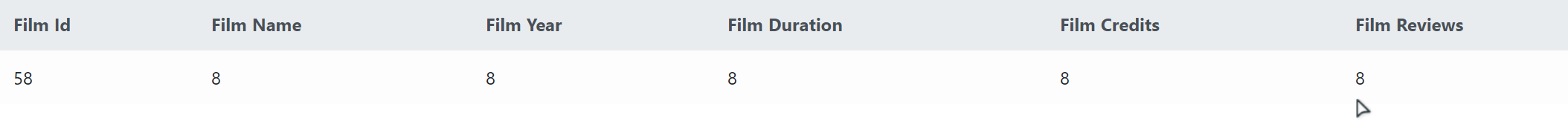
Figure 29: update dialog

Figure 30: updated

### Format retrieval

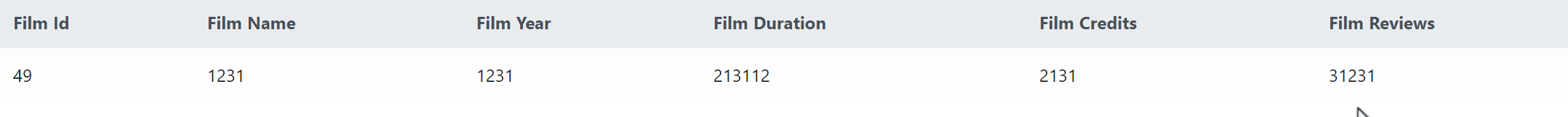


Figure 31:json

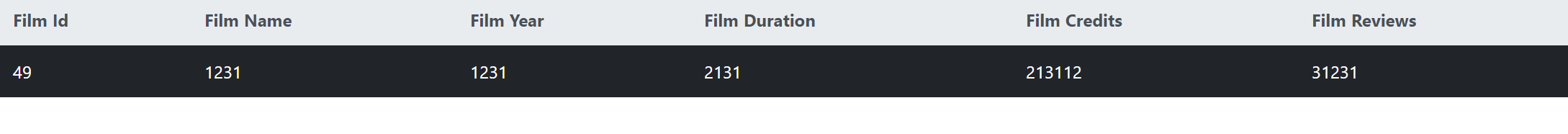


Figure 32:text

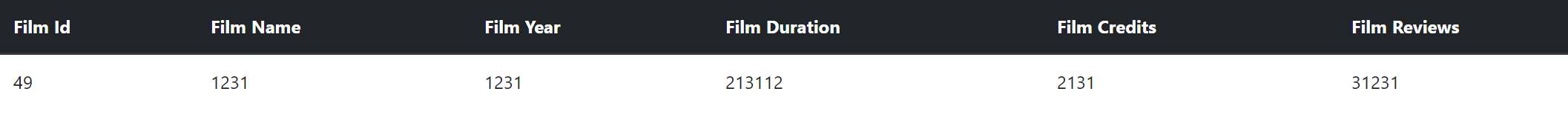


Figure 33: xml

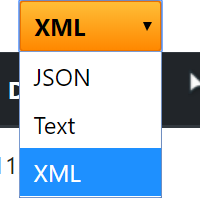


Figure 34: drop down to choose format for crud method retrieval

# Google cloud

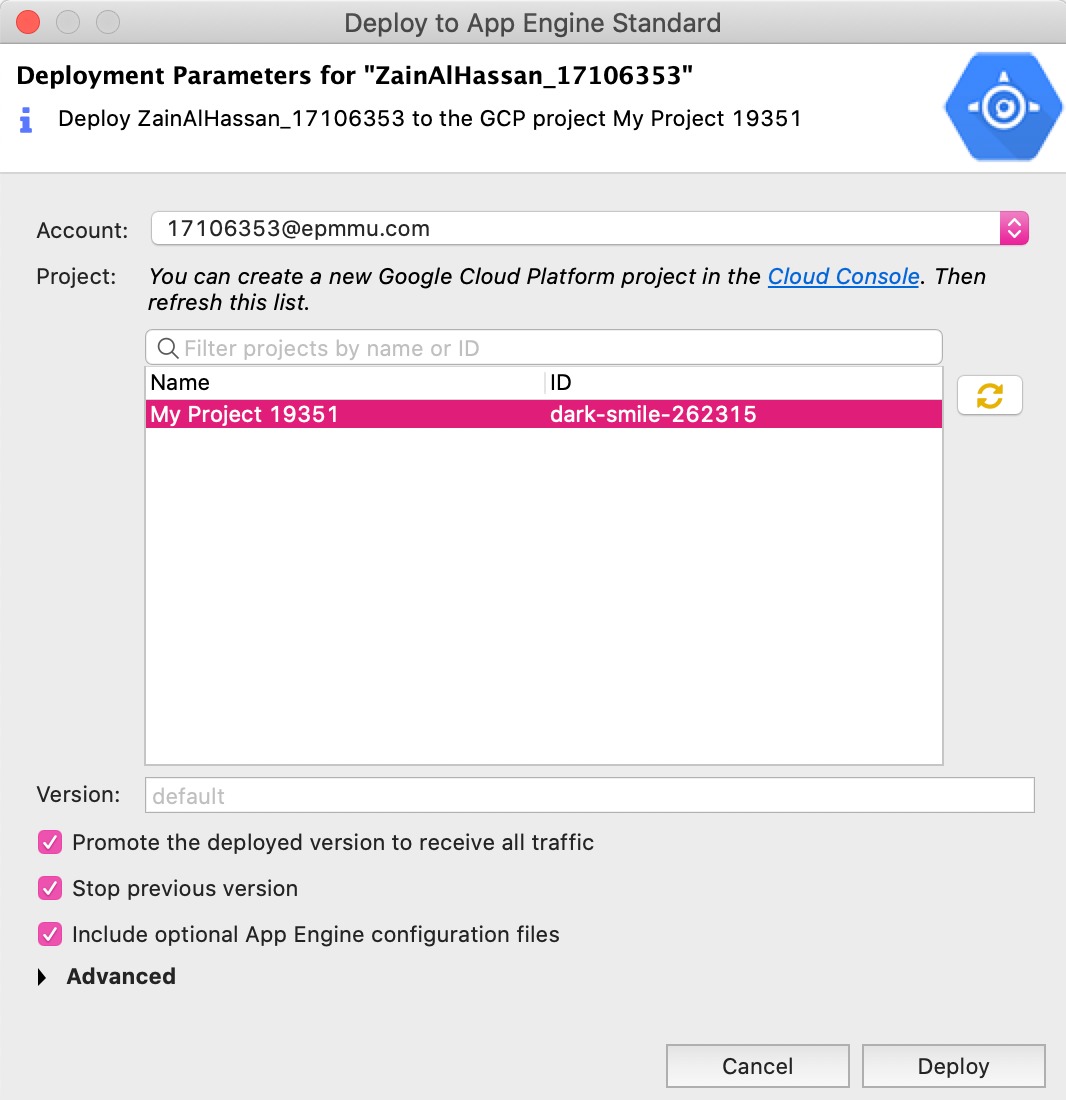


Figure 35:deploying

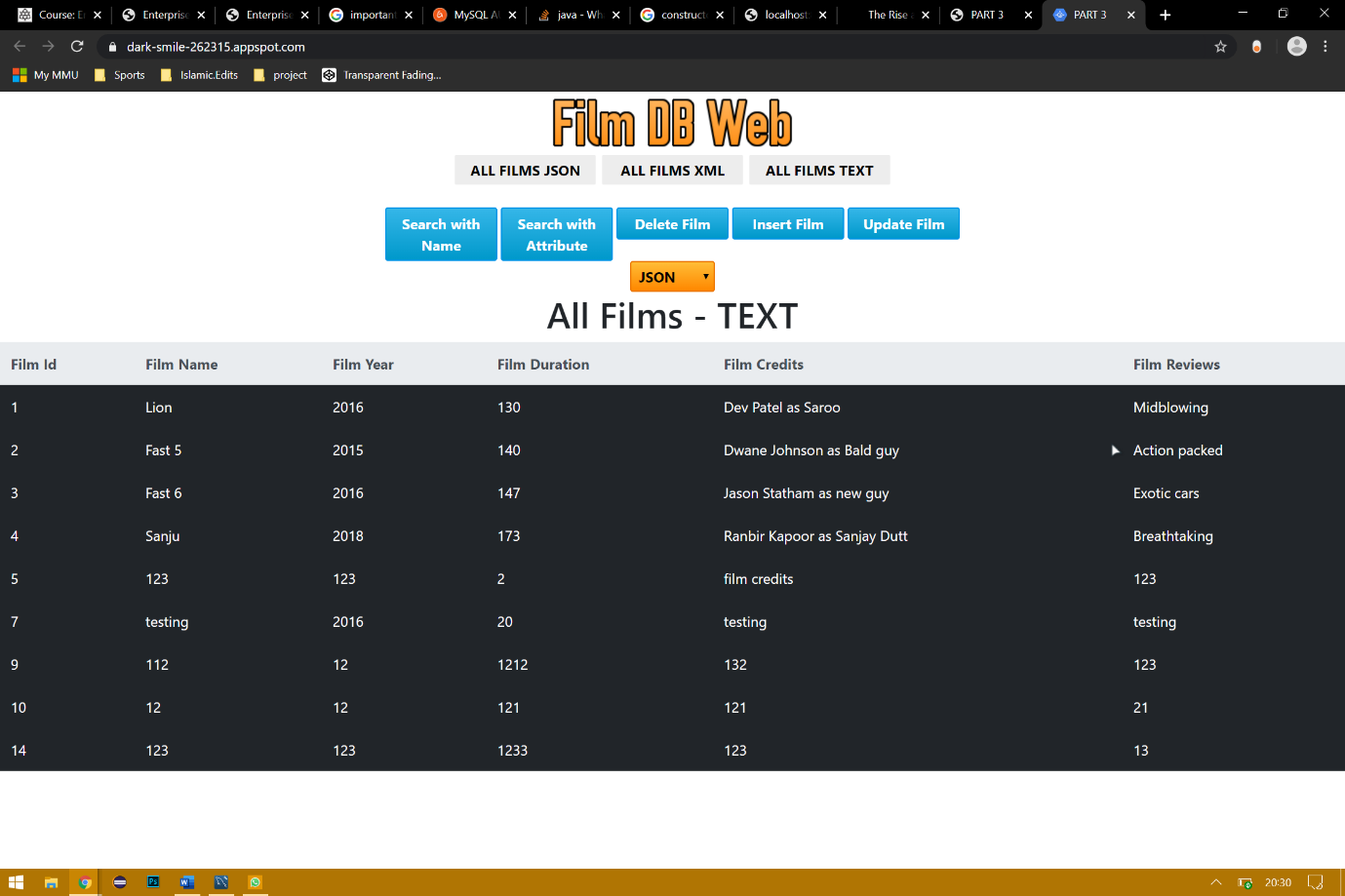


Figure 36:fully deployed web service

# WSDL

A web service heavily based on xml.

