Coursework Report

Alexis Guillonneau
40404292@napier.ac.uk
Edinburgh Napier University - Module Advanced Web Technologies (SET09103)

Abstract

This coursework has for objective to demonstrate the understanding of the Python Flask micro-framework. For this web-application I have decided to choose movies as subject. This collection of movies will contains different information such as the duration, the year of the release, the starring, the producers, etc.

Keywords – Report, Coursework1, Advanced Web Technologies, SET09103, MovieSearch, Edinburg Napier University, Alexis Guillonneau, 40404292

Title

Considering the web-application has for collection movies, I pick a simple title so it be simplier to identify the main purpose. Therefore I use a contraction of two word for the final result **MovieSearch**.

1 Introduction

The web-application MovieSearch is composed of many features but the most important remains the research. However for the purpose of the discovery, ten movies chosen randomly are displayed on a carousel as you can see on figure 1. For the research, a side-bar is proposed to the user on the left side to find movies according a certain category (e.g. Action, Horror, etc). A display of all the movies is also available via the tab "Movies". Another feature is to display for example the movies of a certain producer by clicking on his name via the page detailing a movie.

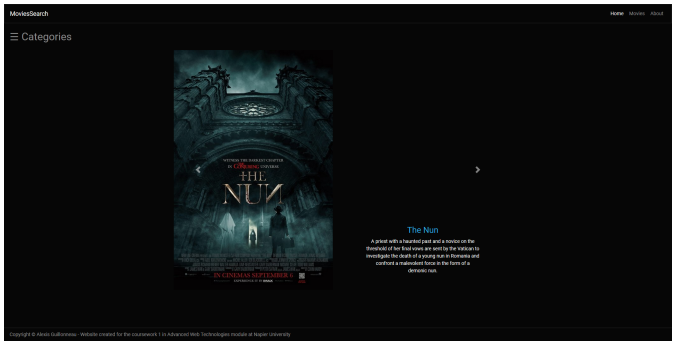

Figure 1: **The Web-application main page** - This main page is mainly composed of a carousel

2 Design

For this web-application, I choose an URL hierarchy which allows me the simpliest way to find different movies according to the chosen criterion (category, year, producer, etc.). So in the first place, we have the main page located at "/". Of it derives all the others pages as you can see in the figure 2. The different URLs such as "/category/<category>", "/year/<year>", etc. will display movies depending of the string of characters chosen (e.g. "/producer/Christopher Nolan" will display the movies produced by Christopher Nolan, figure 5). In the same way "/movies/<movies>" will display all the information about the movie chosen (figure 4). If we go up a notch in the hierarchy, logically the URL "/movies" matches with the tab "Movies", briefly mentioned in section 1. That is the way I will be able to search and display the different information the user wants (figure 7).

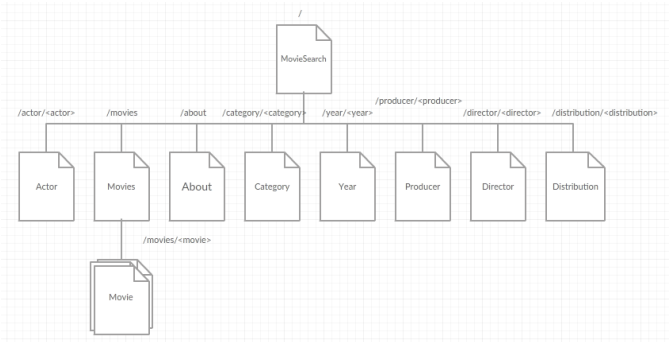

Figure 2: **The Web-application navigation map**

In addition, I have chosen a particular design pattern: the architecture Model-View-Controller. The model is the part that interacts with the database (figure 3). The view, here is our templates, generates output to interact with the user. And the controller is between those two part, it will receive data from the model and will update the view to change information presented to users. Usually, for the model, a database engine such as MongoEngine or SQLAlchemy is used but for our little web-application, SQLite is enough.

It differs from the implementation seen on the workbook, but after some research and some tests, I decided to use the MVC for the coursework.

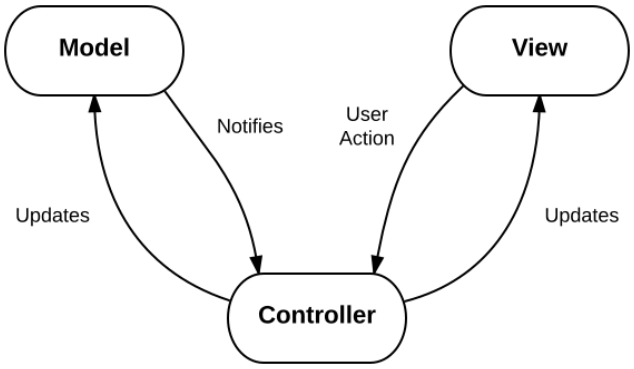

Figure 3: **Model-View-Controller Architecture**

3 Enhancements

Firstly, the feature I would improve in the way to sort the movies in tab "Movies" with dynamic Bootstrap tags. This could improve the user's search experience in a more intuitive way. On another side, I would change the way the database is structured. By creating a relational database with a table for the actors, the producers, etc. In this way, I think it would improve the ability to process large information in the future, because right now there is only few movies in the database but but in a sense of sustainability of the web-application change is needed. To get the list of actors I would have liked to use AJAX to get the information, but I just passed it throught the 'render_template' function of Flask. This is a method I could use more often in this web-appplication.

Secondly, the feature I would add is an account system. Mainly for the use of comments, rating and additions of favorite movies. Always in the idea of personalizing the user experience. So, once connected, for examples, the carousel on the main page could proposed some movies based on the user information. Furthermore, adding more information about movies like related movies, trailers, etc. could be a good way to satisfy the user. Before I mentionned comments about movies, before, I mentioned comments about a movie, but it also means thinking about how to manage the flow of comments. A way of doing this would be adding a column for comments, but for manage all of this, an administrator would be present to handle the stream of comments. It also means an administrator interface for example to add new movie, update or delete data.

4 Critical Evalution

In the web-application I have built, I feel the URL hierarchy work well. Indeed, I allows me to display many information depending of what the user needs. He can access for example to all movies distribued by a certain company, etc. This is an implementation that seems clear and easy to use.

But, the way it is possible to sort the movies in the tab "Movies" is not good as I would like. For the moment, the user can just sort by year and according to a certain actor. in my opinion, this does not allow a real sort of data. A problem also lies in the lack of data included in the database. To fill this gap, a lot of time would be needed but I preferred to spend it in the development of the web-application.

There are also some problems concerning the layout of the carousel for example or the general layout of the movie information. Besides, information on how to use the web-application are missing, this leaves the user alone to discover the different features.

However, the sidebar for the categories is working very well wich it allows to quickly access to the wanted information (figure 6).

For the need of the database, I developed a Python script that allows me to go from the excel file where my information resides to a SQLite file in '.db'. This script is also suitable for relational databases. It is not part of the application but it was useful to me where the evocation in this section.

5 Personal Evaluation

During this coursework, I discover another way to do web-application, it was a great discovery beacause I think Python is a usefull and powerfull programming language so being able to use it is a boon for me. I have encountered many problems in the understanding and use of Jinja2 templates. But after having documented myself through books [1][2][3], using templates became easier. Similarly, the use of Flask's various methods such as 'url_for' took me some time to understand how it works.

During development, I realized that it might be easier and more interesting to use the inheritance offered by Jinja2. For that I managed to have a basic page where I can put the different blocks according to the used routes, this avoids recurring information on every page

I encountered some problems in javascript for sorting movies by year and actor. After a long debugging I realized that was selecting the bad data that came from the Jquery listener. I learned from this to test every output in the function process.

I also think I managed to produce a clean and readable code that could be maintained. Something that I have always had difficulty doing in the different projects that I led.

References

[1] Aggarwal, *Flask Framework Cookbook*. Birmingham,United Kingdom: Packt Publishing Ltd., 2014.

[2] Dwyer, *Flask By Example*. Birmingham,United Kingdom: Packt Publishing Ltd., 2016.

[3] Maia, *Build Web Applications with Flask*. Birmingham,United Kingdom: Packt Publishing Ltd., 2015.

Appendices

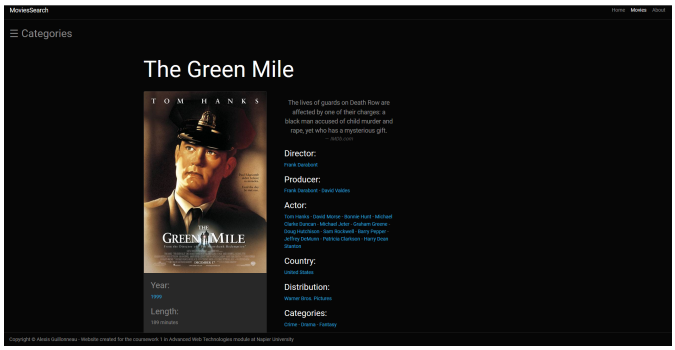

Figure 4: **Layout of the movie details**

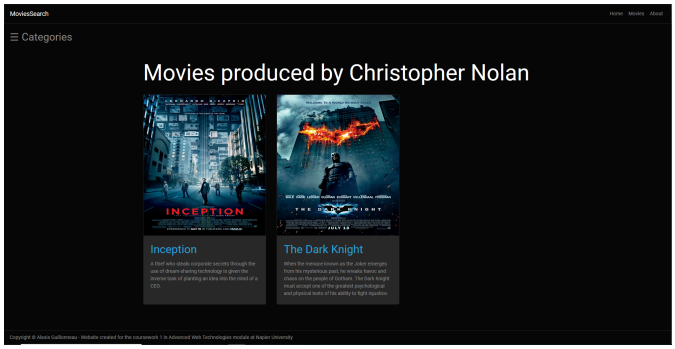

Figure 5: **Christopher Nolan example** - Layout of the movies produced by Christopher Nolan

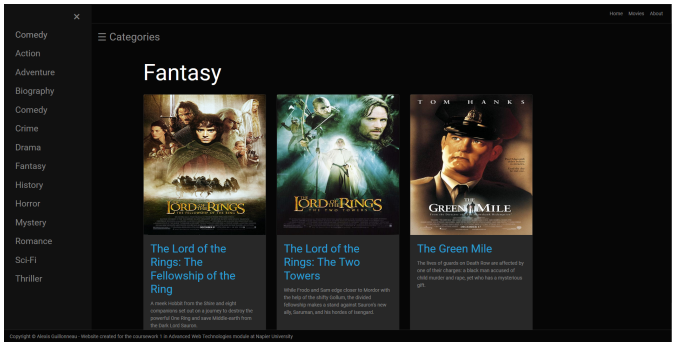

Figure 6: **Example : Fantasy movies** - Layout the category Fantasy

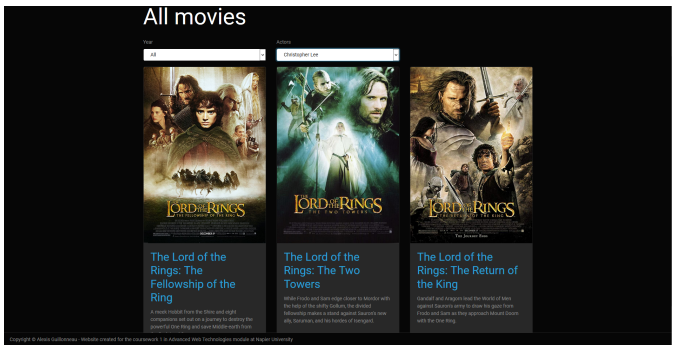

Figure 7: **Layout of the tab "Movies"** - The actor Christopher Lee is selected in this figure