The Movie Generator Project

EDAF90 Web Programming

Malin Cronquist, Anders Klint, Gabrielle Carlsson & Ola Johansson

March 14, 2019

1 Implementations and Conclusions

Originally the idea was to implement a web site that would generate a movie suggestion based on what forecast a weather API would provide. The forecast was supposed to depend on what location the user would enter. However, difficulties in the implementation lead to a Police Station Information API being used instead. A user enters a city, and if the city has a police station they are urged to watch a movie instead of breaking the law. A random movie is then recommended and presented to the user, using the OMDb API.

1.1 Weather turned into Police API

The site was meant to call the Yahoo weather API, which is free of charge to use by individuals and non-profit organizations. It required registration of a project to receive a Consumer Key and a Secret. After that the registered app was requested to be white listed. The Yahoo developer network provided documentation with JavaScript example code for querying the API. The API was OAuth1 protected and needed to be signed with the provided keys. This complected the querying, forcing us to add header information. In the example code this was only provided with jQuery. Angular suggests that the GET request should be done with HttpClient. We made a serious attempt to rewrite the jQuery request to HttpClient code, but struggled with adapting it to being asynchronous. Without any success we decided to change API to one provided by the police authorities with data showing in what Swedish cities police stations are located. This API did not require any authentication and

was therefore easier to request. This somewhat changed the whole purpose of the web site, but because this is an application made only for educational purposes, we thought it was a reasonable decision.

1.2 OMDb API

The OMDb API was used to dynamically fetch data from IMDb.com. In contrast with the weather API, this API was much easier to configure. All that was required was to request a free API key from their website and use it as a parameter in the GET request. After this, movie data from a specific movie could easily be fetched by providing the API key and an IMDb key as request parameters.

However, the goal was to fetch a random movie, and not a specific movie. To solve this, a python script was written to parse the IMDb keys in a GET request using any IMDb list URL and save them to a Json list. The Json list could then be loaded in the front end and used to get a random key by selecting a random index.

In this implementation of the website, a list of the top 250 movies was used, although this can easily be expanded to bigger or more sophisticated lists.

1.3 Learning Angular

We thought it was difficult to get an overview of Angular and its structure. We did not have any experiences of Angular and it would have been helpful to have a lab before or more information on the lectures about Angular. Most of the time went to create the project by following different tutorials, without learning how to code.

The team worked well together, and we helped each other. Most of the code was written together which gave us a good overview of the work we accomplished.

2 Individual statements

- Malin Cronquist: City API, city page and layout. 17 h
- Anders Klint: OMDb API, movie page, python IMDb key script, and routing. 20 h

- Gabrielle Carlsson: City API, city page and layout. 23 h
- Ola Johansson: OMDb API, movie page and routing. 17 h

3 GitHub

Access to our project: The Movie Generator.