Assignment 1 - Personal Plan

Christian Olsson

31 januari 2018

1 Books

1.1 Plan

Since this is the first go at this assignment, there is probably some time needed to get a overview of the existing code, and to understand what is to be done. The actual coding looks fairly straight-forward and should not take too much time.

Get an overview of the existing code: 30 min.
Create the book-object factory: 30 min.
Create book objects for testing: 15 min.
Total: 1h 15min

1.2 Reflect

The book object was created as a factory function in a file 'bookFactory.js' in the dao-directory. In GetBooksResource, a few books was created and printed using console.log.

The existing code was fairly straightforward and did not take long to understand (used 15 minutes for that). The book factory was created using a familiar pattern from previous courses, and took 15 minutes to complete. Modifying the GetBooksResource to log a few books took 8 minutes. Total time 38 minutes.

The actual implementation was much quicker then planned, probably because the existing code was easier to understand than first anticipated. The vagrant reload is very slow, and at least a third of the time spent was used on vagrant issues (not terminating, slow loading etc.).

I have some problems with the editor (Atom), showing all kinds of errors in the existing code supplied due to it not using JavaScript Standard Style.

2 JSON

2.1 Plan

JSON conversion in JavaScript is very simple, using built in functions. This should be a one-line change and should not take more than 15 minutes.

2.2 Reflection

JSON-conversion was made using the built-in function JSON.stringify, and printed using console.log. Elapsed time 10 minutes. Again the vagrant issues was atleast half of the time spent.

3 Web

3.1 Improvement Strategies

The two main annoyances so far is the vagrant issues when restarting the web server, and the error messages from the Atom editor. I will try the following:

- Using the Vagrant SSH to shorten reload times.
- Removing the JavaScript Standard Style linter from Atom, so that only actual errors are displayed in red.

I estimate this to take 30 minutes.

3.2 Plan

The GetBooksResource needs to be changed, and use the API for the assignment. This will probably take some reading up on the API (15 minutes). The changes seems easy enough and should not take more then 15 minutes.

Estimated time 30 minutes.

3.3 Reflection

The Vagrant SSH was easy to fix, just remove the autostart parts of the vagrant file, and then connect to the virtual machine using vagrant ssh. I disabled the linter in Atom, and the false error messages are gone. This took 20 minutes. I think things like these are hard to estimate how long they will take, since the solutions are not always obvious.

It took some 15 minutes for me to navigate the API documentation and find the 'callback' function in the GetBooksResource object (I got stuck using

a return statement), but when that was sorted and in combination with the much shorter vagrant reload times, the implementation took less then 2 minutes.

4 Vision document

4.1 Reflection

I tried to write the vision document using plain text, but I found it tricky to include all the content without getting to specific. There is also not much written in the course literature (at least I didn't find it). The idea of a book library is straight forward enough, but it's hard to find a purpose for a software like this. I made something up that could be of use. To get an overview of what the software should contain, I made a list of stuff that should be included. The entire vision document took 1h 20min to complete.