Learning diary: Niko Kareno
Anytime-course: Software Development Skills: Back-End 2020-21

Introduction:

I've done miscellaneous starter projects using different frameworks and tools, such as Angular and React connected to Firebase and Amplify, but I felt that by using these I did not get a very good understanding of the underlying stack. Working with these tools seemed to be a confusing mix of framework specific templating languages and using plugins.

From this course I'm expecting to get a better understanding of the back-end stack up until the front facing api that I could then use for front end and/or mobile development. I've taken the anytime front end course and currently completing the mobile course too, so I'm hoping to connect the different pieces of the puzzle with this.

Introduction to REST:

I have worked a bit with REST APIs - as a user. From user perspective these seem very intuitive, but this is of course because most of the APIs I've used have been public and built, structured and documented well. The introductory lesson did not really provide much new information to me, but it was a nice re-cap of the concepts.

Node JS:

Other than using node package manager in angular or react projects to install libraries etc., I haven't really used Node before.

Starting off, the CommonJS notation was already a surprise to me. I've had some experience with front end JavaScript development, but had never encountered it before for some reason. The differences were not big, but later on in the course when I was trying to do some more work with the front end side I had some issues with these differences.

Working with the url, path, fs and os was a good reminder. I've used these (especially fs and os) with Python previously, not so much with JS.

Setting up the server was something that I had copy/pasting before without properly understanding what is actually happening. The lesson was explaining it very well and I think I now understand what's happening behind the curtains a bit better. The routing part seems very heavy in this implementation (as said in the video) and I've worked with Angular routers for example so understanding what they actually do is interesting.

All in all, I feel it's good to learn about the different layers of the stack at least on this crash course level. I've been 'Youtube learning' things before and worked my way through tutorials, but often the tutorials are focused on a very specific set of tools and don't explain how and why the routers and static folders work the way they do. This video had a very nice approach of actually explaining these a bit more in depth - or perhaps also gaining more and more experience on these topics helps me to understand these deeper.

I did not do the Heroku deployment at this stage as I decided I'll do it at the end for my own project. That was a huge hassle, but at the end succeeded. After half a day of googling and stackoverflowing I managed to set up the project so that it ran on Heroku as well. Had some issues related to the production vs. dev dependencies in package.json, which were not super clear from the Heroku dashboard error logs. Once I realized to use the 'heroku run bash' and run the start script from there and see 'heroku logs --tail' logs, I managed to get to the right path. I had previously struggled with the environment variables and setting those to Heroku was also something that was not covered in the videos, but google helped me with.

MongoDB

In some other tutorial projects I had already installed and dabbled with MongoDB, but again - not fully understanding many of the details. So getting into the proper usage was interesting. I've done courses on relational databases and MySQL, and have had difficulties in getting my head around the differences between NoSQL and MySQL (because I had been doing these 'copy/paste' tutorials with NoSQL).

Not 100% decided on my personal preference regarding relational on NoSQL databases - I see advantages in both. Somehow the data structures in relational databases seem more intuitive (probably because I'm so used to them), but the flexibility of NoSQL seems promising (although also a bit frightening).

I managed to set up the MongoDB Atlas nicely and it's cool to have the db running in the cloud so easily, still having proper CLI and the Compass GUI to it.

ExpressJS

I was very eager to get to the Express section, to finally actually get the pieces together. The instruction video was very nicely scoped and explained that the purpose of the Express in this context is more on building the API for the database, instead of serving a dynamic website. It was good that a simple UI was built with the express-handlebars to make the usage a bit more concrete, but for me the scope was perfect, because I have previously experimented with other templating frameworks (without proper understanding of the data layer) and it was easy to understand which layer are we working with now.

Also getting better understanding of the routing helped me to understand other frameworks and their structures a bit better. Perhaps the Express lesson could have had a bit more complex data model (more entities) so that the project structure for routing would have been covered in more width. Have to check out some other videos for this perhaps (or experiment on my own).

I was really surprised about the easy use / building of the schemas with Mongoose & MongoDB - as opposed to my previous experiences with Java and MySQL. When moving onto my own project after the Express lesson, I was excited to see that much of the new structures I was creating started working pretty much out of the box with very few issues with the data layer.

Own project

As my own project I decided to work on a very basic example of a to-do app (which seems to be a tutorial favorite on so many cases). The reason for this is that I wanted to focus on building a simple back-end for

an easily comprehensible data model. I was worried if I try to do something too complex, I might get lost in the actual logic of the functionality instead of focusing on the main point - which is the proper handling of a variety of requests against a well defined api. I'm planning to use this same back-end and API for another course. I also added an API page for this own project, where one can test the calls to the api and see the JSON responses.