Lappeenrannan teknillinen yliopisto

Sofware Development Skills

MONTBLANC Emmanuel, 000740166

LEARNING DIARY, Software Development Skills: Full Stack

**LEARNING DIARY**

20.4.2022

I checked the general information on the moodle page. I set up the git repository for the module. I chose to use VS code as it is the IDE I am the most familiar with.

I understood what is node.js and its usage. Installed it.

I learned how to use npm to install packages, Exporting / requiring

I then discovered the npm Path module, that is almost the same as the Path library in Python so that was quite easy to understand.

03.05.2022

I continued the node.js crash course and learned about the modules: url, http, os, as well as how to respond to events and emit events. I then learned how to create a server, how to use scripts to launch nodemon and not having to stop and restart the server everytime. How to use the url to load automatically the file and check for errors.

I then installed Heroku and added my files to it so I can execute it on the server.

10.05.2022

I followed the crash course on mongoDB, so I first installed mongoDB and compass.

I learned how to:

* Create database and collection with the shell or using compass
* Insert a document, or multiple at once. Show, or find specifics rows, count them, sort them. Apply a function to the result of the search
* Update a whole document or just a specific field, increment a field, rename it.
* Delete a document.
* Add a subdocument to a document
* Find a document by a element in array
* Create index
* Do a search by text
* Use operator greater and lesser than to find documents.
* Modify documents using compass

I then created an account to get access to atlas and created a cluster and connected to it with the shell, created a collection there and added a document to the cluster database.

11.05.2022

I discovered what is express and its usage, I Installed postman to make requests easier. I then understood the concept of middleware and how to create one with express, as well as how to use routes to separate different parts of the code.

I learned how to set up a static folder. I learned how to create a basic API, and how to answer to get, post, put and delete requests with an example of a local member list.

I then learned how to use express-handlesbars to render templates. Using it I encountered a small error, ‘exphbs is not a function’ and apparently I needed to use exphbs.engine( … ) instead of exphbs( … ), which was not the case in the crash course.

12.05.2022

I learned how to create an angular workspace and project. How to serve the application, and do changes like the tittle or the stylesheet.

create a new component using the CLI and display it and its properties.

I learned how to edit the data of the component using the two-way data binding with ngModel, which required me to import a module to the application.

I then learn how to display a list using \*ngFor, as well I using condition within the html file with \*ngIf and changing the style of a component based on a condition with angular’s class biding.

I then created a second component, and learned how to pass information to it using @Input.

I then learned how to create services, and use them to refactor the code. I created a first one which was register at the root level to make accessible anywhere in the application and another one that is injected in the first one. I also discovered Observable objects that are used for asynchronous operations.

13.05.2022

I then learned to use the angular router to navigate through components by defining routes, adding links to the templates files and getting parameters from the routes. With this I refactored some parts of the code.

I then modified the code to use HTTP in the app and get the data from a web API. I refactored some services to support post(), put() and delete() to add news features to the components.

17.05.2022

Set up the dependencies, created the app.js file and set a static folder, initialized the express routes and the connection to the database. Created a user model using mongoose schema and added a function to register users. Also learn how to encrypt the data for passwords using bcryptjs.

18.05.2022

I then followed the part on the api authentication and token, and added a function to check the password and create a token when connecting.

I spent a long time on an error ‘Login sessions require session support. Did you forget to use `express-session` middleware?’ and didn’t really understand where it came from, and even with using the source code of project I still got it. In the end I found out that when I commented the line app.use(passport.session()); the error was gone and the authentication still seemed to work.

I then started the angular part of the example project, which required me to uninstall node to reinstall an older version in order to be able to use an old version of angular-cli.

I then initialize the angular source folder, create the routes and components for login, register, profile, dashboard and home, as well as a navigation bar that allows us to go from on page to the other.

19.05.2022

I created the page for the homepage as well as the login page with a form. I created a service to check if the all the fields are filled and if the email is valid. I then discovered flash messages, that allow to create small popups to display warnings.

I then created another service to connect to the backend and add the user to the database when using the register page. I added to this service a function to authenticate the user when on the login page as well as storing the user and his token to the local memory. Next, I added to this service a function to logout, that empties the local memory. I added a function to get the profile of the user by loading the token from the local memory, for the profile page. Following this I added a function to check if the user is logged in using tokenNotExpired from angular2-jwt, to allow me to only show the relevant buttons on the navbar. I then learned how to use canActivate to protect routes and automatically redirect when not authorized. Finally, I built the application with the ’ng build’ command and redirected every route to the index.html file.

20.05.2022

I started to work on my final project. The idea would be to keep the architecture of the mean stack tutorial, and to add features to it to make it a blog website. The first thing I did was to create new collection for articles with an article schema and a route to write them. The article has as attributes: a title, a body text, a writer and a date, I plan on adding comments, but I will add them later. I then tested the creation of article with postman.

I then went on to add a new page for writing articles. I first created a new component ‘write’ with the angular cli and added a form in the component with a submit button. Then I created a service to actually post the article to the database. Once I got the writing part done, I started working on displaying the articles, I chose to display all of them on the dashboard page. In the ngInit of the dashboard component, it will use a service to get all of the articles, and then I can display them using \*ngFor in the html file. I struggle a bit to display the date in a readable way at first, but then I found that with angular you can simply do ‘{{ article.date | date:'short'}}’ to format the date.

21.05.2022

I started to implements comments, the first thing I did was to create the API function to add comments to the database. I had to modify the article schema to add attributes comments which is an array of object, where each comment is composed of a writer, a commenttext and a date. To add a comment to an article, I just take as an input the id of the article, the writer and the comment, I then get the article from the id and do a push on its comments attribute to add the new comment.

I then displayed the comments for each article using another \*ngfor. To be able to comment on the article I added a form to each article with a text input for the comment and a button to submit. When submitting, it gets the id of the article, and call a new function of the article service which will connect to the backend to add the comment to the database. After this, the page reload itself to see the new comment.

22.05.2022

I added buttons to sort the articles by either date or popularity. I defined the popularity as the number of comments an article have. To do so I added a variable in the component that change when clicking on the button, after ngOnInit() is called, refreshing the page and checking the variable to get how to sort the articles.

I then wanted to modify a bit the page to write the articles, so far I just used a form-control input, which was only one line and not really convenient for writing a whole article. I chose to use instead textarea which was more suited for my usage. The only problem was there was no style for it in the css file I used (from the tutorial), so I had to download the css file and modify it to copy the same style as the form-control for the textarea. Finaly I modified the home page so that the register and login button doesn’t show up once loggedin, and instead added a button to go the dashboard.

I then build the application and filmed the video of the project running as well as creating the ReadMe file to explain how to execute the project.