Mobile Applications Development - Coursework 1

Manuel Alaminos Dominguez
40333504@napier.ac.uk
Edinburgh Napier University - Web Technologies(SET08101)

1 Introduction

This is the second and latest coursework of Web Technologies module at Napier University. We had to develop a full-stack blog site. This is, client side and server side with a RESTful API and CRUD operations.

The core technologies that we would use were HTML, CSS, JavaScript for the front-end. For the back-end, we would make use of Node.js as well as Mongo DB for the databases, a NoSQL type of database. As we can see, JavaScript is the main technology chosen here, front-end, back-end but also database, as Mongo DB uses JavaScript syntax, so everything is familiar from top to bottom.

However, we could make use of different frameworks and libraries if we needed to do so. In my case, I had planned on using Angular as the front-end framework, Express micro-library for the back-end, and Mongoose to make things simpler in the database too.

My decision to use the Angular JavaScript framework was driven by two main concerns:

- I will start a placement work on 2nd of May, in which I will be using Angular for the front-end. Therefore, this was a good opportunity to learn and use it.
- Despite its steep learning curve, a JavaScript framework like Angular would facilitate some operations. Keeping a single page application, which would be rendered depending on the actions of the user.

Just for clarification, Angular is not Angular.js, the first Angular developed by Google, also known as Angular 1.x. Angular is Angular 2+, currently with Angular 6.0 in beta, soon to be release, also developed by Google.

The use of Express.js and Mongoose has a very simple explanation. They seem to offer a very straightforward and simple way of doing usual tasks that would otherwise take a little bit more time by reinventing the wheel every time we want to perform simple tasks like routing. They also seem to be very popular and quite the standards these days.

The blog would offer the possibility to log in as a user, and the ability to perform CRUD operations. These are: Create posts, retrieve posts, update posts and remove posts.

Both users and posts would be permanently stored in a Mongo DB database.

Three weeks previous the deadline of this submission I started to plan my study time and created a study road-map, as they were two main things that I did not work with previously, Angular and Node.js. Angular applications are developed with TypeScript as well, a typed superset of JavaScript that compiles to plain JavaScript, which allows me to check for errors previous compilation as well as providing me with static typing, classes and interfaces, something that I also had to learn about. My experience with Mongo DB was limited and mainly theoretical rather than practical. Therefore, my background reading would be quite extents this time around.

The study road-map that I followed was the following:

- Angular-Tutorial: Tour of Heroes
- Mosh Hamedani Node video
- Mosh Hamedani Building RESTful APIs with Node and Express video
- Mosh Angular course from Udemy
- Maximilian MEAN stack from Udemy

Obviously, my background reading had to be extended even further to fully understand different topics. I relied on the following sites:

- Angular official documentation
- RxJS API for Angular
- MongoDB official documentation
- Mongoose official documentation
- Express official guide and documentation

There were some other sites I visited, but the ones mentioned above were the ones I most frequently visited during these three weeks.

2 Software Design

The chosen approach to develop the Blog web-application was the popular and well known Agile software development methodology.

The software design requirements were the following:

User should be able to

- Log in
- Log out
- Add post
- Update post
- Delete post

The web-application should be able to

- Retrieve list of posts
- Check if user is valid and store it in the local storage
- Deliver the user actions to the server

Further in my study plan of the study road-map mentioned in the Introduction section of this report, I started having a better idea of what I could do and how, therefore I started sketching with pencil and paper and drawing some basic wire-frames.

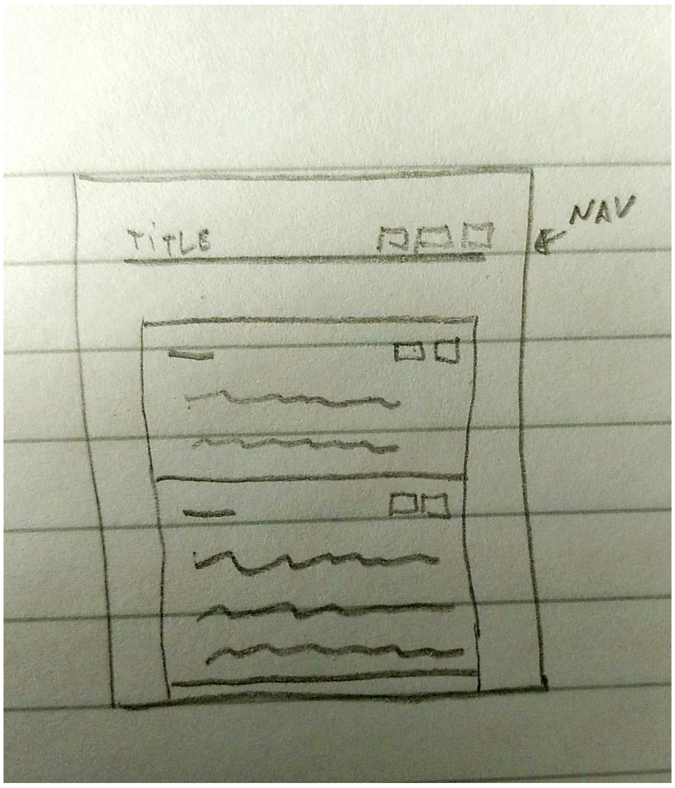


Figure 1: **Sign in page** - sketching

The blog web-application first page that you encounter is the sign in form, once you put your credentials and if they are valid, you will be automatically navigate to the home page.

In the home page you can perform all type of tasks: Add a post, log out, edit a post or delete one. You can also see the list of posts and their content.

When you click on Add or the icon for edit a post, an Angular modal component will appear. This modal contains a form in which you can either create or update a post -depending on what you previously clicked to make this modal appear-. This modal is made of a form, in which the user can set up the Title and Description of a new or existing post. You can either click the Add button to proceed or click the X icon to get back to the home page without modifying anything.

The Delete icon will make another modal appear, this time is to confirm that the user wants to delete an existing post and proceed with this action.

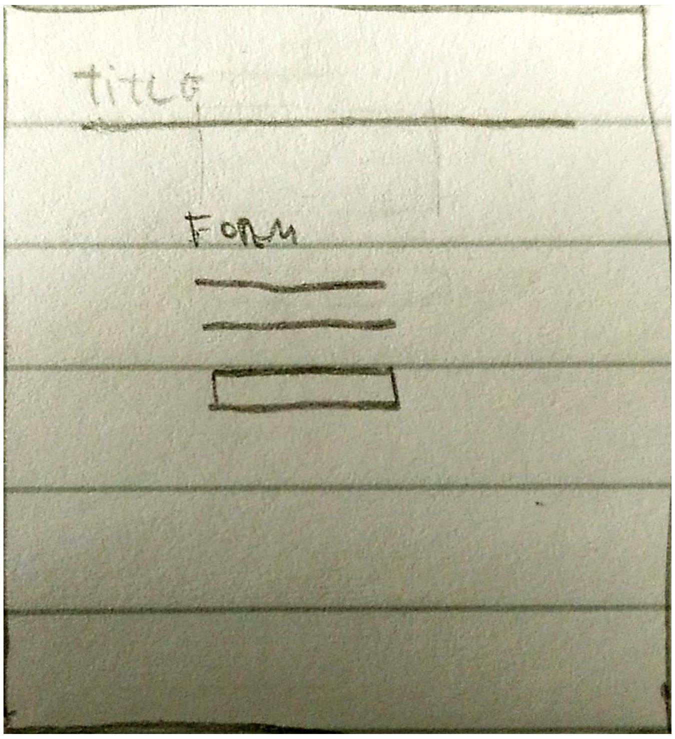


Figure 2: **Sign in page** - sketching

3 Implementation

The blog web application is clearly structured, and it can easily be identified by two main parts: client and server.

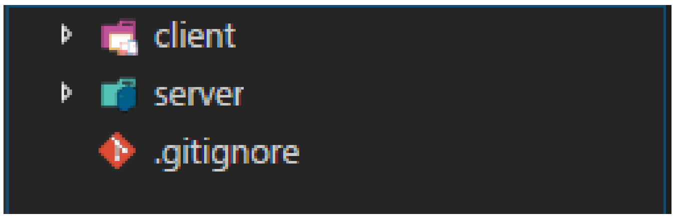


Figure 3: **Folder structure** - root

The client folder is where the Angular front-end application is stored, which contains:

- Its own node modules
- Its own package.json file
- The angular-cli which was used during the creation and development of the blog site
- The front-end application itself with its different components, services and routings

The server is where Node.js lives, which contains the following:

- Its own node modules
- Its own package.json file
- The models for user and post schemas
- The node application itself, from which I imported Express.js, Mongoose, the user and post schemas and

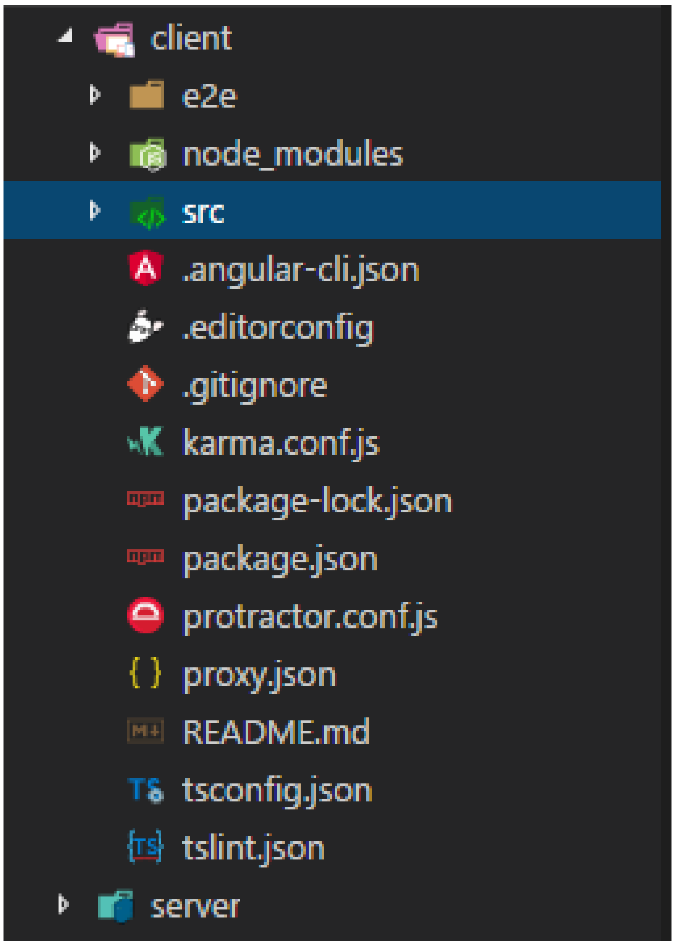


Figure 4: **Folder structure** - client

where the RESTful API is set up waiting for the services in the front-end to call them to proceed with a given action

The Angular CLI is an incredible program that allowed me to easily create the application and set up different components. Some of the used commands were:

- ng new nameOfApp (Creates a new Angular application)
- ng serve / ng serve âĂŞopen (serves the application in localhost:4200 by default)
- ng generate component nameOfComponent / ng g c nameOfComponent (Generates component), which automatically create the different component files (.html, .css and .ts) and import it in the app.module file

Angular automatically compile TypeScript to JavaScript as well, so there was not need for me to manually type âĂIJtsc main.tsâĂİ or anything like this.

MongoDB was set up locally. In Windows 10, I had to run mongod.exe to start the database, and mongo.exe, a shell program that allowed me to create the database, its collections and add users if needed. Despite downloading the MongoDB GUI software, I stayed with the shell version of it.

NPM was used to install and manage all the packages needed: Angular, Express, Mongoose and so on. NPM was also

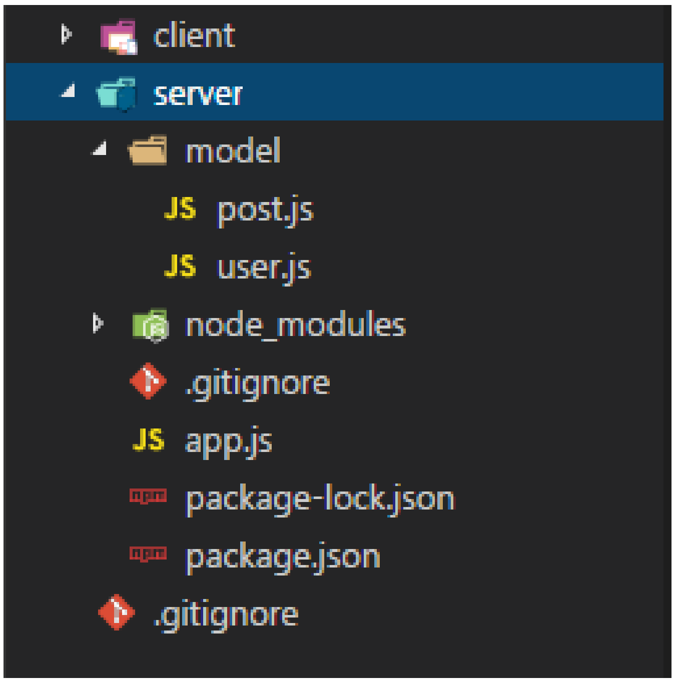


Figure 5: **Folder structure** - server

used to start the application rather than using the command âĂIJng serveâĂİ directly, as I set a proxy and put it in the json file¿ scripts¿ start. So, a simple âĂIJnpm startâĂİ in bash would compile and start the front-end application.

For the server I just had to navigate to the server and run the node application by typing âĂIJnode app.jsâĂİ in bash as well.

4 Critical Evaluation

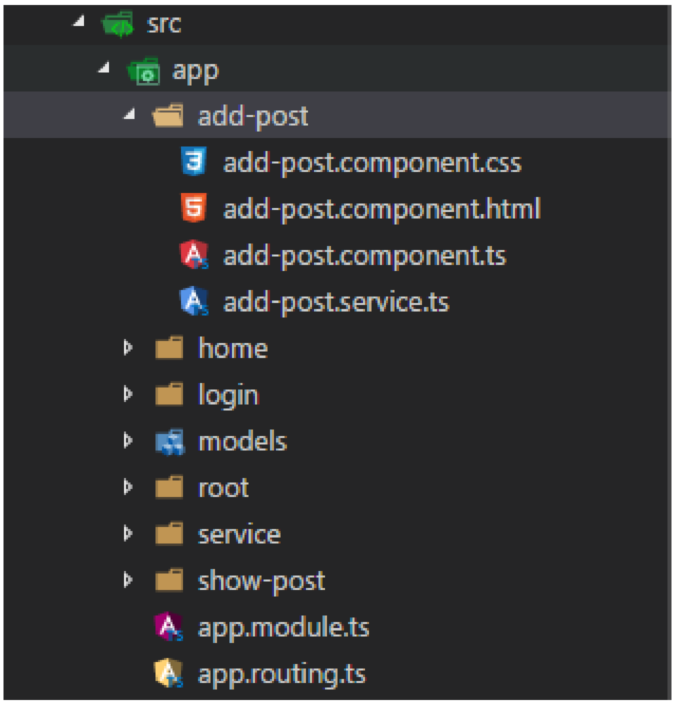
Compared with the software design requirements, the Blog final implementation has accomplished what was expected. Furthermore, it has been implemented with Angular, being this a single page application type of website. And also allows me to log in with different users.

If, however, we compared this blog with a usual blog you would encounter in the Internet, the blog is terribly basic. It lacks the possibility to sign up as a new user (more on that in the personal evaluation), likes, comments, dates, user image, users profile, and more.

Visual wise, the Blog is too simple in my opinion. I focused on the functionality first and wanted to proceed to improve visuals later, but that time never came. More on that in the following Personal Evaluation section.

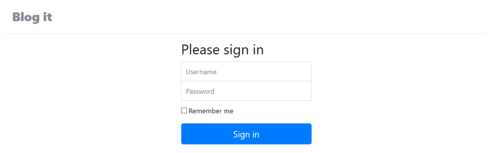
On the positive side, the blog is quite easy to navigate through, and most of the needed actions for the user are being performed within a single page as I intended, with different modals, so it is very easy for a user not to get lost while navigating through different pages of the web application.

There is so much room for improvement. Starting with the possibility to sign up with a new user from the client side.



```
src
  app
    add-post
      add-post.component.css
      add-post.component.html
      add-post.component.ts
      add-post.service.ts
    home
    login
    models
    root
    service
    show-post
    app.module.ts
    app.routing.ts
```

Figure 6: **Folder structure** - angular app



Blog it

Please sign in

Username

Password

Remember me

Sign in

Figure 7: **Signin page**

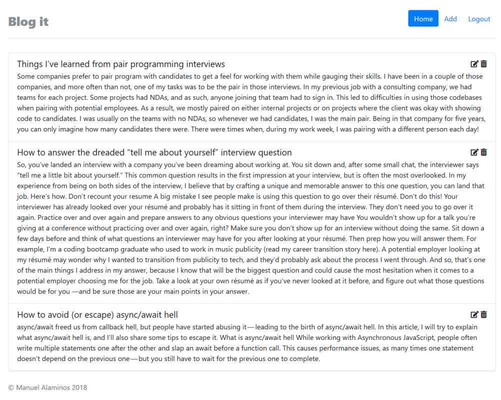
A basic âĂIJlikeâĂİ feature would have been very welcomed. And the design of the website needs improvements to make it much more appealing.

5 Personal Evaluation

When it comes to my personal evaluation during this coursework, I have mixed feelings.

On the one hand, so many technologies used in this coursework were new to me and I had to learn almost from scratch the following ones:

- Back-end web development
- RESTful API design
- Node in general
- Express library
- MongoDB in practice
- Mongoose library



Blog it

Home Add Logout

Things I've learned from pair programming interviews

Some companies prefer to pair program with candidates to get a feel for working with them while gauging their skills. I have been in a couple of those companies, and more often than not, one of my tasks was to be the pair in those interviews. In my previous job with a consulting company, we had teams for each project. Some projects had NDAs, and as such, anyone joining that team had to sign in. This led to difficulties in using those codebases when pairing with potential employees. As a result, we mostly paired on either internal projects or on projects where the client was okay with showing code to candidates. I was usually on the teams with no NDAs, so whenever we had candidates, I was the main pair. Being in that company for five years, you can only imagine how many candidates there were. There were times when, during my work week, I was pairing with a different person each day!

How to answer the dreaded "tell me about yourself" interview question

So, you've landed an interview with a company you've been dreaming about working at. You sit down and, after some small chat, the interviewer says "tell me a little bit about yourself." This common question results in the first impression at your interview, but is often the most overlooked. In my experience from being on both sides of the interview, I believe that by crafting a unique and memorable answer to this one question, you can land that job. Here's how. Don't recount your resume A big mistake I see people make is using this question to go over their résumé. Don't do this! Your interviewer has already looked over your résumé and probably has it sitting in front of them during the interview. They don't need you to go over it again. Practice over and over again and prepare answers to any obvious questions your interviewer may have You wouldn't show up for a talk you're giving at a conference without practicing over and over again, right? Make sure you don't show up for an interview without doing the same. Sit down a few days before and think of what questions an interviewer may have for you after looking at your résumé. Then prep how you will answer them. For example, I'm a coding bootcamp graduate who used to work in music publicity (read my career transition story here). A potential employer looking at my résumé may wonder why I wanted to transition from publicity to tech, and they'd probably ask about the process I went through. And so, that's one of the main things I address in my answer, because I know that will be the biggest question and could cause the most hesitation when it comes to a potential employer choosing me for the job. Take a look at your own résumé as if you've never looked at it before, and figure out what those questions would be for you —and be sure those are your main points in your answer.

How to avoid (or escape) async/await hell

async/await freed us from callback hell, but people have started abusing it—leading to the birth of async/await hell. In this article, I will try to explain what async/await hell is, and I'll also share some tips to escape it. What is async/await hell While working with Asynchronous JavaScript, people often write multiple statements one after the other and slap an await before a function call. This causes performance issues, as many times one statement doesn't depend on the previous one—but you still have to wait for the previous one to complete.

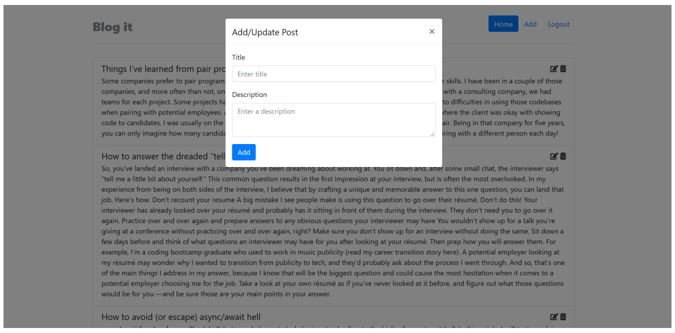
© Manuel Alaminos 2018

Figure 8: **Homepage**

- Angular
- TypeScript

Not only Angular was new to me, but also JavaScript frameworks/libraries as a whole, and the paradigm of single page applications, which I absolutely loved it.

On the other hand, and despite my initial boost in motivation and study plan, I poorly managed my time and hardly developed what first came to my mind of what I wanted to achieve with this exciting project.



Add/Update Post

Title

Enter title

Description

Enter a description

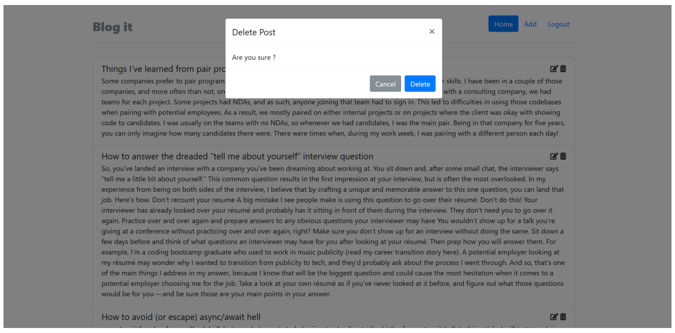
Add

Figure 9: **Homepage**Adding a post

As mentioned in the introduction, my study plan firstly involved Angular. That was mainly my focus the first week and a half. I learned all kind of Angular features, starting from TypeScript, then Angular fundamentals, displaying data and handling events, the use of components, directives, template-driven and reactive forms, consuming http services, routing and navigation, the use of pipes and custom pipes. Angular was very exciting actually!

Before jumping to Maximilian course of the MEAN stack to have a picture of how it all work together, I studied some Node, Express and RESTful APIs design and CRUD operations. Which I think was the right thing to do. And I also found this very entertaining and exciting.

Then, I proceed to do Maximilian Udemy course in which I developed a web application with features I could perfectly use for the coursework project. It had messages instead of posts with title and body of the post, but for the most part it was very similar. It had authentication and authorisation, sign up and sign in components, CRUD operations through a RESTful API design, very good error handling, used of MongoDB and mongoose, and more. And I did it.



Are you sure ?

Things I've learned from pair pro

Some companies prefer to pair program … r skills. I have been in a couple of those companies, and more often than not, on … with a consulting company, we had teams for each project. Some projects had NDAs, and as such, anyone joining that team had to sign in. This led to difficulties in using those codebases when pairing with potential employees. As a result, we mostly paired on either internal projects or on projects where the client was okay with showing code to candidates. I was usually on the teams with no NDAs, so whenever we had candidates, I was the main pair. Being in that company for five years, you can only imagine how many candidates there were. There were times when, during my work week, I was pairing with a different person each day!

How to answer the dreaded "tell me about yourself" interview question

So, you've landed an interview with a company you've been dreaming about working at. You sit down and, after some small chat, the interviewer says "tell me a little bit about yourself." This common question results in the first impression at your interview, but is often the most overlooked. In my experience from being on both sides of the interview, I believe that by crafting a unique and memorable answer to this one question, you can land that job. Here's how. Don't recount your resume A big mistake I see people make is using this question to go over their résumé. Don't do this! Your interviewer has already looked over your résumé and probably has it sitting in front of them during the interview. They don't need you to go over it again. Practice over and over again and prepare answers to any obvious questions your interviewer may have You wouldn't show up for a talk you're giving at a conference without practicing over and over again, right? Make sure you don't show up for an interview without doing the same. Sit down a few days before and think of what questions an interviewer may have for you after looking at your résumé. Then prep how you will answer them. For example, I'm a coding bootcamp graduate who used to work in music publicity (read my career transition story here). A potential employer looking at my résumé may wonder why I wanted to transition from publicity to tech, and they'd probably ask about the process I went through. And so, that's one of the main things I address in my answer, because I know that will be the biggest question and could cause the most hesitation when it comes to a potential employer choosing me for the job. Take a look at your own résumé as if you've never looked at it before, and figure out what those questions would be for you —and be sure those are your main points in your answer.

How to avoid (or escape) async/await hell

Figure 10: **Homepage**Deleting a post

After this, I was ready to start my blog and implement most of the things I learned the previous two weeks. I did not want to use Max course and modify it because I wanted to reinforce my learning but now I think it might have been a mistake, a week left was not enough for me to develop everything I wanted.

Everything looks easy when you are following a tutorial and doing small mini-challenges, but when you are on your own in front of an empty project, things change.

A simple sign up features took me a whole day to develop it, to finally end deprecating it because it was throwing me an error and crashing the server, despite working and creating a new user in the database.

I realised that I had basic knowledge of how everything works but not a deep understanding of how everything truly works. My lack of debugging skills in an Angular application as well as in the back-end, really frustrated me, as I really like debugging when it comes to static web sites. Many times, I was not only feeling that I did not know how to fix something -which is fine- but feeling that I could not know how to fix it -which was not fine-.

Angular itself is known for being one of the most difficult JavaScript frameworks/libraries currently. And it really took me a couple of weeks to truly understand how the services communicate with the server and, despite being in the front-end, it does not deal with the user actions directly, that is the job of the component.ts.

So, the way I see it:

- Component.ts deals with user actions. It does not need to know anything that happened after that
- Service.ts validate the information and interact with the API on the server

At the end, this separation of concerns clicked, however, it was not the only thing that took me a while to understand. Many of Angular features like data binding and two-way data binding were not an easy thing to get my head around, and this is just one example of many.

All in all, I feel very happy for everything I have learned and all the challenges I faced and overcome. But I am very disappointed at myself for not managing my time efficiently and smartly and not being able to properly finish the coursework, rushing at the end, and leaving a blog that leaves a lot to desire, functional and visual wise, even when keeping in mind many of these technologies were unknown to me at the beginning of the coursework.

I would say that I did an okay job.