MEAN Stack in Ionic

Kieran O'Halloran

B.Sc.(Hons) in Software Development

April 10, 2019

Final Year Project

Advised by: Dr John Healy

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract This project was developed as my final year project in level 8, B.Sc. (Hons.) in Software Development in GMIT. The purpose of this project was to develop a cross-platform mobile application Using Ionic and the MEAN stack of technologies (MongoDB, Express.js, Angualr.js and Node.js). The application is a Job Advertising app which allows users to create an add for a job they want to get done and the user can also view all other adds created by other users. The users can also message each other through the application.The app consists of a 3-tier architecture, a front-end, back-end and middle tier. The front end consists of our ionic application, the middle tier consists of the node server which contains our API and builds our express app. The backend is our MongoDB database for storing jobs. Ionic handles all of the User Interface and allows the app to be cross platform. The node server acts as a bridge between the database and the app, it's hosted online using Heroku so the mobile application can access the API as Mobile devices cannot run the node server we have created. Our database is connected to a mlab deployment which is also connected to our Heroku deployment. This all brings the app together with our frontend connected to our hosted node middle tier so it can access the data from the database by consuming the API. Our API is a RESTful API which sends JSON data back and forth between the database and the server and also back and forth between the Mobile application and the server.

Authors This project was developed by Kieran O'Halloran in order to achieve a level 8 B.Sc. (Hons.) in Software Development.

Chapter 1

Introduction

Chapter 2

Context

- Provide a context for your project.
- Set out the objectives of the project
- Briefly list each chapter / section and provide a 1-2 line description of what each section contains.
- List the resource URL (GitHub address) for the project and provide a brief list of the main elements at the URL.

2.1 Filler

2.1.1 More filler

2.2 Filler

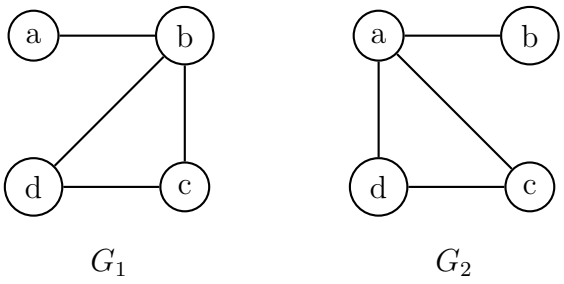
Chapter 3

Methodology

About one to two pages. Describe the way you went about your project:

- Agile / incremental and iterative approach to development. Planning, meetings.
- What about validation and testing? Junit or some other framework.
- If team based, did you use GitHub during the development process.
- Selection criteria for algorithms, languages, platforms and technologies.

Check out the nice graphs in Figure 3.2, and the nice diagram in Figure **??**.



G_1 G_2

Figure 3.1: Nice pictures

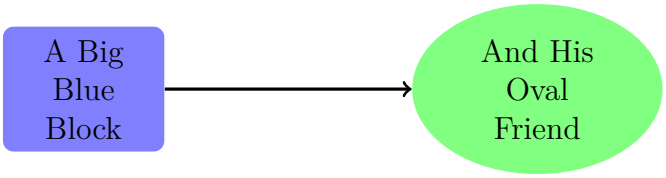


Figure 3.2: Nice pictures

Chapter 4

Technology Review

In this section, I am going to talk about the different technologies I incorporated into this project. I have used a variety of new technologies which I felt would work well in my application. In this section, I will talk about the technologies I choose and my reasons for choosing these technologies. For the backend development, I used the MEAN stack, for front-end development I chose Ionic Framework for cross platform app development so I could build the app on Android, IOS and Windows Phone. I then used Node.js for the server side aspect of the application, the node js server was then hosted on Heroku. Firstly, I am going to talk about the MEAN stack, then about Ionic and Heroku and then finishing off with GitHub.

4.1 MEAN-Stack

What is the MEAN stack? A straight to the point explanation would be that the MEAN-Stack is a free and open-source JavaScript software stack for building dynamic web sites and web applications. Mean Stack is a combination of four popular and highly efficient Javascript libraries, namely MongoDB, Express.js, Angular JS and Node.js. (Shortened down to M for Mongo, E for Express, A for Angular and N for Node).

Mongo DB can be used to store documents in the JSON format, these JSON queries are then handled by Express JS and Node.js on the server side. Angular JS on the frontend is then fed these JSON documents. With the same language on both the client side and the server side. These two technologies work extremely well together and integration between these two environments is seamless and very subtle.[1]

Because all components of MEAN stack support programs written in JavaScript, MEAN applications can be written in one language for both

server-side and client-side execution environments. In order to completely understand the reasons why MEAN stack is so widely used, I will talk about each of its components in detail below.

4.1.1 MongoDB

The MEAN stack comes with a NoSql database technology called MongoDB. It is important that we understand what a NoSQL database is before going into further detail on MongoDB. NoSQL stands for (Not only Sequential Query Language). It is a database that provides a mechanism for storage and retrieval of data which is modelled by means other than the tabular relations used in relational databases.

NoSQL databases have been proven to be the solution to what is known as Big Data as they follow a schema-less data model. A NoSQL database provides increased scalability and flexibility compared to relational databases. Studies show that in recent years developers and organisation have experienced a sharp rise in the volume of user data and products that have to be stored in databases.

NoSQL databases started gaining popularity in the 2000's when companies began investing and researching more into distributed databases [2]. NoSQL databases are widely used to store and retrieve very large amounts of data using a key-value format. These types of databases have emerged as the best choices that suite modern mobile and web development. So now that its clear what NoSql is we are going to talk about MongoDB in more detail.

MongoDB is a schema-free document database written in C++ and developed in an opensource project by the company 10gen Inc [3]. The name mongo is extracted from the word humongous. According to its developers, the main goal of MongoDB is to close the gap between the fast and highly scalable key-value-stores and feature-rich traditional RDBMSs. It provides high availability , high performance, and automatic scaling and allows data insertion without a predefined schema.

MongoDB is one of several database types to arise in the mid-2000s under the NoSQL banner. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents. Documents comprise sets of key-value pairs and are the basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables. A record in MongoDB is composed of field and value pairs and are similar to JSON objects. The value of field may consists of arrays, and array of documents or other documents.

MongoDB maintains data consistency in the sense that one write operation to the data in the database allow subsequent read operations. They use a locking mechanism that contributes to increased execution time as the number of update operation increases. [8] [10].

MongoDB supports dynamic schema design, allowing the document a collection to have different fields and structures. The database uses a document storage and data interchange format called BSON, which provides a binary representation of JSON-like documents. MongoDB also uses Automatic sharding wich enables data in a collection to be distributed across multiple systems for horizontal scalability as data volumes increase.[4].

4.1.2 How Mongo Works In My Application

In my application, I used mongo to store information from my jobs section. I wanted people to be able to add a job and post so other users could then also see that job. An example of how I set up my endpoints in the node server is shown here:

```
app.post('/api/jobs', function(req, res)
```

Next, I created a Job provider which was a Typescript file that contained three methods getJobs, createJob, and deleteJobs. The getJob function sends a get request to my Heroku server that will then return my job data. The createJob function accepts a job object as a parameter and then posts that to the same endpoint. While my deleteJob function will make a request to the API to delete it. A small extract of my getJobs function is shown below:

```
getJobs(){

if (this.data) {
return Promise.resolve(this.data);
}

return new Promise(resolve => {

this.http.get('https://kierantradie.herokuapp.com/api/jobs')
.map(res => res.json())
.subscribe(data => {
this.data = data;
resolve(this.data);
});
```

```
});
}
```

I used mongoose to connect my application to mlabs. Mongoose is a MongoDB Object Document Mapper (ODM) for Node. It provides the user with a simple validation and query API to help you interact with the MongoDB database.

```
mongoose.connect('mongodb://heroku_lz2bt2wt:kabs1g4ubvtolmbjkvold9inkd@ds227185.
    if (error) console.error(error);
    else console.log('Mongo Connected');
});
```

4.1.3 Express

Express is another component of MEAN Stack. Express is a nodejs asynchronous based web framework. Express.js builds on the underlying capability of Node, by providing a web application server framework. Express.js is a Node.js web application server framework, designed for building single-page, multi-page and cross platform hybrid web applications and it gives Node.js a more realistic website structure that is not present when using Node by itself. For my application express will allow me to create routes for the REST API I will be creating.[5]

4.1.4 Node.js

Node.js is a Javascript runtime built on Chrome's V8 JavaScript Engine. Node.js uses an event driven, non-blocking I/O model that makes it lightweight and efficient web server environment, ideal for constructing a web-service API's. Node.js package ecosystem, "npm", is the largest ecosystem of open source libraries in the world. For my application node will be my server which will sit between the frontend of the application and the MongoDB database.[6]

4.1.5 AngularJs

The final component making up the mean stack is Angularjs. Defined in Angulars official documentation[7] - AngularJS is a structural framework for dynamic web apps. Angular allows the user to use HTML as their front-end language and lets you extend HTML's syntax to express an application's component clearly. Angular also has a data binding and dependency injection which eliminates much of the code you currently have to write. This

all happens within the browser, making it a perfect partner for any server technology.

AngularJS simplifies application development by presenting a higher level of abstraction to the developer. Like most types of abstraction, it comes at a cost of flexibility. In other words, not every app is a good fit for AngularJS. AngularJS was built with the CRUD application in mind and this was another reason why I thought this technology would be a good fit for my application as I wanted to be able to add and delete data to a database. [8]

There are a number of reasons why angular js is so popular and some of which include how Angularjs structures the source code by following the Model View Controller. The second reason is Angularjs ability to do two-way data binding. It decreases the amount of code that is written to keep the model and view in agreement. Angularjs models are old java object (POJO), therefore it is quite simple to change or append properties without any major complications. Finally, and probably the most important feature that Angular js has to offer is dependency injection. Dependency injection is a software design pattern that deals with how components get hold of their dependencies. The angular injector subsystem is in charge of creating components, resolving their dependencies and providing them to other components as requested [9].

After speaking in some detail about each component of the mean stack you should have a clear understanding of this technology and how it works and benefits applications.

4.2 RESTful API

A RESTful API is an application program interface (API) that uses the HTTP requests GET, POST, PUT, and DELETE data. A RESTful API, which is also referred to as a RESTful web service is based on representational state transfer (REST) technology[10].

A RESTful web service is based on representational state transfer (REST) technology. It is an API that communicates with HTTP requests to GET, PUT, POST and DELETE data and then links to the four fundamental database operations - CREATE, READ, UPDATE, DELETE. An API for an application is basically code that allows two software programs to communicate with each another. In my case the api is created in my node server file and this then links to my database where I am able to apply the HTTP requests.

4.2.1 How RESTful API's work

The API basically takes different parts of a transaction to make a number of small modules. These modules then target a specific underlying part of the transaction. As a result, developers are provided with a lot of flexibility.

The RESTful API uses GET for read and idempotent requests to retrieve a resource, POST for write requests which create a resource, PUT to change the state of or update a resource and DELETE to remove it. All calls are presumed to be stateless which means nothing can be retained by the RESTful service between executions. As a result, REST is suited to cloud applications because stateless components can be freely redeployed if something fails. [11]

The reason for this is that requests can be directed to an instance of a component and therefore there is nothing kept that needs to be remembered by the next transaction.

For these reasons REST is largely preferred for web/mobile use. The RESTful model can also be extremely helpful in the cloud as using APIs to bind services is as simple as controlling how the URL is decoded.

4.3 Ionic

Ionic is a complete open-source SDK for hybrid mobile app development. It is built on Angular. Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by using Cordova.

Services and features: Ionic provides all the functionality which can be found in native mobile development SDKs. Users can build their apps, customize them for Android or iOS, and deploy through Cordova. Ionic includes mobile components, typography, interactive paradigms, and an extensible base theme.
Besides the SDK, Ionic also provides services that developers can use to enable features, such as push notifications, A/B testing, analytics, code deploys, and automated builds.
Ionic also provides a powerful command-line interface (CLI), so developers can get started with creating a project with a simple command. The CLI also allows developers to add Cordova plugins and additional front-end packages [**?**], enable push notifications, generate app Icons and Splash screens, and build native binaries.

Supported platforms: Ionic is focused on building for modern Web standards and for modern mobile devices. For Android, Ionic supports Android 4.1 and up. For iOS, Ionic supports iOS 7 and up. Ionic 2 supports the Universal Windows Platform for building Windows 10 apps. Ionic Framework, powered by Angular.js, supports BlackBerry 10 apps.While trying to understand Ionic better I followed along with some tutorials I found on this blog post [12].

Installation: Ionic is an npm module and requires Node.js.

- Install Ionic Code: npm install -g ionic
  First, install Node.js. Then, install the latest Ionic command-line tools in your terminal. Follow the Android and iOS platform guides to install required tools for development.

- Start an App: ionic start myApp tabs

- Run your App: cd myApp
  ionic serve
  Most applications can be built in browsers using ion services. When you are ready to deploy the application to a real device, you can review the deployment guide.

4.3.1 Typescript

As previously mentioned one of the main difference between Ionic 1 and 2 is the fact that Ionic 2 uses Typescript instead of javascript. Microsoft created TypeScript with its first public release in October 2012 but Typescript has only become more popular in web development since angular and ionic added it to their 2.0 frameworks. So to get a better grasp of Typescript we are going to talk about it here in more detail. Basically, TypeScript is a superset of JavaScript that compiles into Java, which means it behaves identical to JavaScript but with some extra features added in. So you don't run TS on your web server, ultimately it's all JavaScript. TypeScript also allows developers access to powerful tools for writing modern JavaScript.[13] [14]

4.4 Firebase

For my login page I needed to handle user Authentication. To do this I decided to use Firebase Authentication.

Firebase Authentication provides backend services, SDKs, and ready to use UI libraries to authenticate users of your application. It supports authentication using passwords, popular federated identity providers like Google, Facebook and Twitter.[15]

Firebase Authentication integrates with other Firebase services, and it competes with industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with a custom backend.

A user is able to sign in to a Firebase app by either using FirebaseUI as a complete drop-in auth solution or by using the Firebase Authentication SDK to manually integrate one more of the sign-in methods into the application. For my application I used the firebase SDK. By doing this we were able to choose which sign in methods we wanted to add to our application. I decided to use just email and password authentication. Firebase Authentication also handles sending password reset emails. The firebase SDK also comes with a reset password feature which sends the user an email with a link to reset there password. This is a good security measure as the user will need to have access to their personal email account before they can reset their password.[16]

4.5 Heroku

In my application, I needed somewhere to deploy it to. After a lot of research and experimentation, I decided to use Heroku[17]. Heroku hosts my Node server with my RESTful API. My Heroku deployment is then connected to mLab which is a fully managed cloud database service that hosts MongoDB databases and connects them to services like Heroku [18]. This then allowed me to run the application with out any intervention.

This was the most difficult part of the project to get working as I had a lot of trouble getting my database to properly connect with Heroku but after extensive research and a lot of testing I figured out that I needed to point the node.js server to the build of the application and then add the URL of our Heroku instance in replace of the localhost URL.

4.6 GitHub

GitHub is a web-based collaboration platform for software developers, delivered through a software-as-a-service (SaaS) business model which allows you to host and review code along with managing projects. Github first launched in 2008 and was founded on Git which is an open source code management system created by Linus Torvalds to make software builds faster.[19]

Git works by storing source code from projects and tracking all changes made to that code to a repository. Repositories can be made either public or private so developers can share their code. It is a great tool for developers when collaborating projects as it provides users tools for managing changes from different developers. I found this to be greatly beneficial in my development.

GitHub works by using git commands to push projects up to its website [20]. The process behind this is straight forward. First the user needs to use git bash to navigate to the folder they want to upload. This folder then needs to be initialized by using the command:

```
$ git init
```

This command creates an empty Git repository which has a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. A HEAD file is also created and this file references the HEAD of the master branch. The next command you have to use is:

```
$ git add .
```

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit.

Once the files are added you need to run:

```
$ git commit -m""
```

This stores the current contents of the index in a new commit along with a log message from the user describing the changes. And then finally run:

```
$ git push origin master
```

to push all the files to the master branch of your repository.

Other features that GitHub has to offer are its ability to fork, pull and merge from someone's repository. A fork is essentially a copy of a repository that allows developers to make modifications without affecting the original code. If the developer would like to share the modifications, they can send a pull request to the owner of the repository. The owner can then decide after reviewing the modifications if they would like to pull the modifications into the repository. They then have the option to accept the modifications and merge them with the original repository.[21]

Another feature of Github that I found to be very helpful is the commit history. Everytime you make a change to your code and push it up to Github you are creating a commit. A commit is Githubs way of tracking the changes

you have made to the project since your last commit. I found this useful because I could easily see what was pushed up and exactly what was added or removed. You are also able to download the project from a commit at any point, so for example you added features to the project that you no longer want you can revert back to any point in your commit history. I used this feature on a few occasions as I often encountered problems after doing some work on the project which resulted in issues and errors and It saved me a lot of time as a result.

Chapter 5

System Design

As many pages as needed.

- Architecture, UML etc. An overview of the different components of the system. Diagrams etc... Screen shots etc.

Column 1	Column 2
Rows 2.1	Row 2.2

Table 5.1: A table.

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportuni-ties in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and ob-jectives (a few lines).
- Highlight your findings from the evalua-tion section / chapter and any opportuni-ties identified.

Bibliography

[1] M. Stack, "https://www.sitepoint.com/introduction-mean-stack/."

[2] NoSql, "https://en.wikipedia.org/wiki/nosql."

[3] . Gen, "https://www.mongodb.com/press/10gen-announces-company-name-change-mongodb-inc."

[4] MongoDB, "http://searchdatamanagement.techtarget.com/definition/mongodb."

[5] Express, "https://expressjs.com/."

[6] NodeJS, "https://nodejs.org/en/."

[7] Angularjs, "https://docs.angularjs.org/guide/di."

[8] W. is AngularJS, "https://docs.angularjs.org/guide/introduction."

[9] A. features, "https://code.tutsplus.com/tutorials/5-awesome-angularjs-features–net-25651."

[10] R. Api, "https://en.wikipedia.org/wiki/representational$_s tate_t ransfer$."

[11] RESTful-API, "http://searchcloudstorage.techtarget.com/definition/restful-api."

[12] J. Morony, "https://www.joshmorony.com/."

[13] Typescript, "http://whatpixel.com/is-typescript-worth-learning/."

[14] Microsoft and google collaborate on typescript, "https://techcrunch.com/2015/03/05/microsoft-and-google-collaborate-on-typescript-hell-has-not-frozen-over-yet/."

[15] F. Authentication, "https://firebase.google.com/docs/auth/."

[16] Firebase, "https://firebase.google.com/."

[17] Heroku, "https://www.heroku.com/about."

[18] mlab, "https://mlab.com/."

[19] GitHub, "https://github.com/."

[20] G. commands, "https://education.github.com/git-cheat-sheet-education.pdf."

[21] G. Operations, "http://searchitoperations.techtarget.com/definition/github."