|  |
| --- |
| Full Stack Web Development Specialization Capstone Project |
| Application Design and Software Structure |
| Course Manager |

|  |
| --- |
| Francisco Villar  4/30/2017 |

1. Introduction

While developing an app, you go though many different tutorial videos in order to get the skills you need. That’s why I want to create Courses Manager, an app that tracks all the different courses/videos that a user sees to avoid watching a video multiple times if I don’t want to.

In Course Manager the user will be able to login and register to the website. Also, the user will be able to View, Add, Edit and Delete courses from their profile. Finally, and more important, the user will be able to access the about us page.

2. Design and Implementation

2.1 The REST API Specification

The following are the endpoints of my REST API for each of the routes:

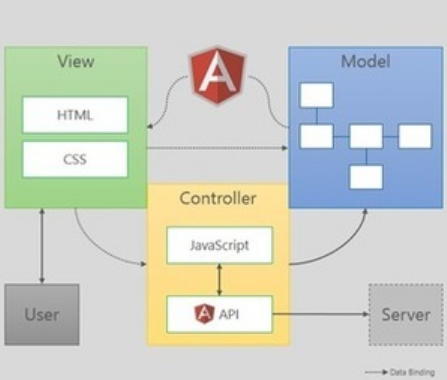
* **/users**
  + get('/'): Get all users in the system.
  + get('/login/:username'): Get the user object, including username and password, so it can process the login process.
  + post('/'): When registering a new user to the system, a user object is saved in the database.
* **/courses**
  + get('/'): Get all the courses associated with that user.
  + get('/:id'): Get an specific course using the course id.
  + post('/'): When adding a new course, a course object needs to be stored in the database.
  + put('/:id'): Update a course given an Id and an course object.
  + delete('/:id'): Delete a course given an Id.
* **/authors**
  + get('/'): Retrieve all the authors existing on the database.
  + get('/:id'): Get an specific author given an author Id.
  + post('/'): Add a new author by providing an author object.

2.2 Front-end Architecture Design

In both of the projects, web and mobile projects, the architectural pattern used is MVC using AngularJS, which according to Matt Milner in “AngularJS: MVC implementation” is

Implemented in JavaScript and HTML. The view is defined in HTML, while the model and controller are implemented in JavaScript. There are several ways that these components can be put together in AngularJS but the simplest form starts with the view.

The difference between the web application and the mobile application is that as it is going to be seen on the file structures, the mobile application will have some wrappers and plugins needed so the code is transformed to the mobile device using Cordova and ionic framework. This can be represented for both projects as follows based on Yad Faeq diagram:



First, I proceed to explain the file structure of my web project. In here, the main folder will be public which contains the JavaScript files in the javascripts sub-folder and the different HTMLs in the views sub-folder.

/courseManager

/bin

/node\_modules

/public

/images

/javascripts

app.js

controllers.js

services.js

/stylesheets

bootstrap.min.css

site.css

/views

aboutUs.html

course.html

footer.html

header.html

home.html

login.html

newCourse.html

register.html

index.html

/routes

authors.js

courses.js

index.js

users.js

app.js

package.json

Second, the mobile files structure, which is similar to the web project, but in this case the www folder will contain the HTML file in the templates sub-folder and the JavaScript files in the js sub-folder.

/courseManagerIonic

/hooks

/node\_modules

/plugins

/routes

authors.js

courses.js

index.js

users.js

/www

/img

/js

app.js

controllers.js

services.js

/css

site.css

/lib

/templates

aboutUs.html

course.html

home.html

login.html

menu.html

newCourse.html

register.html

index.html

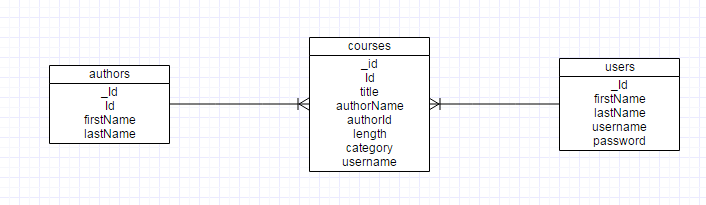
app.js

package.json

2.3 Database Schemas, Design and Structure

The database in use will be a NoSQL DB, more specifically MongoDB. The database will be divided in three collections: authors, courses, and users. Each of these collections will have the following document structure.

* authors:
  + \_id: MongoDB auto-generated Id.
  + Id: Author’s system Id.
  + firstName: Author’s first name.
  + lastname: Author’s last name.
* courses:
  + \_id: MongoDB auto-generated Id.
  + Id: Courses’ system Id.
  + title: Tutorial/video title.
  + authorName: The name of the author associated with the course.
  + authorId: The Id of the author associated with the course.
  + length: How long is the video/tutorial.
  + category: User can categorize video/tutorial by a subject, i.e. angular, mongodb, etc.
  + username: The username associated with the course or how actually logged the course.
* users:
  + \_id: MongoDB auto-generated Id.
  + firstName: User’s first name.
  + lastName: User’s last name.
  + username: User’s user name used to login to the app.
  + password: User’s password used to login to the app.



2.4 Communication

Both web and mobile projects uses the factory in their services.js. In here, it is used the ngResource to communicate with the back-end. In more details, here is how it is going to look:

* Login:
  + Request URL:http://localhost:portNo/users/login/username
  + Request Method: GET
  + Request Parameter: username, which is the username of the user object that we want to retrieve.
* Registration
  + Request URL:http://localhost: portNo /users
  + Request Method: POST
  + Request Payload: {"firstName":"first","lastName":"last","username":"qa","password":"123"}.
* Get User Courses
  + http://localhost: portNo /courses?username=username
  + Request Method: GET
  + Request Query: username, which is the logged in user
* Get Specific Course
  + Request URL:http://localhost: portNo /courses/courseId
  + Request Method: GET
  + Request Parameter: courseId, which is the ID of the course that we want to retrieve
* Get Authors
  + Request URL:http://localhost: portNo /authors
  + Request Method: GET
* Update a course
  + Request URL:http://localhost: portNo /courses/courseId
  + Request Method: PUT
  + Request Parameter: courseId, which is the ID of the course that we want to update
  + Request Payload: {"\_id":"5859309f3188b5cbab814dde","Id":2216,"title":"Computer System Engineering","authorName":"John Smith","authorId":2,"length":"4:21",  
    "category":"Computer Science","username":"fv"}.
* Add a new course
  + Request URL:http://localhost: portNo /courses
  + Request Method: POST
  + Request Payload: {"title":"course title","authorId":1,"length":"10:34","category":"Testing","username":"fv"}.
* Delete a course
  + Request URL:http://localhost: portNo /courses/ courseId
  + Request Method: DELETE
  + Request Parameter: courseId, which is the ID of the course that we want to delete

3. Conclusions

In this project I want to put in practice all the knowledge acquired on the specialization, that is why I want to do a simple proof of concept that could help me assemble all the different pieces in one. Moreover, I would like to create a project that can be a CRUD boilerplate for other projects I have in mind. More specifically this project will help me create from scratch all the different layers of the program.

In this document, we have gone through out all the different layers of the design and architecture of the Course Manager. First, on how the REST API end points are going to look like. Second, how the projects’ file structure are organized, for both web and mobile projects, and know which design pattern we are going to use in the project, which is MVC. Third, we designed the database in a NoSQL environment and give more structure to the documents and collections. Finally, we discussed on how each API call is going to be and how the message will look like in terms of the request URL, Method and Payload.

4. References

Wireframe diagrams done with <https://www.gliffy.com>

AngularJS: MVC implementation (Matt Milner): <https://www.pluralsight.com/blog/software-development/tutorial-angularjs-mvc-implementation>

Which should I choose for a new web application, AngularJS or Flux React, and why? (Yad Faeq): <https://www.quora.com/Which-should-I-choose-for-a-new-web-application-AngularJS-or-Flux-React-and-why>