SE31520 Assignment 2

Owen Garland - owg1@aber.ac.uk - Version 0.3

Introduction

This report details my progress with the Developing Internet-Based Applications CSA assignment. I will justify all of my design decisions as well as analysing how those decisions effected the project. You can view the project, Heroku & TravisCI integration at https://github.com/bag-man/SE31520-Assignment-2.

Architecture

The design and architecture of this project really wasn't up to me, the project is a standard rails project which means it is structured exactly as rails projects are always structured.

When the project was initially given to me, the API endpoints were not abstracted away from the human web end points. This really isn't an issue, as it makes the development of an API quite intuitive for someone who is already familiar with the website. Simply add .json to the end of the resource that you want to get a .json feed.

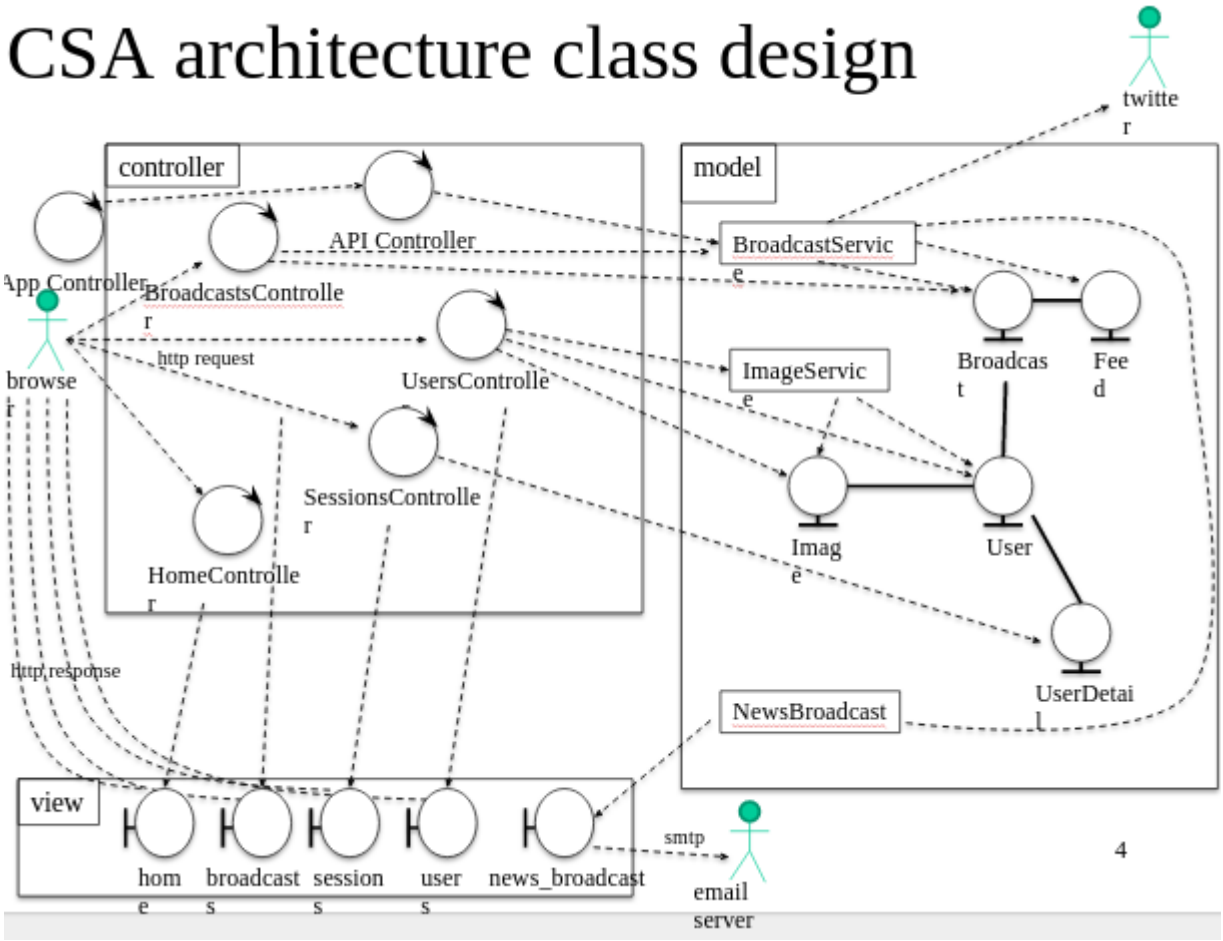
However a requirement for this project was to abstract the API endpoint out to a separate URI at /api/, this was done by creating a new set of controllers in an API sub-directory, and removing all of the API end points from the original controllers and moving them over to the new ones.

After that I moved the jbuilder views from the original views folder to the api folder so that the json could be built. I also had to add a new namespace to the routes file that directed requests to /api/ to the correct locations.

This does have the benefit of abstracting the human web from the computer friendly part of the web, and minimises the risk of non technical users accidentally accessing a JSON endpoint and thinking that the site is malfunctioning.

UML

I have lightly modified the UML diagram provided in the lecture slides, I've show the API controller interacting with the Broadcast model and Application Controller.



Implementing ActionCable to this project was fairly straight forward, I simply had to read the documentation for how ActionCable should be implemented in a Rails project and add the necessary files and functions to my project.

I found the implementation of ActionCable to be fairly poorly documented online however and it did take a while to find a resource that did provide enough of the information required. It was also very difficult when compared to SocketIO which has a lot simpler implementation. I created some coffeescript that would run on the client and a div for the output to be displayed. I then added a broadcast channel, and an ActionCable connection controller to the server side of the application. To get it to all link up I had to create a broadcast relay job and update the broadcast model to run the job when a new broadcast was made.

Once I had the ActionCable successfully added to my project I ran into an issue where it wouldn't run correctly on the 'thin' webserver that the project had been setup to use. After a few hours of Googling I was able to determine that this was the issue and that switching to the 'puma' webserver would solve these issues.

RESTful client

This client was very simple, all it had to do was perform one GET and one POST request. I quickly developed a short python script that could perform these functions, then added in a minimal CLI using the docopt library.

I chose to use python for this script as it is a language that I am more familiar with than Ruby, and for scripts this size python seems like a very popular choice as it is included in pretty much all major distributions and OS X.

Cucumber tests

Feature driven tests were something rather new to me. I am very experienced in writing unit tests for web sites, and found the concept of a feature test quite counter intuitive. Unit tests are written to be atomic so that they can rule out all variables that might effect the output of the test.

Feature tests are the opposite of this as they test a whole feature rather than a single function, I can see this being used as a smoke test before deployment to ensure that key features are working, this combined with passing unit tests would give you a very good indicator that the project was doing everything it should.

However I do not see cucumber tests as a valid substitute for unit tests at all. There were some unit tests given to us for the user class, however they didn't pass when they were given to us, nor were we instructed to fix them.

Screencast

Please see demo.mp4 in this folder for the screencast that shows Cucumber tests passing, Heroku, TravisCI, and the use of the CLI client and ActionCable.

Flair

To add some flair to this project I have used version control to develop the application, this has allowed me to branch code off and look at different revisions for regressions. At one stage I had an issue caused by my naming a function `update`, which I was unaware was a reserved function in rails, this caused a whole host of errors. To fix this I did a git bisect across the project which essentially allowed me to do a binary search through all of the commits, until I reached the one that had introduced the regression. Ideally I would have had unit tests that I could have added to the git bisect, but these weren't included in the project. However manually chekcing for the regression didn't take long and I had quickly narrowed down the area of code to look at for the bug.

I also linked my Github repository to Heroku, and added TravisCI to the project as well. This means that when I commit code, TravisCI will run my cucumber tests, and if they pass Heroku will deploy the code so the latest stable version of the application can be seen online. I then added a couple of badges to the README.md of the project so that you can easily get to the TravisCI tests. I also added one to code climate which evaluates the state of the code in the repository.

Build Status Code Climate

You can view the site on Heroku here, the login details are `admin:admin`.

For my CLI application I used docopt which allowed me to first write the help message for the

application, and then docopt would parse this message to generate the code that would handle these arguments. This means I have a very nice standardised CLI without having to write any argument parsing logic.

Evaluation

I have been interested in developing web applications for many years and have spent my industrial year making NodeJS applications, so I felt that this module would be a good place for me to show my knowledge and expertise in this area. Unfortunately I didn't really get a long with Ruby, nor Rails. Both presented challenges to me, with Ruby itself I had issues understanding the syntax, it bothers me that some parts of the syntax are optional and other parts just seem like they were intended to confuse.

For example, when calling a function you can use the following equally.

```
function param1, param2

function(param1, param2)
```

An easy fix for this would be to include a linter configuration in the project that enforces one set style of coding Ruby, this would have the added benefit of highlighting any syntax errors in the editor so that you could see them before having the application error on you.

I have used frameworks that allow for scaffolding before and haven't been a fan. I like to know what I am creating and how it all links together. With some frameworks this isn't too much of a problem but I felt with Rails there was a lot of magic holding it all together, and it was really unclear how some files linked together unless you had intimate knowledge of how rails uses its file structure. This magic is always apparent and makes you feel rather helpless when it comes to debugging. I'm sure that if you learn how Rails works in detail then you will be able to make the most of this magic, but having something not work because a file wasn't named correctly is very demoralising.

The thing that stood out most to me between rails and node was the lack of an equivalent to a "package.json" from a node project. This file will list out every package (and it's version) that is used by the project, as well as the node version the project is designed to use, and meta data about the project, such as license, authors name and version etc. It also includes a list of all of the commands that can be ran on the project, build, test, watch etc, and what each of those commands actually does.

It is rather frustrating as I have demonstrated knowledge of everything required for this assignment numerous times in my own side projects, just using different technologies. My usual stack utilises NodeJS, Express, MongoDB and SocketIO. You can see an example of my stack here, it uses unit tests, continuous integration, linting, minifying, and templating among other things. I struggled to find motivation for this project and I think I won't have done anywhere near as well on

this assignment as I should have. Not out of a lack of understanding of the concepts but just sheer apathy to figure out how implement features in an alien technology, that doesn't appear to offer any real benefits over the ones that I am already familiar with. I don't know what mark to expect for this assignment and I wouldn't like to hazard a guess.

I can see the value in Cucumber tests as a smoke test to make sure that key features work before deployment, but don't see them as a replacement for unit testing. I found writing the Cucumber tests rather difficult as it was hard to not think of them as unit tests. In my tests I created a Before to clear and populate the database for example, and I think that is the unit test way of thinking; perhaps I should have used the rails db:reset in my test script instead.

The biggest issue I have had with this project was to do with the broadcasts and how they related to feeds. In my head I imagined that a broadcast object would store in it, the content of the broadcast and the feeds that it had been selected to be broadcast in. This would be the standard way of doing things if the project was using a NoSQL database like MongoDB. It would also be a fairly sensible way of storing the data in a SQL database as well, you would simply store a boolean for each feed type. This would reduce the complexity of the data structure making it easier to develop with. It doesn't make much sense to me to introduce the added complexity of the many to many relationship when the feeds don't require any extra data associated with them.