Documentation: CA 4 MEAN –> (Mongo- Express-Angular-Node)

21.11.2014

Names:

Boyko Surlev

Nikolaj Desting

Peter Tomascik

Design description of the overall system functionality

1. Back-end

Our program is built on top of NodeJs which is using Express template. By using this template we were forced to use predefined structures, which make the code of our application more robust and easier to navigate in. Our code has three big parts in which second one takes care about back-end stuff. The structure is following: *db.js* connects and works with Mongo Database; *dataLayer.js* represents an interface between our Node and Mongo Database. It contains the functions which communicate with database and pass the JSONs further inside our application into the *REST_api.js*. This part is the last in the back-end. It handles the requests from the client side and by using the methods from interface it takes out whatever data are needed.

2. Front-end

Folder *Public -> App* includes all view files we have. By using AngularJS has each view at least one html and one JavaScript file which are communicating together. This is creating nice single page application with different routes, based on functionality you want to use. JavaScript files communicate further with *REST_api*. After developing the functionality of our project we struggled with implementing BootStrap files and code into it so it has a user-friendly environment.

Description of how we solved part 4

This part was done by Boyko. He finished first half of this part completely. Only problem which is not solved is dealing with headings. Otherwise is this functionality fully accessible.

Description of the back-end tests

As mentioned before we separate our logic to back and front end part. This decision allows us to test our *ends* separately and by that decision we were able to proof logic part of our project. Testing back-end took less time than front-end. One part of our business logic includes interface. First we had to understand the concept so we do not mixed it up with Java sort of interface. Interface we have includes all methods which are middle part between REST_api and our database. We tested all methods (see below) with only positive results.

getWiki(title)

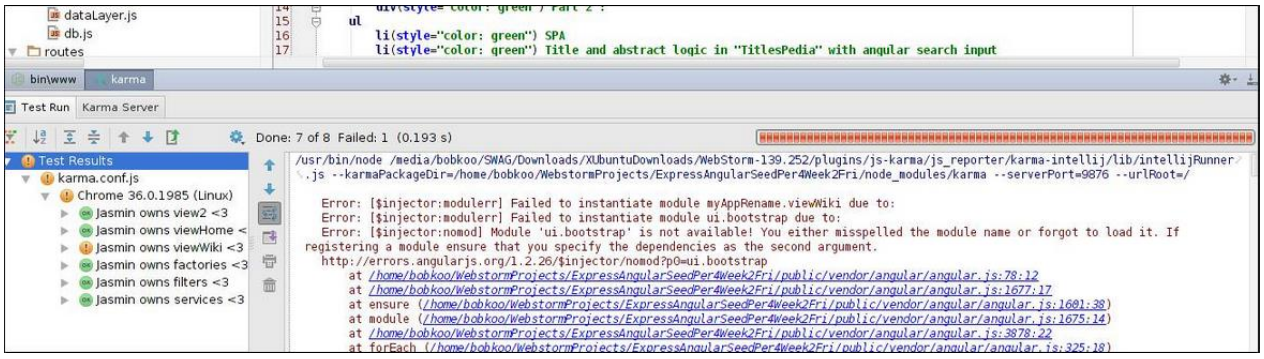
findWiki(searchString)

getCategories()

getWikisWithCategory(category)

Description of the front-end tests

For testing front-end of our application we used Jasmine test framework with the Karma test runner. A good thing is that we did not need to spend any time with configuration of testing environment so we could clearly focus only on testing. It took us while to grab the concept and fully understand how it works but by the end we managed to have 5 working tests. We have to admit that it was big struggle with bootstrap and angular module and as you can see on the picture bellow program printed some unexpected error. As it printed out there are some irregularities with some inner module of the program. However it was a big struggle it teaches a lot and gives a basic overview how to work with this framework.



Who did what?

Who	What
Boyko	Part #I,II,III,IV a),bonuses and tests
Nikolaj	GitHub + Azure + design
Peter	Part#I,II (majority done by Boyko) + design + documentation

Description of what we added

What	Description
bonus #1	finished completely
bonus #2	finished completely
ordered table + search	most of them based on some unique parameter
back button	on most pages
Accordion	rendering tables