Sparehands.ie

Final Project Report

Sam Power

A00226315

A final report submitted in part fulfillment of the degree Ericsson Masters in Applied Software Engineering in the school of Engineering with the supervision of Sean Kennedy.

School of Engineering

Athlone Institute of Technology

December 5th 2015

Table of Contents

Introduction

Scope Of Project

Develop a Web Application for a topic of your choice using Single Web Page Application (SPA) architecture. The application must use adhere to the architecture shown below. The application should:

- Use RESTful web services
- Have CRUD actions for one or more entities
- Have the ability to upload either images or files (depending on your application) from the client to the server.
- Use the following features in the UI
  - Glyphicons
  - Navigation bar using tabs
  - Progress bar
  - Dialog boxes
  - Have a responsive layout for two screen sizes (Grid system)
- Use some form of authentication and login
- The application must be fully tested and a high level of test automation is expected.
- In addition to these features your project will also have an individual aspect:
  - *React is a JavaScript library for creating user interfaces by Facebook and Instagram. React views can be used with Backbone.js. Document how to use React.js and evaluate its usefulness.* *Link to Example*

Aims and Objectives

Sparehands will be a single page web service that will allow registered users to create and apply to small jobs. The owners of advertised jobs will also be able to accept or reject applicant to jobs they have created. I aim to use ReactJS to render some front end elements and document how this was achieved.

Product Backlog

Story ID	Story Description
1	As an unregistered user I want to be able to create an account.
2	As a registered user I want to be able to login to my account.
3	As a registered user I want to be able to edit my account.
4	As a registered user I want to be able to delete my account.
5	As a registered user I want to be able to create a job.
6	As a registered user I want to be able to edit a job I created.
7	As a registered user I want to be able to delete a job I created.
8	As any user I want to be able to view jobs.
9	As any user I want to be able to filter the jobs by any search term.
10	As a registered user I want to be able to apply to a job.
11	As registered user I want to be able to reject/accept applications.
12	As a registered user I want to be able to view all my jobs.
13	As a registered user I want to be able to view all my applications.
14	As a user, each query I make should take less than 2 seconds.

Sprint 0

Sprint Goal

End to end functionality for all user related CRUD with queries taking less than 2s and a base design done for the UI.

Definition of Done

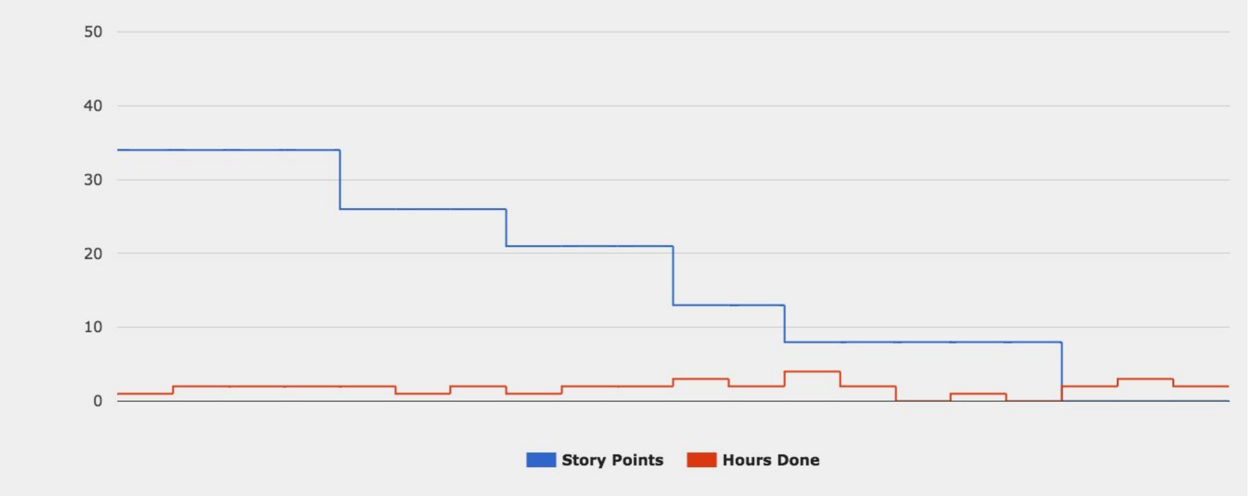
1. Clean code produced.
2. Unit tested (>=70% code coverage).
3. Builds without errors.
4. System test completed.
5. HTML/CSS validated (if applicable).
6. Queries take less than 2 seconds.
7. ReactJS is used throughout.

Sprint Plan

The following user stories were committed to be completed upon finishing Sprint 0.

- **User Story #1** - As an unregistered user I want to be able to create an account.
- **User Story #2** - As a registered user I want to be able to login to my account.
- **User Story #3** - As a registered user I want to be able to edit my account.
- **User Story #4** - As a registered user I want to be able to delete my account.
- **User Story #5** - As a registered user I want to be able to create a job.

Burndown Chart



In the range of October 5th - October 24th 2015

Test Strategy

Unit Test

AUTOMATED

70% code coverage

Tools: Junit version 4.12

Mockito version 1.9.5

Integration Test

MANUAL

One integration test per story

Server side on its own (WS, DAO, DB)

Tools: PAW Version 2.2.3 (App Store)

Arquillian version 1.1.8

System Test

MANUAL & AUTO

One positive and one negative selenium test per story

Tools: Selenium version 2.46.0

Performance Test

MANUAL

Acceptance Test

MANUAL

Multiple per story (Given-When-Then)

User Story Elaboration

Number	**Priority**		**Name**
1	1		As an **UNREGISTERED USER** I want to be able to **CREATE** an **ACCOUNT**.
	Narrative		
	As an	**I want to... (When)**	**So that... (Then)**
	Unregistered User	Create an account	I can use that account to access the CRUD on the site.
	Acceptance criteria		
	Scenario 1 – All User details entered correctly		
	Given	**When**	**Then**
	● Valid Details.	● The user is trying to register.	● The user datatable will be updated.
	Scenario 2 – User email is already registered		
	Given	**When**	**Then**
	● All Details but email registered already.	● The user is trying to register.	● The user datatable is not updated. ● Prompt is shown saying why.
	Scenario 3 – User password does not match criteria		
	Given	**When**	**Then**
	● All Details but password does not match criteria.	● The user is trying to register	● The user datatable is not updated. ● Prompt is shown saying why.
	Scenario 4 – Any empty fields when registering		
	Given	**When**	**Then**
	● Empty fields when registering account.	● The user is trying to register.	● The user datatable is not updated. ● Prompt is shown saying why.

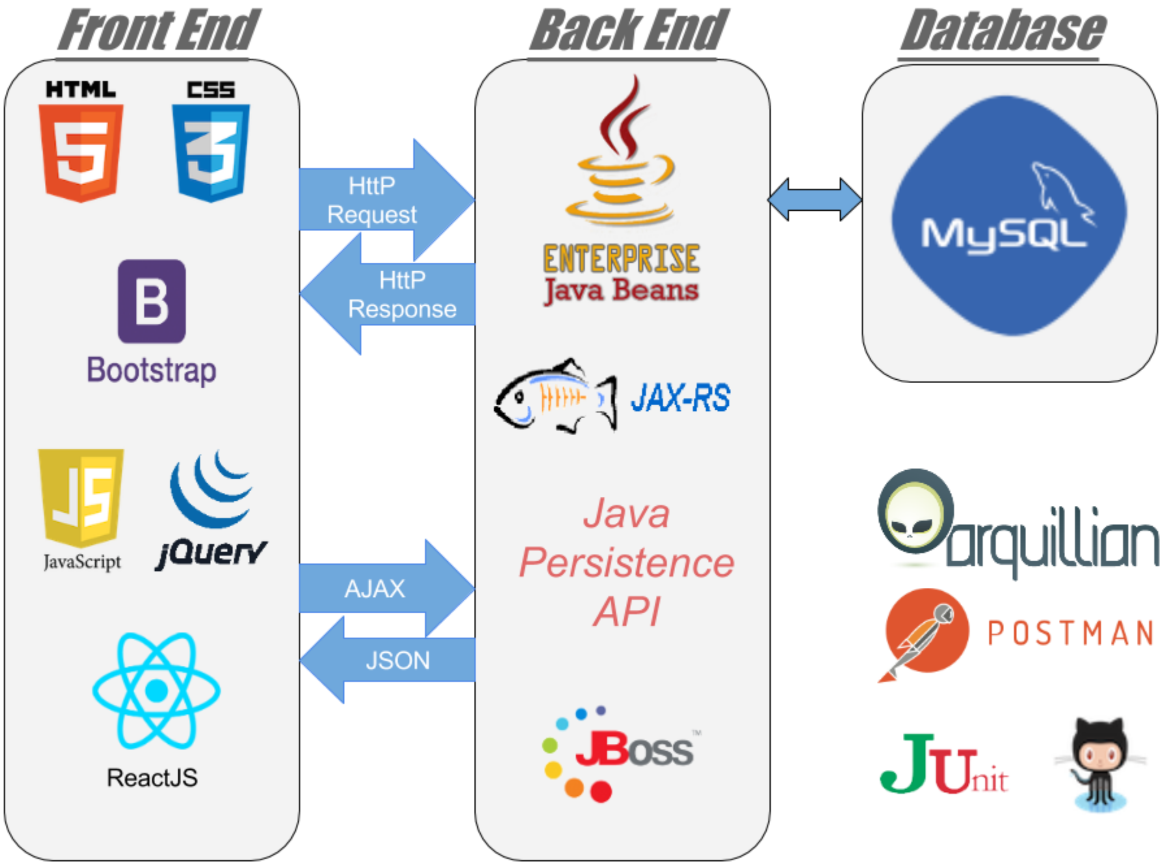
Number	**Priority**		**Name**
2	2		As a **REGISTERED USER** I want to be able to **LOGIN** to my **ACCOUNT**
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Login	I can use the account to access the CRUD on the site.
	Acceptance criteria		
	Scenario 1 – All User details entered correctly		
	Given	**When**	**Then**
	● Valid Details.	● The user clicks login.	● The user is logged in.
	Scenario 2 – Wrong email or password entered		
	Given	**When**	**Then**
	● Invalid Details	● The user clicks login.	● The user is not logged in. ● A prompt is shown saying why.

Number	**Priority**		**Name**
3	3		As a **REGISTERED USER** I want to be able to **EDIT** my **ACCOUNT**.
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Edit my details	Update the database.
	Acceptance criteria		
	Scenario 1 – When all fields edited are valid and new.		
	Given	**When**	**Then**
	● Valid Details.	● The user clicks save details.	● The user datatable is updated.
	Scenario 2 – Wrong email or password entered		
	Given	**When**	**Then**
	● Invalid Details	● The user clicks save details.	● The user datatable is not updated. ● A prompt is shown saying why.

Number	**Priority**	**Name**	
4	4	As a **registered user** I want to be able to **DELETE** my **ACCOUNT**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Delete my account.	Remove it from the database.
	Acceptance criteria		
	Scenario 1 – When valid password is entered.		
	Given	**When**	**Then**
	● Valid password entered.	● The user clicks delete account.	● The user datatable is updated. ● The users data is removed.
	Scenario 2 – Wrong password is entered.		
	Given	**When**	**Then**
	● Invalid password entered	● The user clicks delete account.	● The user datatable is not updated. ● A prompt is shown saying why.

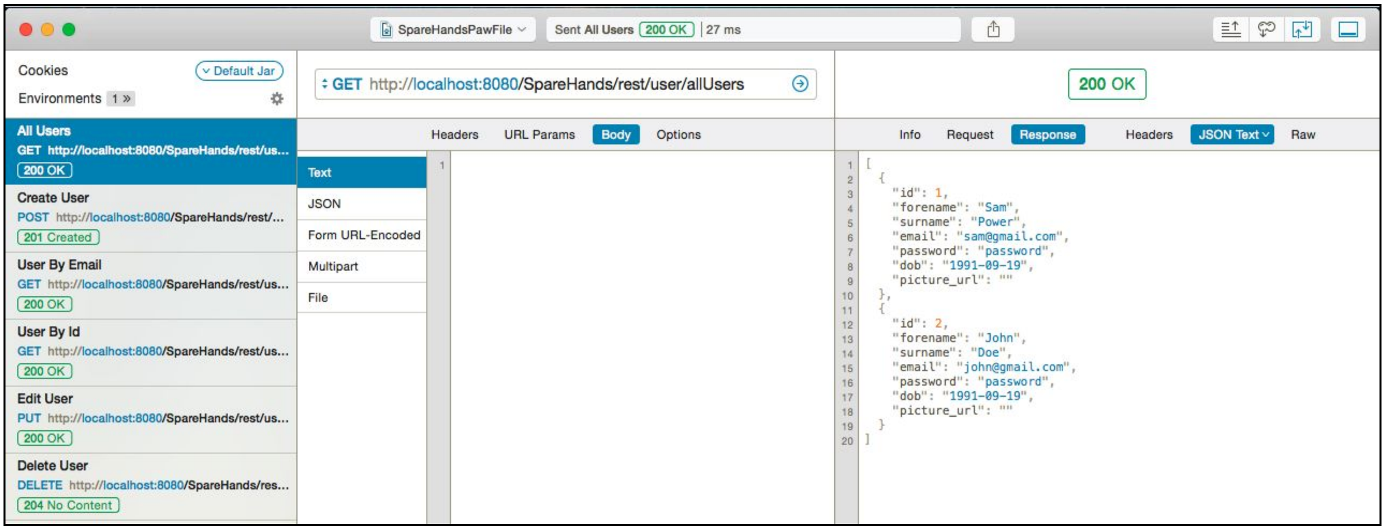
Number	**Priority**	**Name**	
5	5	As **Registered User** I want to be able to **CREATE** an **JOB**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Create a new job.	Users can see and respond to this job.
	Acceptance criteria		
	Scenario 1 – All fields are valid.		
	Given	**When**	**Then**
	● Valid Details	● The user clicks add job.	● The job datatable is updated.
	Scenario 2 – Any empty fields empty.		
	Given	**When**	**Then**
	● Empty Fields.	● The user clicks add job.	● The job table is not updated. ● A prompt is shown saying why.

System Architecture



Test Results

Integration Testing - User CRUD (PAW)



All Users
GET http://localhost:8080/SpareHands/rest/us...
200 OK

Create User
POST http://localhost:8080/SpareHands/rest/...
201 Created

User By Email
GET http://localhost:8080/SpareHands/rest/us...
200 OK

User By Id
GET http://localhost:8080/SpareHands/rest/us...
200 OK

Edit User
PUT http://localhost:8080/SpareHands/rest/us...
200 OK

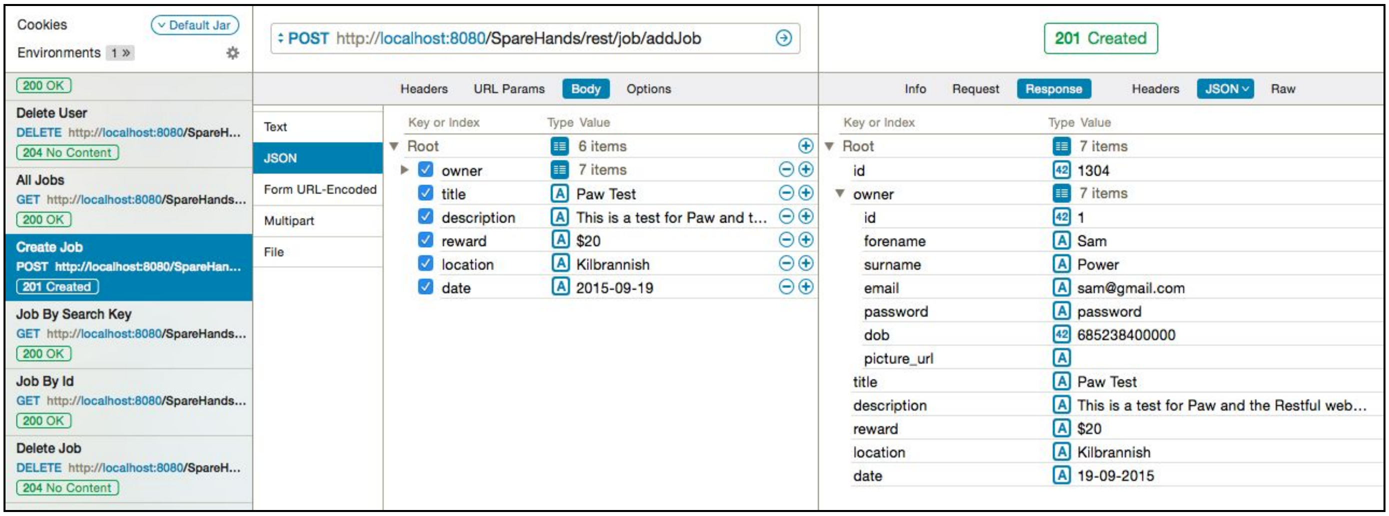
Delete User
DELETE http://localhost:8080/SpareHands/res...
204 No Content

GET http://localhost:8080/SpareHands/rest/user/allUsers

200 OK

```
[
  {
    "id": 1,
    "forename": "Sam",
    "surname": "Power",
    "email": "sam@gmail.com",
    "password": "password",
    "dob": "1991-09-19",
    "picture_url": ""
  },
  {
    "id": 2,
    "forename": "John",
    "surname": "Doe",
    "email": "john@gmail.com",
    "password": "password",
    "dob": "1991-09-19",
    "picture_url": ""
  }
]
```

Integration Testing - Create a Job (PAW)



200 OK

Delete User
DELETE http://localhost:8080/SpareH...
204 No Content

All Jobs
GET http://localhost:8080/SpareHands...
200 OK

Create Job
POST http://localhost:8080/SpareHan...
201 Created

Job By Search Key
GET http://localhost:8080/SpareHands...
200 OK

Job By Id
GET http://localhost:8080/SpareHands...
200 OK

Delete Job
DELETE http://localhost:8080/SpareH...
204 No Content

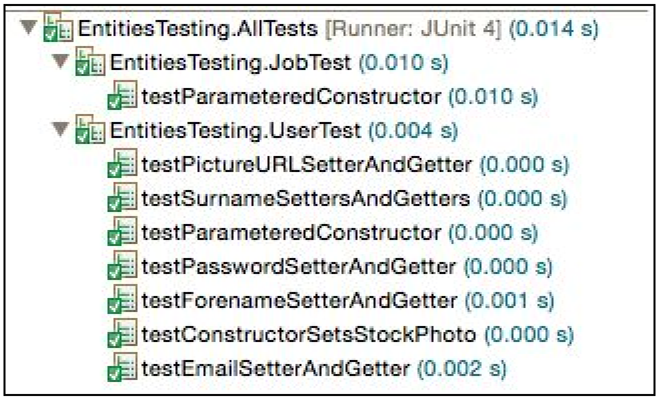
POST http://localhost:8080/SpareHands/rest/job/addJob

201 Created

Key or Index	Type	Value
▼ Root	≡	6 items
▶ ☑ owner	≡	7 items
☑ title	A	Paw Test
☑ description	A	This is a test for Paw and t...
☑ reward	A	$20
☑ location	A	Kilbrannish
☑ date	A	2015-09-19

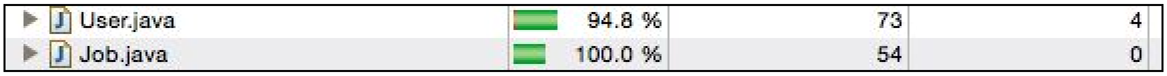
Key or Index	Type	Value
▼ Root	≡	7 items
id	42	1304
▼ owner	≡	7 items
id	42	1
forename	A	Sam
surname	A	Power
email	A	sam@gmail.com
password	A	password
dob	42	685238400000
picture_url	A	
title	A	Paw Test
description	A	This is a test for Paw and the Restful web...
reward	A	$20
location	A	Kilbrannish
date	A	19-09-2015

jUnit Tests (8/8)



```
▼ EntitiesTesting.AllTests [Runner: JUnit 4] (0.014 s)
  ▼ EntitiesTesting.JobTest (0.010 s)
      testParameteredConstructor (0.010 s)
  ▼ EntitiesTesting.UserTest (0.004 s)
      testPictureURLSetterAndGetter (0.000 s)
      testSurnameSettersAndGetters (0.000 s)
      testParameteredConstructor (0.000 s)
      testPasswordSetterAndGetter (0.000 s)
      testForenameSetterAndGetter (0.001 s)
      testConstructorSetsStockPhoto (0.000 s)
      testEmailSetterAndGetter (0.002 s)
```

jUnit Coverage



```
▶ User.java     94.8 %    73    4
▶ Job.java     100.0 %    54    0
```

Manual Tests ()

<table>
<tr><td>#</td><td colspan="2">Priority</td><td>Name</td></tr>
<tr><td>1</td><td colspan="2">1</td><td>As an <b>UNREGISTERED USER</b> I want to be able to <b>CREATE</b> an <b>ACCOUNT</b>.</td></tr>
<tr><td rowspan="11"></td><td colspan="3">Narrative</td></tr>
<tr><td>As an</td><td>I want to... (When)</td><td>So that... (Then)</td></tr>
<tr><td>Unregistered User</td><td>Create an account</td><td>I can use that account to access the CRUD on the site.</td></tr>
<tr><td colspan="3">Acceptance criteria</td></tr>
<tr><td colspan="3">Scenario 1 – All User details entered correctly</td></tr>
<tr><td>Given</td><td>When</td><td>Then</td></tr>
<tr><td>● Valid Details.</td><td>● The user is trying to register.</td><td>● The user datatable will be updated.</td></tr>
<tr><td colspan="3">Scenario 2 – User email is already registered</td></tr>
<tr><td>Given</td><td>When</td><td>Then</td></tr>
<tr><td>● All Details but email registered already.</td><td>● The user is trying to register.</td><td>● The user datatable is not updated.
● Prompt is shown saying why.</td></tr>
</table>

	Scenario 3 – User password does not match criteria		
	Given	**When**	**Then**
	● All Details but password does not match criteria.	● The user is trying to register	● The user datatable is not updated. ● Prompt is shown saying why.
	Scenario 4 – Any empty fields when registering		
	Given	**When**	**Then**
	● Empty fields when registering account.	● The user is trying to register.	● The user datatable is not updated. ● Prompt is shown saying why.

#	**Priority**	**Name**	
2	2	As a **REGISTERED USER** I want to be able to **LOGIN** to my **ACCOUNT**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Login	I can use the account to access the CRUD on the site.
	Acceptance criteria		
	Scenario 1 – All User details entered correctly		
	Given	**When**	**Then**
	● Valid Details.	● The user clicks login.	● The user is logged in.
	Scenario 2 – Wrong email or password entered		
	Given	**When**	**Then**
	● Invalid Details	● The user clicks login.	● The user is not logged in. ● A prompt is shown saying why.

#	**Priority**	**Name**	
3	3	As a **REGISTERED USER** I want to be able to **EDIT** my **ACCOUNT**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Edit my details	Update the database.
	Acceptance criteria		
	Scenario 1 – When all fields edited are valid and new.		
	Given	**When**	**Then**
	● Valid Details.	● The user clicks save details.	● The user datatable is updated.
	Scenario 2 – Wrong email or password entered		
	Given	**When**	**Then**
	● Invalid Details	● The user clicks save details.	● The user datatable is not updated. ● A prompt is shown saying why.

#	Priority	Name	
4	4	As a **registered user** I want to be able to **DELETE** my **ACCOUNT**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Delete my account.	Remove it from the database.
	Acceptance criteria		
	Scenario 1 – When valid password is entered.		
	Given	**When**	**Then**
	● Valid password entered.	● The user clicks delete account.	● The user datatable is updated. ● The users data is removed.
	Scenario 2 – Wrong password is entered.		
	Given	**When**	**Then**
	● Invalid password entered	● The user clicks delete account.	● The user datatable is not updated. ● A prompt is shown saying why.

#	Priority	Name	
5	5	As **Registered User** I want to be able to **CREATE** a **JOB**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Create a new job.	Users can see and respond to this job.
	Acceptance criteria		
	Scenario 1 – All fields are valid.		
	Given	**When**	**Then**
	● Valid Details	● The user clicks add job.	● The job datatable is updated.
	Scenario 2 – Any empty fields empty.		
	Given	**When**	**Then**
	● Empty Fields.	● The user clicks add job.	● The job table is not updated. ● A prompt is shown saying why.

Sprint 1

Sprint Goal

Full completion of project.

Definition of Done

1. Clean code produced.
2. Builds without errors.
3. System test completed.
4. HTML/CSS validated (if applicable).
5. Queries take less than 2 seconds.
6. ReactJS is used throughout.

Sprint Plan

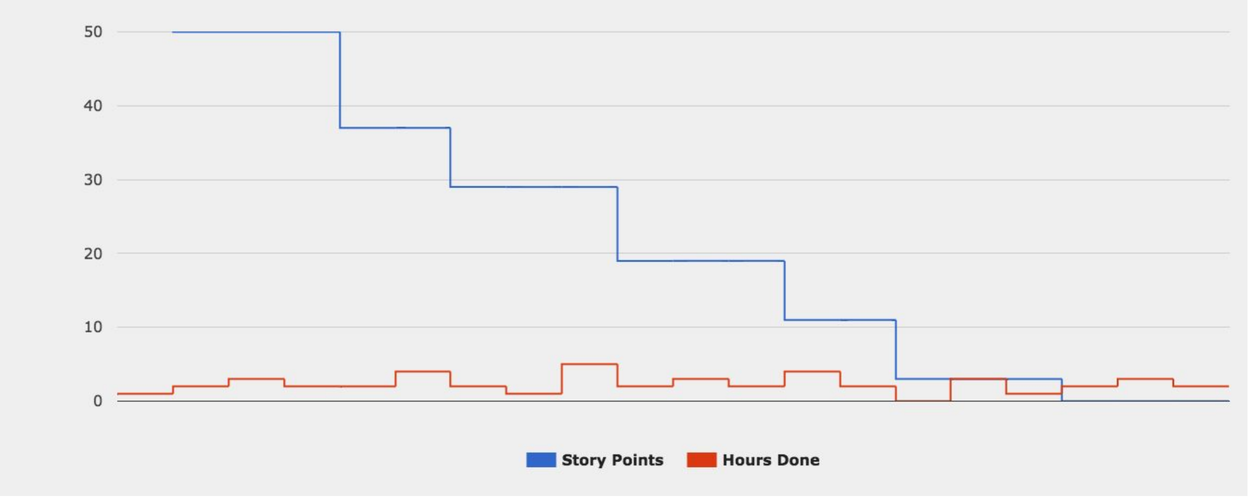
The following user stories were committed to be completed upon finishing Sprint 0.

- **User Story #7** - As a registered user I want to be able to delete a job I created.
- **User Story #8** - As any user I want to be able to view jobs.
- **User Story #9** - As any user I want to be able to filter the jobs by any search term.
- **User Story #10** - As a registered user I want to be able to apply to a job.
- **User Story #11** - As registered user I want to be able to reject/accept applications.
- **User Story #12** - As a registered user I want to be able to view all my jobs.
- **User Story #13** - As a registered user I want to be able to view all my applications.
- **User Story #13** - As a user, each query I make should take less than 2 seconds.

Also the following user story was removed for the reason that people may edit the job information after accepting an applicant, abusing the system.

- **User Story #6** - As a registered user I want to be able to edit a job I created.

Burndown Chart



In the range of November 11th - November 30th 2015

Test Strategy

Unit Test

AUTOMATED

70% code coverage

Tools: Junit version 4.12

Mockito version 1.9.5

Integration Test

MANUAL

One integration test per story

Server side on its own (WS, DAO, DB)

Tools: Postman (Switched from PAW)

Arquillian version 1.1.8

System Test

MANUAL & AUTO

One positive and one negative selenium test per story

Tools: Selenium version 2.46.0

Performance Test

MANUAL

Acceptance Test

MANUAL

Multiple per story (Given-When-Then)

User Story Elaboration

Number	Priority	Name	
7	7	As **Registered User** I want to be able to **DELETE** an **JOB** I created.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Delete an job.	Advert is removed from database.
	Acceptance criteria		
	Scenario 1 – Valid password is entered.		
	Given	**When**	**Then**
	● Valid Password	● The user clicks delete job.	● The job table entry is deleted.
	Scenario 2 – Invalid password is entered.		
	Given	**When**	**Then**
	● Invalid Password	● The user clicks delete job.	● The job table entry is not deleted.

Number	Priority	Name	
8	8	As **ANY USER** I want to be able to **VIEW JOBS**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	User	View all the Adverts	I can find one I want.
	Acceptance criteria		
	Scenario 1 – User clicks view jobs.		
	Given	**When**	**Then**
	● Any User	● The user clicks view jobs.	● All jobs are displayed.

Number	Priority	Name	
9	9	As **ANY USER** I want to be able to **FILTER** the **JOBS** by some **QUERY**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	User	Filter the jobs shown	I can get a more detailed list of jobs.
	Acceptance criteria		
	Scenario 1 – Search Query is valid.		
	Given	**When**	**Then**
	User	● The user enters any query.	● The jobs displayed are filtered.

Number	Priority	Name	
10	10	As a **REGISTERED USER** I want to be able to **APPLY** to an **JOB.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Respond to an job.	I can be considered for the job.
	Acceptance criteria		
	Scenario 1 – User logged in.		
	Given	**When**	**Then**
	● Registered User	● The user clicks Apply to Work.	● Applicant User's id is added to the job database entry. ● Advert Owner User is notified.
	Scenario 2 – User not logged in.		
	Given	**When**	**Then**
	● Unregistered User	● The user clicks Apply to Work.	● User is prompted to login!

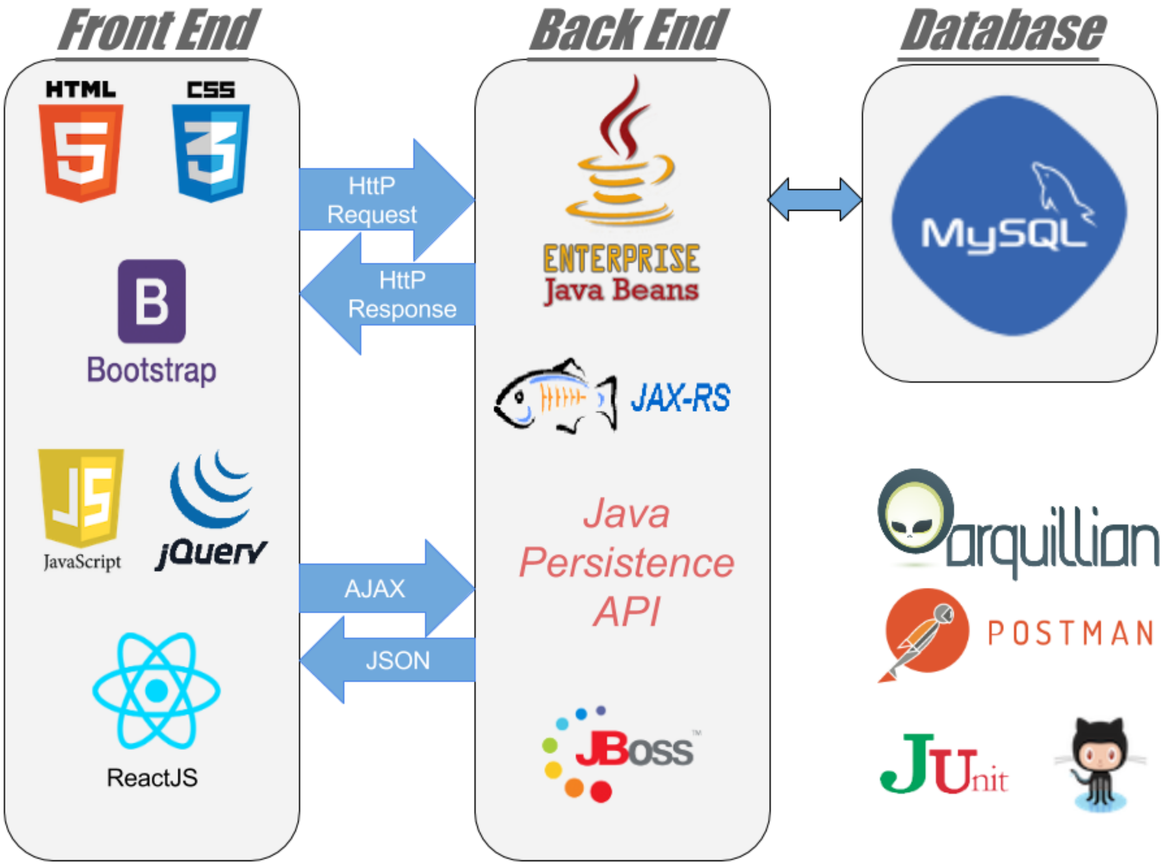
Number	Priority	Name	
11	11	As **Registered User** I want to be able to **REJECT/ACCEPT APPLICATIONS**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Respond to applications to an job I own.	Accept or Reject applicants.
	Acceptance criteria		
	Scenario 1 – Registered User clicks Accept		
	Given	**When**	**Then**

	● Registered User	● The user clicks accept Applicant.	● Applicant is notified. ● Applicants ID is moved from Applied column to Accepted column in Advert DB entry.
	Scenario 2 – Registered User clicks Reject		
	Given	**When**	**Then**
	● Registered User	● The user clicks accept Applicant.	● Applicant is notified. ● Applicants ID is moved from Applied column to Rejected column in Advert DB entry.

Number	**Priority**	**Name**	
12	12	As a **REGISTERED USER** I want to be able to **VIEW** all my **JOBS.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	View all the jobs I have posted	I can manage them.
	Acceptance criteria		
	Scenario 1 – User clicks My Adverts.		
	Given	**When**	**Then**
	● Registered User	● User clicks My Adverts	● User is shown their current jobs.

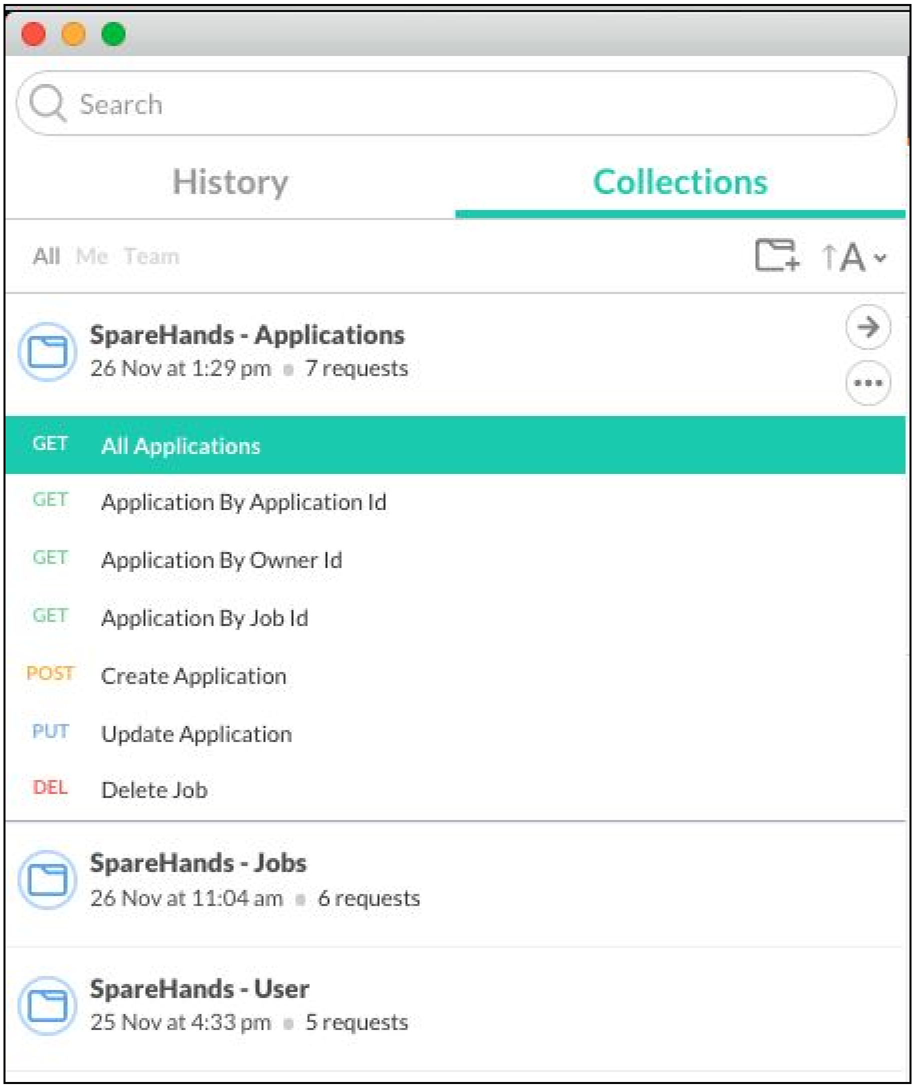
Number	**Priority**	**Name**	
13	13	As **REGISTERED USER** I want to be able to **VIEW** all my **APPLICATIONS.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	View my applications.	I can manage them.
	Acceptance criteria		
	Scenario 1 – User clicks My Applications		
	Given	**When**	**Then**
	● Registered User	● User clicks My Applications	● User is shown their current applications.

System Architecture



Test Results

Integration Testing - Full Application API Related Calls.



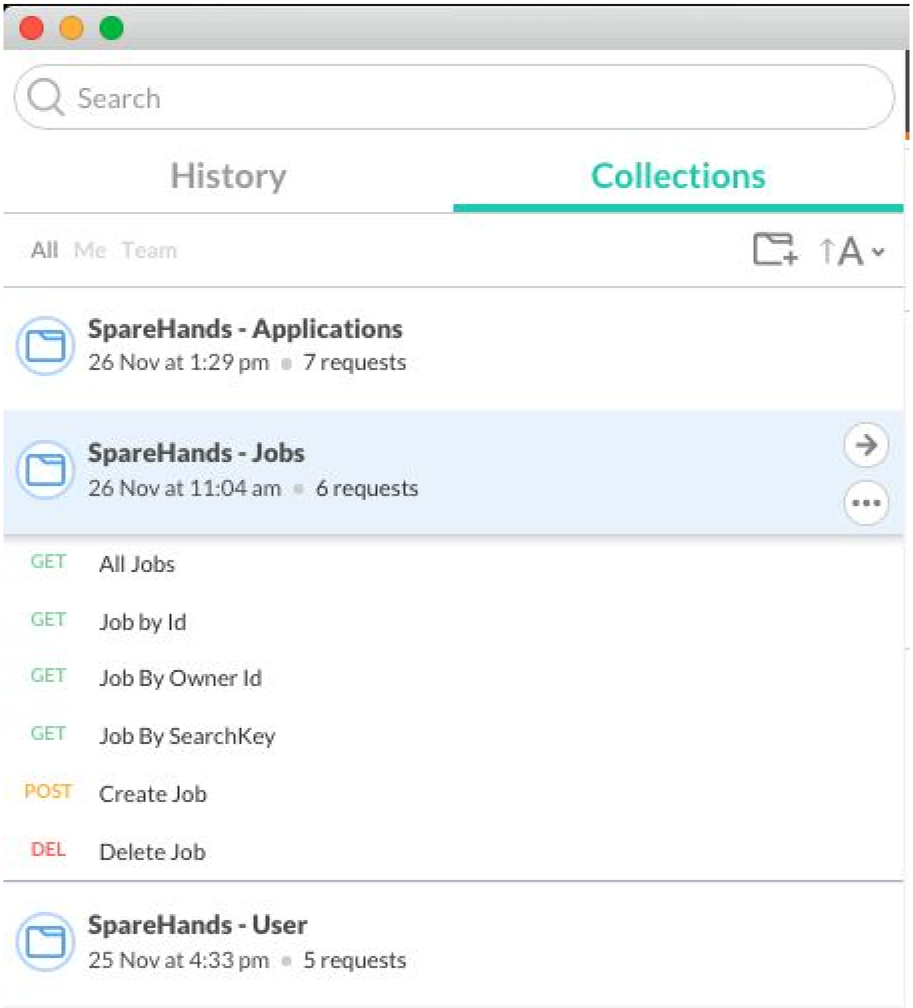
SpareHands - Applications
26 Nov at 1:29 pm • 7 requests

GET All Applications
GET Application By Application Id
GET Application By Owner Id
GET Application By Job Id
POST Create Application
PUT Update Application
DEL Delete Job

SpareHands - Jobs
26 Nov at 11:04 am • 6 requests

SpareHands - User
25 Nov at 4:33 pm • 5 requests

Integration Testing - Full Job API Related Calls.



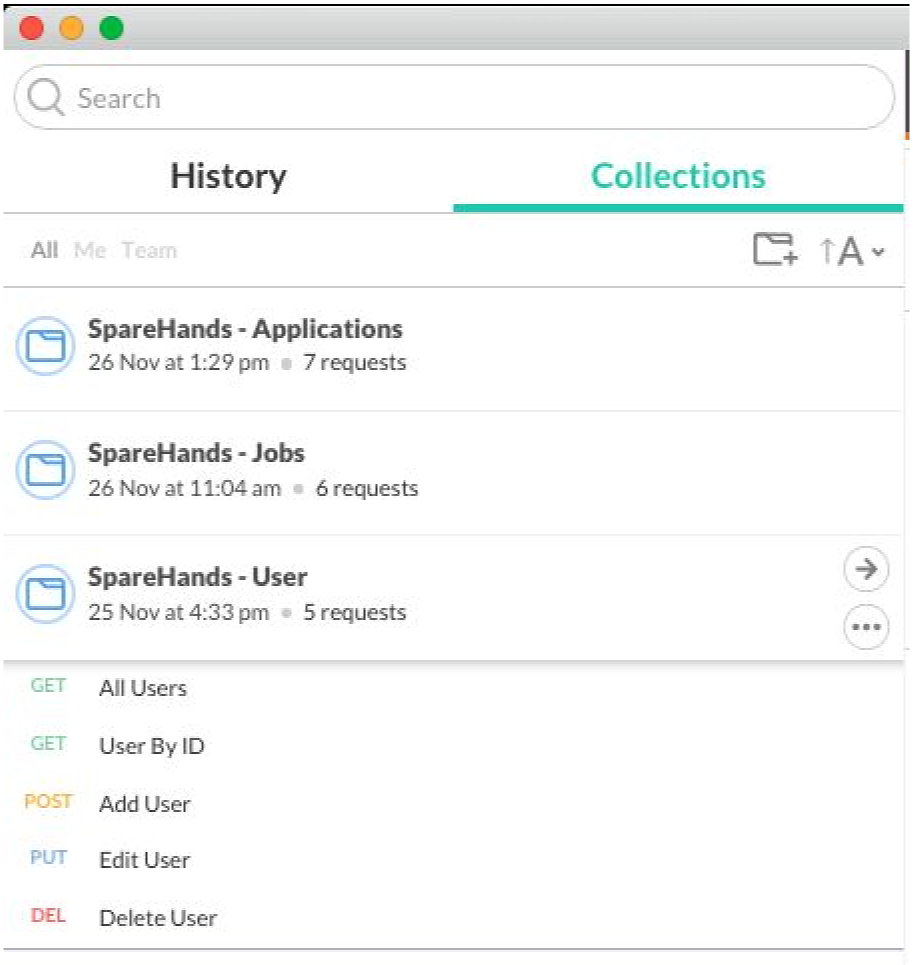
SpareHands - Applications
26 Nov at 1:29 pm • 7 requests

SpareHands - Jobs
26 Nov at 11:04 am • 6 requests

GET All Jobs
GET Job by Id
GET Job By Owner Id
GET Job By SearchKey
POST Create Job
DEL Delete Job

SpareHands - User
25 Nov at 4:33 pm • 5 requests

Integration Testing - Full User API Related Calls.



Search

History

Collections

All Me Team

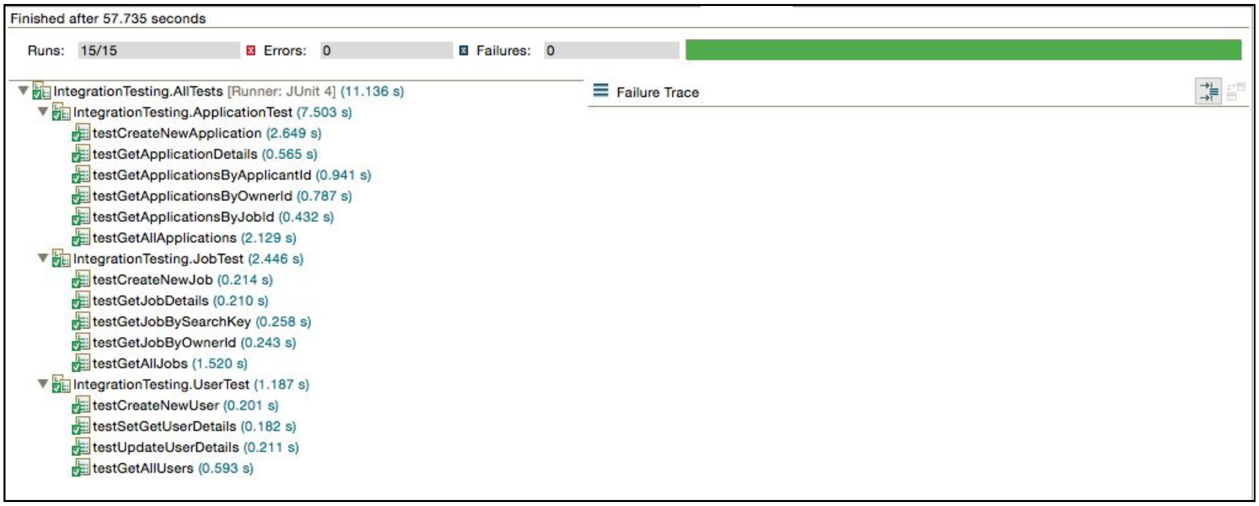
SpareHands - Applications
26 Nov at 1:29 pm • 7 requests

SpareHands - Jobs
26 Nov at 11:04 am • 6 requests

SpareHands - User
25 Nov at 4:33 pm • 5 requests

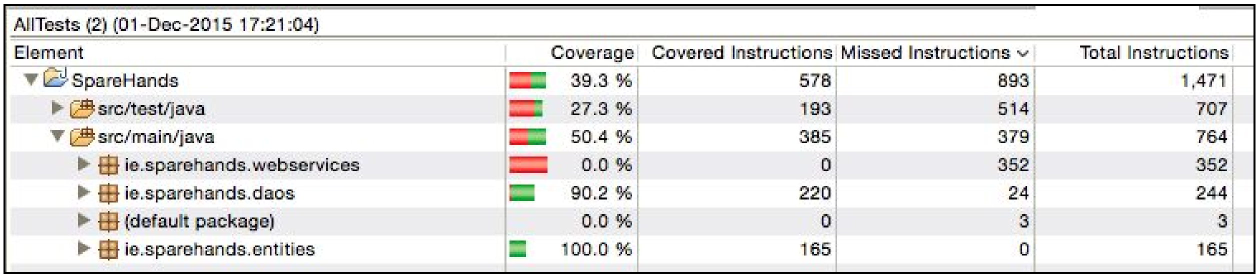
GET All Users
GET User By ID
POST Add User
PUT Edit User
DEL Delete User

Arquillian Testing of DAO’s and Entities (15/15)



```
Finished after 57.735 seconds

Runs: 15/15    Errors: 0    Failures: 0

▼ IntegrationTesting.AllTests [Runner: JUnit 4] (11.136 s)
  ▼ IntegrationTesting.ApplicationTest (7.503 s)
      testCreateNewApplication (2.649 s)
      testGetApplicationDetails (0.565 s)
      testGetApplicationsByApplicantId (0.941 s)
      testGetApplicationsByOwnerId (0.787 s)
      testGetApplicationsByJobId (0.432 s)
      testGetAllApplications (2.129 s)
  ▼ IntegrationTesting.JobTest (2.446 s)
      testCreateNewJob (0.214 s)
      testGetJobDetails (0.210 s)
      testGetJobBySearchKey (0.258 s)
      testGetJobByOwnerId (0.243 s)
      testGetAllJobs (1.520 s)
  ▼ IntegrationTesting.UserTest (1.187 s)
      testCreateNewUser (0.201 s)
      testSetGetUserDetails (0.182 s)
      testUpdateUserDetails (0.211 s)
      testGetAllUsers (0.593 s)
```

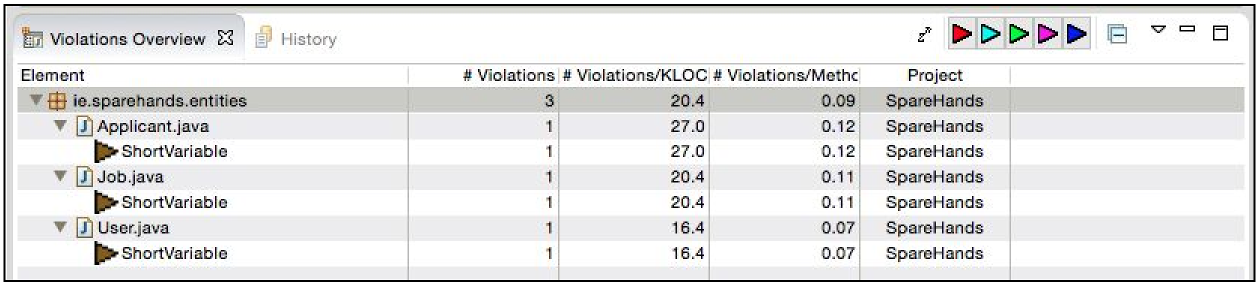
Arquillian Coverage of DAO’s and Entities (50.4% of entire project)



AllTests (2) (01-Dec-2015 17:21:04)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ SpareHands	39.3 %	578	893	1,471
▶ src/test/java	27.3 %	193	514	707
▼ src/main/java	50.4 %	385	379	764
▶ ie.sparehands.webservices	0.0 %	0	352	352
▶ ie.sparehands.daos	90.2 %	220	24	244
▶ (default package)	0.0 %	0	3	3
▶ ie.sparehands.entities	100.0 %	165	0	165

PMD Testing of Project



Element	# Violations	# Violations/KLOC	# Violations/Methc	Project
▼ ie.sparehands.entities	3	20.4	0.09	SpareHands
▼ Applicant.java	1	27.0	0.12	SpareHands
ShortVariable	1	27.0	0.12	SpareHands
▼ Job.java	1	20.4	0.11	SpareHands
ShortVariable	1	20.4	0.11	SpareHands
▼ User.java	1	16.4	0.07	SpareHands
ShortVariable	1	16.4	0.07	SpareHands

*Note that these Short Variables are named “Id” which directly link to the Database variables.

Manual Test

#	Priority	Name	
7	7	As **Registered User** I want to be able to **DELETE** a **JOB** I created.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Delete an job.	Advert is removed from database.
	Acceptance criteria		
	Scenario 1 – Valid password is entered.		
	Given	**When**	**Then**
	● Valid Password	● The user clicks delete job.	● The job table entry is deleted.
	Scenario 2 – Invalid password is entered.		
	Given	**When**	**Then**
	● Invalid Password	● The user clicks delete job.	● The job table entry is not deleted.

#	Priority	Name	
8	8	As **ANY USER** I want to be able to **VIEW JOBS**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	User	View all the Adverts	I can find one I want.
	Acceptance criteria		
	Scenario 1 – User clicks view jobs.		
	Given	**When**	**Then**
	● Any User	● The user clicks view jobs.	● All jobs are displayed.

#	Priority	Name	
9	9	As **ANY USER** I want to be able to **FILTER** the **JOBS** by some **QUERY**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**

	User	Filter the jobs shown	I can get a more detailed list of jobs.
	Acceptance criteria		
	Scenario 1 – Search Query is valid.		
	Given	**When**	**Then**
	User	● The user enters any query.	● The jobs displayed are filtered.

#	**Priority**	**Name**	
10	10	As a **REGISTERED USER** I want to be able to **APPLY** to a **JOB.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Respond to an job.	I can be considered for the job.
	Acceptance criteria		
	Scenario 1 – User logged in.		
	Given	**When**	**Then**
	● Registered User	● The user clicks Apply to Work.	● Applicant User's id is added to the job database entry. ● Advert Owner User is notified.
	Scenario 2 – User not logged in.		
	Given	**When**	**Then**
	● Unregistered User	● The user opens a job info modal.	● Apply button is not visible.

#	**Priority**	**Name**	
11	11	As **Registered User** I want to be able to **REJECT/ACCEPT APPLICATIONS**.	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	Respond to applications to an job I own.	Accept or Reject applicants.
	Acceptance criteria		
	Scenario 1 – Registered User clicks Accept		
	Given	**When**	**Then**
	● Registered User	● The user clicks accept Applicant.	● Applicant is notified. ● Applicants ID is moved from Applied column to Accepted column in Advert DB entry.

	Scenario 2 – Registered User clicks Reject		
	Given	**When**	**Then**
	● Registered User	● The user clicks accept Applicant.	● Applicant is notified. ● Applicants ID is moved from Applied column to Rejected column in Advert DB entry.

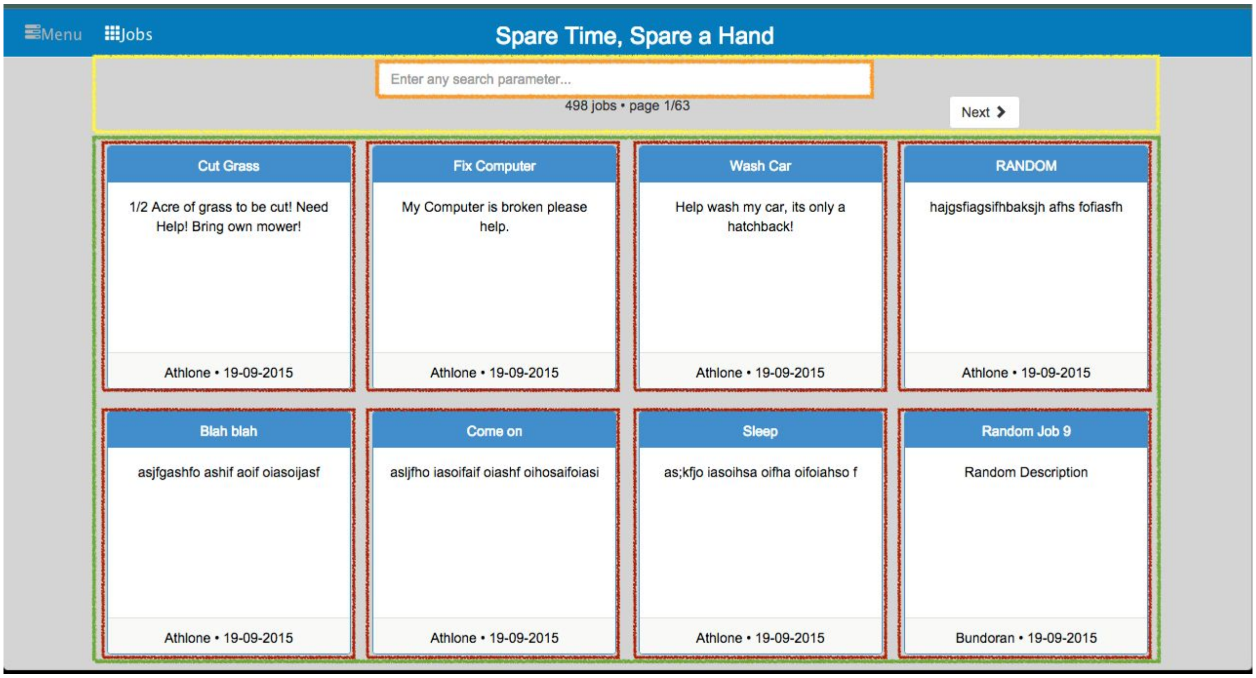
#	**Priority**	**Name**	
12	12	As a **REGISTERED USER** I want to be able to **VIEW** all my **JOBS.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	View all the jobs I have posted	I can manage them.
	Acceptance criteria		
	Scenario 1 – User clicks My Adverts.		
	Given	**When**	**Then**
	● Registered User	● User clicks My Adverts	● User is shown their current jobs.

#	**Priority**	**Name**	
13	13	As **REGISTERED USER** I want to be able to **VIEW** all my **APPLICATIONS.**	
	Narrative		
	As a	**I want to... (When)**	**So that... (Then)**
	Registered User	View my applications.	I can manage them.
	Acceptance criteria		
	Scenario 1 – User clicks My Applications		
	Given	**When**	**Then**
	● Registered User	● User clicks My Applications	● User is shown their current applications.

Research Topic - ReactJS

ReactJS is a javascript library used for building the front end of a web based application that was created by Facebook and Instagram. The aim of this library is to make building a large application with data that changes over time an easier task.

React is centered around the creation of one or more components which contain a render method that returns HTML to a specified HTML element. ReactJS provides JSX which allows for the embedding of HTML directly into the Javascript. These components are reusable in the same sense as you would see in any OOP language. An example of this is seen below in the rendering of the list of jobs seen on the homepage of Spare Hands as there are a number of Job List Item components within the Job List component.



Cut Grass
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Fix Computer
My Computer is broken please help.
Athlone • 19-09-2015

Wash Car
Help wash my car, its only a hatchback!
Athlone • 19-09-2015

RANDOM
hajgsfiagsifhbaksjh afhs fofiasfh
Athlone • 19-09-2015

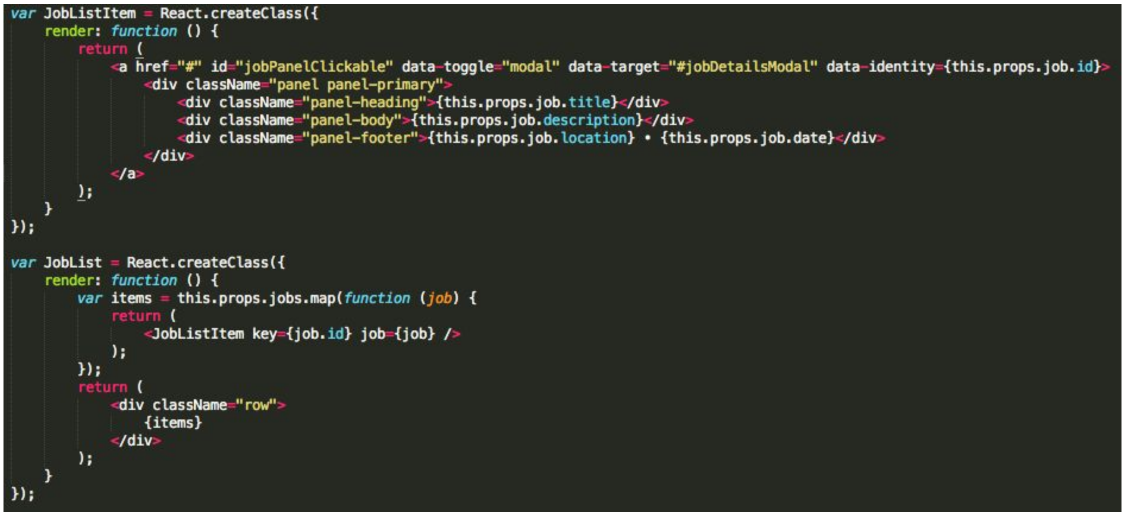
Blah blah
asjfgashfo ashif aoif oiasoijasf
Athlone • 19-09-2015

Come on
asljfho iasoifaif oiashf oihosaifoiasi
Athlone • 19-09-2015

Sleep
as;kfjo iasoihsa oifha oifoiahso f
Athlone • 19-09-2015

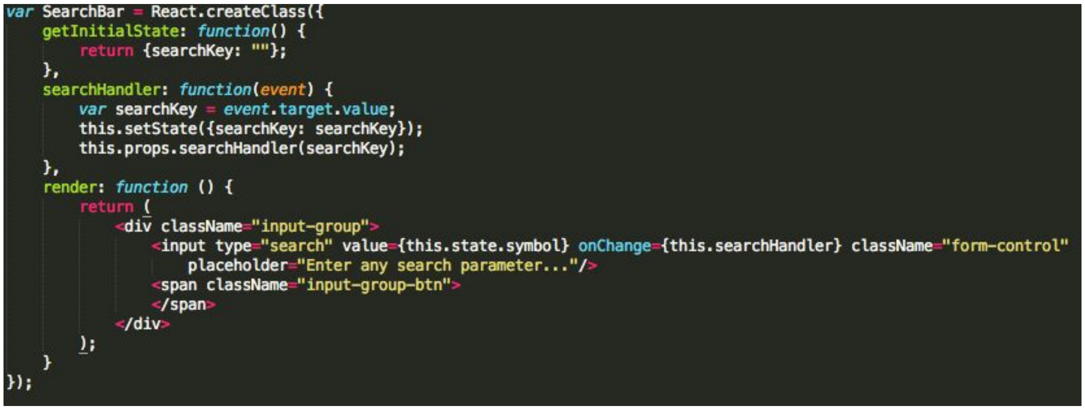
Random Job 9
Random Description
Bundoran • 19-09-2015

- **Orange Outline -** Search Bar Component
- **Yellow Outline -** Paginator Component
- **Green Outline -** Job List Component
- **Red Outline -** Job List Item Component



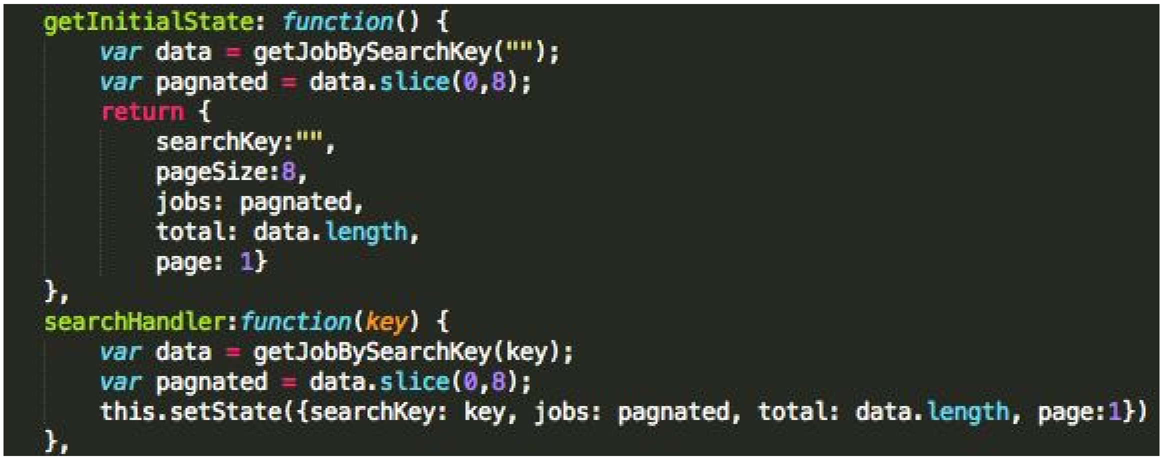
```
var JobListItem = React.createClass({
    render: function () {
        return (
            <a href="#" id="jobPanelClickable" data-toggle="modal" data-target="#jobDetailsModal" data-identity={this.props.job.id}>
                <div className="panel panel-primary">
                    <div className="panel-heading">{this.props.job.title}</div>
                    <div className="panel-body">{this.props.job.description}</div>
                    <div className="panel-footer">{this.props.job.location} • {this.props.job.date}</div>
                </div>
            </a>
        );
    }
});

var JobList = React.createClass({
    render: function () {
        var items = this.props.jobs.map(function (job) {
            return (
                <JobListItem key={job.id} job={job} />
            );
        });
        return (
            <div className="row">
                {items}
            </div>
        );
    }
});
```

Another benefit of ReactJS is that the onclicklisteners can be declared easily with the code itself. An example of this is shown in the Search Bar component. Below we see that rather than declaring our listener using Javascript or jQuery we can easily do so while rendering the <input/> by adding the onChange={function()} to the declaration. Any change will result in the searchHandler function being run which in turn passes the new state back up to the component which rendered this one allowing it to then update itself.



```
var SearchBar = React.createClass({
    getInitialState: function() {
        return {searchKey: ""};
    },
    searchHandler: function(event) {
        var searchKey = event.target.value;
        this.setState({searchKey: searchKey});
        this.props.searchHandler(searchKey);
    },
    render: function () {
        return (
            <div className="input-group">
                <input type="search" value={this.state.symbol} onChange={this.searchHandler} className="form-control"
                    placeholder="Enter any search parameter..."/>
                <span className="input-group-btn">
                </span>
            </div>
        );
    }
});
```

Below is a snippet from the JobsGrid component renders the JobList component. The array of objects from which the JobList renders is declared initially as all jobs but upon a change in the SearchBar it will be updated to a more refined list.

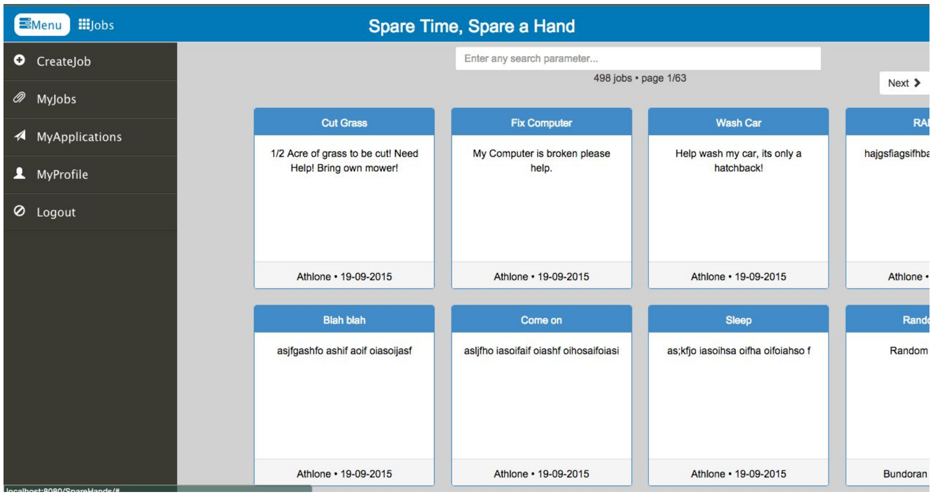


```
getInitialState: function() {
    var data = getJobBySearchKey("");
    var pagnated = data.slice(0,8);
    return {
        searchKey:"",
        pageSize:8,
        jobs: pagnated,
        total: data.length,
        page: 1}
},
searchHandler:function(key) {
    var data = getJobBySearchKey(key);
    var pagnated = data.slice(0,8);
    this.setState({searchKey: key, jobs: pagnated, total: data.length, page:1})
},
```

In conclusion I feel that ReactJS is a very powerful and readable language. At the moment my understanding of it is not quite where I would like it to be but I feel that I have the basics understood very well. I think that it is a very useful language when building any site that wants to implement live updates to it’s UI in the result of a change in its underlying data.

Implementation

HTML, CSS and Bootstrap were used extensively throughout this project and can be seen any of the screenshots below. Javascript and jQuery were used in large to run a function in the event of an element being clicked and in hiding and showing the user the relevant and permissible views. A key feature of this is the slide out menu that can only be accessed by users that are logged in.



Cut Grass
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Fix Computer
My Computer is broken please help.
Athlone • 19-09-2015

Wash Car
Help wash my car, its only a hatchback!
Athlone • 19-09-2015

RA
hajgsfiagsifhba
Athlone •

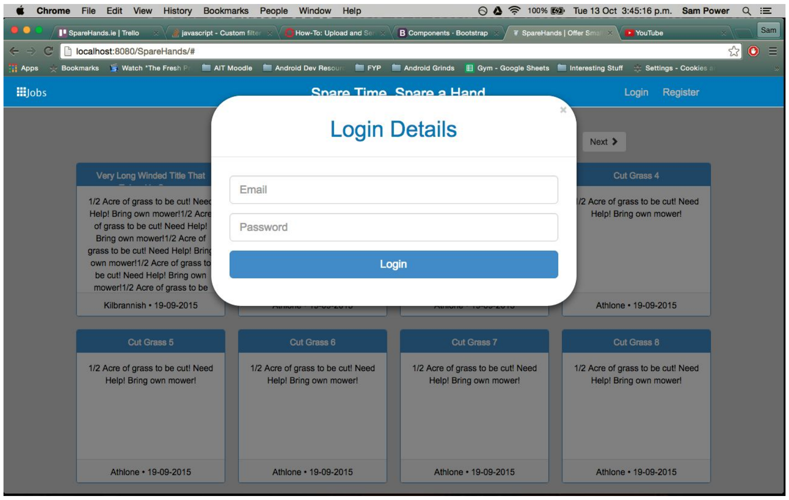
Blah blah
asjfgashfo ashif aoif oiasoijasf
Athlone • 19-09-2015

Come on
asljfho iasoifaif oiashf oihosaifoiasi
Athlone • 19-09-2015

Sleep
as;kfjo iasoihsa oifha oifoiahso f
Athlone • 19-09-2015

Rand
Random
Bundoran

Modals were also used to register, login, create a job, view a jobs details and manage a jobs applications as shown below.


Chrome File Edit View History Bookmarks People Window Help
100% Tue 13 Oct 3:45:16 p.m. Sam Power
SpareHands.ie | Trello
javascript - Custom filte
How-To: Upload and Se
Components · Bootstrap
SpareHands | Offer Sma
YouTube
Sam
localhost:8080/SpareHands/#
Apps Bookmarks Watch "The Fresh P AIT Moodle Android Dev Resou FYP Android Grinds Gym - Google Sheets Interesting Stuff Settings - Cookies
Jobs
Spare Time, Spare a Hand
Login
Register
Login Details
×
Email
Password
Login
Next

Very Long Winded Title That
1/2 Acre of grass to be cut! Need Help! Bring own mower!1/2 Acre of grass to be cut! Need Help! Bring own mower!1/2 Acre of grass to be cut! Need Help! Bring own mower!1/2 Acre of grass to be cut! Need Help! Bring own mower!1/2 Acre of grass to be
Kilbrannish • 19-09-2015

Cut Grass 4
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

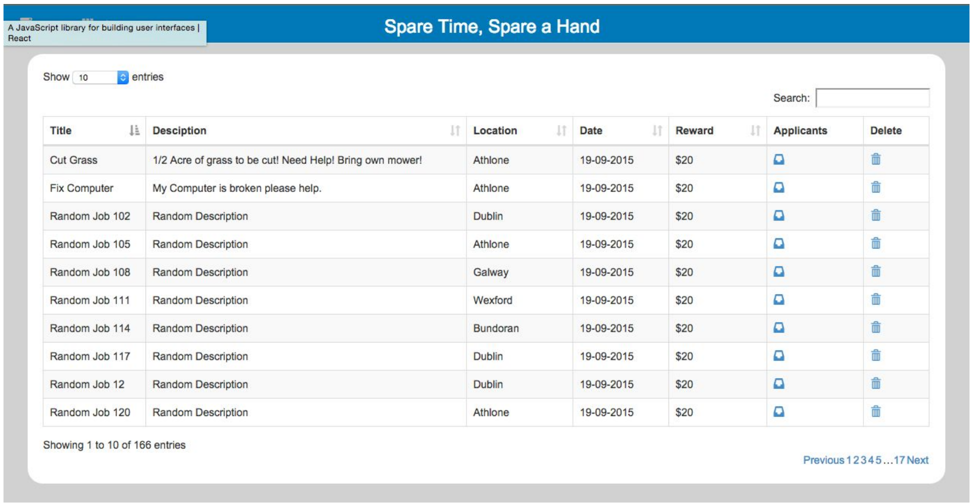
Cut Grass 5
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Cut Grass 6
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Cut Grass 7
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Cut Grass 8
1/2 Acre of grass to be cut! Need Help! Bring own mower!
Athlone • 19-09-2015

Datatables were another technology used, specifically to show the user their own list of jobs or applications.



A JavaScript library for building user interfaces | React

Spare Time, Spare a Hand

Show 10 entries

Search:

Title	Desciption	Location	Date	Reward	Applicants	Delete
Cut Grass	1/2 Acre of grass to be cut! Need Help! Bring own mower!	Athlone	19-09-2015	$20	📥	🗑
Fix Computer	My Computer is broken please help.	Athlone	19-09-2015	$20	📥	🗑
Random Job 102	Random Description	Dublin	19-09-2015	$20	📥	🗑
Random Job 105	Random Description	Athlone	19-09-2015	$20	📥	🗑
Random Job 108	Random Description	Galway	19-09-2015	$20	📥	🗑
Random Job 111	Random Description	Wexford	19-09-2015	$20	📥	🗑
Random Job 114	Random Description	Bundoran	19-09-2015	$20	📥	🗑
Random Job 117	Random Description	Dublin	19-09-2015	$20	📥	🗑
Random Job 12	Random Description	Dublin	19-09-2015	$20	📥	🗑
Random Job 120	Random Description	Athlone	19-09-2015	$20	📥	🗑

Showing 1 to 10 of 166 entries

ReactJS was used to render the home page (All Jobs) along with displaying error messages in all of the modals.

The backend was built using JAX-RS, EJBs, JPA and run on a jBoss server with a SQL Database and it was tested using Postman and Arquillian. I used Github as my version control and Trello.com to keep track of the tasks on my Scrum Board.

Evaluation and Conclusion

The initial aim of this project was to develop a web application using SPA architecture while also adhering to the Scope of the Project and Aims & Objectives sections at the beginning of this report. I believe that I have met all of these requirements. There are changes with my project which I would love to implement if there was more time available. Of I were beginning again I would definitely focus more on implementing the front end with ReactJS. I believe that this would lead to an improvement in the speed and efficiency of the site and also the readability and reusability of the code. I would also implement server side pagination using Backbone so as to make it a more scalable website. I feel that at the moment it is good enough for a small user base but if it the database was to grow to a much larger size then it would begin to suffer.

In conclusion I feel that I have met all requirements asked of me and that I have learned a great deal in the process of completing this project and I will be continuing to use ReactJS as I have developed an affinity for the library.