QA Spring Boot Project

By Farzan Chowdhury

Introduction

The Task:

- We were told to create a CRUD application with the utilisation of supporting tools, methodologies and technologies that encapsulate all core modules covered during training.

What I chose:

- I chose my topic to be cats.

Contents

Planning

Brief:

- Cat - A small domesticated carnivorous mammal with soft fur, a short snout, and retractable claws. It is widely kept as a pet or for catching mice.

Why I chose it:

- Because they were easy to implement within the project requirements using all CRUD functionalities.

How it relates to me:

- I have always wanted a cat but could never get one so I thought I'd use it in this project instead.

Planning the project

Approaching the project:

- First, I read the project specifications.
- Broke down the project scope.
- Created a Kanban board on Jira.
- Created a risk assessment.
- Made an ERD to represent my database.
- Made a git repository and initialised my repository on my computer.
- Check my project met the requirements of the MVP.

Following the plan:

- I was able to follow my project very well because I completed all the project requirements within the deadline.

Technologies

Learned Technologies:

- Jira
- Git/GitHub
- MySQL
- Java
- Maven
- MockMVC
- HTML
- CSS
- JavaScript

Learned technologies: JIra

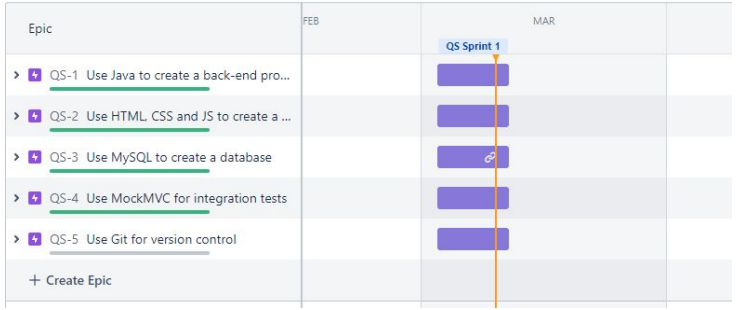
I used Jira to plan the project.

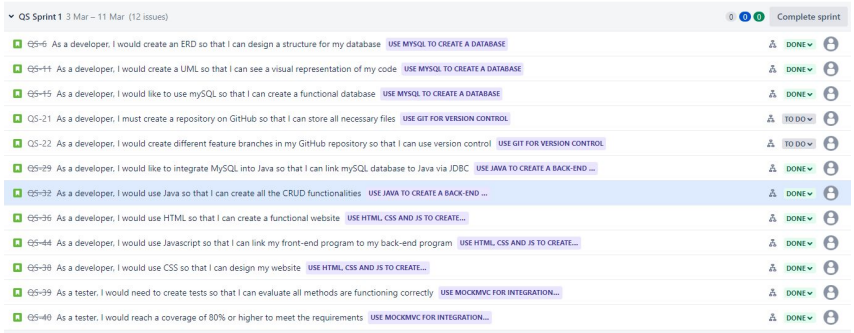
Working with it:

- I learned to structure all of my tasks.
- Good time management.

Even better if:

- Used more blockers.
- Add extra user-stories and child issues to meet the extension task.





Learned technologies: Git/GitHub

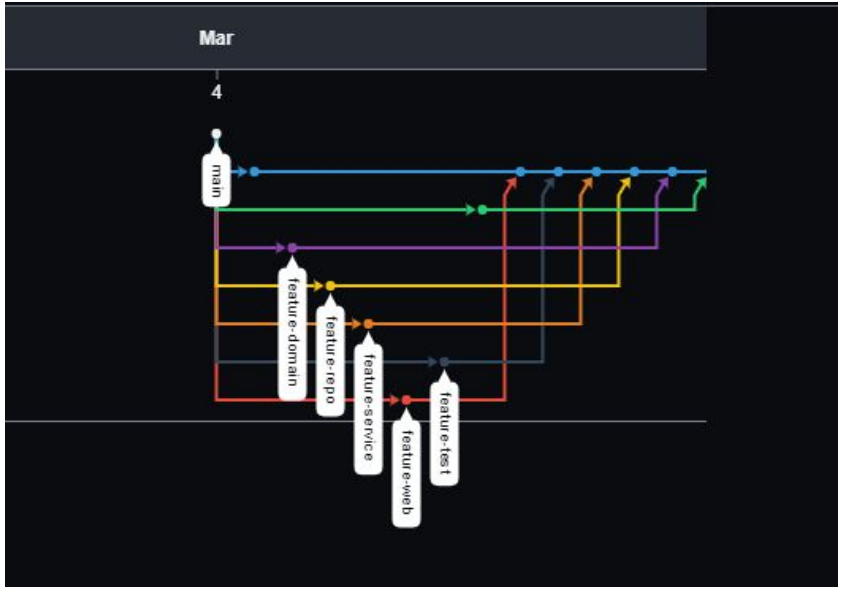
I used Git for version control.

Working with it:

- I learned to make multiple feature branches for different tasks.
- I learned to push any completed files to repository.

Even better if:

- I become more confident in using Git.



Learned technologies: MySQL

I used MySQL for my database.

Working with it:

- I learned to make tables with attributes.
- I learned to add any data to the tables.

```
1  create database if not exists spring;
2  use spring;
3  create table cat (id integer auto_increment, age integer, name varchar(255), breed varchar(255), primary key (id));
4
5  insert into cat (age, breed, name) values (12, 'Persian', 'John');
6  insert into cat (age, breed, name) values (5, 'Bengal', 'Jimmy');
7
8  select * from cat;
9
10
```

Learned technologies: Java

I used Java for backend application with Spring Boot Framework.

Working with it:

- I learned to implement multiple functional methods.
- I learned to create any required classes, files and interfaces.

Even better if:

- Made more functional methods.

```java
@Test
void testCreate() throws Exception {
    Cat testCat = new Cat(null, "Jones", "Sphynx", 15);
    String testCatAsJSON = this.mapper.writeValueAsString(testCat);
    RequestBuilder req = post("/create").contentType(MediaType.APPLICATION_JSON).content(testCatAsJSON);

    Cat testCreatedCat = new Cat(3, "Jones", "Sphynx", 15);
    String testCreatedDCatAsJSON = this.mapper.writeValueAsString(testCreatedCat);
    ResultMatcher checkStatus = status().isCreated();
    ResultMatcher checkBody = content().json(testCreatedDCatAsJSON);

    this.mvc.perform(req).andExpect(checkStatus).andExpect(checkBody);
}

@Test
void testGetAll() throws Exception {
    RequestBuilder req = get("/getAll");
    List<Cat> testDogs = List.of(new Cat(1, "John", "Persian", 12), new Cat(2, "Jimmy", "Bengal", 5));
    String testGetAllAsJSON = this.mapper.writeValueAsString(testDogs);
    ResultMatcher checkStatus = status().isOk();
    ResultMatcher checkBody = content().json(testGetAllAsJSON);

    this.mvc.perform(req).andExpect(checkStatus).andExpect(checkBody);
}

@Test
void testGetById() throws Exception {
    RequestBuilder req = get("/get/1");
    Cat testCats = new Cat(1, "John", "Persian", 12);
    String testGetByIdAsJSON = this.mapper.writeValueAsString(testCats);
    ResultMatcher checkStatus = status().isOk();
    ResultMatcher checkBody = content().json(testGetByIdAsJSON);

    this.mvc.perform(req).andExpect(checkStatus).andExpect(checkBody);
}
```

Learned technologies: MockMVC

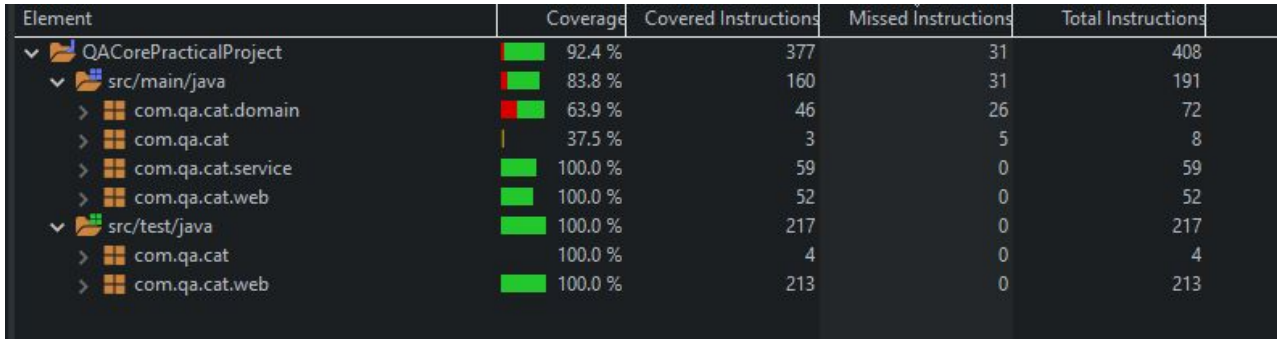
I used MockMVC for testing.

Working with it:

- I learned to test all functional methods.

Even better if:

- Added unit testing for the classes that weren't covered.



Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
QACorePracticalProject	92.4 %	377	31	408
src/main/java	83.8 %	160	31	191
com.qa.cat.domain	63.9 %	46	26	72
com.qa.cat	37.5 %	3	5	8
com.qa.cat.service	100.0 %	59	0	59
com.qa.cat.web	100.0 %	52	0	52
src/test/java	100.0 %	217	0	217
com.qa.cat	100.0 %	4	0	4
com.qa.cat.web	100.0 %	213	0	213

Learned technologies: HTML, CSS and Javascript

I used HTML, CSS and JavaScript for frontend application.

Working with it:

- I learned to create a webpage using my HTML.
- I learned to create working functions on JavaScript(used fetch api).

Even better if:

- Better Styling of my webpage using CSS.
- Added more functions for the user.

```
<body onload="showCats()">
  <div class="container">
        <h1>Your Favourite Cat Catalogue!</h1>
        <h3>Add New Cat</h3>
        <form class="add-cat-form">
          <div class="form-group">
            <label for="name">Cat Name</label>
            <input type="text" class="form-control" id="catName" />
          </div>
          <div class="form-group">
            <label for="breed">Cat Breed</label>
            <input type="text" class="form-control" id="catBreed" />
          </div>
          <div class="form-group">
            <label for="age">Cat Age</label>
            <input type="text" class="form-control" id="catAge" />
          </div>
          <div class="form-action-buttons">
            <input
              type="button"
              id="create"
              value="Create"
              onclick="createCat()"
            />
          </div>
          <div class="form-group">
            <label for="catById">Input ID of Cat</label>
            <br /><input id="catId" type="text" class="form-control" />
          </div>
          <div class="form-action-buttons">
            <input type="button" value="submit" onclick="getCatById()" />
          </div>
```

Demonstration

Summary

What went well:

- All tasks and user-stories completed on Jira.
- All operations completed such as create,read,getbyid, update and delete for cats.
- Created a fully functional frontend which link with my database and backend.

What didn’t go well:

- Better time management.
- Achieving higher testing coverage for the project.
- More methods for backend.
- Making the webpage function well.
- Completing the extension task.

Thank you for listening!

Any questions?