IMS PROJECT

Pete Hutchison

Introduction

- I’m Pete!
- I started learning about software development in August this year.
- This is my second QA software-related Bootcamp.

Journey to the project:

What have I learned?

1. **SOURCE CONTROL**
   How to make use of source control with Git & GitHub

2. **DATABASES & SQL**
   Using MySQL, SQL language -> DDL, DML

3. **JAVA**
   Beginner and intermediate features of Java

4. **OOP & SOLID PRINCIPLES**
   Understanding principles and best practices

5. **JDBC, JUNIT, & MAVEN**
   Integrating databases and testing into one final build

6. **JIRA**
   Sprints, user stories, story points, MoSCoW, etc.

Approach to the project
What have I used?

1. SOURCE CONTROL
   Git & GitHub implemented using the feature-branch model

2. SCRUM BOARD
   Jira used to track progress and task prioritisation

3. DATABASE MANAGEMENT
   Local instance of MySQL Server

4. BACK-END PROGRAMMING
   Java used as the back-end language throughout

5. BUILD TOOL
   Project was put together with Eclipse using Maven

6. TESTING
   JUnit & Mockito were used to construct tests

Timeline

DAY 1
Jira board, risk assessment & setting up repo

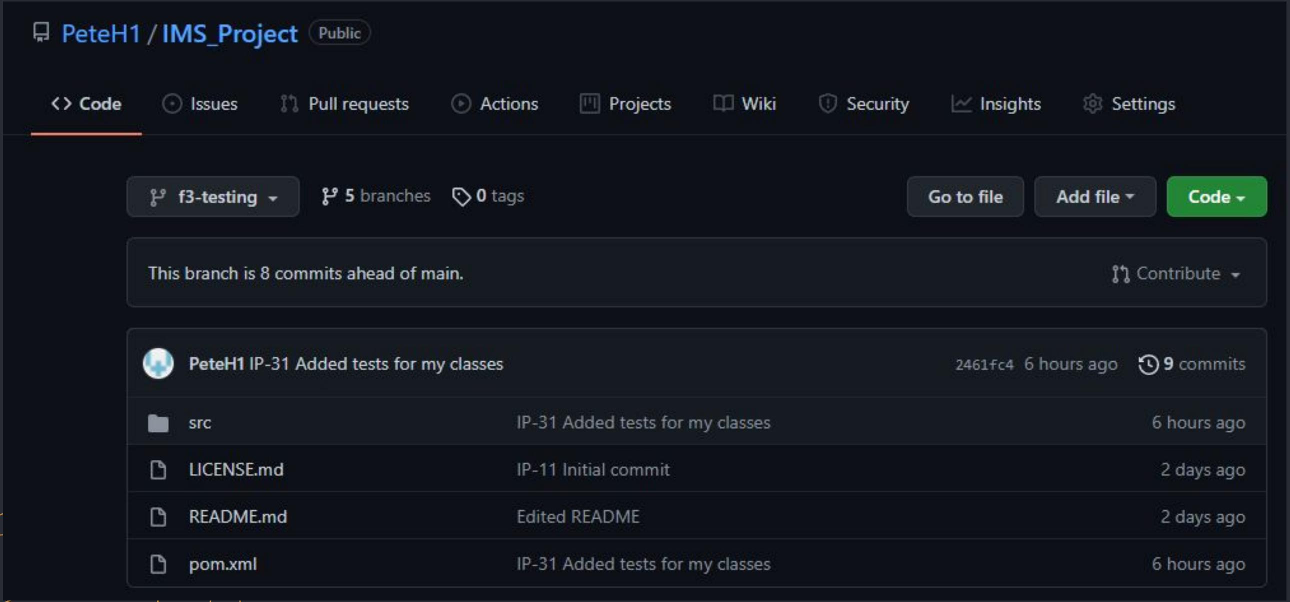
DAY 2
Setting up databases, writing core code

DAY 3
Adding core functionality, starting testing

DAY 4
Additional testing, create diagrams & documentation

DAY 5
Presentation, final changes to code & documentation

Version Control



PeteH1 / IMS_Project Public

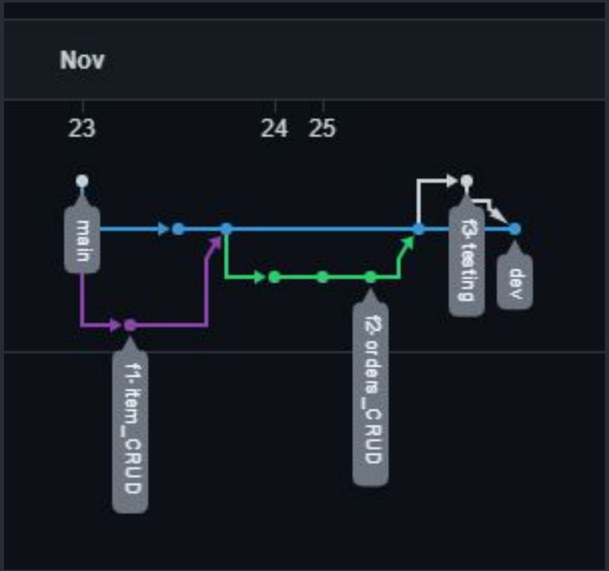
Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

f3-testing | 5 branches | 0 tags | Go to file | Add file | Code

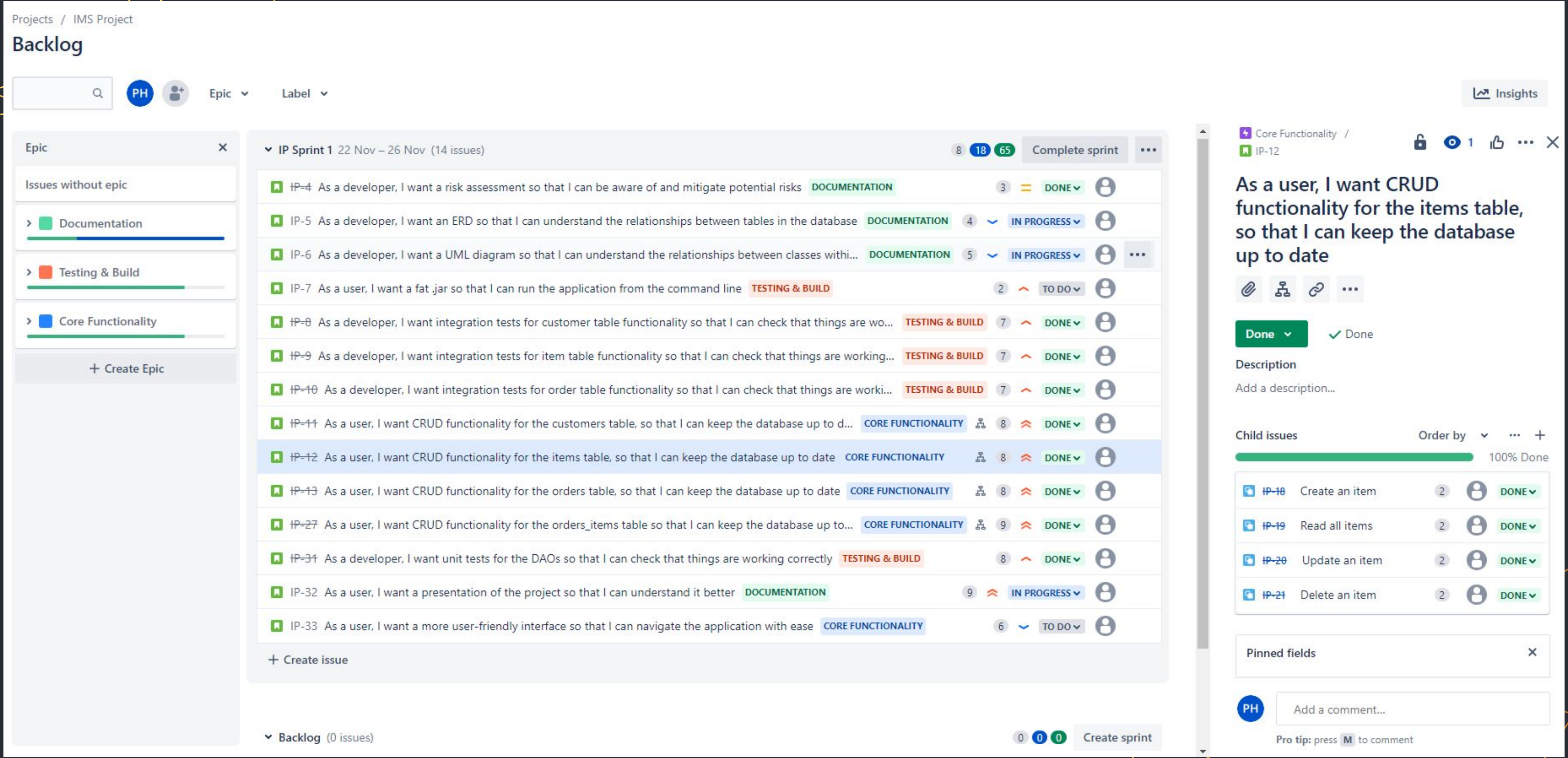
This branch is 8 commits ahead of main. Contribute

PeteH1 IP-31 Added tests for my classes 2461fc4 6 hours ago 9 commits

src	IP-31 Added tests for my classes	6 hours ago
LICENSE.md	IP-11 Initial commit	2 days ago
README.md	Edited README	2 days ago
pom.xml	IP-31 Added tests for my classes	6 hours ago



Jira



Projects / IMS Project

Backlog

PH · Epic · Label · Insights

Epic

- Issues without epic
- Documentation
- Testing & Build
- Core Functionality
- + Create Epic

IP Sprint 1 22 Nov – 26 Nov (14 issues) · 8 · 18 · 65 · Complete sprint

~~IP-4~~	As a developer, I want a risk assessment so that I can be aware of and mitigate potential risks	DOCUMENTATION	3	DONE
IP-5	As a developer, I want an ERD so that I can understand the relationships between tables in the database	DOCUMENTATION	4	IN PROGRESS
IP-6	As a developer, I want a UML diagram so that I can understand the relationships between classes withi...	DOCUMENTATION	5	IN PROGRESS
IP-7	As a user, I want a fat .jar so that I can run the application from the command line	TESTING & BUILD	2	TO DO
~~IP-8~~	As a developer, I want integration tests for customer table functionality so that I can check that things are wo...	TESTING & BUILD	7	DONE
~~IP-9~~	As a developer, I want integration tests for item table functionality so that I can check that things are working...	TESTING & BUILD	7	DONE
~~IP-10~~	As a developer, I want integration tests for order table functionality so that I can check that things are worki...	TESTING & BUILD	7	DONE
~~IP-11~~	As a user, I want CRUD functionality for the customers table, so that I can keep the database up to d...	CORE FUNCTIONALITY	8	DONE
~~IP-12~~	As a user, I want CRUD functionality for the items table, so that I can keep the database up to date	CORE FUNCTIONALITY	8	DONE
~~IP-13~~	As a user, I want CRUD functionality for the orders table, so that I can keep the database up to date	CORE FUNCTIONALITY	8	DONE
~~IP-27~~	As a user, I want CRUD functionality for the orders_items table so that I can keep the database up to...	CORE FUNCTIONALITY	9	DONE
~~IP-31~~	As a developer, I want unit tests for the DAOs so that I can check that things are working correctly	TESTING & BUILD	8	DONE
IP-32	As a user, I want a presentation of the project so that I can understand it better	DOCUMENTATION	9	IN PROGRESS
IP-33	As a user, I want a more user-friendly interface so that I can navigate the application with ease	CORE FUNCTIONALITY	6	TO DO

+ Create issue

Backlog (0 issues) · 0 · 0 · 0 · Create sprint

Core Functionality / IP-12

As a user, I want CRUD functionality for the items table, so that I can keep the database up to date

Done ✓ Done

Description

Add a description...

Child issues · Order by · 100% Done

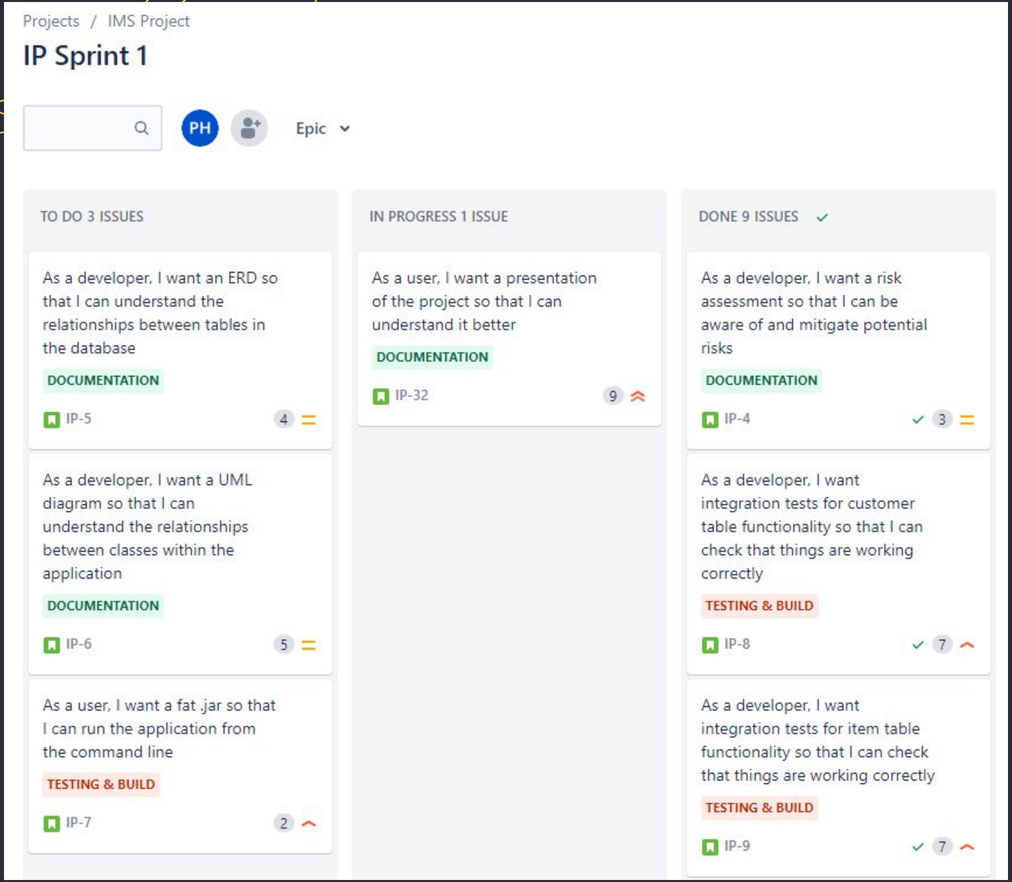
~~IP-18~~	Create an item	2	DONE
~~IP-19~~	Read all items	2	DONE
~~IP-20~~	Update an item	2	DONE
~~IP-21~~	Delete an item	2	DONE

Pinned fields

PH Add a comment...

Pro tip: press M to comment

Jira



Projects / IMS Project

IP Sprint 1

PH Epic

TO DO 3 ISSUES

As a developer, I want an ERD so that I can understand the relationships between tables in the database
DOCUMENTATION
IP-5 4

As a developer, I want a UML diagram so that I can understand the relationships between classes within the application
DOCUMENTATION
IP-6 5

As a user, I want a fat .jar so that I can run the application from the command line
TESTING & BUILD
IP-7 2

IN PROGRESS 1 ISSUE

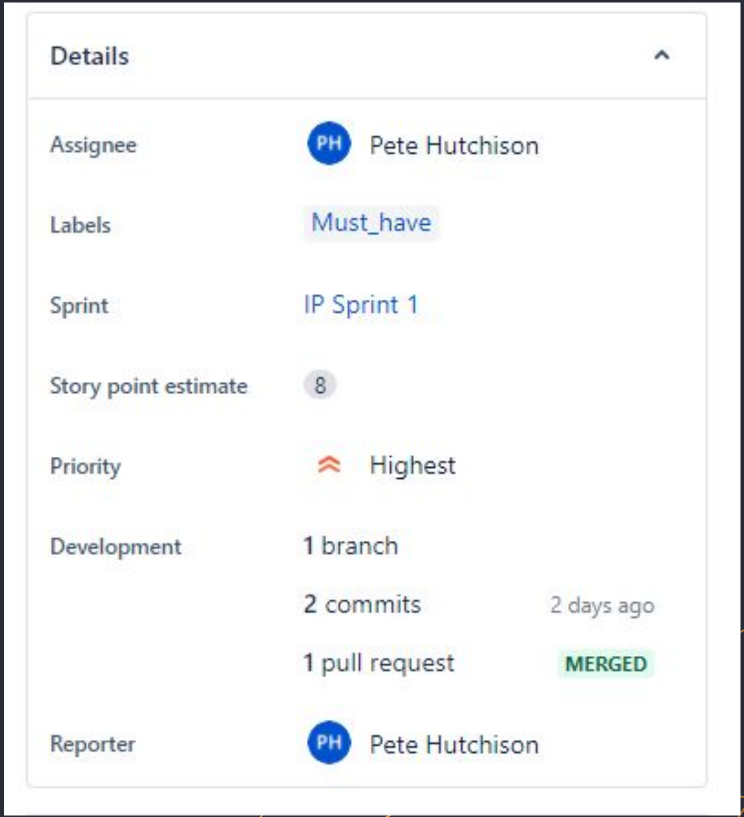
As a user, I want a presentation of the project so that I can understand it better
DOCUMENTATION
IP-32 9

DONE 9 ISSUES ✓

As a developer, I want a risk assessment so that I can be aware of and mitigate potential risks
DOCUMENTATION
IP-4 ✓ 3

As a developer, I want integration tests for customer table functionality so that I can check that things are working correctly
TESTING & BUILD
IP-8 ✓ 7

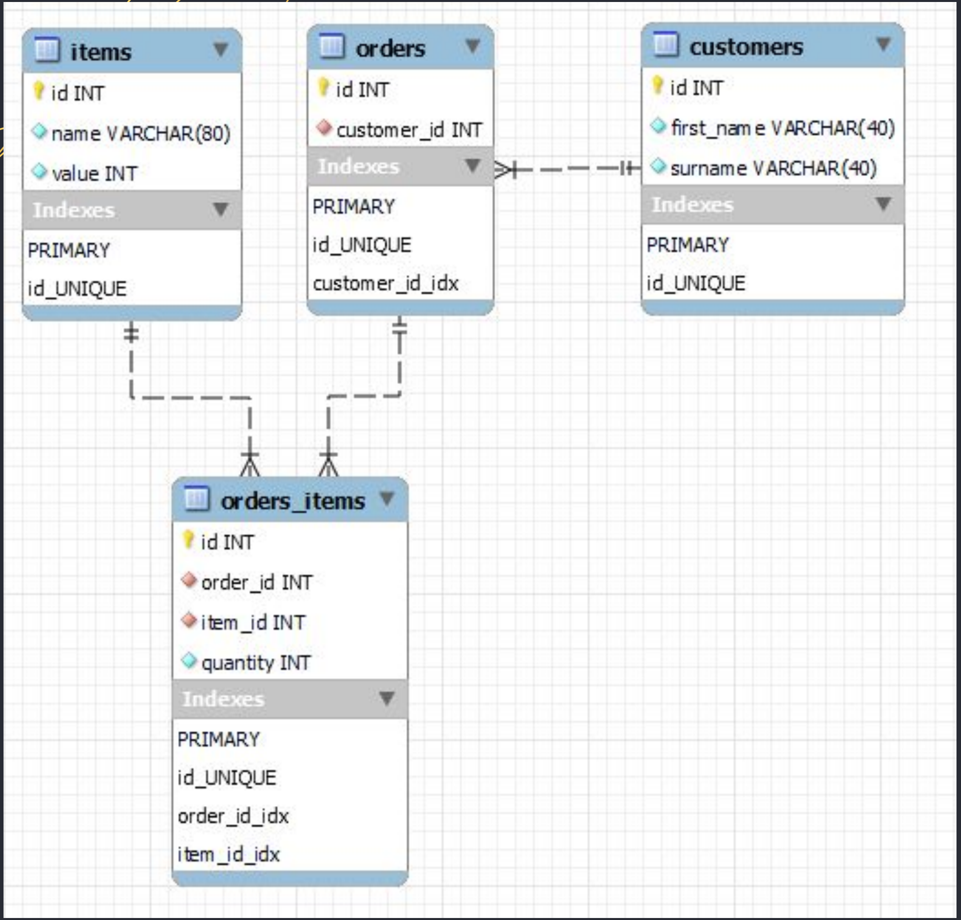
As a developer, I want integration tests for item table functionality so that I can check that things are working correctly
TESTING & BUILD
IP-9 ✓ 7



Details

Assignee	PH Pete Hutchison	
Labels	Must_have	
Sprint	IP Sprint 1	
Story point estimate	8	
Priority	Highest	
Development	1 branch	
	2 commits	2 days ago
	1 pull request	MERGED
Reporter	PH Pete Hutchison	

Project Structure



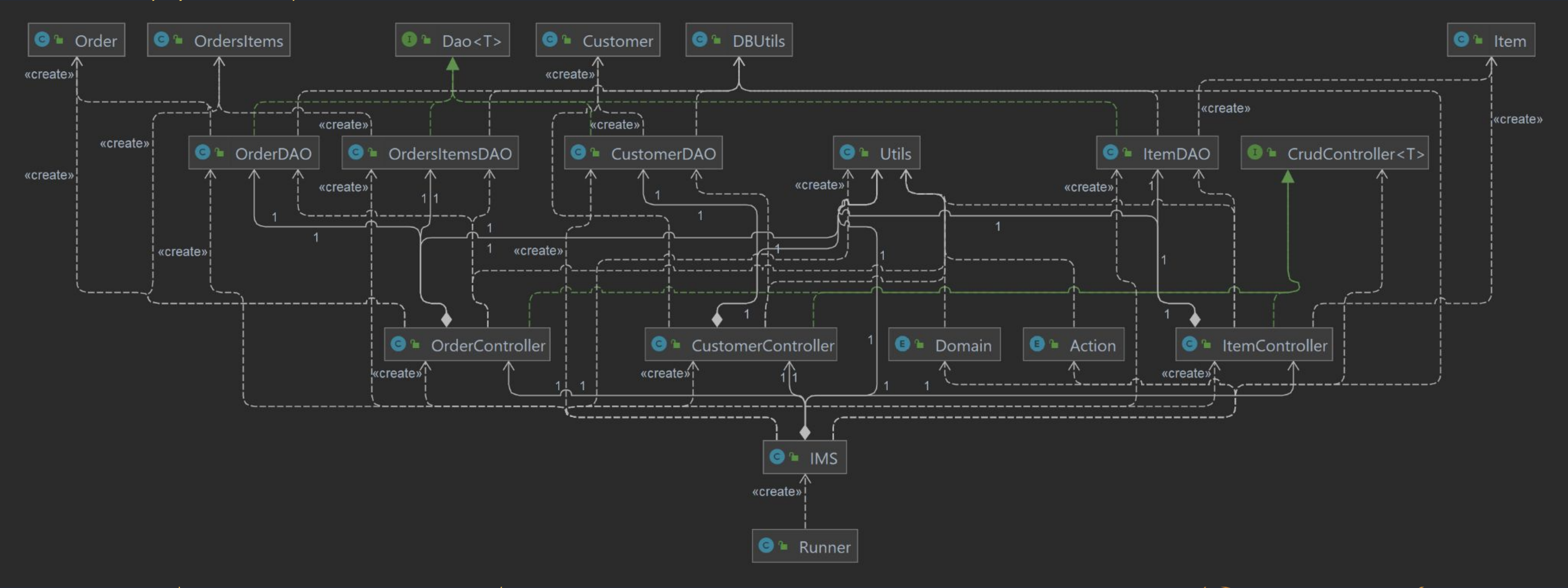
Database Structure:

Set up to meet the specification exactly.

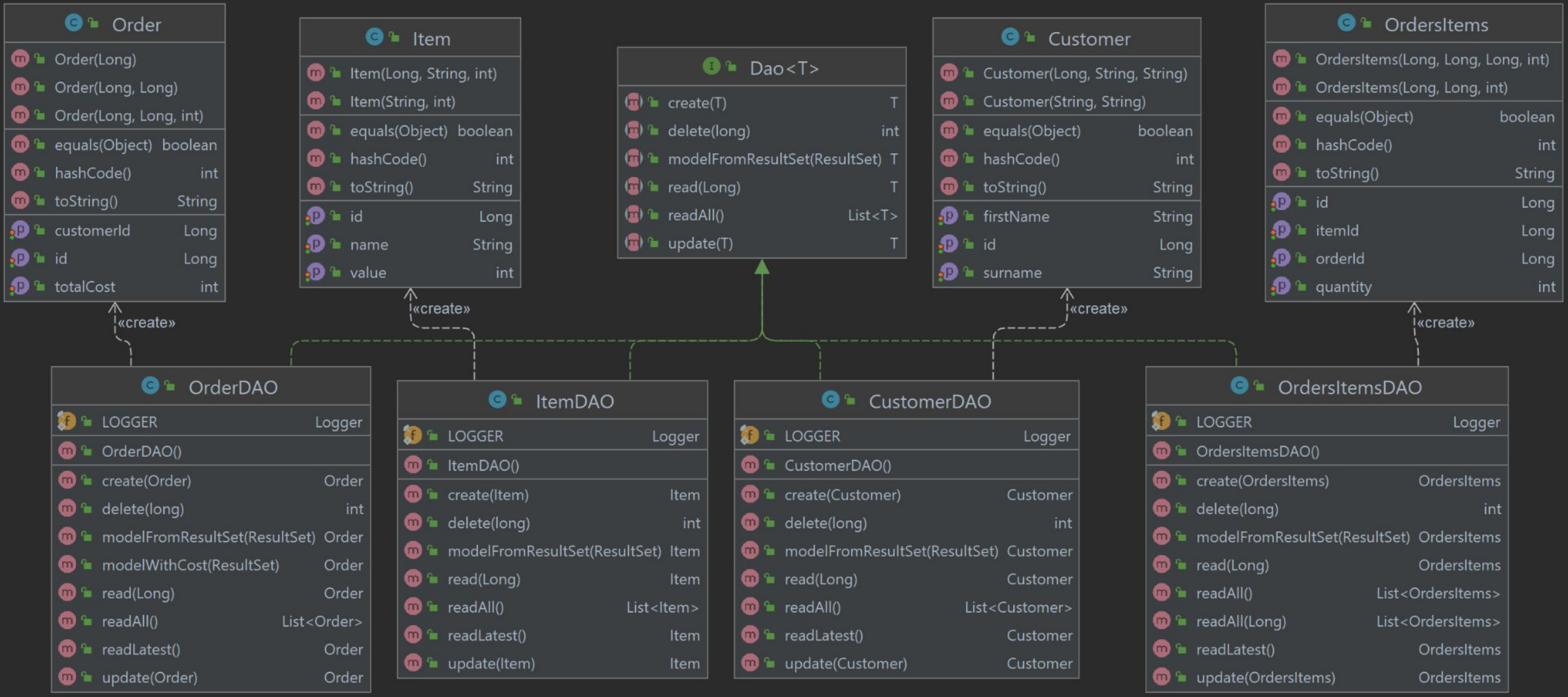
Customers table structure within project template.

orders_items table needed to handle many-to-many relationship between items & orders tables.

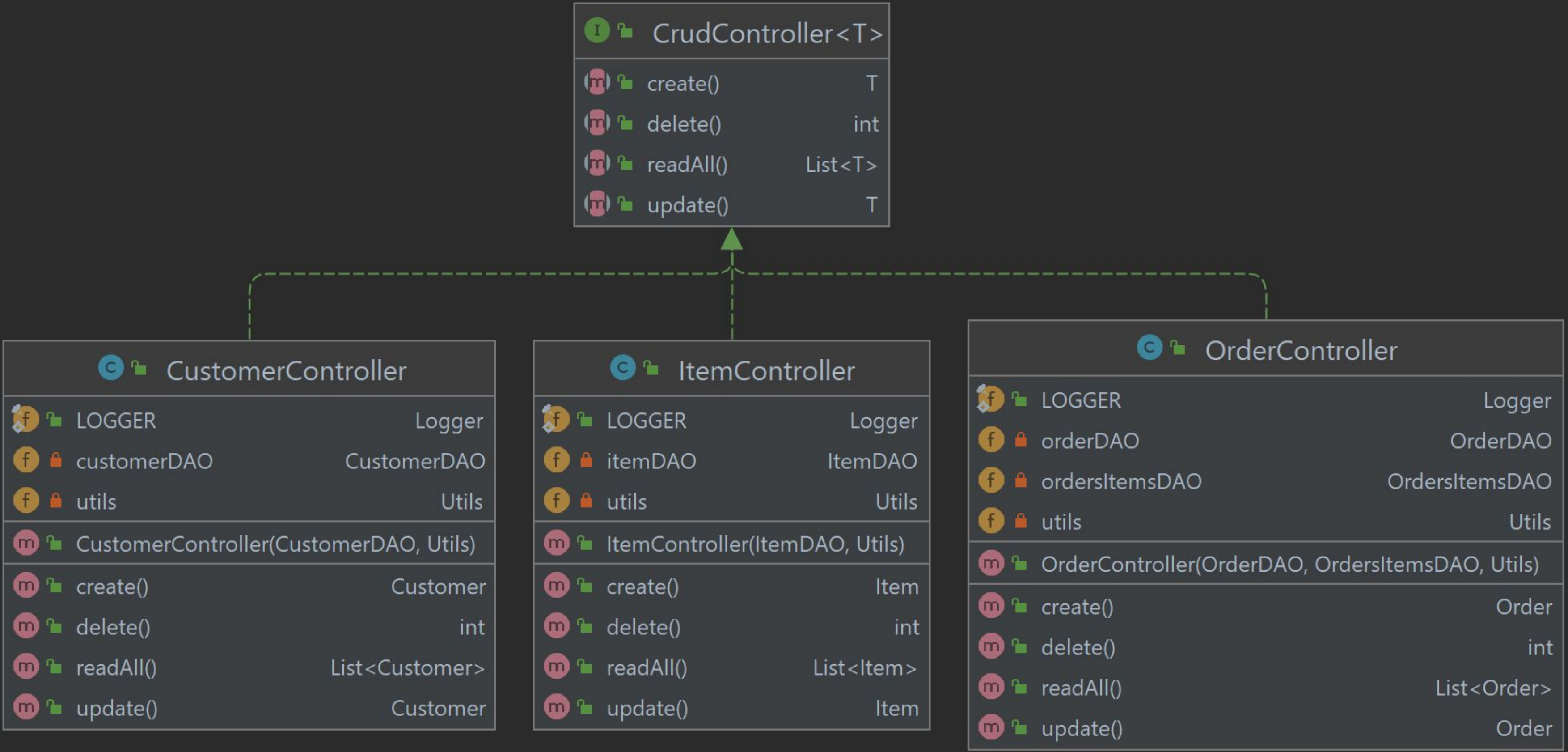
Project Structure



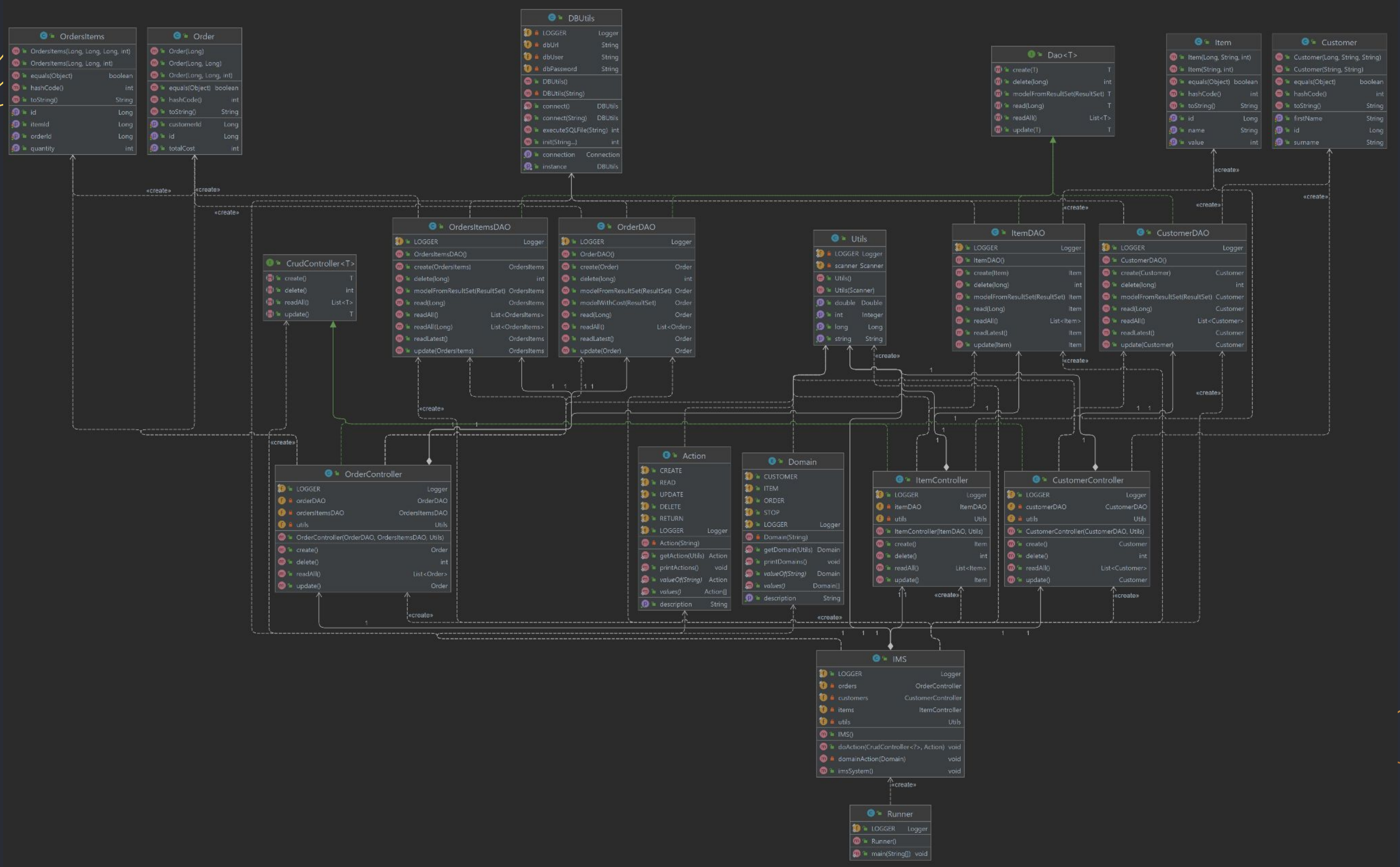
Project Structure



Project Structure



Project Structure



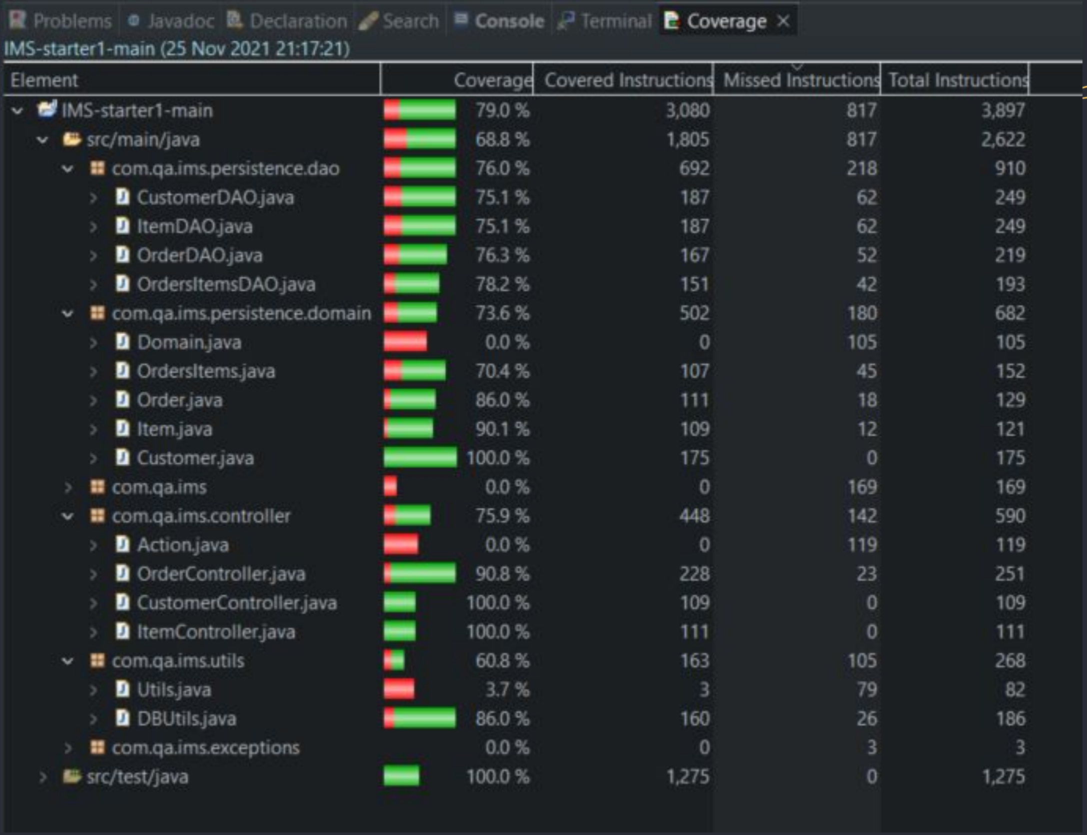
Testing
(subject to change)

Currently 69% test coverage

All core functionality tested

Tests return no errors or failures

Some classes from the template are untested, as are some exception catch blocks within DAOs - this might change later today.

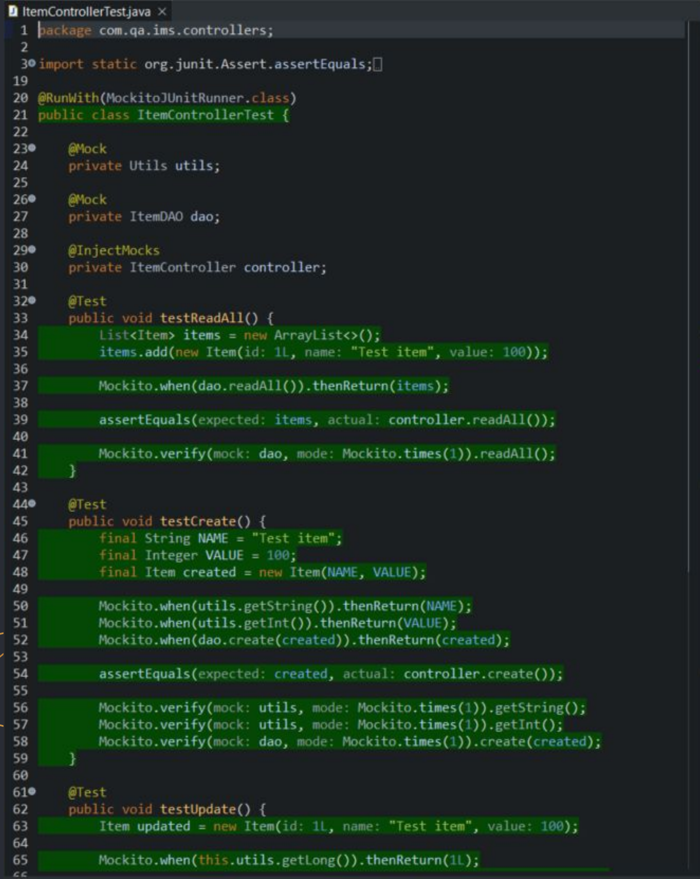


IMS-starter1-main (25 Nov 2021 21:17:21)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
IMS-starter1-main	79.0 %	3,080	817	3,897
src/main/java	68.8 %	1,805	817	2,622
com.qa.ims.persistence.dao	76.0 %	692	218	910
CustomerDAO.java	75.1 %	187	62	249
ItemDAO.java	75.1 %	187	62	249
OrderDAO.java	76.3 %	167	52	219
OrdersItemsDAO.java	78.2 %	151	42	193
com.qa.ims.persistence.domain	73.6 %	502	180	682
Domain.java	0.0 %	0	105	105
OrdersItems.java	70.4 %	107	45	152
Order.java	86.0 %	111	18	129
Item.java	90.1 %	109	12	121
Customer.java	100.0 %	175	0	175
com.qa.ims	0.0 %	0	169	169
com.qa.ims.controller	75.9 %	448	142	590
Action.java	0.0 %	0	119	119
OrderController.java	90.8 %	228	23	251
CustomerController.java	100.0 %	109	0	109
ItemController.java	100.0 %	111	0	111
com.qa.ims.utils	60.8 %	163	105	268
Utils.java	3.7 %	3	79	82
DBUtils.java	86.0 %	160	26	186
com.qa.ims.exceptions	0.0 %	0	3	3
src/test/java	100.0 %	1,275	0	1,275

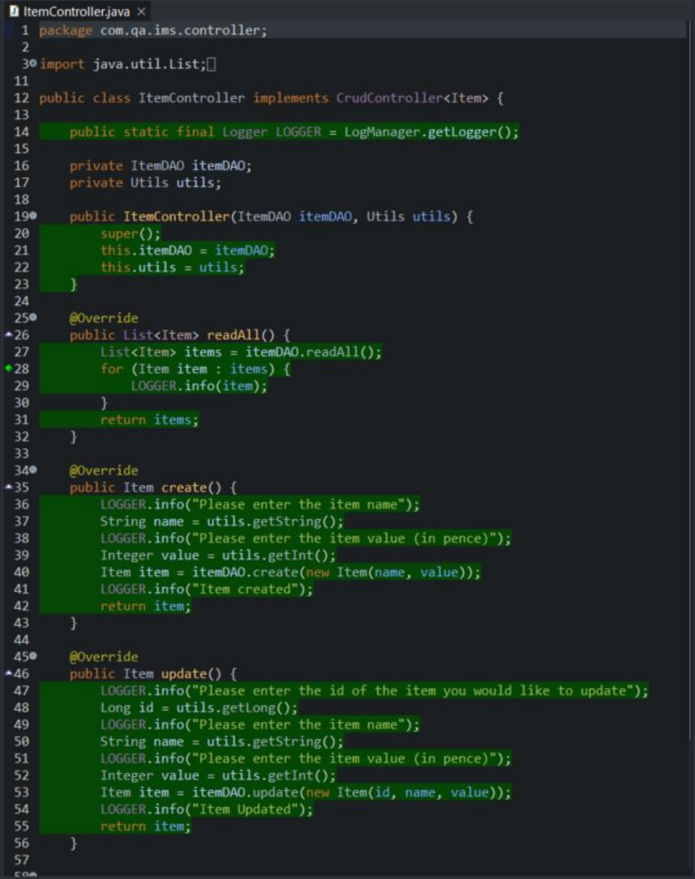
Testing

(subject to change)



ItemControllerTest.java

```java
package com.qa.ims.controllers;

import static org.junit.Assert.assertEquals;

@RunWith(MockitoJUnitRunner.class)
public class ItemControllerTest {

    @Mock
    private Utils utils;

    @Mock
    private ItemDAO dao;

    @InjectMocks
    private ItemController controller;

    @Test
    public void testReadAll() {
        List<Item> items = new ArrayList<>();
        items.add(new Item(id: 1L, name: "Test item", value: 100));

        Mockito.when(dao.readAll()).thenReturn(items);

        assertEquals(expected: items, actual: controller.readAll());

        Mockito.verify(mock: dao, mode: Mockito.times(1)).readAll();
    }

    @Test
    public void testCreate() {
        final String NAME = "Test item";
        final Integer VALUE = 100;
        final Item created = new Item(NAME, VALUE);

        Mockito.when(utils.getString()).thenReturn(NAME);
        Mockito.when(utils.getInt()).thenReturn(VALUE);
        Mockito.when(dao.create(created)).thenReturn(created);

        assertEquals(expected: created, actual: controller.create());

        Mockito.verify(mock: utils, mode: Mockito.times(1)).getString();
        Mockito.verify(mock: utils, mode: Mockito.times(1)).getInt();
        Mockito.verify(mock: dao, mode: Mockito.times(1)).create(created);
    }

    @Test
    public void testUpdate() {
        Item updated = new Item(id: 1L, name: "Test item", value: 100);

        Mockito.when(this.utils.getLong()).thenReturn(1L);
```



ItemController.java

```java
package com.qa.ims.controller;

import java.util.List;

public class ItemController implements CrudController<Item> {

    public static final Logger LOGGER = LogManager.getLogger();

    private ItemDAO itemDAO;
    private Utils utils;

    public ItemController(ItemDAO itemDAO, Utils utils) {
        super();
        this.itemDAO = itemDAO;
        this.utils = utils;
    }

    @Override
    public List<Item> readAll() {
        List<Item> items = itemDAO.readAll();
        for (Item item : items) {
            LOGGER.info(item);
        }
        return items;
    }

    @Override
    public Item create() {
        LOGGER.info("Please enter the item name");
        String name = utils.getString();
        LOGGER.info("Please enter the item value (in pence)");
        Integer value = utils.getInt();
        Item item = itemDAO.create(new Item(name, value));
        LOGGER.info("Item created");
        return item;
    }

    @Override
    public Item update() {
        LOGGER.info("Please enter the id of the item you would like to update");
        Long id = utils.getLong();
        LOGGER.info("Please enter the item name");
        String name = utils.getString();
        LOGGER.info("Please enter the item value (in pence)");
        Integer value = utils.getInt();
        Item item = itemDAO.update(new Item(id, name, value));
        LOGGER.info("Item Updated");
        return item;
    }

```

Demonstration

Sprint review

What did I complete?

On course to complete everything in the deliverables checklist (MVP) for the project.

Core functionality & testing achieved using tools required by project specification.

Some documentation (including README) and a final build to be submitted later today.

What got left behind?

User-friendliness (particularly visual aspect) of application was not improved beyond initial template.

Additional functionality: extra information in tables, adding items to orders on creation, etc. (I did not include this in the sprint, however)

Sprint retrospective

What went well?

Use of Jira to organise work - regularly referring back to deliverables checklist to make sure I was on track to complete the project.

Use of various resources to help solve problems.

Writing & troubleshooting tests was a smoother process than writing the core functionality for the application.

What could be improved?

Dedicate more time to viewing and understanding code when working on existing projects in future.

Dedicate more time to choosing the structure of my code.

Conclusion

Pleased to have met the targets set out within the deliverables checklist.

The project has been a valuable learning experience.

I look forward to bringing what I've learnt to the next project!

THANKS!

DO YOU HAVE ANY QUESTIONS?

petehutchison.atlassian.net

github.com/PeteH1/IMS_Project