To Do List Project

CLAES ALFONSO

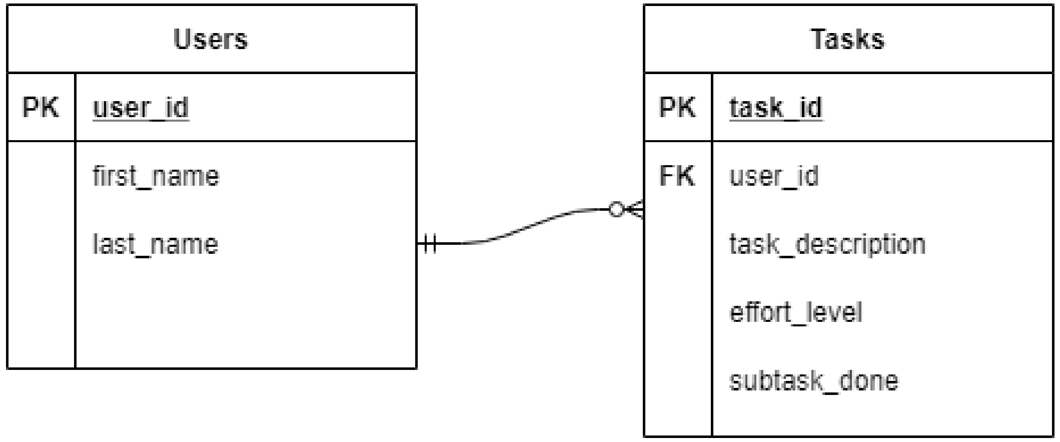
Introduction

- **Project Objective:** Develop a web application using HTML and JavaScript to interact with a SQL database through a Java based API.
- The following technologies were used for this project:
  - **Version Control System**: Git
  - **Source Code Management**: GitHub
  - **Project Management** : Jira
  - **Database Management System**: MySQL (Spring Boot H2 instance)
  - **Back-End Programming Language**: Java
  - **Front-End Programming Language**: HTML, JavaScript
  - **Build Tool**: Maven
  - **Unit Testing**: JUnit
  - **Automated Testing**: Selenium

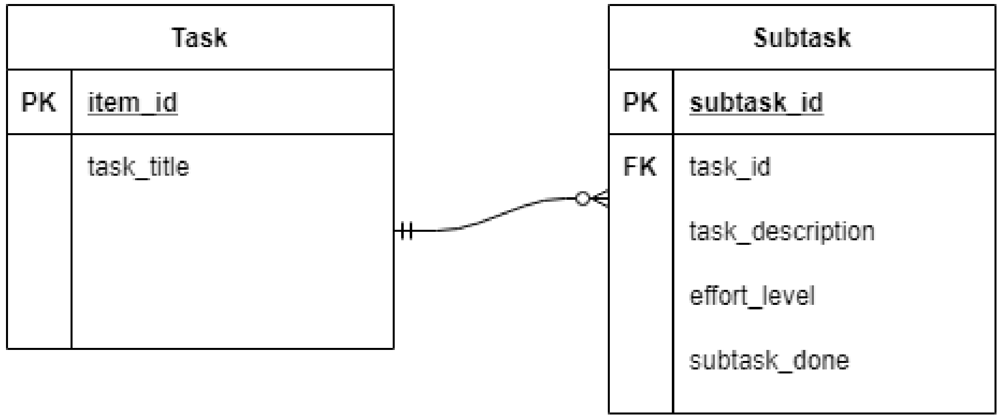
Modelling the SQL Database

- An Entity Relationship Diagram was used to model the relationships between database tables and entries.
- Initially was going to model a User with a list of tasks
- Chose Tasks and Subtasks to simplifiy

Initial ERD diagram

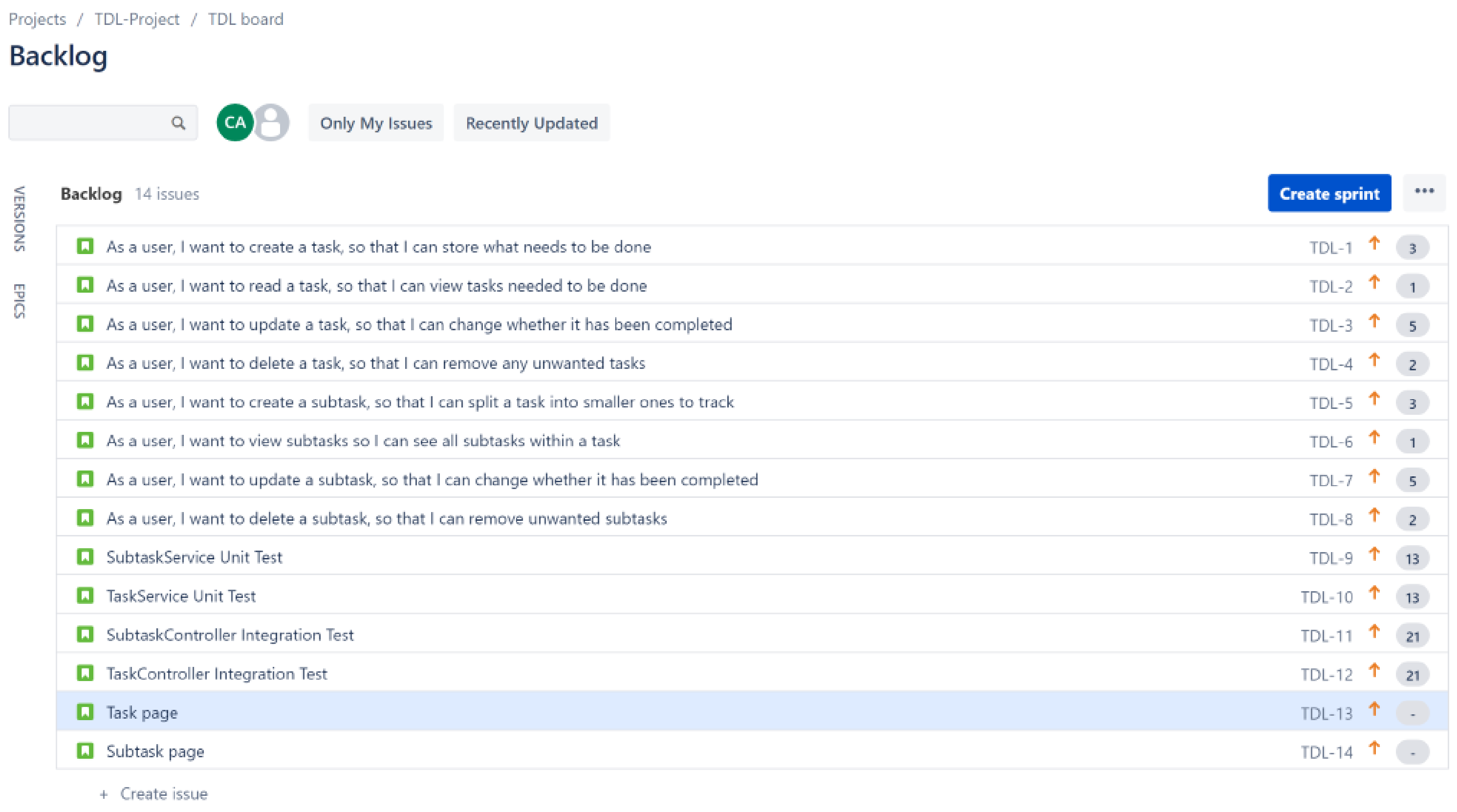


Final ERD diagram



Project Management Overview

- Project was completed over two sprints
- First sprint focused on the back end development
- Second sprint focused on the front end and integration with the back end

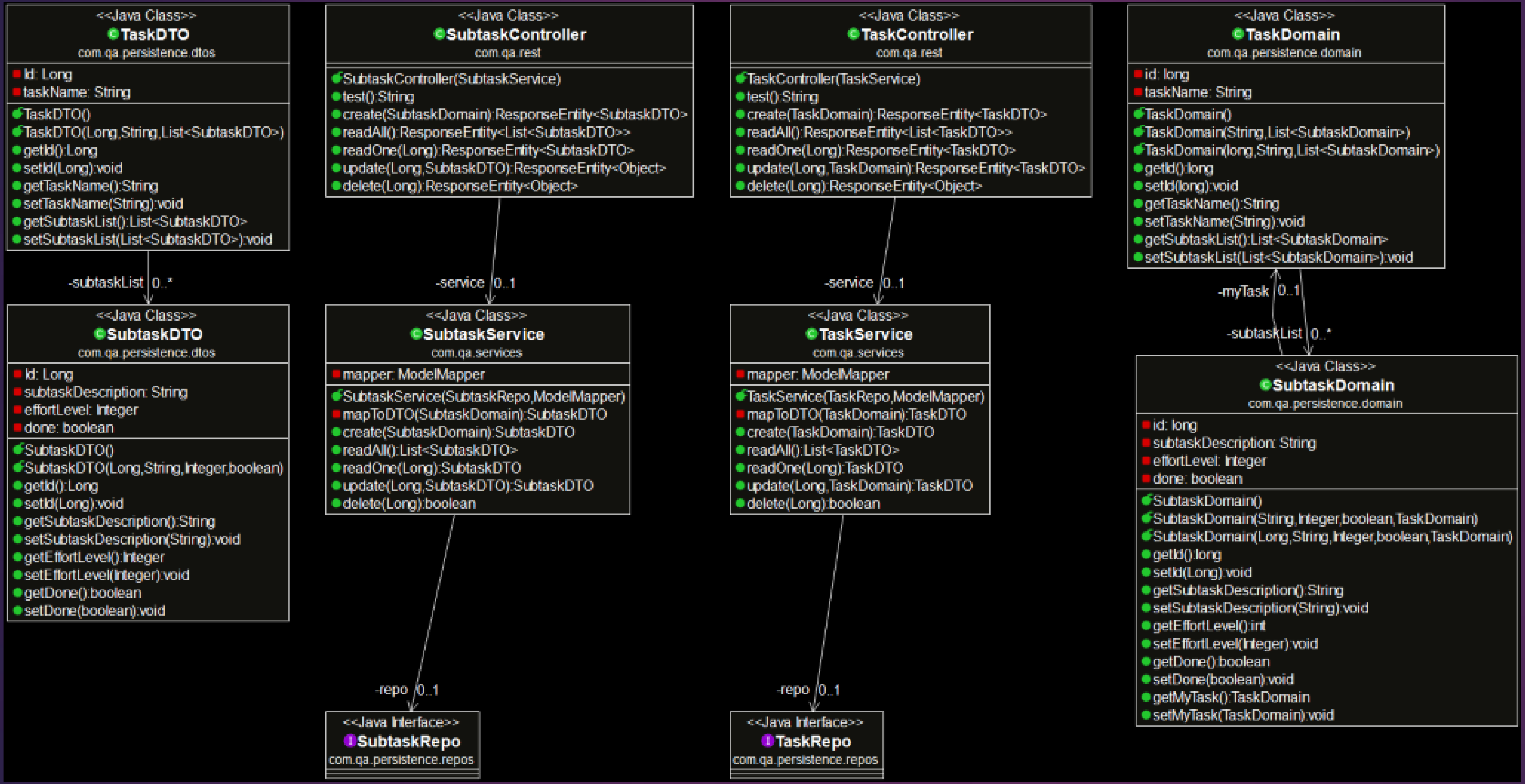

Projects / TDL-Project / TDL board

Backlog

Backlog 14 issues

As a user, I want to create a task, so that I can store what needs to be done	TDL-1	↑	3
As a user, I want to read a task, so that I can view tasks needed to be done	TDL-2	↑	1
As a user, I want to update a task, so that I can change whether it has been completed	TDL-3	↑	5
As a user, I want to delete a task, so that I can remove any unwanted tasks	TDL-4	↑	2
As a user, I want to create a subtask, so that I can split a task into smaller ones to track	TDL-5	↑	3
As a user, I want to view subtasks so I can see all subtasks within a task	TDL-6	↑	1
As a user, I want to update a subtask, so that I can change whether it has been completed	TDL-7	↑	5
As a user, I want to delete a subtask, so that I can remove unwanted subtasks	TDL-8	↑	2
SubtaskService Unit Test	TDL-9	↑	13
TaskService Unit Test	TDL-10	↑	13
SubtaskController Integration Test	TDL-11	↑	21
TaskController Integration Test	TDL-12	↑	21
Task page	TDL-13	↑	-
Subtask page	TDL-14	↑	-

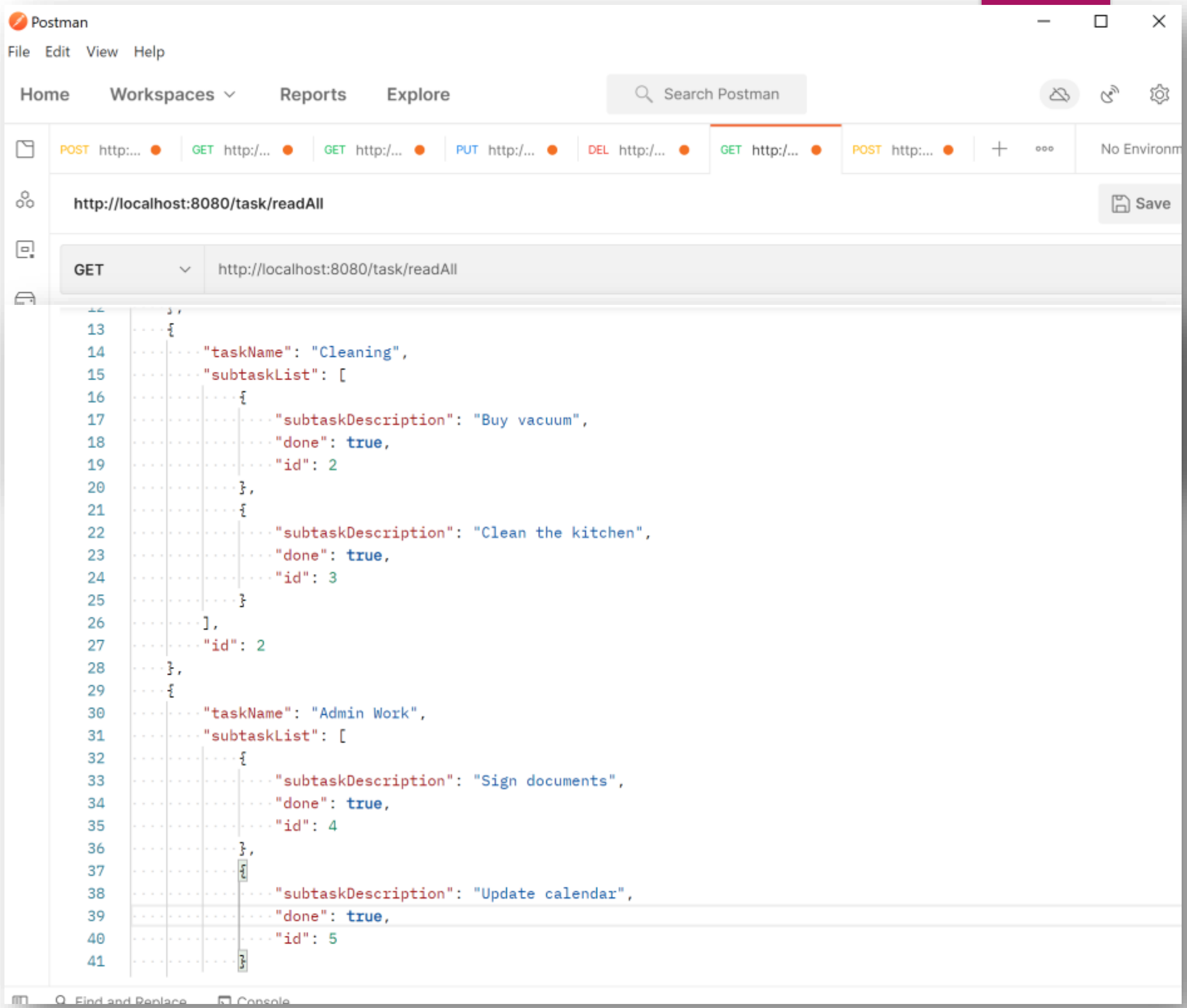
+ Create issue


<<Java Class>>
TaskDTO
com.qa.persistence.dtos
ld: Long
taskName: String
TaskDTO()
TaskDTO(Long,String,List<SubtaskDTO>)
getId():Long
setId(Long):void
getTaskName():String
setTaskName(String):void
getSubtaskList():List<SubtaskDTO>
setSubtaskList(List<SubtaskDTO>):void
-subtaskList 0..*
<<Java Class>>
SubtaskDTO
com.qa.persistence.dtos
ld: Long
subtaskDescription: String
effortLevel: Integer
done: boolean
SubtaskDTO()
SubtaskDTO(Long,String,Integer,boolean)
getId():Long
setId(Long):void
getSubtaskDescription():String
setSubtaskDescription(String):void
getEffortLevel():Integer
setEffortLevel(Integer):void
getDone():boolean
setDone(boolean):void
<<Java Class>>
SubtaskController
com.qa.rest
SubtaskController(SubtaskService)
test():String
create(SubtaskDomain):ResponseEntity<SubtaskDTO>
readAll():ResponseEntity<List<SubtaskDTO>>
readOne(Long):ResponseEntity<SubtaskDTO>
update(Long,SubtaskDTO):ResponseEntity<Object>
delete(Long):ResponseEntity<Object>
-service 0..1
<<Java Class>>
SubtaskService
com.qa.services
mapper: ModelMapper
SubtaskService(SubtaskRepo,ModelMapper)
mapToDTO(SubtaskDomain):SubtaskDTO
create(SubtaskDomain):SubtaskDTO
readAll():List<SubtaskDTO>
readOne(Long):SubtaskDTO
update(Long,SubtaskDTO):SubtaskDTO
delete(Long):boolean
-repo 0..1
<<Java Interface>>
SubtaskRepo
com.qa.persistence.repos
<<Java Class>>
TaskController
com.qa.rest
TaskController(TaskService)
test():String
create(TaskDomain):ResponseEntity<TaskDTO>
readAll():ResponseEntity<List<TaskDTO>>
readOne(Long):ResponseEntity<TaskDTO>
update(Long,TaskDomain):ResponseEntity<TaskDTO>
delete(Long):ResponseEntity<Object>
-service 0..1
<<Java Class>>
TaskService
com.qa.services
mapper: ModelMapper
TaskService(TaskRepo,ModelMapper)
mapToDTO(TaskDomain):TaskDTO
create(TaskDomain):TaskDTO
readAll():List<TaskDTO>
readOne(Long):TaskDTO
update(Long,TaskDomain):TaskDTO
delete(Long):boolean
-repo 0..1
<<Java Interface>>
TaskRepo
com.qa.persistence.repos
<<Java Class>>
TaskDomain
com.qa.persistence.domain
id: long
taskName: String
TaskDomain()
TaskDomain(String,List<SubtaskDomain>)
TaskDomain(long,String,List<SubtaskDomain>)
getId():long
setId(long):void
getTaskName():String
setTaskName(String):void
getSubtaskList():List<SubtaskDomain>
setSubtaskList(List<SubtaskDomain>):void
-myTask 0..1
-subtaskList 0..*
<<Java Class>>
SubtaskDomain
com.qa.persistence.domain
id: long
subtaskDescription: String
effortLevel: Integer
done: boolean
SubtaskDomain()
SubtaskDomain(String,Integer,boolean,TaskDomain)
SubtaskDomain(Long,String,Integer,boolean,TaskDomain)
getId():long
setId(Long):void
getSubtaskDescription():String
setSubtaskDescription(String):void
getEffortLevel():int
setEffortLevel(Integer):void
getDone():boolean
setDone(boolean):void
getMyTask():TaskDomain
setMyTask(TaskDomain):void

UML Diagram: Sprint 1

Microtesting

- Used Postman to test the back end API
- This made it easier to check if requests were working as they should with the API



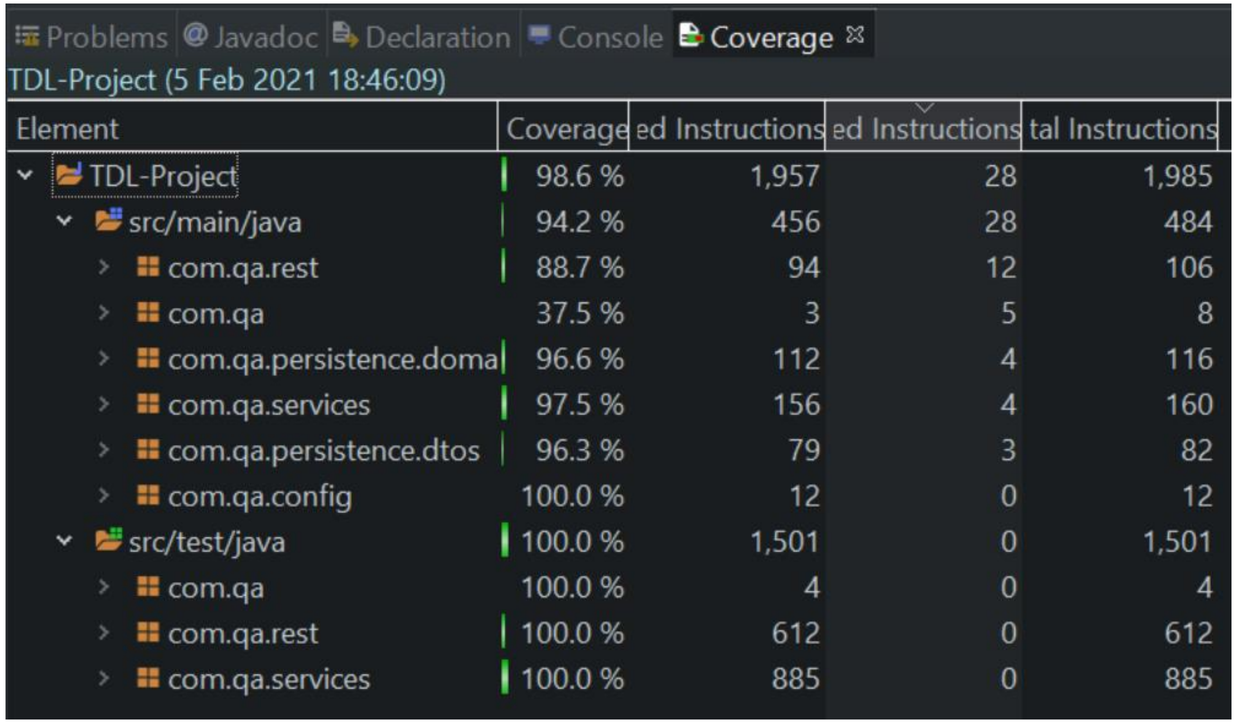
http://localhost:8080/task/readAll

GET http://localhost:8080/task/readAll

```
    {
        "taskName": "Cleaning",
        "subtaskList": [
            {
                "subtaskDescription": "Buy vacuum",
                "done": true,
                "id": 2
            },
            {
                "subtaskDescription": "Clean the kitchen",
                "done": true,
                "id": 3
            }
        ],
        "id": 2
    },
    {
        "taskName": "Admin Work",
        "subtaskList": [
            {
                "subtaskDescription": "Sign documents",
                "done": true,
                "id": 4
            },
            {
                "subtaskDescription": "Update calendar",
                "done": true,
                "id": 5
            }
```

End of Sprint 1

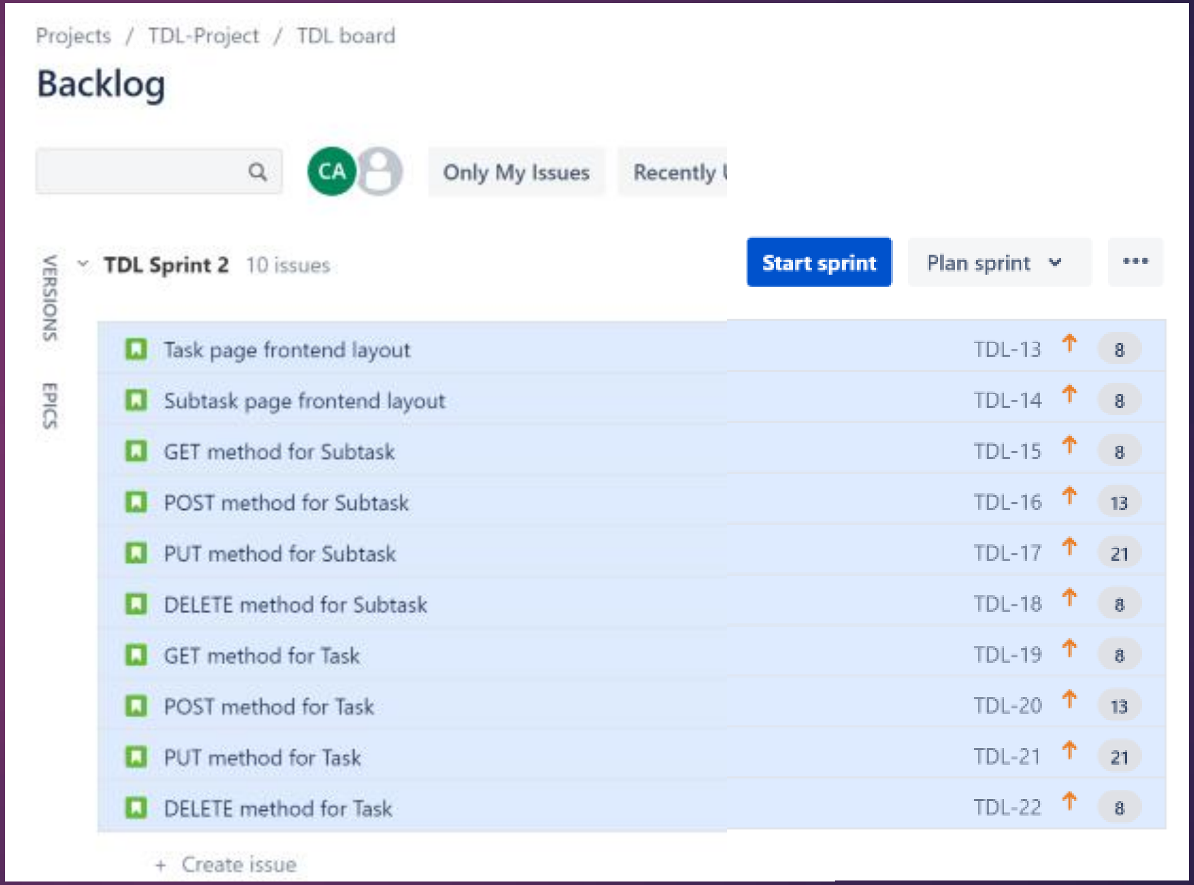
- By the end of Sprint 1, I had completed almost all backend development and testing
- Test coverage above 94%


Problems | Javadoc | Declaration | Console | Coverage

TDL-Project (5 Feb 2021 18:46:09)

Element	Coverage	ed Instructions	ed Instructions	tal Instructions
TDL-Project	98.6 %	1,957	28	1,985
src/main/java	94.2 %	456	28	484
com.qa.rest	88.7 %	94	12	106
com.qa	37.5 %	3	5	8
com.qa.persistence.doma	96.6 %	112	4	116
com.qa.services	97.5 %	156	4	160
com.qa.persistence.dtos	96.3 %	79	3	82
com.qa.config	100.0 %	12	0	12
src/test/java	100.0 %	1,501	0	1,501
com.qa	100.0 %	4	0	4
com.qa.rest	100.0 %	612	0	612
com.qa.services	100.0 %	885	0	885

Start of Sprint 2



Projects / TDL-Project / TDL board

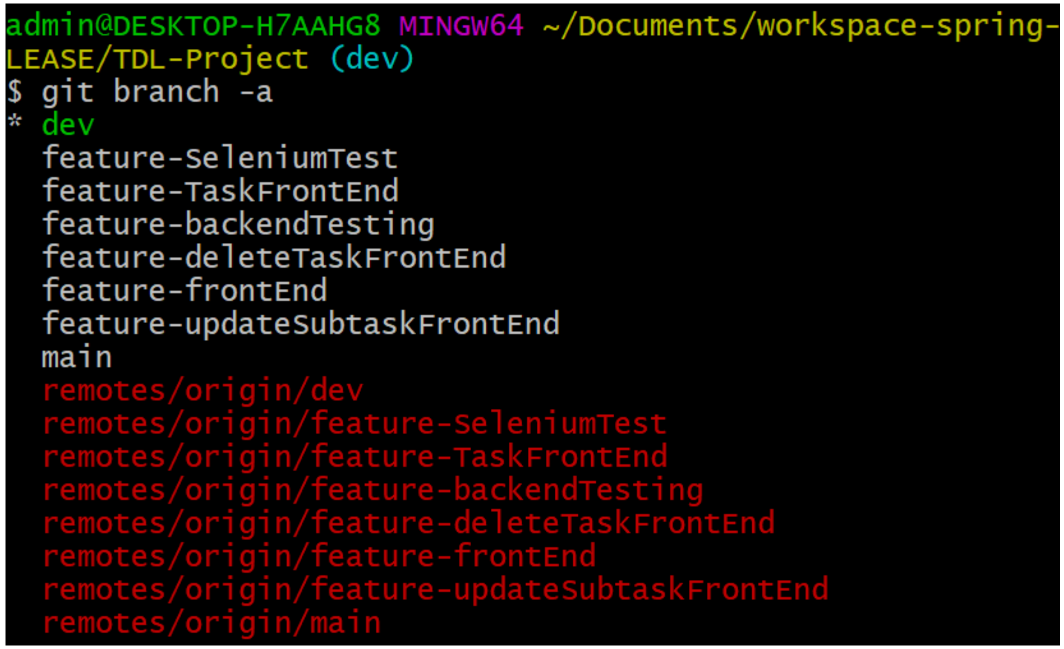
Backlog

TDL Sprint 2 10 issues

Task page frontend layout	TDL-13	8
Subtask page frontend layout	TDL-14	8
GET method for Subtask	TDL-15	8
POST method for Subtask	TDL-16	13
PUT method for Subtask	TDL-17	21
DELETE method for Subtask	TDL-18	8
GET method for Task	TDL-19	8
POST method for Task	TDL-20	13
PUT method for Task	TDL-21	21
DELETE method for Task	TDL-22	8

Continuous Integration

- Git used for version control.
- Regular commits and pushes were used to mitigate losing work.
- The feature-branch model was used to separate certain functions into different branches.
- Project pushed to a GitHub repository

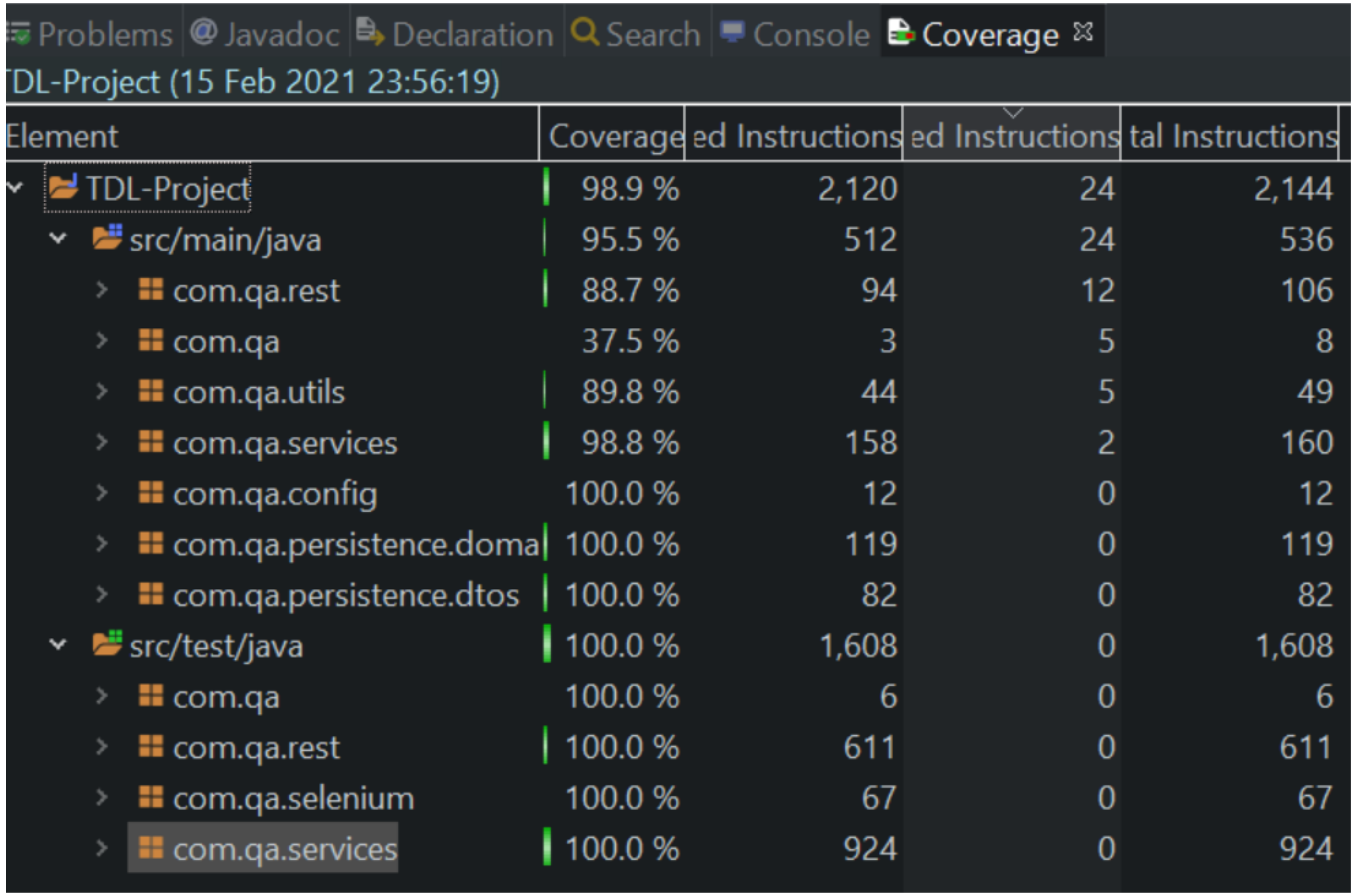


```
admin@DESKTOP-H7AAHG8 MINGW64 ~/Documents/workspace-spring-
LEASE/TDL-Project (dev)
$ git branch -a
* dev
  feature-SeleniumTest
  feature-TaskFrontEnd
  feature-backendTesting
  feature-deleteTaskFrontEnd
  feature-frontEnd
  feature-updateSubtaskFrontEnd
  main
  remotes/origin/dev
  remotes/origin/feature-SeleniumTest
  remotes/origin/feature-TaskFrontEnd
  remotes/origin/feature-backendTesting
  remotes/origin/feature-deleteTaskFrontEnd
  remotes/origin/feature-frontEnd
  remotes/origin/feature-updateSubtaskFrontEnd
  remotes/origin/main
```

Live Demonstration

Testing

- 95.5% coverage
- Integration tests helped the most with increasing test coverage



TDL-Project (15 Feb 2021 23:56:19)

Element	Coverage	ed Instructions	ed Instructions	tal Instructions
TDL-Project	98.9 %	2,120	24	2,144
src/main/java	95.5 %	512	24	536
com.qa.rest	88.7 %	94	12	106
com.qa	37.5 %	3	5	8
com.qa.utils	89.8 %	44	5	49
com.qa.services	98.8 %	158	2	160
com.qa.config	100.0 %	12	0	12
com.qa.persistence.doma	100.0 %	119	0	119
com.qa.persistence.dtos	100.0 %	82	0	82
src/test/java	100.0 %	1,608	0	1,608
com.qa	100.0 %	6	0	6
com.qa.rest	100.0 %	611	0	611
com.qa.selenium	100.0 %	67	0	67
com.qa.services	100.0 %	924	0	924

Project Sprint Review

MoSCoW Approach

Must Have	Should have	Could have	Won't have this time
Basic CRUD backend functionality ✔	Friendly user interface ✔	Total effort amount for a Task ✗	Task categories/tags ✗
Task name ✔	Minimum 80% test coverage ✔	Move Subtask to another Task ✗	User accounts with login ✗
Subtask description ✔			
Functional front end to interact with API and database ✔			

Conclusion - Sprint Retrospective

What Went Well:	What could be improved:
Completed a working application with all MVP requirements and additional functionality.	Keep track of Jira board daily
Completing some difficult methods independently.	Cleaner code, separate large functions into smaller pieces
Understanding of HTML and JavaScript development	Using branches for intended features

Thank you for listening

Questions?