Tests Final Assignment - Gutenberg Application

Zygimantas Pranka
Plamen Getsov
Emil Klausen
Diana Strele

Cphbusiness Computer Science
2018

Table of Content

Requirements

- ## User stories

Purpose of a user story is to get a clear and high-level understanding of the requirement from the given task. This makes it easier for the user to give a specific requirement and for the developer to meet this requirement without having to “guess” various parts which can end up in misunderstanding from both parts.

For user and for us as business User Stories will solve the problem of wasting time on developing something that isn't required, but was miscommunicated and unclear. When a user story is precise and clear, it reduces the chance of misunderstanding therefore time and money is saved and work is done faster. As well as products developed can be presented and approved faster, making the whole development process smoother.

- User Story 1

As a user
I want to be able to search for city mentioned in the book,
So that I can find all book titles and authors with the given city name mentioned in the book.

- User Story 2

As a user,
I want to be able to get all the cities from a given book
So that I can plot it onto a map

- User Story 3

As a user
I want to search for an author
So that I get a map with all cities mentioned in the authors books

- User Story 4

As a user
I want to to search for a geolocation
So that I can get all the books in the vicinity of the given geolocation

Before releasing feature, it needs to go through Acceptance Testing. Acceptance Tests goes under the Behaviour Test Development as the developers need to know what tests have to pass in order to call development to be done. Acceptance tests are tests that describes the system behaviour and / or functionality that needs to be passed in order to determine the system or functionality to be finished. These tests focuses more individually on the app, as in our case we have acceptance tests for the User Story 1 as see below.

1. Acceptance Criteria:

As a user, I expect to get back a list with all book titles with corresponding authors that mention this city.

2. Acceptance Test Case:

When searching for London, I get back Charles Dickens book list containing ‘Oliver Twist’.

3. Scenario: A user searches for a city, which returns all book titles and corresponding authors that mentions the cities

Arrange

Given: Opening the project page

Act

When: Entering a <city>

Assert

Then: I should get a book related to that city in result

★ *Note: see the User Stories 2,3 and 4 Acceptance Criteria in Appendix.*

During the Sprint iterations we can measure how far have we come with the given requirements of a User Story. Therefore we have created a Product Backlog as seen below.

	Story	Estimation	Priority
1.	As a user I want to be able to search a book by its title	10h	1
2.	As a user I want to search books by author	6h	2
3.	As a user I want to search for books with certain cities	6h	3
4.	As a user I want to see all cities mentioned by author (In his books)	6h	4
5.	As a user I want to get all the locations in a book in a map	5h	5

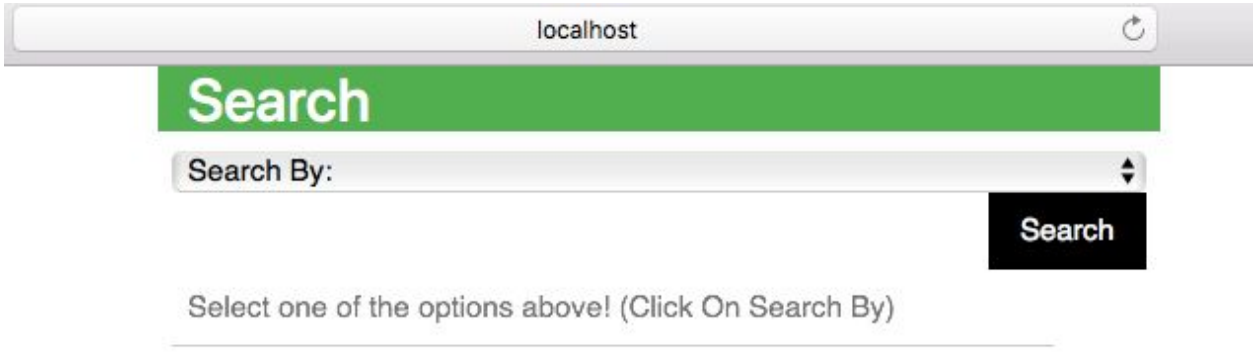
Product Backlog of User Stories

System Design

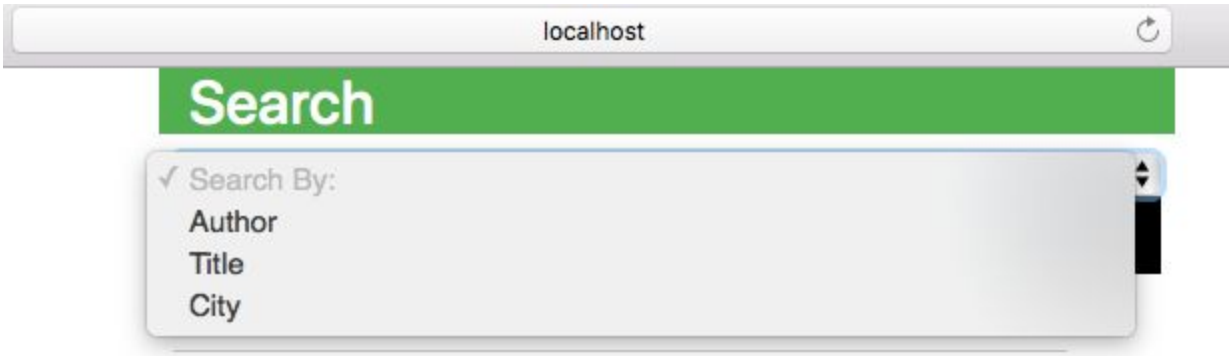
- ## Functional requirements

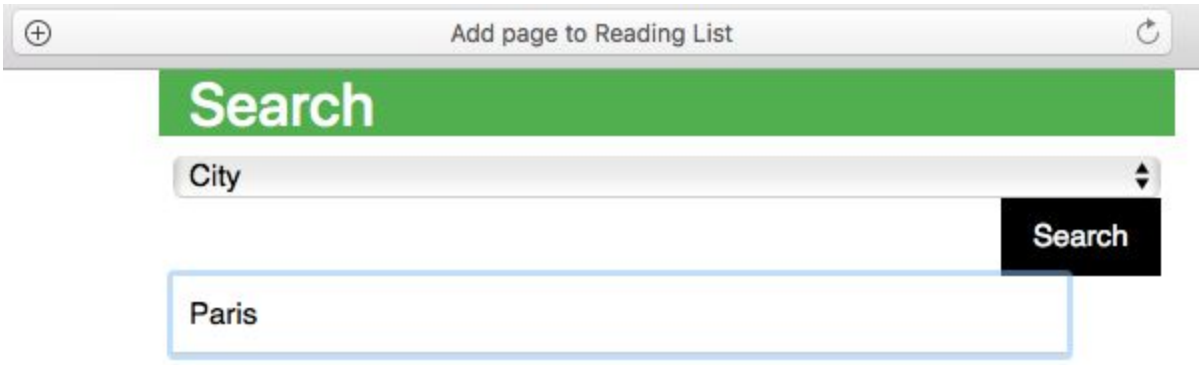
During the first Sprint, we defined specific behaviours and functions of our system:

- Simple web interface

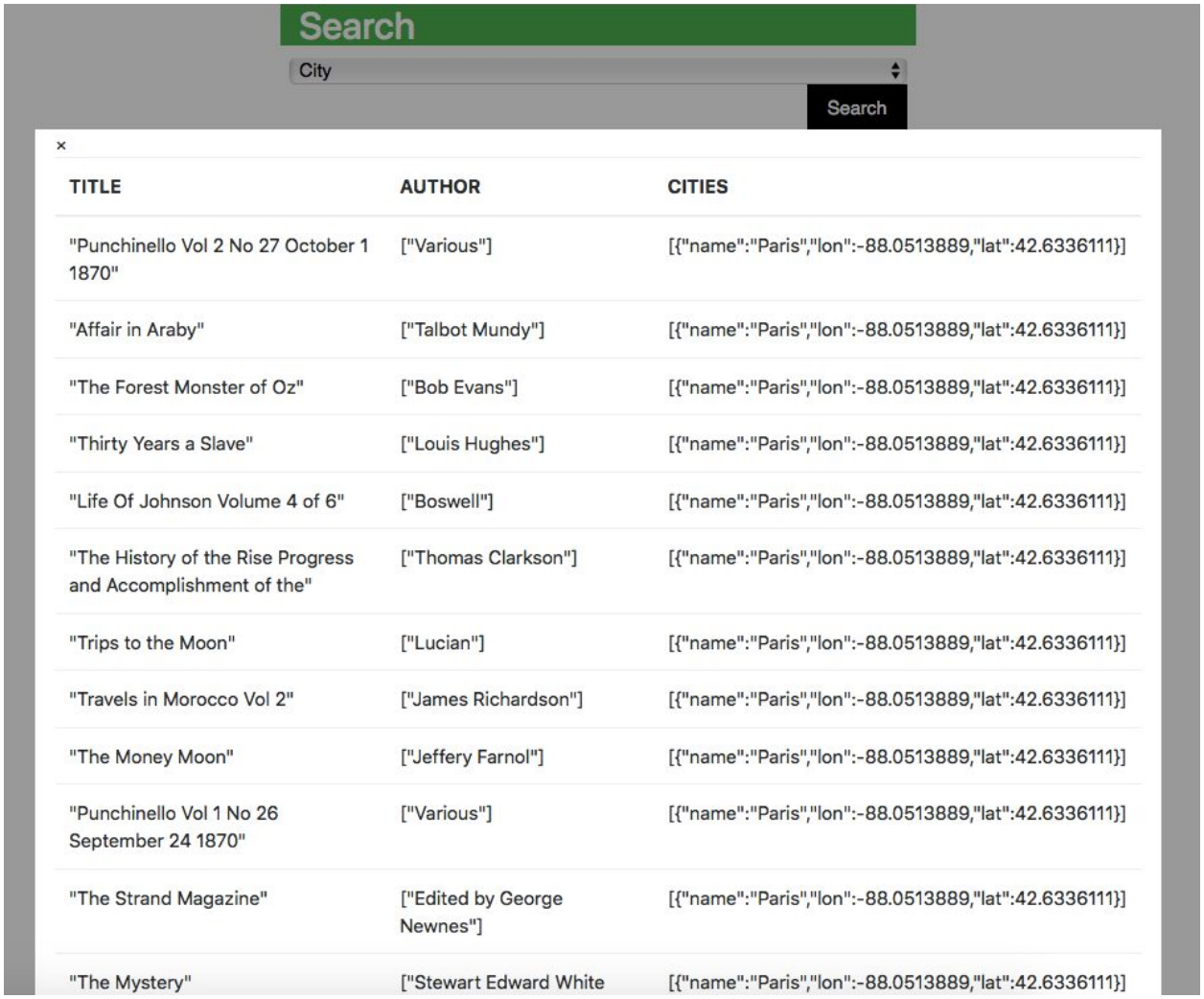


- Search tab with search button





- Results displayed in a list



Behavior:

- What are the business rules?

A user must be able to pick what to search by: title, author or city.
The user must be able to give input on his picked category.

- What will users do before and after using this feature?

Users do not need to login or have any prerequisites to use the system. They can simply enter the page, get information and close it.

- What's the worst thing that could happen when someone uses it? (exposes risks)

Worst thing would probably be that someone drops the table and we would have to recreate the database.

- What is the best thing that can happen?

That everything works perfectly. All queries are executed quickly.

Architecture and Design of Code Metrics

- ## Non-functional requirements

These requirement specifies criteria that can be used to judge the operation of a system, rather than specific behaviors.

Such as: the project is coded in Java, the query speed is fast, the project runs on any browser, etc.

- Quality Attributes

Quality Attributes are realized non-functional requirements, used to evaluate the performance of a system. To define if the software has good quality, we need to look at systems architecture, ease of use, efficiency, extensibility and reusability.

Architecture is a key property of a software system, it describes an internal structure of the system. A good and simple system structure makes it easy to understand the system as well as it is easier to make changes, test, implement and maintain.
Ease of use is also an important aspect as it should be friendly, and therefore will become more popular to use. To help making usage easier is documentation, if the software contains explanatory documentation, the users might not even need the training on how to operate the system.
Efficiency is ability to produce a high performance software with use of available hardware and software resources.
Extensibility allows to accommodate changes to design or technical constraints with minimum effect to the system.
Reusability is important so that the software can be reused in others parts, and therefore save time, money and hardware use.
.

- Could this affect performance?
    - How will we test for that?

We are using Jmeter for checking performance.

- Could the story introduce any accessibility issues?

No, we do not have any other roles, except for the user.

- What quality attributes are important for this feature, for the context?

For queries, we value mainly efficiency and reusability - coding-wise.

Testing

● Testable code

Behaviour Driven Development (BDD) is based on Test Driven Development (TDD) and it helps product owners and developers to narrow the gap between production. One of the most famous tools that supports BDD is Cucumber, which we will discuss later on.

Test strategies:

- JUnit for Automated Testing
- White box Testing with Cucumber and Selenium
- Performance Tests for “fast enough” queries
- Test with small dataset to check functionality

- Do we need a lot of exploratory testing?
    - Should we write a high level exploratory testing charter for where to focus testing for this feature/epic?

  No , we have rather used a simple dataset to test the actual functionality that does not differ respectively to larger data.
    - Do we have the right data to test this?

  Yes, we have tested it in both development and actual database.
    - Does it require updating existing tests, or adding new ones?

  No, we have not had the experience of adding tests because of datasets.
- Is there any learning curve for a new technology?

Cucumber was new to us, some of Selenium’s properties as well. We have tried Jmeter with dubious success. The learning curve for Cucumber was not steep.

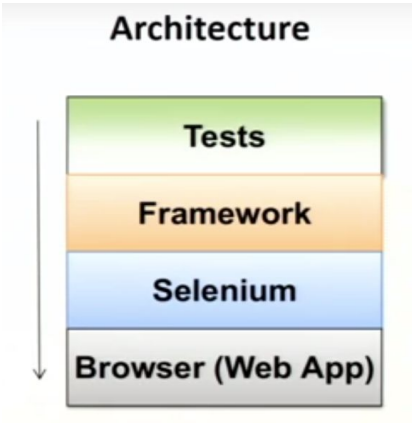
● Automated tests

Using automated tests helps us to execute test case suite, save time and enter test data into the system under the test to compare expected and actual results. This helps us to reduce test cases to be run manually. Automated testing is valuable for the development, containing Unit Testing, Integration Testing and User Acceptance tests. These belong to black box automation tests. In Black box we test everything as a User would use it without knowing the code.

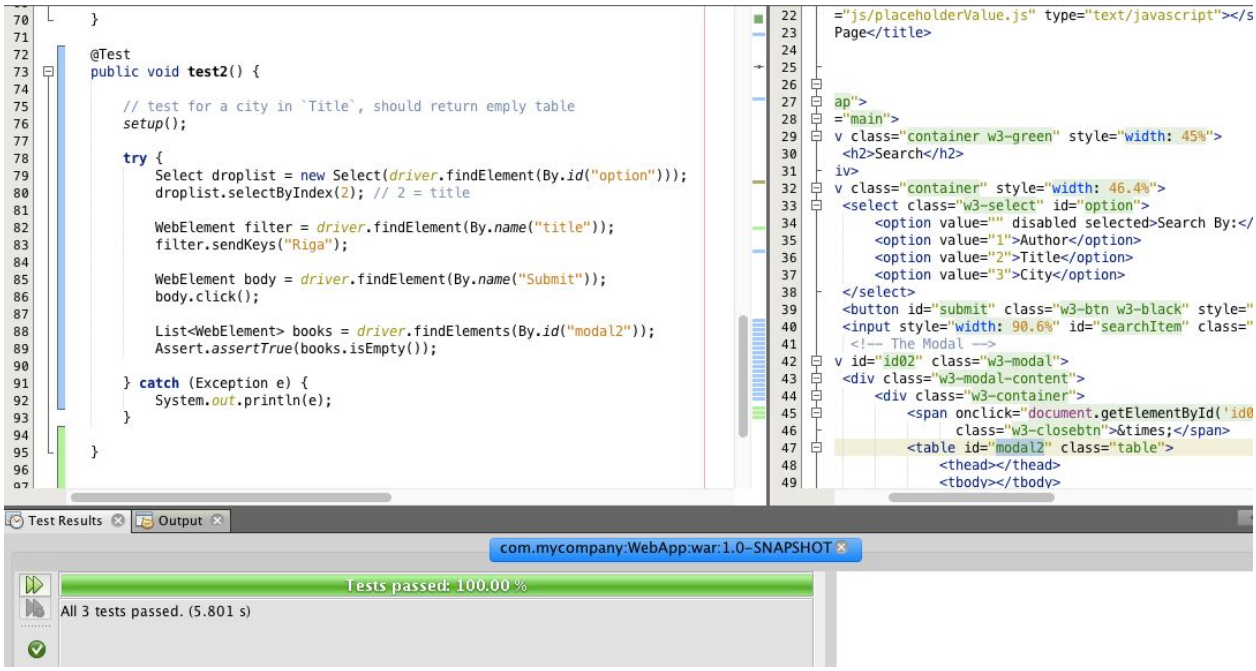
As we are working in Agile environment with sprints. Automated testing is very grateful instead of using manual testing. Therefore as the project continues to build, automating tests is time saving and makes sure that the system keeps working as expected.

- Selenium tests

Selenium software allows us to do operations on websites, so we can test the interactions happening on the web application over and over automatically. The architecture of using Selenium is, that we have tests that utilizes the Framework which uses Selenium and then the Selenium will touch the browser. Below figure displays the interaction between the layers.



Architecture of automated testing with Selenium



Selenium test for wrong input.

- Cucumber tests

For defining our test cases, we use Cucumber tool, which is between Behaviour Driven Development and Acceptance Driven Development.
As cucumber is an open source tool framework for test automation, we have used this tool to test our test cases.

Example of Acceptance Test Case from User Story 1:

When searching for London, I get back Charles Dickens book list containing ‘Oliver Twist’.

Cucumber uses a language called Gherkin, which is a plain language to describe a test case. It can be written in any coding language as it isn't technical, Gherkin is just a language that Cucumber uses to define its test cases in a human readable way and helps to enforce firm requirements.

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running stepdefs.TestRunner
Feature: Get Books By City
    As a User
    I want to type in a city
    So that I get a list of books in return

  Scenario Outline: Enter Valid City name # get_books_by_city.feature:6
    Given Opening the page
    When Entering '<city>'
    Then I should get results

    Examples:
Starting ChromeDriver 2.38.552518 (183d19265345f54ce39cbb94cf81ba5f15905011) on port 23492
Only local connections are allowed.
May 25, 2018 12:55:30 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS

  Scenario Outline: Enter Valid City name # get_books_by_city.feature:13
    Given Opening the page                # StepDefinitions.user_opens_page()
    When Entering 'Paris'
    Then I should get results             # StepDefinitions.user_gets_all_books_with_mentioned_city_name()

1 Scenarios (1 undefined)
3 Steps (1 skipped, 1 undefined, 1 passed)
0m4.002s


You can implement missing steps with the snippets below:
```

Figure 1
Test output for User Story 1.

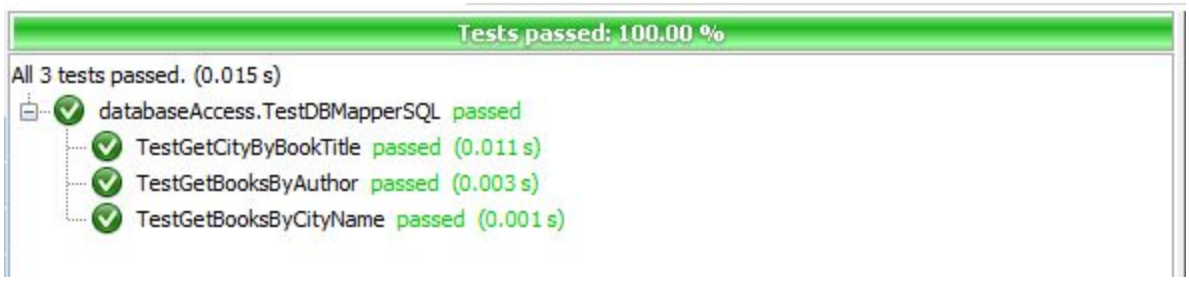
As seen in Figure 1, Gherkin uses step keywords as a standard format to define test cases: Given, When and Then. This file is located in plain text feature file, extended with `.feature` and can contain multiple scenarios. Each scenario and feature has to be able to execute independently. After defining our test case with the step keywords, we need to hook it up with the code steps to do the automated tests.

- Unit tests

In Unit tests we isolate a specific function to ensure that it works as expected and will go either green or red (in most cases if used in an ide)
The more unit tests, the better, as it will ensure that each little section is doing it's part and is doing it correctly.

```
@Test
public void TestGetBooksByCityName() throws SQLException {
    List<Book> list = new ArrayList();

    try {
        Connection connection = con.getConnection();
        Statement stmt = connection.createStatement();
        String cityName = "Paris";
        String query = "SELECT DISTINCT b.bookid, title, city, name FROM Books b, "
                + "Cities c, Authors a WHERE c.city = '" + cityName + "' AND b.bookid = "
                + "c.bookid AND a.bookid = b.bookid;";
        ResultSet res = stmt.executeQuery(query);
        while (res.next()) {
            String title = res.getString("title");
            String name = res.getString("name");
            System.out.println("Name: " + name + " Title: " + title);
            list.add(new Book(title, name));
        }

        assertEquals(list.get(1).getTitle(), "The Picture of Dorian Gray");
        assertEquals(list.get(1).getAuthor(), "Oscar Wilde");

    } catch (Exception e) {

    }
}
```

Unittest User Story 1, getting books list by given city name



All Unittest output

Organization

- ## Team work

As this project was developed in team of four, each member played equal part in the team. In Agile team, size doesn't matter, everyone participate and have role to play, we all switched tasks and roles of decision making and jobs to handle. We communicated daily via Messenger and Skype to agree on time to meet in person and on issues that occured when working remotely. Daily meetings were held in the morning and we were all four present and worked on project. While one member was working on front end, one on cucumber tests, one on unit tests and last on report, this was switched around and code reviewed repeatedly.

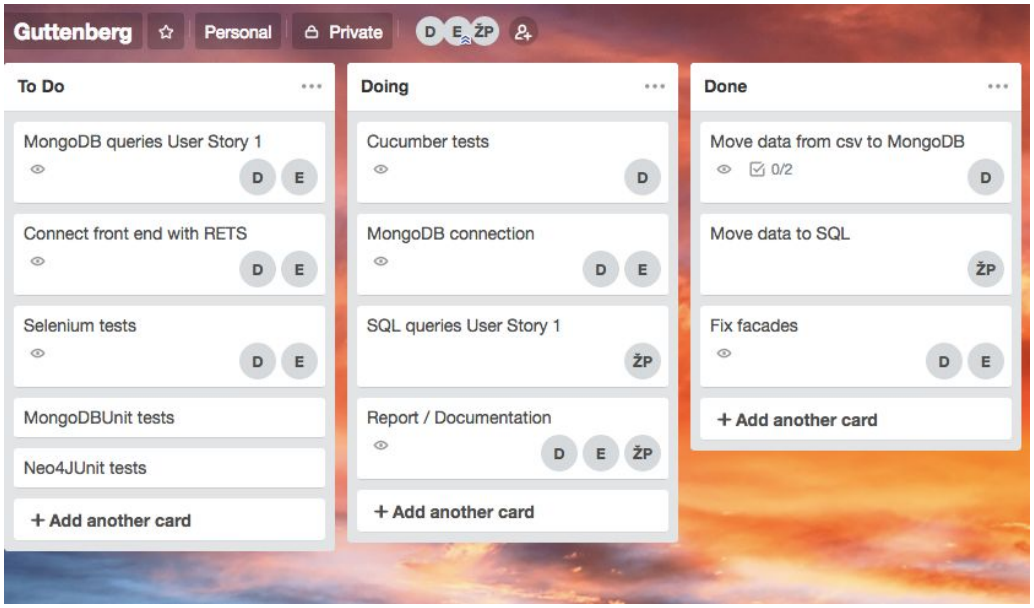
During the first Sprint, we established certain rules for the team to follow to reduce unclearaty and misunderstandings within the development process.

Coding standards:

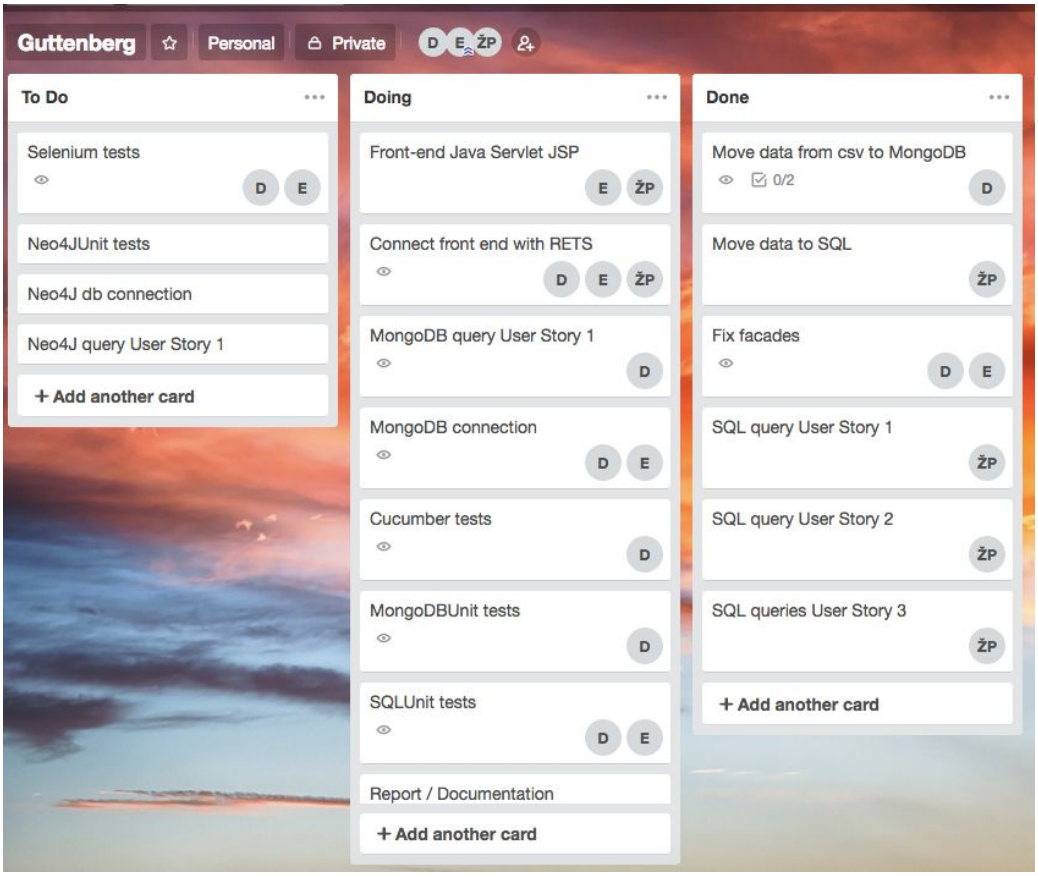
- Write with camelcase function names.
- Write with capital class names.
- Variables with lowercase.
- Package organisation according to functionality/layers
- Written in Java

- ## Agile process

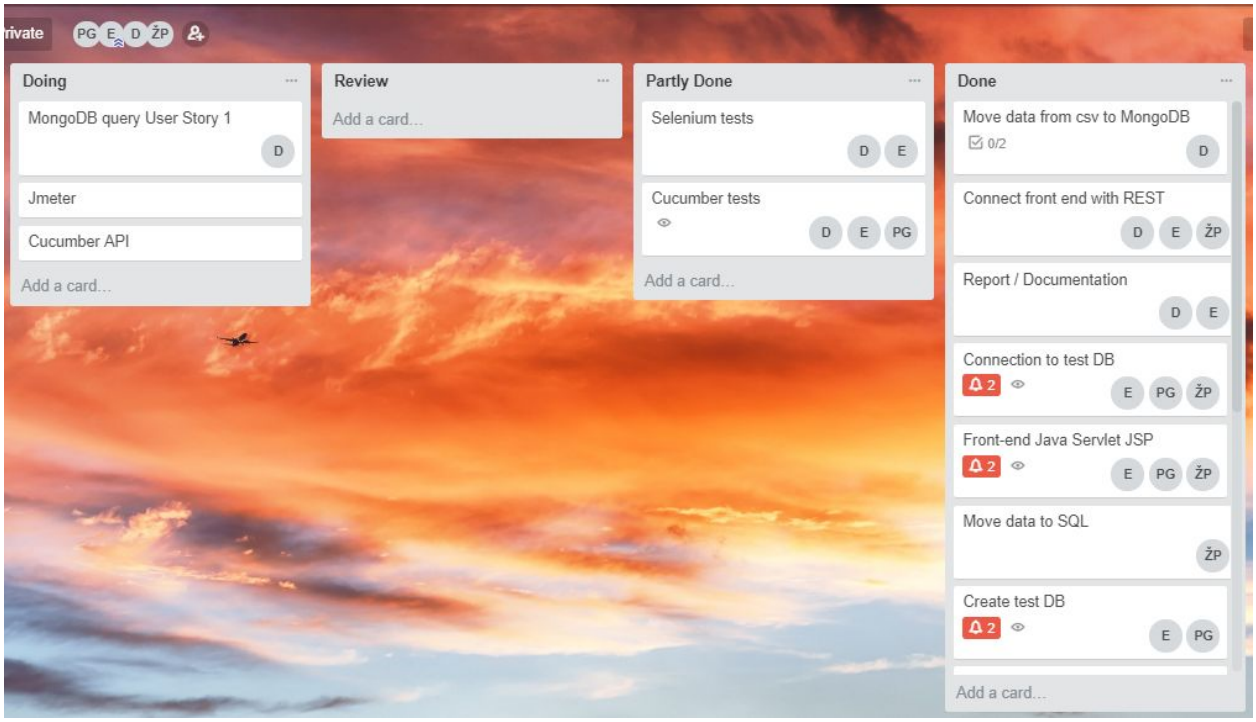
Without effective Agile Teams, composed and motivated individuals, organizations cannot achieve the business benefits of Lean-Agile development. Team should be empowered, self-organizing, and self-managing. We are accountable to deliver results that meet the Customer’s needs and expectations.
For keeping track of the tasks we used Trello board, where we assigned tasks and moved work that was done between sections. This helped us to keep good overview of the project and prioritize tasks. Adapting to change, having clear requirements, overcoming exactimates and missed deadlines by adjusting for unknown or unforeseen problems when they arise is part of working in agile processes.



Sprint 2 Trello board



Sprint 3 Trello board



Final Trello result

Appendix

User Story 2:
Given a book title, application plots all cities mentioned in this book onto a map.

1. Acceptance criteria:

As a user, I expect to get back a plot with all cities mentioned in the given book onto a map.

2. Acceptance Test Case:

When searching for 'Oliver Twist' it displays all cities mentioned in the book on the map.

3. Scenario Outline:

    Arrange

        Given: There are cities in the book

    Act

        When: The user searches

    Assert

        Then: plot all the cities on a map

User Story 3:
Given an author name, application lists all books written by that author and plots all cities mentioned in any of the books onto a map.

1. Acceptance criteria:

As a user I expect to get back a plot with all cities mentioned in all of a specific authors books.

2. Acceptance Test Case:

When searching for Stephen King, It will display a map of all of the cities mentioned in his books.

3. Scenario Outline:

    Arrange

        Given: The author exists

    Act

        When: The user searches

    Assert

        Then: return a map of all cities mentioned in given authors books

User Story 4:
Given a geolocation, application lists all books mentioning a city in a vicinity of the given geolocation.

1. Acceptance Criteria

As a user i expect to get all cities within a certain area of a given geolocation

2. Acceptance test case

Plotting in “48.8566° N, 2.3522° E” I expect to get Paris and the surrounding cities.

3. Scenario Outline:

    Arrange

        Given: the coordinates exist

    Act

        When: user searches for geolocation

    Assert

        Then: return nearby cities of geolocation