**Project Plan**

***TickieSystem***

*Invividual Project-*

*Semester III*

|  |
| --- |
| **Date : 27.11.2020** |
| **Version : 3.0** |
| **State : Unfinished** |
| **Author : Aleksandar G** |

#### Version history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| V 1.0 | **08.10.2020** | **Aleksandar G** |  | **Unfinished** |
| V 2.0 | 06.11.2020 | **Aleksandar G** |  | **Unfinished** |
| V 3.0 | 27.11.2020 | **Aleksandar G** |  |  |

Contents

[1. Architectural Design 3](#_Toc57400885)

[2. Database Design 4](#_Toc57400886)

[3. Design Desicions 4](#_Toc57400887)

[3.1 Back-end Framework 4](#_Toc57400888)

[3.2 Front-end Framework 4](#_Toc57400889)

[4. User stories 5](#_Toc57400890)

[5. Wireframes 5](#_Toc57400891)

[6. Testing 8](#_Toc57400892)

[6.1 Frontend Testing 8](#_Toc57400893)

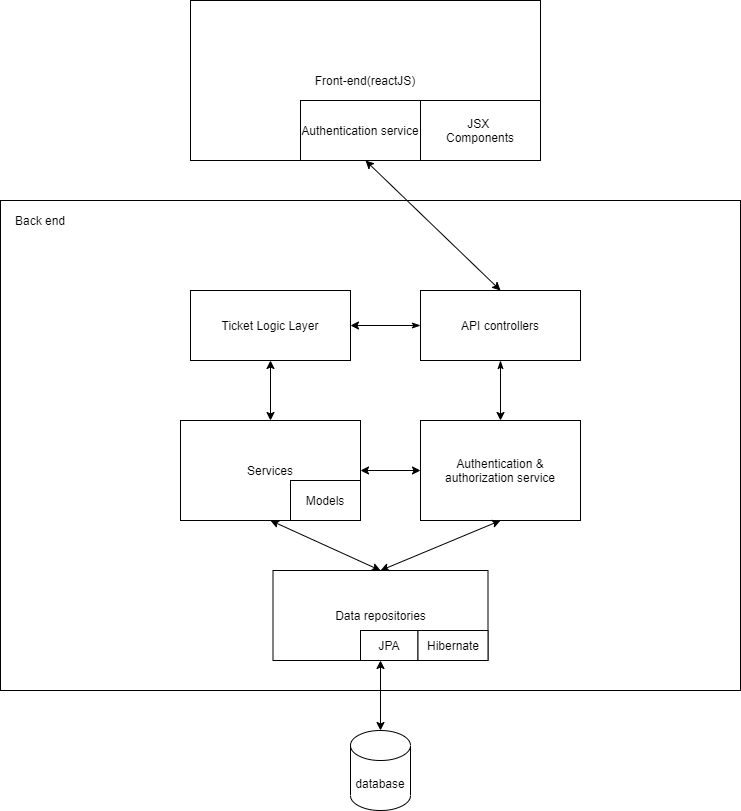
[6.2 Back end Tesing 8](#_Toc57400894)

[7. Quality Metrics 8](#_Toc57400895)

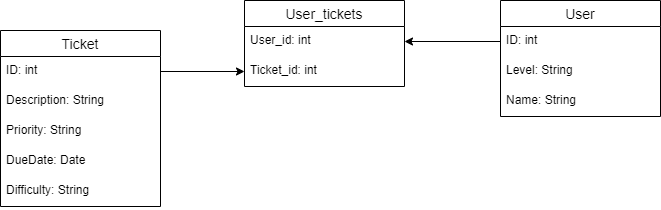
[8. CI 9](#_Toc57400896)

[9. Authentication and Authorization 10](#_Toc57400897)

# Architectural Design



# Database Design



# Design Desicions

## Back-end Framework

I chose Spring boot over jersey bacause the documentation of Spring boot is way better than the documentation of jersey and moreover, Spring-boot in the most famous back end java framework ,therefore, it has a lot more resources and tutorials online.

## Front-end Framework

I chose React.js as my front end framework because it is again the most wide spread front end framework out there. It is not that hard to go into the performace in good and the syntax is elegant. There ar a lot of qualoty tutorials online for it and the documentation is ok.

# User stories

As a senior developer I want to be able to see all the tickets in the system so I can select tickets which are the most critical **90**:

* All the tickets should be visible to a senior developer
* The importance level of the ticket should be visible to all developers

As a developer I want to be able to filter the ticket by importancy so I can select the most crucial one and prioritize them **90**:

* The filtering system should work with all importance levels.

As an intern I want to see the tickets only for my level so I can only focus on them **70**:

* The system should be able to automatically filter the visible tickets depending on the user’s level in the system.

As a developer I want to assign myself to a ticket so this way the others can see that this ticket is taken already and they can work on another one **70**:

* Every user should be able to assign himself to an available ticket
* A taken ticket should disappear from the system.
* Every senior developer should be able to see which ticket is taken and by whom it is taken

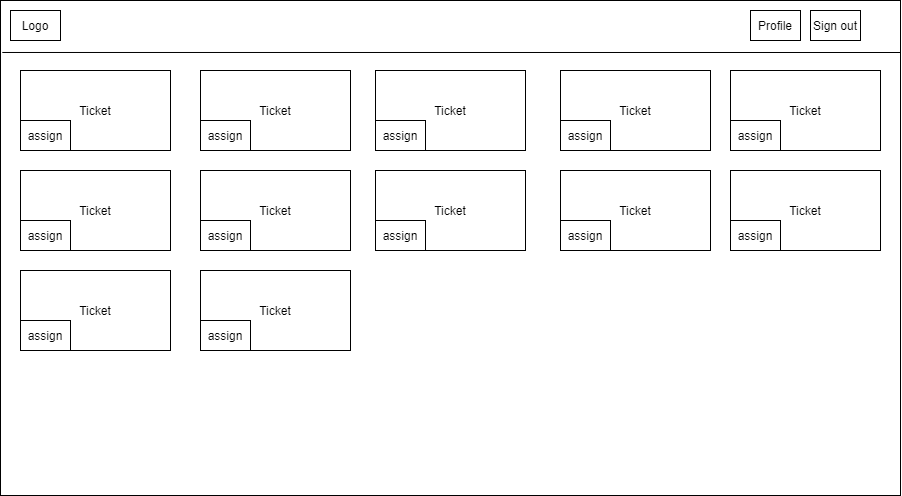
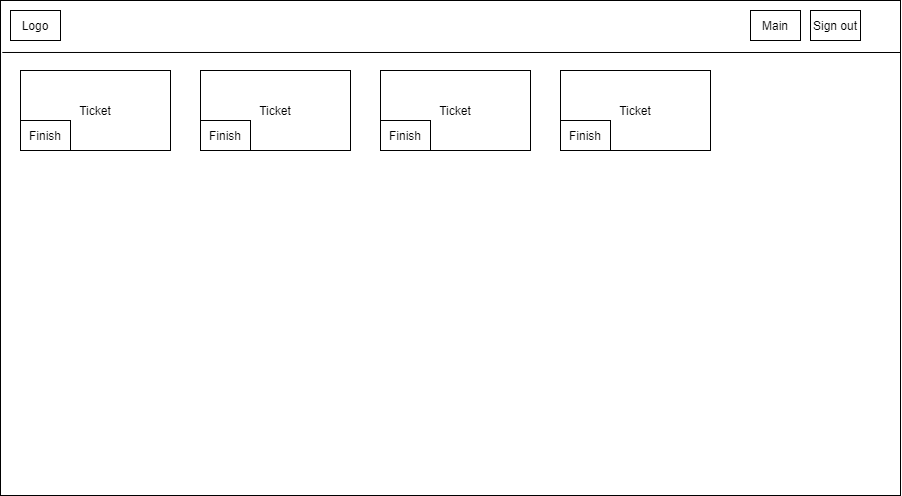
As a developer I want to bee able to close a ticket, in this way my senior can see that this bug has been dealt with **80**:

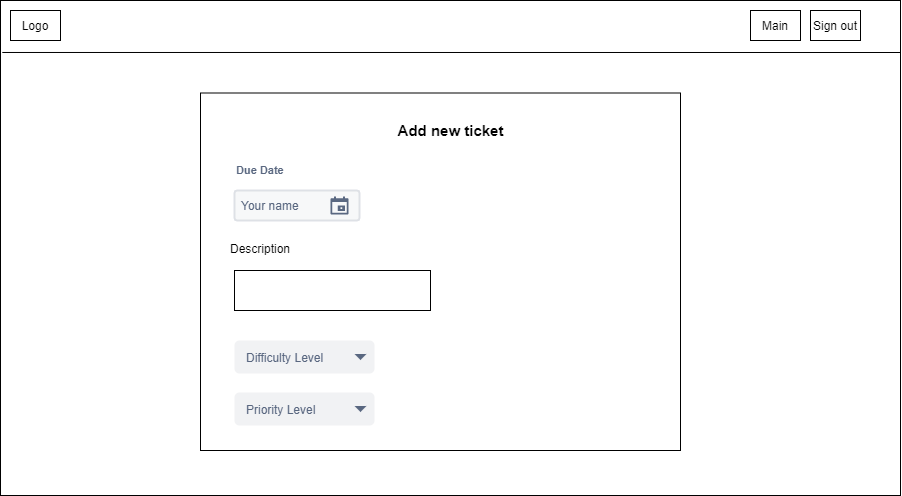
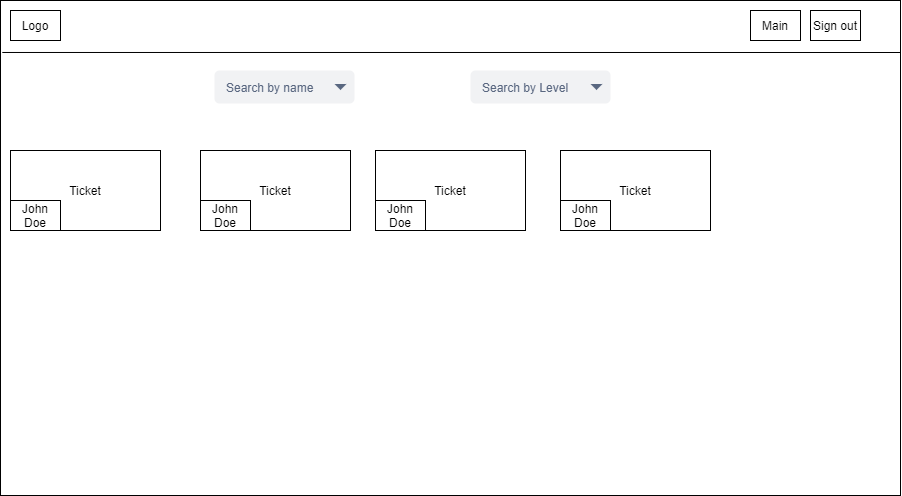
* Every developer should be able to close a ticket he assigned himself to.
* Senior dev can see and the closed tickets

As a developer in the company I need a login so that I can login to my profile and see all of tickets I took **90**:

* Every developer in the company is required to login to the system with username and password
* Every developer should be able to see all the tickets he took, but only when he is logged in the system.

# Wireframes





# Testing

## Frontend Testing

For the front end tesing I am going to perform Unit Tesing for the rendering of componennt and Ingtegretion tesing to see if the logic of the componenct is executed propenly. In order to do this I will use Jest.

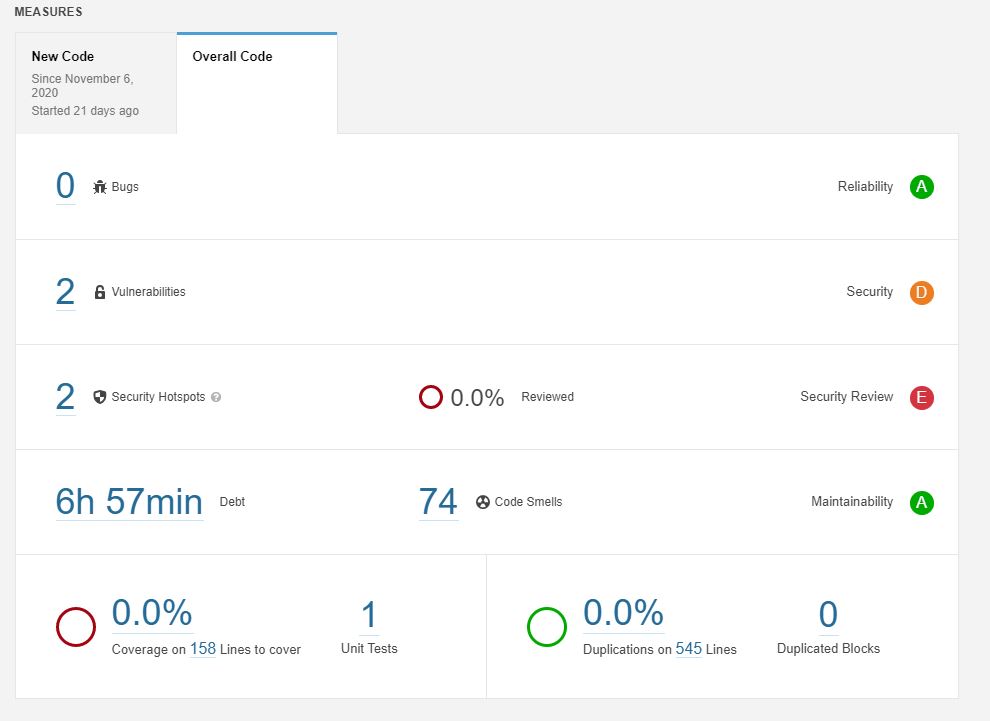
So far I have Unit Testing for the rendering operation of the components

## Back end Tesing

For the back end tesing I am going to perform regretion tesing, integretion testing and unit tesing.

I already have test methods for the Services and repositories and I am working on the Api testing. I perform those tests using Mockito and Junit.

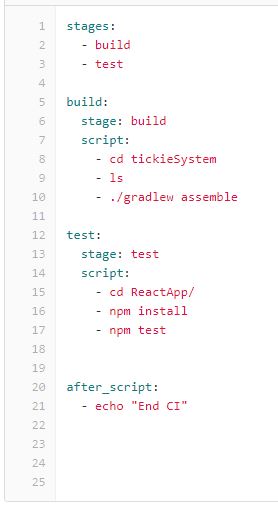
# Quality Metrics



\*I have tests in the front end as well as is the back end, so I dont know why in sonarQube it says 0.0% code coverage

# CI





# Authentication and Authorization

For the authentication I don’t use JWT, however I use something simular.

When I user clicks on the button log in a request in being sent to the back end withe the username and passsword of the user. The back end checks a user with is username and password exists and the sends a “*200 OK”* status if is it ok.

The if the status is 200 I generate a authentication tocken. Then I put the authentication tocken in the header of the axios config and the username in session storage. This is how then I can see authorize different request user the header is the request from the user.