Design Document

Zhaklin Yanakieva

Student number: 3811468

Date: 14/11/2020

Table of Contents

**C41**

C1 diagram and explanation1.1

C2 diagram and explanation ……………………………………………………………………………………………………….1.2

C3 diagram and explanation ……………………………………………………………………………………………………….1.3

C4(UML diagram and explanation ..…………………………………………………………………………………………….1.4

**Database design diagram2**

**Justification of frontend3**

**Justification of backend4**

**DOT framework5**

**CI/CD setup explanation5**

**Sonarqube setup explanation6**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision history | Revision class | Comments |
| Sprint 2 | 1.0.0 | Major | Initial activity |
| Sprint 3 | 2.0.0 | Minor | Updated diagrams and explanation to them |
| Sprint 4 | 3.0.0 | Major | Updated format-APA style/design; explanation to diagrams/CI set-up/Sonarqube(pre/post) |
| Sprint 5 | 4.0.0 | Major | Test plan |
| Sprint 6 | 5.0.0 | Minor | Change of CI/CD table |

**C1 Explanation:**

**Diagram

Description automatically generated**On the diagram above, there are three different roles for users of the website. One for the guest-user, who can only see the home page, the other – aggregated user, who is the actually logged-in user who has most of the functions, and the admin, who can do the functions for the website support, such as uploading videos, deleting users, etc.

**Why MySql and H2:**

Diagram

Description automatically generatedI chose to use MySql for storing the data because of its high availability and quick-start capability, which means that there are features self-management capabilities like auto restart, space expansion and automatic configuration changes for ease of management. It also comes with a comprehensive set of migration tools and a fully loaded graphical management suite. Furthermore, H2 as a mock database, which will be useful for the creation of the unit tests.

**Explanation:**

The diagram above displays the connection between the back and front end through a restful API. The backend is made on Java, using the framework Springboot, and the frontend – on React.js.

The line between, which represents the connection between the front and the back end, is without arrows because it just shows the connectivity. This type of relationship is called “association relationship” because it represents inter-process communication.

The backend send information to the data layer which is also called “data flow”, and in this case, a line with an arrow is required.

C3

Chart, box and whisker chart

Description automatically generated

**Explanation:**

Layered Architecture is all about the separation of concerns, encapsulating and decoupling the code. Layering means that the code has to be grouped by its functional role within the application.

In the diagram above, the backend layers are separated as follows:

1. There are the controllers, which play the role of the business layer, which need to be separated by the data layer.
2. Therefore, the logic layer appears, which consists of services in which all the methods are created.
3. Finally, the data layer is connected to the logic layer.

Diagram

Description automatically generated

**Justification for the front-end framework of choice: [[1]](#footnote-1)**

According to the research I made, React.js is easier to be learned for beginner developers. One of the main concerns developers have is choosing a framework (or library) that is not confusing and can be implemented in a way the learner can understand it. React is easy to grasp for developers who are familiar with Javascript. However, even if I am not that good in Javascript, React can be the right place to start my learning process. Unlike Angular, React holds a smooth learning curve.

In React, your application comprises of components. Ideally, it is started by building small components like buttons, checkboxes, dropdowns, menus, etc. and create wrapper components around these smaller components. And as going on writing the higher level wrapper components, a single root component and several hierarchical components are created. Now, here’s a no brainer: each component in React has its own logic, so the component may be re-used.

I tried to make test projects using the three frameworks: React.js; Vue.js and Angular.js, and after experiencing the work with them, I realized that it would be more convenient for my project to use React.js. Not only is it more understandable for a beginner with this frameworks, but the error that occur while implementing a code, happen to be found more often on the internet. Also, I chose React.js because the versions are updated automatically while the ones of Angular are done manually which will waste more time.

**Justification for the back-end framework of choice:[[2]](#footnote-2)**

* Autoconfiguration:

1. Developers can automatically configure their Spring application and also the framework gives the chance of changing the configuration based on the dependencies the user lists instead of them. For example, when there is “MySQL” listed as a dependency, it will configure your Spring application with the “MySQL connector” included. Yet, if the user wants to add a custom configuration, the user can create a class that overrides the default configuration for your “MySQL connector”.

* Standalone:

1. There’s no need to deploy your application to a web server. You simply enter the run command to start the application.

* Opinionated:

1. On the official page, we find that Spring Boot decides for you which defaults to use for the configuration. Also, it decides which packages to install for the dependencies a user requires and this setup helps developers to get started quickly on their projects.

* Better documentation:

1. The how-to pages of the spring boot framework are better explained.
2. There is more information about the errors that may occur during the process of learning spring boot.

**Authoization and Autthentication explanation:**

For the implementation of the authorization and the authentication I use OAuth(an authorization framework for REST/APIs). It enables apps to obtain limited access to a user's data without giving away a user's password because it is hashed. This framework is guaranteed to be the most secure flow because you can authenticate the client to redeem the authorization grant, and tokens are never passed through a user-agent.

**Backend:[[3]](#footnote-3)**

Spring Security:

– WebSecurityConfig is the most important feature of the security implementation, which provides HttpSecurity configurations.

– UserDetailsService interface has a method to load user by username and returns a UserDetails object that Spring Security can use for authentication and validation.

– UserDetails class contains the credentials of a user (such as: username, password, authorities) to build an Authentication object.

– UsernamePasswordAuthenticationToken gets {username, password} from login request, AuthenticationManager will use it to authenticate a login account.

– AuthenticationManager to validates UsernamePasswordAuthenticationToken object.

– AuthenticationEntryPoint will catch unauthorized error and return a 401 when Clients access protected resources without authentication.

– UserRepository & RoleRepository to work with Database, will be imported into AuthController, which handles signup/login requests by OncePerRequestFilter class (an [[4]](#footnote-4)abstract class from Springboot, which deals with request to API), and TestController, which has accessing protected resource methods with role based validations.

**Frontend:[[5]](#footnote-5)**

– The Header component is a container with React Router, which has the navbar of the website and contains the routes to every page.

– Login & Register components have a form for data submission. They call methods from auth-service to make login/register request.

– auth-service methods use axios to make HTTP requests. It also stores or gets JWT from Browser Local Storage inside these methods. (which will be advanced by storing the JWT in a cookie instead)

– Profile component displays user information, such as role/username/hashed password, etc., after the login.

– board-admin-component, board-moderator-component, board-user-componentcomponents will be displayed by state user-roles, which sets the roles of each user of the website. User.service is used to access protected resources from Web API and it uses auth-header() helper function to add JWT to HTTP header. Auth-header() returns an object containing the JWT of the currently logged in user from Local Storage. In order the services with the CRUD methods to work, as a parameter to each of them, { headers: authHeader() } needs to be added.

**DOT frameworks:**

**“What”:**

"application domain": the purpose of the project is mainly to entertain the audience(users).

"available work": Frontend framework – React.js, Backend framework – Springboot, Security frameworks – included in Springboot, Unit and Integrational testing

"innovation domain”: in this part, I focus on improving the quality of my website and give the users an opportunity to sit and go through some tennis matches in their spare time.

**“Why”:**

In the research, I am trying to optimize the fit between my product and the application context and I need to assure that my product is up to contemporary quality standards. In order to convey the research on the software application, I have to use all the expertise available to create my product. First thing is to start researching the "available work" domain until I get to the “innovational domain” and finish it with all dot-frameworks ready for the project.

**“How”:**

The DOT-Framework has 5 research strategies: Library, Field, Lab, Showroom, Workshop

In the “Matchpoint” project, the frameworks I use are Library by reading tutorials, documentation and watching videos on how to implement certain tasks. Then, I do the Lab research, where I test the product. Also, I do the Showroom research, when the product is presented to people to express their feedback. Finally, Workshop research is done by prototyping, designing and co-creation activities that are all ways to gain insights.

**CI/CD configuration explanation:**

The first step I took to create the CI on my project was to read the documentation of how to install and configure it. Then, I created two yml files – in the folder of my intelij project where I use the build/test/sonarqube stages, and in the main git repository folder, in order to run the CI correctly by redirecting to the other yml and in this case, the commands will be run correctly.

The first step for hosting the project on the docker server was to install docker and then, create images and containers for both the back and front end, for the database as well. The commands for the front and back end:

1. docker build -t –name . – this is for building the project and the “--name” is to replace it with the name of the image that is soon to be created. The dot in the end is put instead of a tag. This command is valid for both back and front end.
2. docker run -it -p 3000:80 --rm --name dockerizeReact frontend – this command is for running the container for the frontend and the first name is for the container and the second one – the name of the image.
3. docker run -it -p 8080:80 --rm --name – this command is run for the backend container.
4. docker run -p 3306:3306 -d --name mariadb -eMARIADB\_ROOT\_PASSWORD=Password123! mariadb/server:10.4 – this command is used for creating and running the docker database.

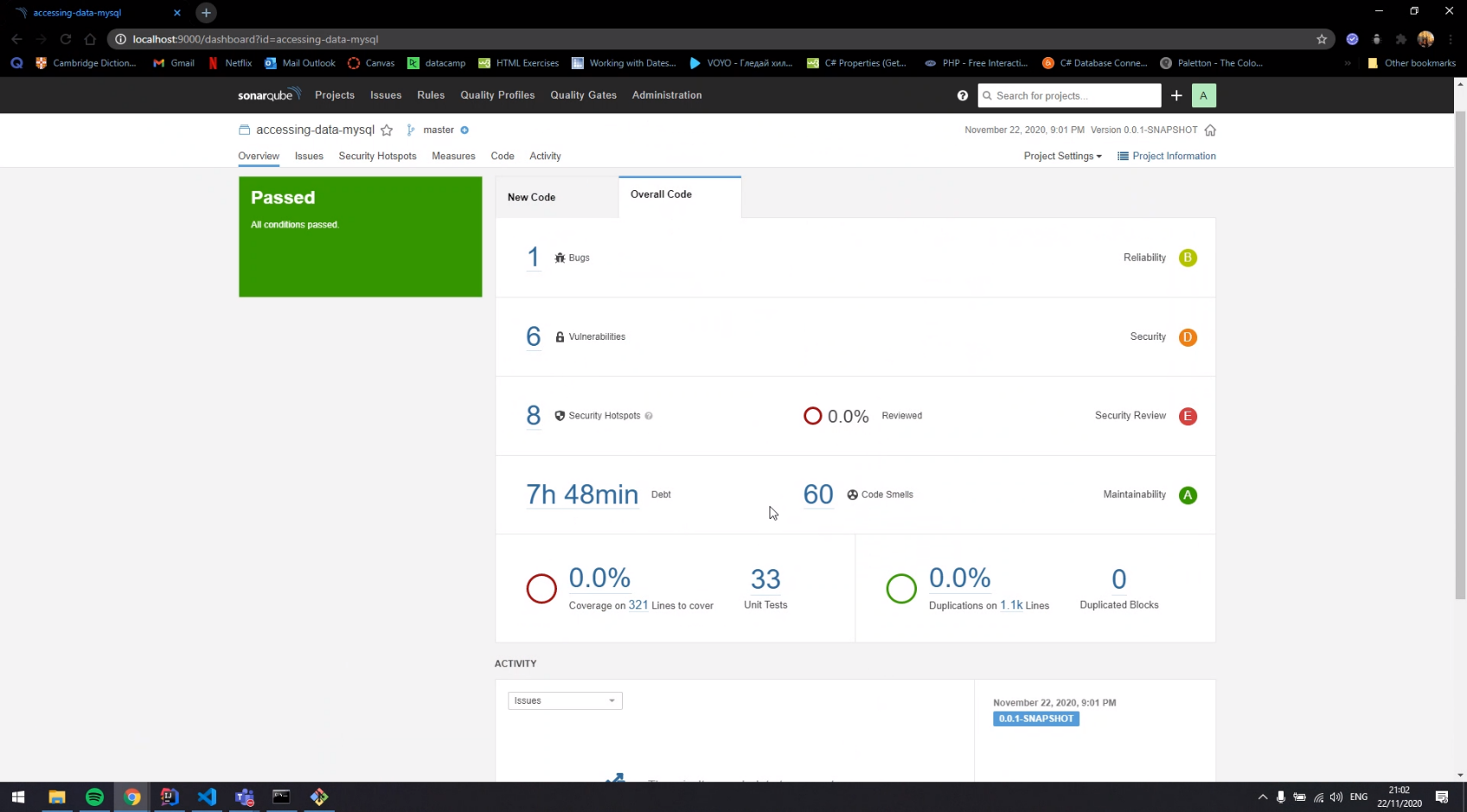
Diagram, schematic

Description automatically generated

**Quality** **metrics** **explanation:**

“Sonarqube” is a program used to evaluate the quality of a code. Firstly, in order to install “Sonarqube”, a zip file needs to be downloaded from the official sonarqube website. Then, the downloaded file has to be unarchived in the C:disk and afterwards the “StartSonarqube.bat” file in the bin folder should be opened so as to run the Sonarqube. A new tab will open in the browser where a login is required and then, a project needs to be created. After that, in the terminal in Intelij, the command “gradlew sonarqube” has to be run. In the end, the screenshot below displays what should appear after the whole explanation.

**Before:**

**After:**

**Graphical user interface

Description automatically generatedCypress:**

**Test plan**

During the process of creating an application, there are usually some errors that may occur and in order for such unpleasant situations not to happen, developers have to make tests of the functionalities of the app.

**What tests:**

Backend (BE): In the BE, where I am using Spring boot, I decided to make unit tests and integrational tests. The unit tests I made for the models, so as to make sure that each of the properties there are working property. For the controllers I made the integration tests, which are used to check the methods that are implemented there, such as “updateUser” or “deleteUser”.

Frontend (FE): In the FE, for which I am using React JS, I used cypress to check if everything works properly. There is a “home-spec” file, where I store all of the cypress tests. This file needs to be run and then, I can see if the tests are successful or not. First of all, I created a test for the connection with the link and if can be visited. Then, tests for the elements, such as buttons and inputs, were made and in the end, the registration and the login were tested.

|  |  |  |  |
| --- | --- | --- | --- |
| Nº | Action | Expected results | Comment |
| 1 | The system must establish a connection | The website is started and the connection between the BE and FE is established properly. | The project will not run into production until this is criteria is met. |
| 2 | Enter a valid username and password and click the login button | After logging in, to redirect to the profile page, where the user sees their information | The project will not run into production until this is criteria is met |
| 3 | The admin must be able to do all of the service functionalities | The admin must be able to do the CRUD functions without any error occurring. | This requirement is met, however, there is one CRUD function (for products) that needs to be improved in the FE |

# Bibliography

Backend, B. (2018). Retrieved from https://bezkoder.com/spring-boot-react-jwt-auth

Canvas. (n.d.). Retrieved from https://fhict.instructure.com/

Frontend, B. (2018). Retrieved from https://bezkoder.com/react-jwt-auth/

JS, R. (2016). Retrieved from https://www.c-sharpcorner.com/article/what-and-why-reactjs/

Springboot. (2019). Retrieved from https://stackify.com/what-is-spring-boot/

1. JS, R. (2016). Retrieved from https://www.c-sharpcorner.com/article/what-and-why-reactjs/ [↑](#footnote-ref-1)
2. Springboot. (2019). Retrieved from https://stackify.com/what-is-spring-boot/ [↑](#footnote-ref-2)
3. Backend, B. (2018). Retrieved from https://bezkoder.com/spring-boot-react-jwt-auth [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. Frontend, B. (2018). Retrieved from https://bezkoder.com/react-jwt-auth/ [↑](#footnote-ref-5)