Architecture document

17.11.2020

This is a document containing information about the design about how the project is going to operate. It will look in depth about database connection and classes nested inside the code how exactly communicate with each other. The project is about a online community specifically about windsurfing and its lovers.

Summary

This application would contain information about different people selling equipment for windsurfing. On the other hand there will be an open space for users to chat with each other about their experience in windsurfing as well as sharing their favorite spots.

Content

1.User Stories

2.Why I chose those software to work with

3.Design Document

4.CI setup

5.Additional

User Stories:

As a neutral user I want to be able to see the content of the front page without a registration. -90%(done)

-When you go to the Home(default) page URL you are able to go through every post and have a look at it as well as the comments.

As a user I would want to be able to create an account so that I can be able to enter the app. – 80%(done)

-When you reach the URL of the home page you could use the Registration form by clicking on Sign Up button on the right of the navigation bar.

As a user I would want to be able to log in having an account so that the app recognizes me every time I enter– 80%(done)

-When you already managed to register you can log in using your credentials when clicking on log in on top right corner.

As a user I would want to be able to log out so that I can change accounts– 80%(done)

-When you have entered the Website you have the authority to log out. The navigation bar is different for users who have logged in and it has the option to log out.

As an admin I would want to add or remove content so that I can manage any irrelevant posts. -70%

When you go on a post as a user you are able to delete each and every post.

As an admin I would want to be able to manage other’s posts as well so that I have full control over the users. -60%

As a windsurfer I would want to look at others’ people posts so that I can look for equipment. -60%(done)

-When a user enters someone elses post they have the possibility to add comments underneath it.

As a windsurfer I would want to be possible to write in the chat so that I can contact others via a chat when buying a equipment. -30%

As a windsurfer I would want to communicate with other people from the same region in a group chat so that I can buy what I want from nearby. -30%

As a user I would want to search people and equipment by Region so that I can look up local offers. -40%

As a user I would want to contact/chat to only people from specific country so that we can communicate easier. -20%

As a user I would want to be possible to change my region at any moment so that I can see more posts. -40%

As a windsurf user I would want to look for specific equipment only so that I can search only what I need. -35%

As a user I may want to update any of my posts at any time so when I decide to add more info about item it is visible to everyone. -50%

-When you click on the button addPost on top of the page you are able to add content as a User such as a new post.

As a user I want to be able to delete a certain post so that it is visible that my item is sold. -50%(done)

-When you go to a post which you have posted you have to ability to delete it , however if you don’t have the author rights of this post you are not able to remove it.

As a user I would want to create a new post so that at any time I can add new content. -45%(done)

-When I enter the Website as a registered user you have the authority to Add Posts when click on the top of the page.

As an admin I would want to be able to delete other’s accounts so that any harmful people are removed. -20%

1.1 Why I chose working with Maven and Spring Boot on first place?

* **Maven** is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process like ANT. ... **maven** make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.
* Both build systems provide built-in capability to resolve dependencies from configurable repositories. Both are able to cache dependencies locally and download them in parallel.
* As a library consumer, Maven allows one to override a dependency, but only by version. Gradle provides customizable [dependency selection](https://docs.gradle.org/current/userguide/dependency_management.html#component_selection_rules) and [substitution rules](https://docs.gradle.org/current/userguide/dependency_management.html#sec:module_substitution) that can be declared once and handle unwanted dependencies project-wide. This substitution mechanism enables Gradle to build multiple source projects together to create [composite builds](https://docs.gradle.org/current/userguide/composite_builds.html).
* Maven has few, built-in dependency scopes, which forces awkward module architectures in common scenarios like using test fixtures or code generation. There is no separation between unit and integration tests, for example. Gradle allows [custom dependency scopes](https://docs.gradle.org/current/userguide/dependency_management.html#sub:configurations), which provides better-modeled and faster builds.
* Maven dependency conflict resolution works with a shortest path, which is impacted by declaration ordering. Gradle does full conflict resolution, selecting the highest version of a dependency found in the graph. In addition, with Gradle you can declare versions as strictly which allows them to take precedence over transitive versions, allowing to [downgrade a dependency](https://docs.gradle.org/current/userguide/dependency_downgrade_and_exclude.html#sec:enforcing_dependency_version).
* As a library producer, Gradle allows producers to [declare `api` and `implementation` dependencies](https://docs.gradle.org/current/userguide/java_library_plugin.html#sec:java_library_separation) to prevent unwanted libraries from leaking into the classpaths of consumers. Maven allows publishers to provide metadata through [optional dependencies](https://maven.apache.org/guides/introduction/introduction-to-optional-and-excludes-dependencies.html), but as documentation only. Gradle fully supports [feature variants and optional dependencies](https://docs.gradle.org/current/userguide/feature_variants.html).

1.2 Why I chose Spring booth

Advantages of **Spring Boot**:

It has plenty of powerful tools which easily nest in your application .

It reduces lots of development time and increases productivity. It avoids writing lots of boilerplate Code, Annotations and XML Configuration. It is very easy to integrate **Spring Boot** Application with its **Spring** Ecosystem like **Spring** JDBC, **Spring** ORM, **Spring** Data, **Spring** Security etc.

1.3 Why I chose React for my core frontend?

ReactJS is a JavaScript library, open sourced by Facebook in 2013, which is great for building huge web applications where data is changeable on a regular basis.

**When it comes to choosing a framework for your frontend react is maybe the most handy decision due to its variable and flexible components.**

Used sources:

<https://www.lambdatest.com/blog/top-javascript-frameworks-for-2019/>

<https://blog.techmagic.co/reactjs-vs-angular-vs-vuejs-what-to-choose-in-2020/>

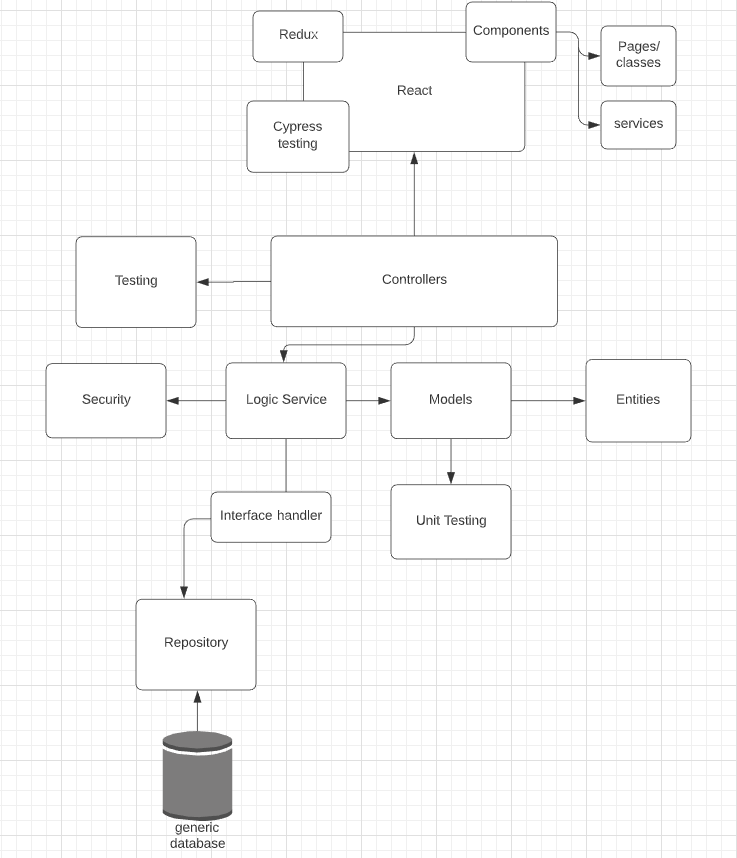
<https://reactjs.org/>

<https://gradle.org/maven-vs-gradle/#:~:text=Gradle%20allows%20custom%20dependency%20scopes,dependency%20found%20in%20the%20graph>.

Design Document

2. Design of the architecture behind the project and how different functionalities and components such as Entities are connected

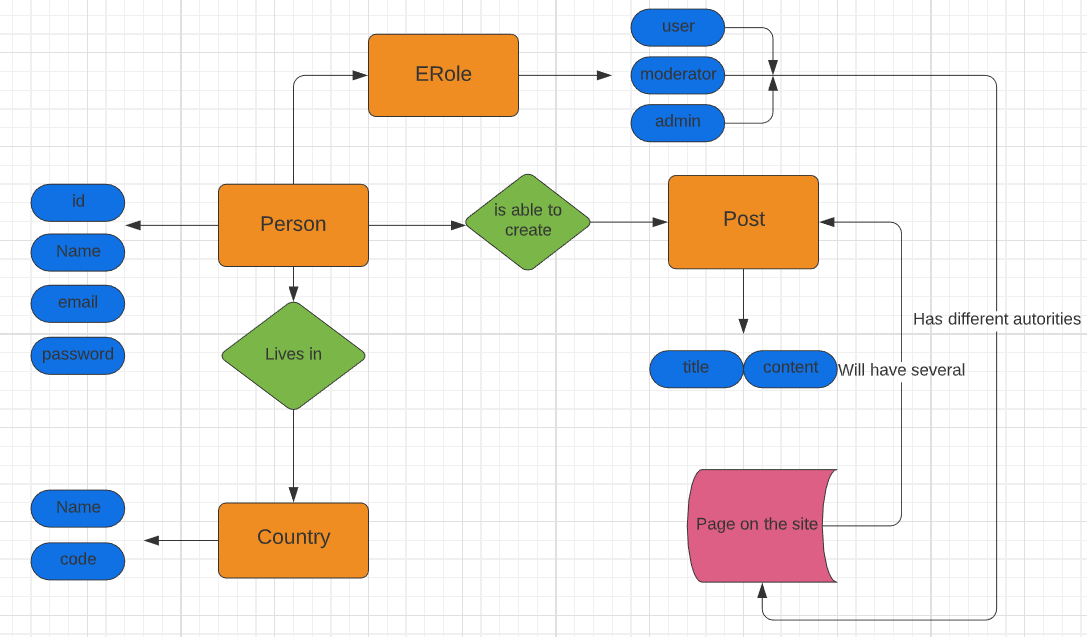
High Level Architecture



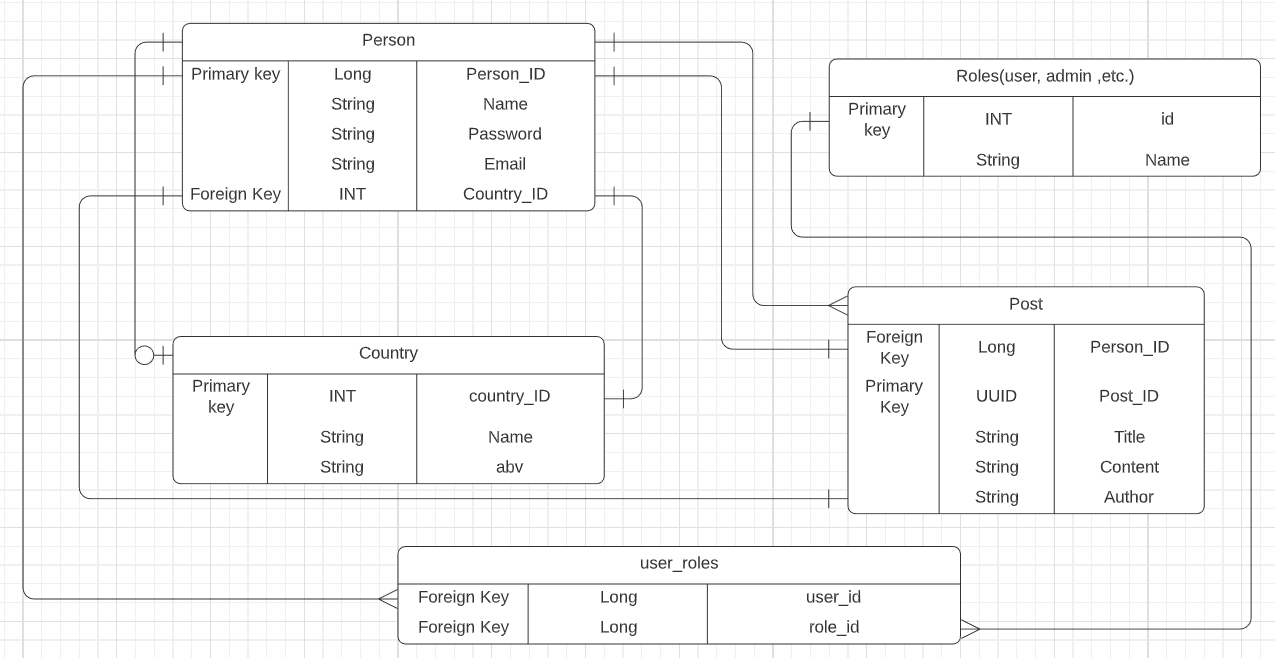
Let us take for instance a cycle of a data representing a person.

Firstly it has to be given as an argument by someone (User Interface), then if the data is correct according to output then it is taken by the controller which receives it, thus the service layer can check it out for issues such as an existing properties etc. Also prevents from harmful attempts. When fulfilling all the properties of a model it is then available for storage in the database.

2.1Entity Relationship Diagram



* 1. Database Design



Description :

Person is the representation of a user( the customer who is going to use the application/software). Each user will be able to create as many posts as they want . A post object will have variables such as Title, Content , Author , Time, however only those three mentioned above will be stored in database.

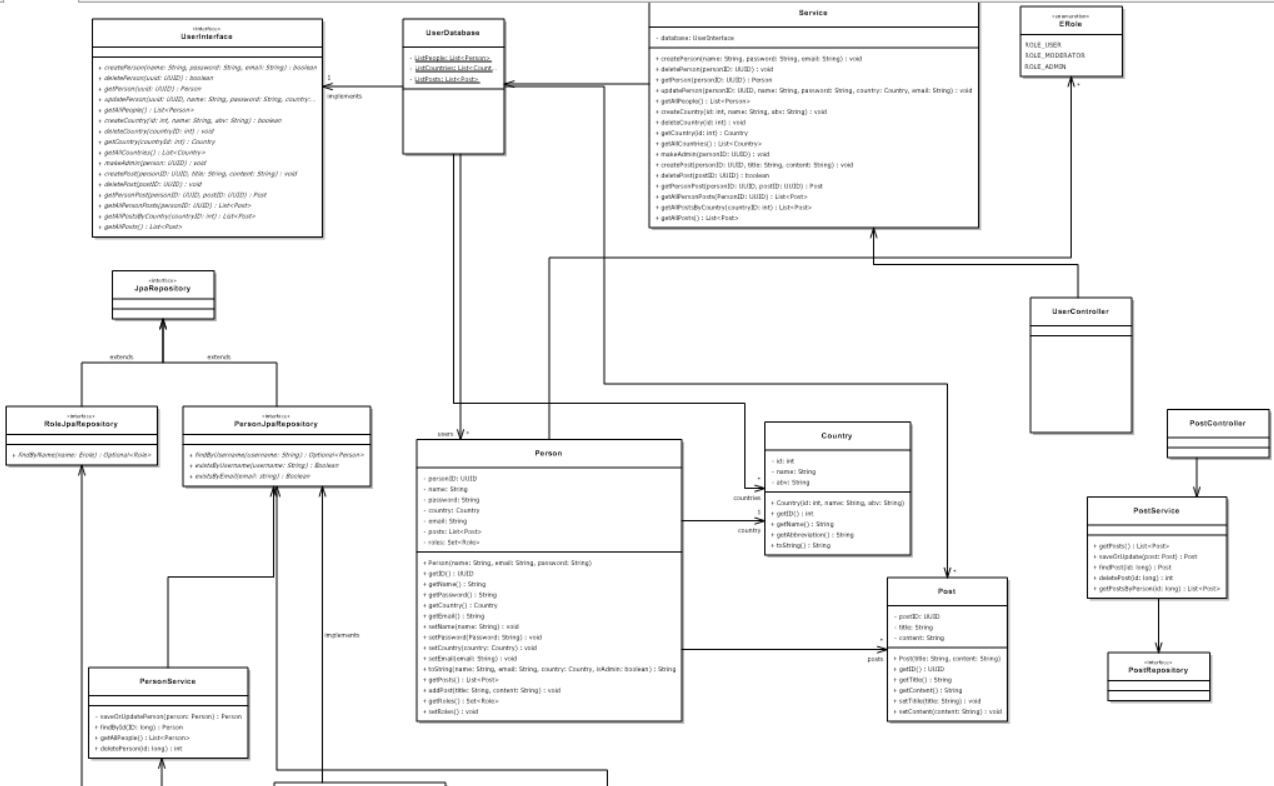
The person ID will serve as a foreign key in most of the others tables. Each user will have a country which he will be able to change anytime he wants still a neutral one will be assigned when creating a user.

The country Primary key will serve as a foreign key in the Person table.

To differ between users there is a Roles table created which will take responsibility of storing all the variable roles such as moderator , ordinary user and admin.

In addition a new table of user\_roles will store each user and his possible roles.

3.UML diagram



I have represented only some of the most essential classes and interfaces so you could get a little more detailed idea about the connectivity between classes and how they interact .

What is important is that SOLID is used and even thought on first sight it is not clear to distinct each class has its own purpose and there are also service classes which take responsibility about the logic that takes place between each Entity and the Controllers themselves.

If we cold easily describe the URL there are a few layers that contain what is needed.

-the Entity layer (describes only the modules classes)

- the repository layer(which ate basically a interfaces that extends with severe methods a Database connections)

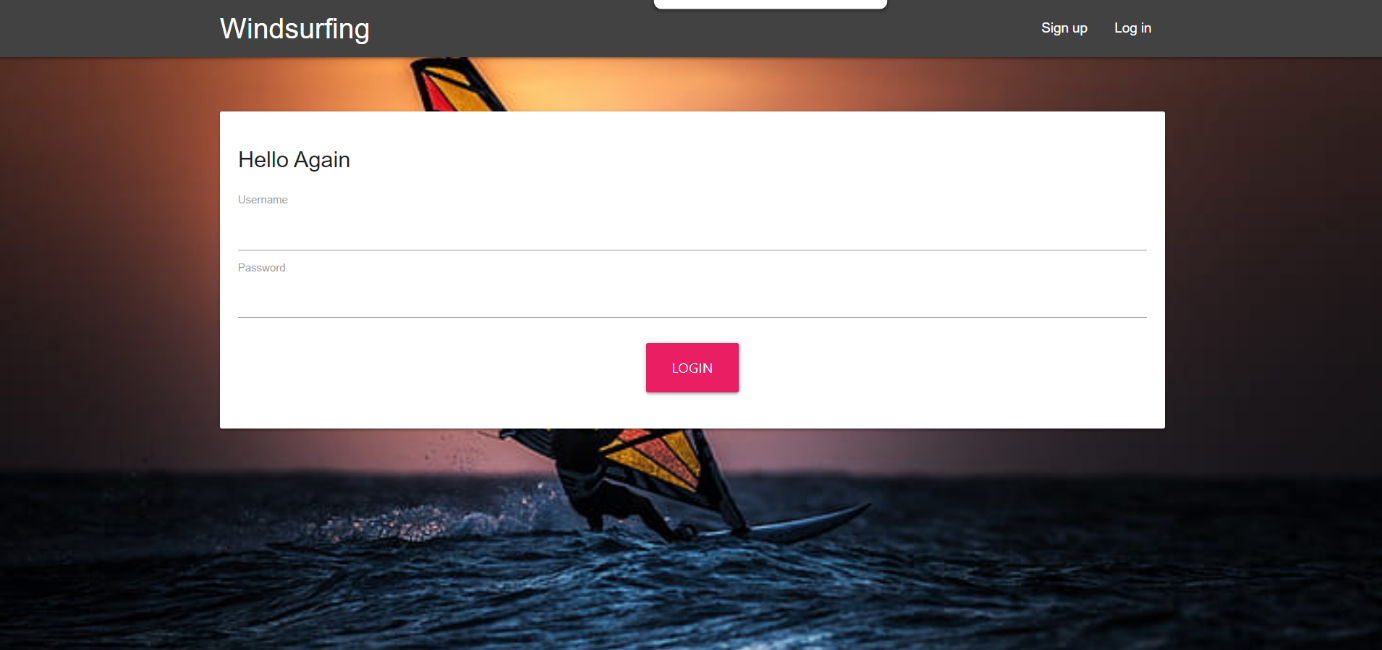
- The Service class(which basically implements the logic and inputs the entities inside the repositories/Database)

-also the Controller layer which mainly gives feedback weather everything has gone well inside the service when asking for input by users.)

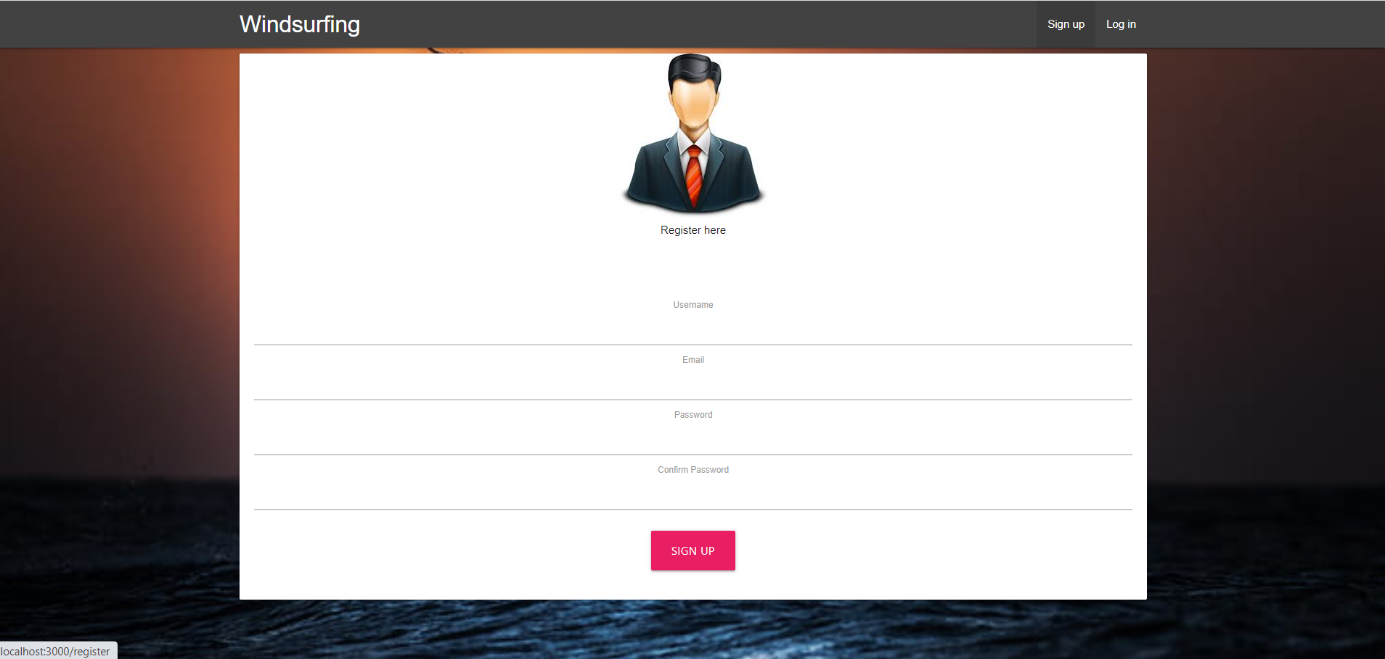
However the Authentication and Authorization are divided from the main functionalities but implement the same methodology as well as the services.

4.Wireframe

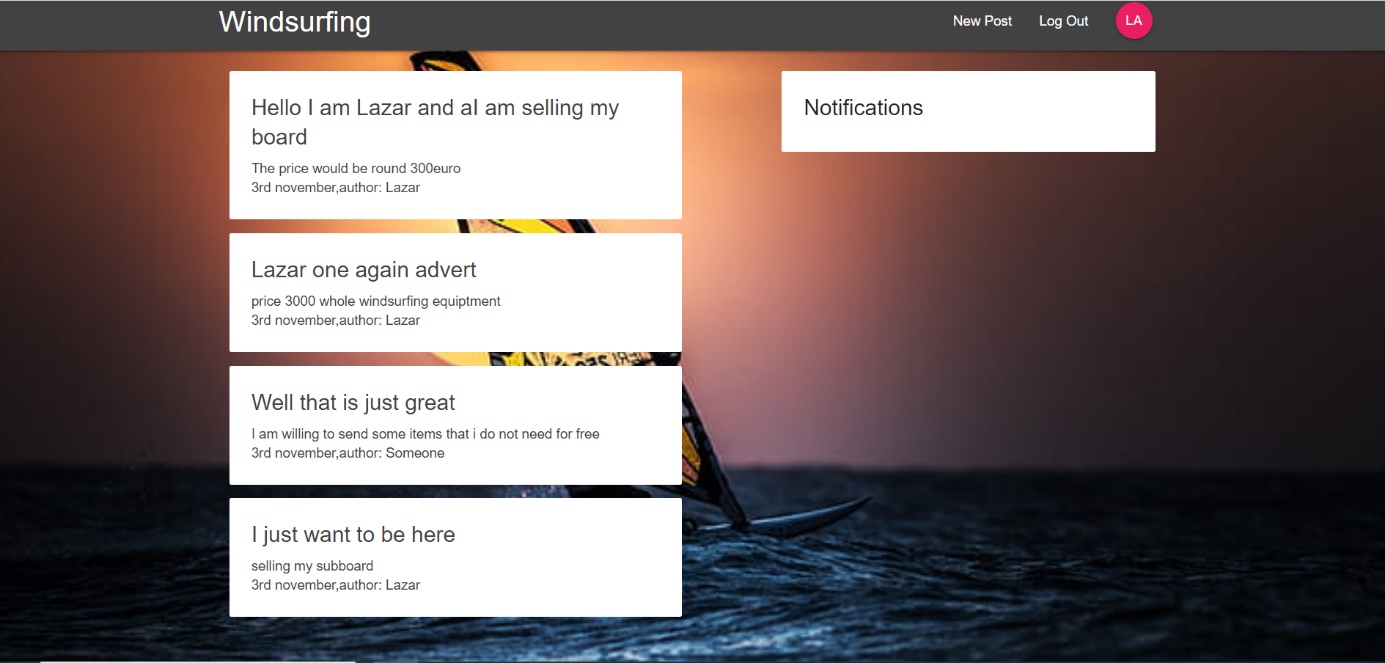
1.Log in page



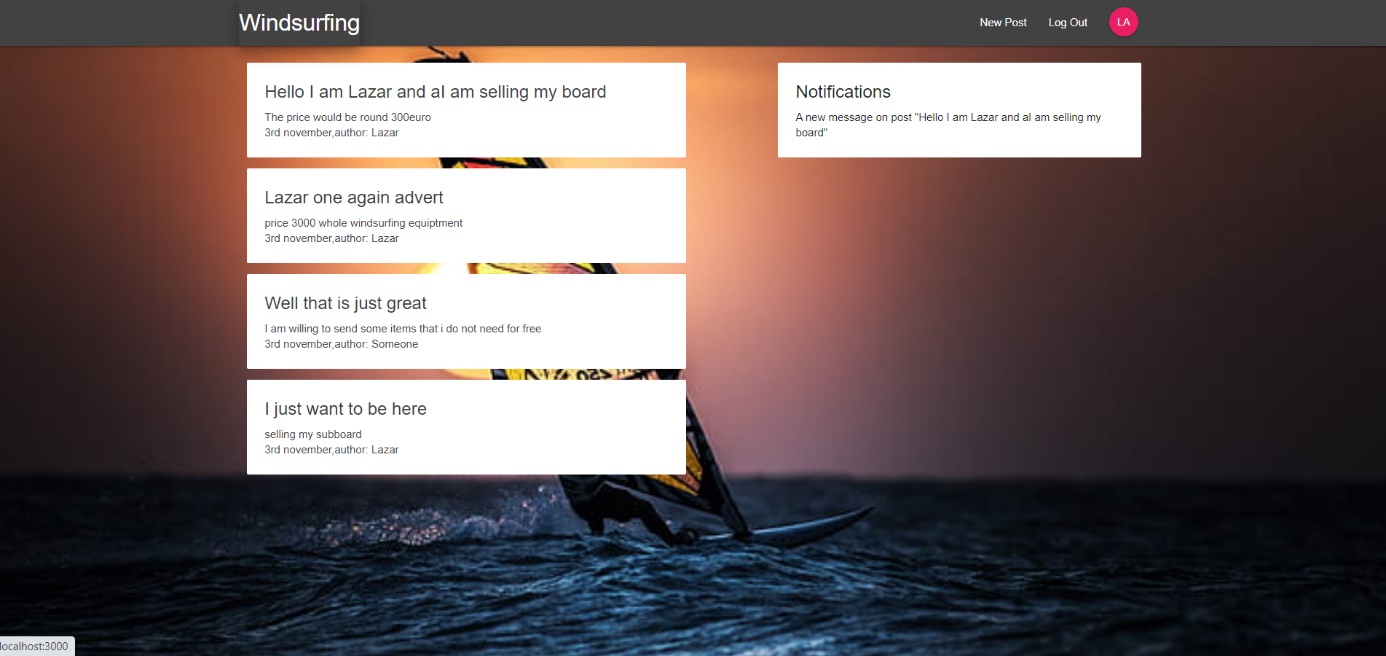
2.Register page



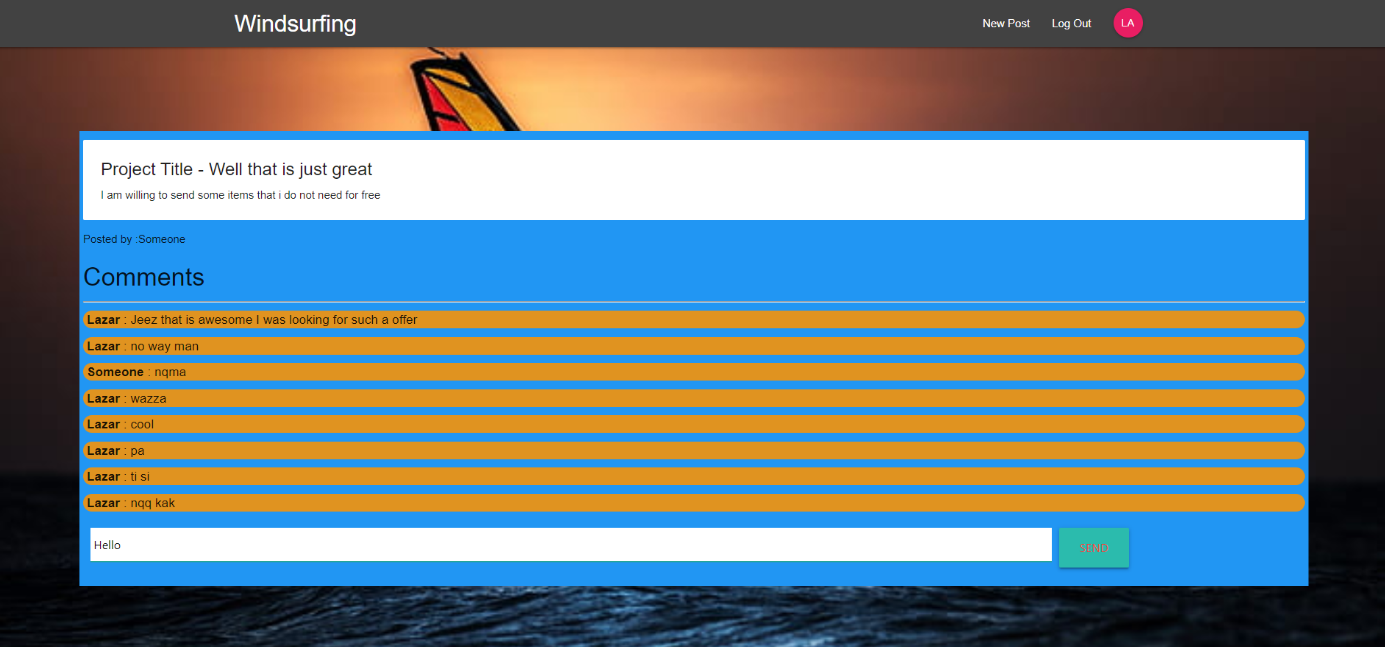
3.Home page



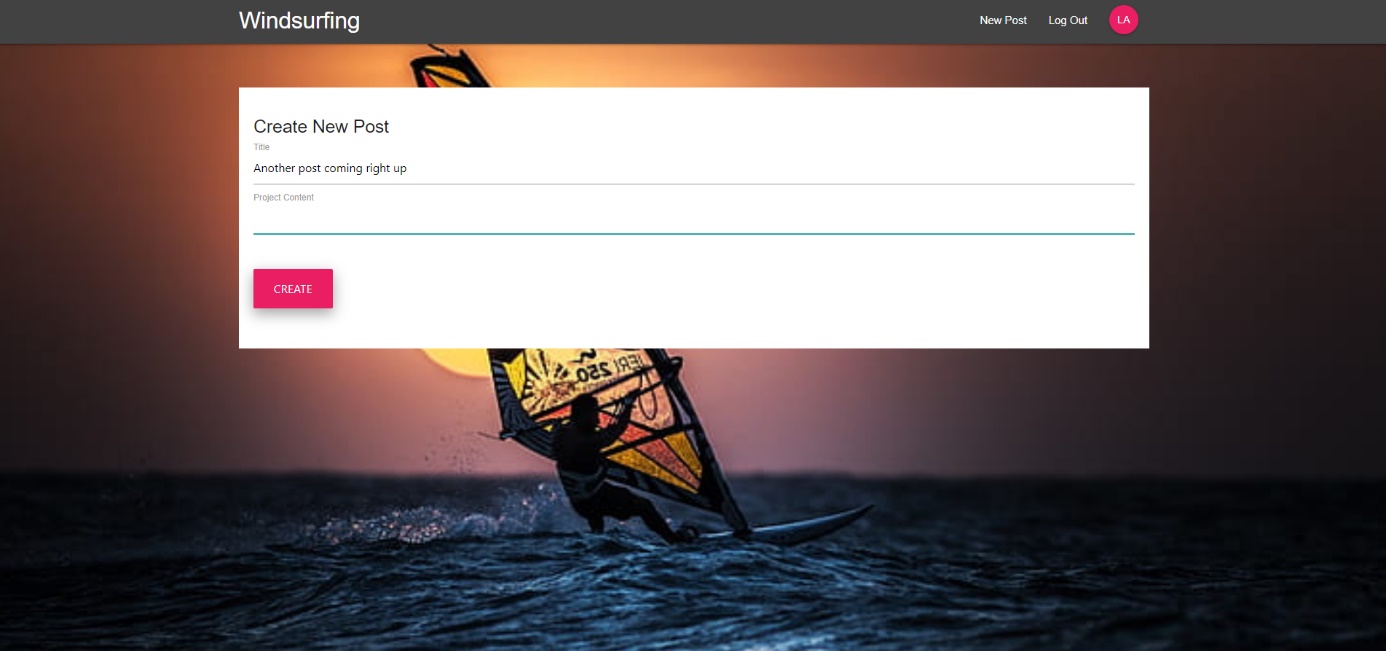
4.1 Receiving a notification



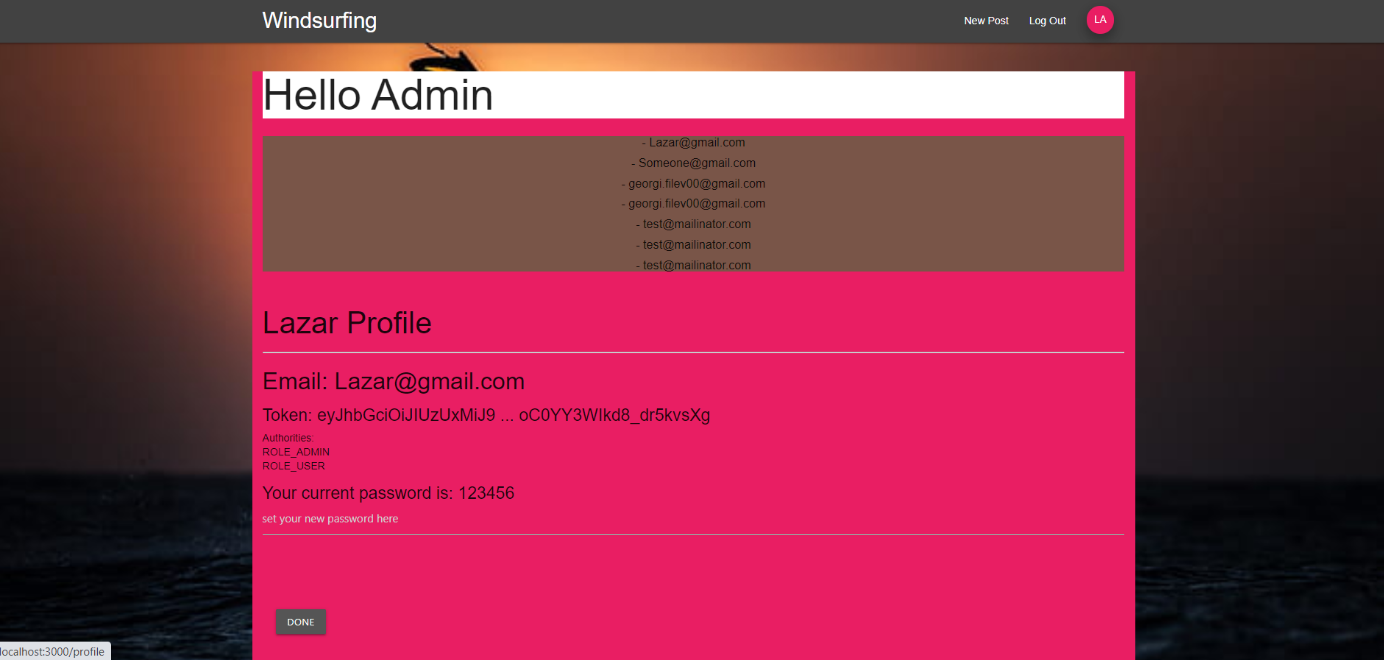
5.Comment on a different post



5.Create a post new Post page



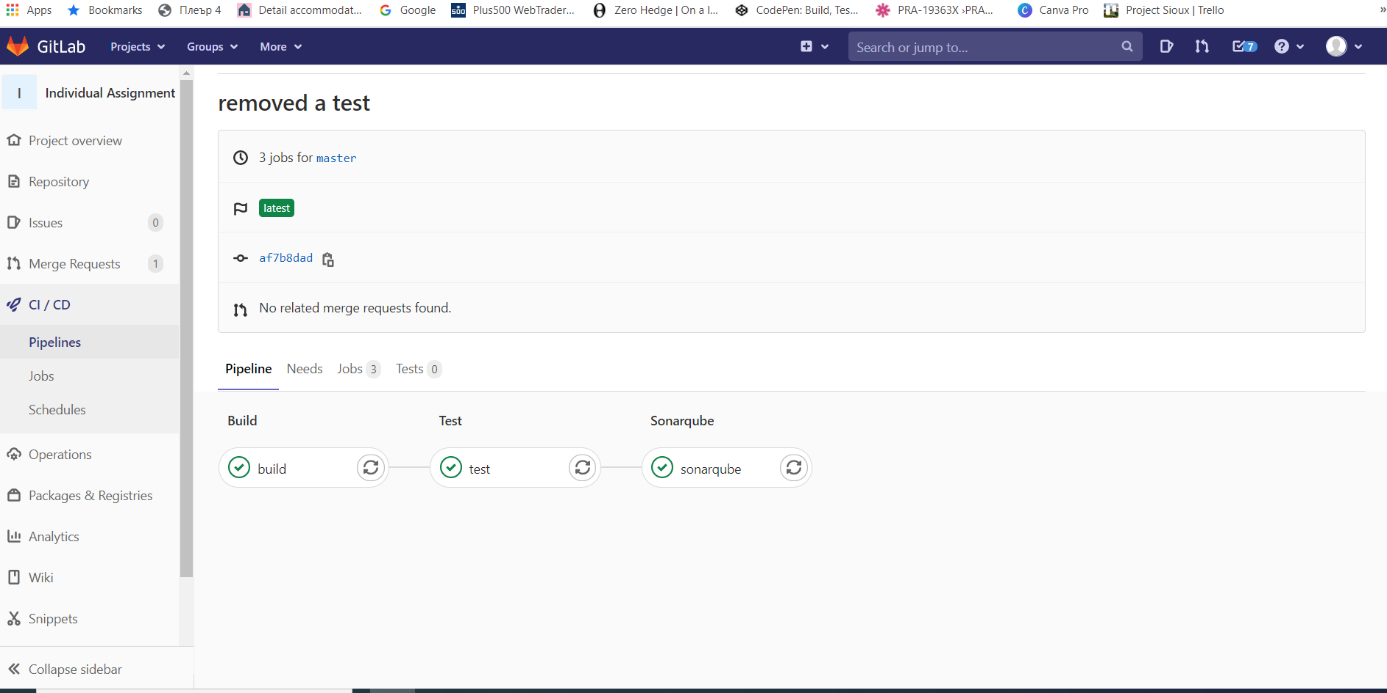
6.Settings/Adjusting page



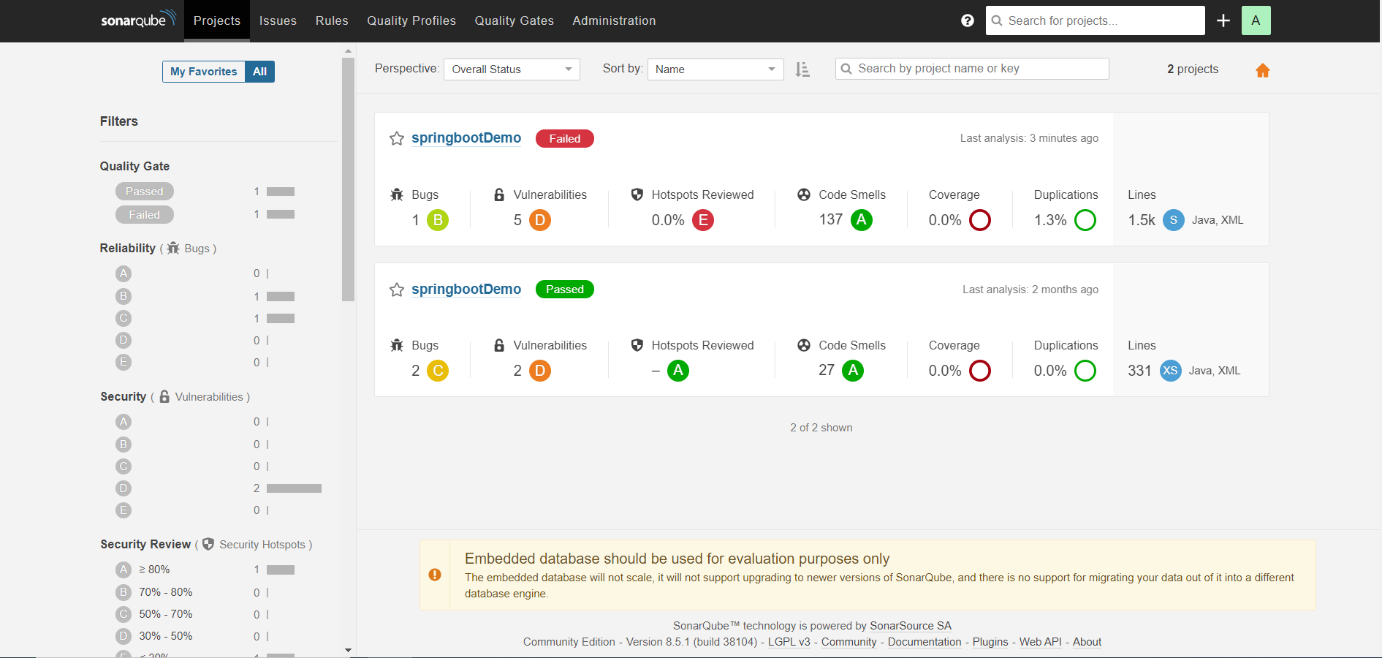
Additional :

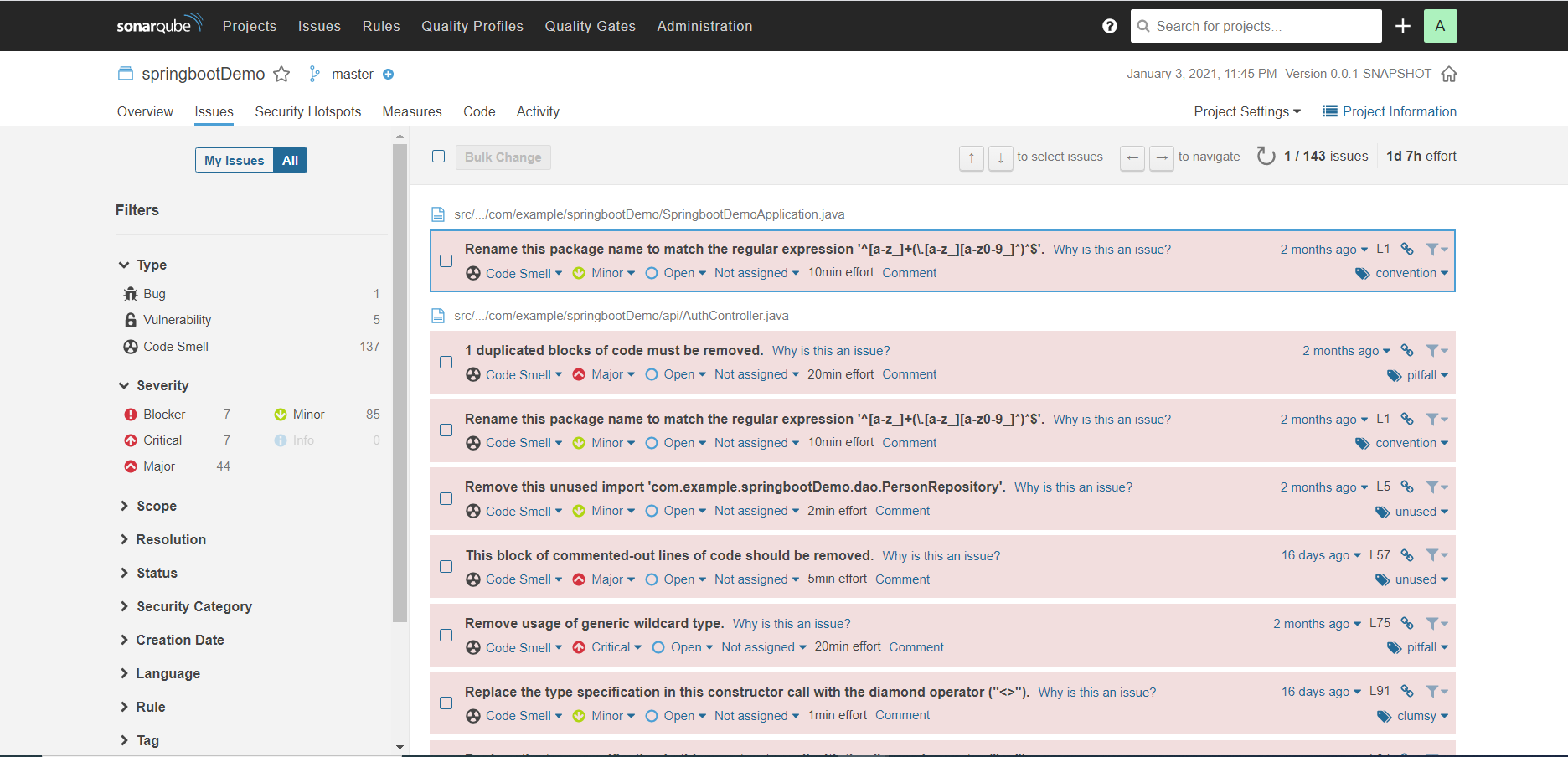
- my git repository:

<https://git.fhict.nl/I431788/individual-assignment.git> - to check my work

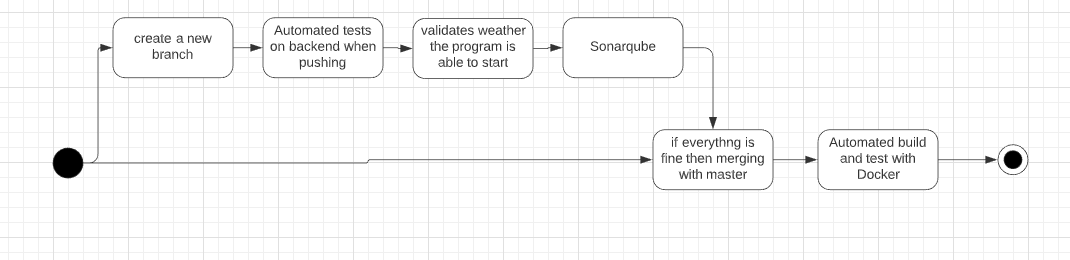


-Sonarqube working correctly as well





CI setup



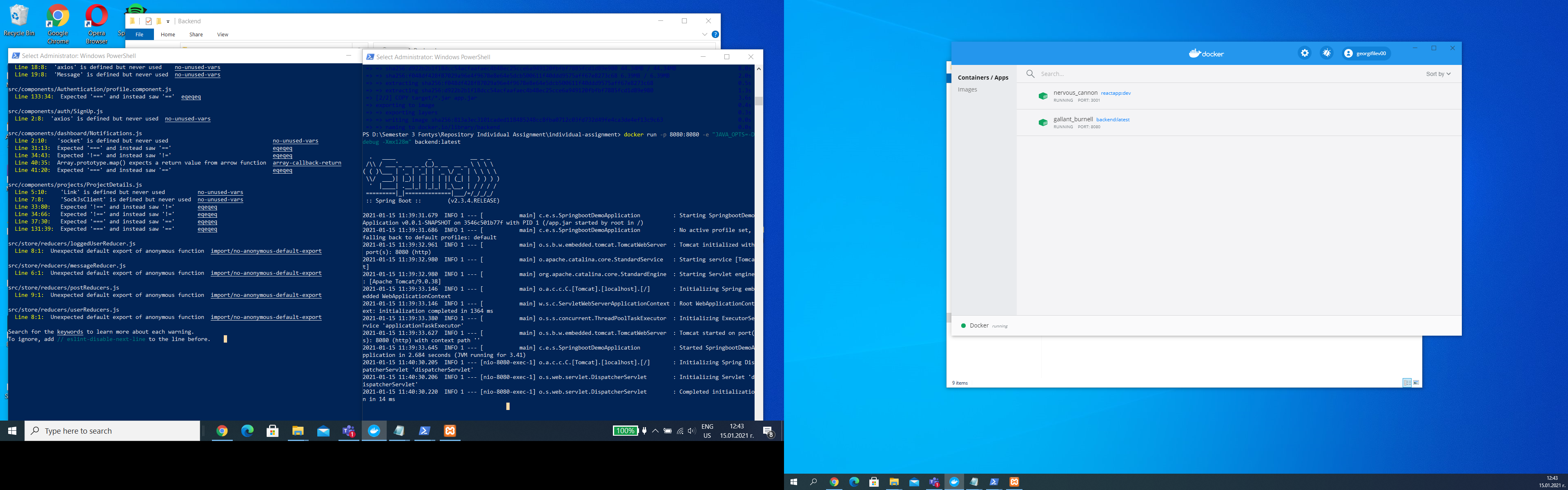
Explanation: When Developing anything which will be merged to the main code it first have to pass every unit test build in the solution before deploying to the Repository, However it is possible that there is some sort of issue which might not pass when automated tests in the GitLab run. When everything goes fine additional test using sonarqube is furthermore exploring the code and give feedback. Then if no issues are found a fusion/merge to the main branch could be done.

I have implemented Mockito tests as well as unit test which take responsibility about different components in the backend. However in the Frontend I am using cypress which looks weather all functions lead to supposed pages/actions.

After the CI by using Docker it takes responsibility about the Continues Deployment.

Docker





Dockerfile for backend :

*FROM adoptopenjdk/openjdk11:alpine-jre*

*ARG JAR\_FILE=target/\*.jar*

*COPY ${JAR\_FILE} app.jar*

*ENTRYPOINT ["java","-jar","/app.jar"]*

Dockerfile for frontend:

*# pull the base image*

*FROM node:alpine*

*# set the working direction*

*WORKDIR /app*

*# add `/app/node\_modules/.bin` to $PATH*

*ENV PATH /app/node\_modules/.bin:$PATH*

*# install app dependencies*

*COPY package.json ./*

*COPY package-lock.json ./*

*RUN npm install*

*# add app*

*COPY . ./*

*# start app*

*CMD ["npm", "start"]*