# Web Platform Development 2 M3I322955 – Group Report

## Bartosz Gajda – S1631175 Mateusz Hawrot – S1633727, Group I

Code Repository - https://github.com/bartoszgajda55/WPD2  
Deployed Application - https://wpd2.herokuapp.com/

## Introduction

The purpose of the project was to create the milestone planner application using Java programming language. The functionalities required for the project were to provide for the user authentication, which will allow creating an account and log-in. Furthermore, the application shall provide addition and subtraction of milestones. There shall be the ability to share the milestones to a different user. For the project, we decided to use Spring Boot 2 framework, integrated with Thymeleaf 3 template enging, and implementation of Model-View-Controller architecture.

## Link Sharing Implementation

Designing and implementing the Link/Project sharing feature was the single most difficult task in the whole project. The solution that was suggested in the project specification, which is sharing the unique link for a Project didn’t seem for us as a feasible solution. The main problem is that discovering the links and the sharing mechanism would have to be done outside of the main application. The Users would have to exchange the links using external messenger or mailer. This makes the application only partially functional, and do not support the users in process of effective managing the projects.

The solution that we have discovered as being most user friendly, while being a part of the application was the Project Sharing mechanism. The foundation of this method lays in the ability of the Project to be shared. Each of Project entity in the database, has a field that stores the list of users that have a shared access to it. This allows the application and Project owner to easily manage the other Users which should an access to certain Project.

The Project creator has admin like permission to manage the Project. There isn’t any mechanism built in to grant such permission to other Users. The Users which have the Project shared with them have only read permission to it. They can’t edit or delete the project. That is not the case though in the Milestones. This feature of the Project is open to be managed by any User. He or she can add, edit and delete the Milestones as needed. Every Milestone has also a field called *completed* associated with it. This indicates whether the Milestone should be checked as completed or not. Rest of the implementation is pretty similar to rest of the application.

The implementation of sharing mechanism starts with Project Model. The model consists of couple of fields, one of each is the *sharedWith* which is an array of Object Identifiers. Each id is equal to one User who has a read access to the Project. The next element is the Project Repository, which handles the most basic operations on the database. Another element is the Project Service. This element is responsible for handling all of the more complicated business operations. Next part if the Project Controller. The controller has a task of intercepting the HTTP requests, processing them and sending the response as appropriate.

In the end the solution that we have implemented proved to be a very good and very useful feature. Adding the Users to the Project is very easy to understand and fast to get up and running with. The User can always see which projects are shared with him and start collaborating right away.

## Persistence Mechanism

The persistence layer of the application was one of the main requirements. While choosing the right technology we tried to focus on the compatibility with the already chosen Spring Boot, and the learning curve of it. Among the many options there, including the ones suggested during the labs, we have decided to go with MongoDB. MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata [[1]](#footnote-2).

This database is known for great integration with Spring Boot Data package. The Mongo can seamlessly work our framework of choice. The upside of this choice is also that we had some previous experience using it. This biggest advantage of MongoDB is:

* First and foremost, it is very easy to install and setup the MongoDB.
* The very basic feature of MongoDB is that it is a schema-less database. No schema migrations anymore. Since MongoDB is schema-free, your code defines your schema.
* The ability to derive a document-based data model is one of the most attractive advantages of MongoDB. Because, the way it stores the data in the form of BSON (Binary JSON), ruby hashes etc, helps to store the data in a very rich way while being capable of holding arrays and other documents.
* The document query language supported by MongoDB plays a vital role in supporting dynamic queries.
* Very easy to scale.
* Due to the structuring (BSON format - key value pair) way of the data in MongoDB, no complex joins are needed.
* Performance tuning is easy compared to any relational databases.[[2]](#footnote-3)

Our database setup has been developed while having the simplicity in mind. After conducting the object-oriented analysis phase, the main entities identified are: User, Project and Milestone.

The User entity models of course the person interacting with the application. The person has the ability to log in, log out and create an account. The account consists of following fields: id, email, password, first name, last name, bio and list of Projects.

The Project entity models the project object that groups the milestones in thematic group. The project entity allows the owner to create, edit and delete it according to needs. The project entity is built using following fields: id, title, description, start date, end date, list of milestones and list of shared Users.

The Milestone object models the single task that is supposed to be carried out as a part of Project. A single milestone is always associated with certain Project, therefore there is not a possibility to create a standalone Milestone. The Milestone entity consists of the following: id, title, content, deadline and status.

## Application Functionality and Test Reports

Thymeleaf 3 was integrated into Spring Boot application. It is a server-side template engine, which is capable of processing HTML, CSS, JavaScript and CSS files.

Thymeleaf injects its login to the template files without affecting the template from being used as a design prototype.

In order to display the values on the website, properties written in HTML file need to be externalized and replaced with values taken from a certain object. Those fragments usually are called messages, which have a key that identifies them. Those key need to be assigned inside *the:text* attribute.

The example shown below demonstrates the structure of thymeleaf engine inside the HTML code.

The simple form handles the data between plain text passed into form and controller which deals with the data.

Using *the:object* structure, the type of object being created is specified in the form. Moreover, the type of action that will be executed in the Controller is specified by *th:action* structure.



The application functionality implemented into the application allows creating an account, therefore, log-in using the details provided in the registration form.

The user is provided with the ability to create a project that consists of the title, description start and end date (which are optional fields). The project will not be created if there is an existing project consist of the same title in the database. Functionality to edit the project details were implemented, therefore the user may change the title, description, start or end date for each of the projects.

Into each of the project milestones can be added. Milestones can be fully managed by the owner, which mean they can be added, deleted or progress status may be changed. The ability to edit the milestone was not implemented, as it was not specified. The toggle slider implementation provides the functionality to change the milestone status between *to-do* into *done*. Simple javascript code was implemented to handle that process.

## Security Measurements

Implementation of Model-View-Controller provides a key design principle, which makes the application open for extension but closed for modification. Spring MVC framework creates a *DiespatcherServlet* which passes the requests to the handler. The default handler is based on the @Controller and marks as @*RequestMapping* annotations, which binds the value of the query parameter. It enables to create RESTful application, by using @*PathVariable* annotation.

Spring Boot Security dependency was implemented into the project and provides security for all HTTP endpoint with basic authentication. It is fully configurable, which allows allocating the permission to the application bits.

The Authentication Manager was implemented into the project. It is an extension within the Spring Security framework that passes tokens to each of Authentication Providers, which determine whether the token is valid or not. There are three possible returns for each token passed through the Authentication Manager, that is *authenticated*, *unauthenticated* or *cannot be authenticated*. Authenticated means that input was verified and represent valid data. Furthermore, the user will gain access to display the projects.

Unauthenticated will be returned if the data processed are not represented the valid principle, therefore access will be rejected.

The status cannot be authenticated will be returned if the data requested for authentication will not be passed to *ProviderManager*. An authentication exception will be thrown in that case.

## Reflection on the tools used in the development

Git was used for version control. It enabled the functionality of managing the iteration of the project, keep each implemented functionality on a separate branch, which allows to keep the application up and running on the master branch, while an error occurs on during implementation of new features.

GitHub Platform was used to store the code in the cloud. It enabled sharing the code between group members and merging the branches into one if no error or conflict occurred. GitHub also provides the feature to review the code stored into the repository. It allowed us to mark the error and bug, which progressed towards application development

Kanban board provided on the GitHub platform was used in order to indicate the progress of the project or status of a particular task. Each branch might be added as a task on the board.

Heroku Platform that offers deployment of the code in the cloud was used in order to deploy the application.

During the development, the application Robo 3T was used. It is a free tool available on all desktop platform, which allows reviewing the records stored in MongoDB.

1. https://en.wikipedia.org/wiki/MongoDB [↑](#footnote-ref-2)
2. https://www.studytonight.com/mongodb/advantages-of-mongodb [↑](#footnote-ref-3)