**Bwise**

Final Year Project 2017

Riona Greally – G00325504

Vaida Albekyte – G00328909

# Introduction/background:

Bwise is a Web application for users to track their day-to-day spending. We are creating this app as part of our Final year project. We sat down together to brainstorm ideas and decided to go with a budgeting application so its users could track their spending daily, weekly, monthly and yearly.

# The Name: “Bwise”

Once given go ahead for our app from our Supervisor Dr John Healy, we decided we needed to decide on a name for our budgeting application. We came up with several different names but decided to go with the name Bwise.

Why we decided on Bwise :

* B stands for Budget -> *”Budget Wise”*
* The idea behind the application is to **Be Wise** with our money and spending.

# Aims:

By the time we complete the project, we aim for our application to include some features such as:

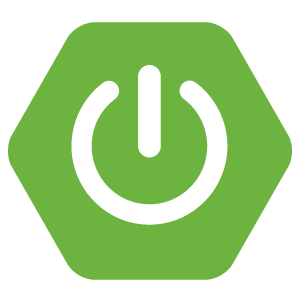
* We aim to create an application for a user to be able to track all their spending’s in one place. The User can enter the amount and description which would be saved to a database and then be displayed on a table along with the date and time stamp and ID number.
* Secondly, we want the user of the application to be able to update or delete any of the transactions saved in the database.
* Calculate all the transactions inputted by the User to give a total amount of their expenditure.
* Allow the user to be able to search their spending’s by date and also, allow the user to create a PDF file of Results that they can print if they so wish.

# Technology Used & Why:

We decided to create our *‘BWise‘* Application using **Spring Boot with Java** along with **MySql** database, along with a **Bootstrap** design. We chose these Technologies for our application for many different reasons.

**Spring Boot:**

Spring Boot is among the new features of Spring Framework version 4.0. The main objective of Spring Boot is to provide quick and simple java applications using an embedded server, Tomcat.

Spring Boot is not a framework to write applications, it is a feature that helps you build, package and deploy your Spring application with minimal configurations.

We chose to use Spring Boot as it is one of the most popular Java frameworks in the market. Before this semester, we hadn’t used the framework Spring before and decided it would be a good technology to expand on and gain some experience in.

**MySql:**

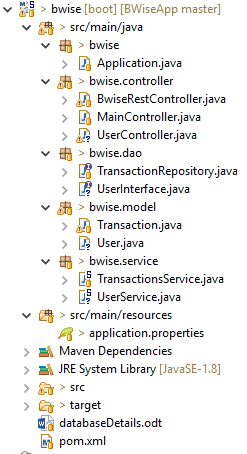
MySql is an open-source relational database management system based on Structured Query Language (SQL). It is ranked as one of the most popular database engines along with Oracle. It is written in C and C++, and runs virtually on all platforms like Windows, Linux etc.

We picked to work with MySql for our database for several different reasons, such as:

* It is ranked very high in use in Industry
* We have prior experience in working with MySql databases
* Wide range of documentation available
* It has several different tools we could use to work with our database, like MySql Workbench

# Architecture of Solution:

Here is the structure of our Spring Boot Application:



We have separate package folders for the different types of classes. We have done this in order to keep our project folder neat and tidy and also for it to be easy to understand to anyone looking at the application for the first time.

The POM (Project Object Model) file, is the core of the project configuration in Maven. It is a single file that contains the information for in which we want to build the project. This is where we inject the required dependencies needed for our application to work the way we want.

Some of the dependencies we injected are:

* Spring-boot-starter-web
* Spring-boot-starter-data-jpa
* Mysql-connector-java

**Application.java**

This is a single Java file that contains a main method. Here we are stating our project at a Spring Boot application using the *@SpringBootApplication* annotation.

*@SpringBootApplication* indicates a configuration class that declares one or more *@Bean* methods and also triggers the auto-configuration and component scanning.

*SpringApplication.run(Application.****class****, args); is in the main() method of the class. The main() uses this line to launch the application.*

**Application.properties:**

Here we are referencing properties such as the view paths and database details.

This line states the Spring MVC view prefix, meaning /WEB-INF/jsp/ is the path in which the view is located.

spring.mvc.view.prefix=/WEB-INF/jsp/

Here states the Spring MVC view suffix, meaning the the type of view file is and ends in ‘.jsp’.

spring.mvc.view.suffix=.jsp

We are also telling Spring the details of our database we are working with in here. We do this with the following lines:

spring.datasource.url:jdbc:mysql://localhost/transactionsdb?useSSL=false

spring.datasource.username:root

spring.datasource.password:

spring.datasource.driver:com.mysql.jdbc.Driver

These lines are describing the URL and name of the database being used and the username and password in which the database is under.

**MainController.java:**

This class is annotated with @Controller, indicating this class serves the role of a controller. This class handles the HTTP Requests using @RequestMapping() methods.

@GetMapping() and @PostMapping() are shortcut annotations for the  @RequestMapping(method = RequestMethod.\*). note: ‘\*’ being GET in a Get Request or POST in a post request).

The code inside the () of the Mapping annotation (eg. @GetMapping("/update-transaction")), specifies the URL path in which the request method will take place.

**Transaction.java & User.java:**

These are model classes and are each referenced to the tables ‘t\_transactions’ and ‘t\_users’ in our database by configuring it using the @Entity annotation.

**Database:**

Our database is a MySql Relational Database Management System. We created it and worked on it using the MySql Workbench tool which is a desktop app to aid database developers.

**Database name:**  *‘transactionsdb’*

**Table: t\_transactions**

**Columns:**

|  |  |
| --- | --- |
| **id** | int(11) AI PK |
| amount | int(11) |
| type | varchar(45) |
| date\_transfered | datetime |

**Table: t\_users**   
  
**Columns:**

|  |  |
| --- | --- |
| **acc\_id** | int(11) AI PK |
| username | varchar(12) |
| password | varchar(8) |
| first\_name | varchar(20) |
| last\_name | varchar(20) |

# Limitations:

In the process of the project, the main limitation we came across was that there are limited resources to Spring Boot resources and guidelines compared to other features. This is because Spring Boot is recent feature in the Spring framework.

Also we felt limited as it was a new technology for us. This meant it took time to learn and understand most features.

# Known Bugs:

The main known bug in our application is the Log in. We got a registration form working where the users’ details are saved to the database, however we did not get the full login to work, meaning the application is a single user system.

Also, another bug in the application is that the password in which the user can register is not encrypted, it is just hard-coded, which is not good security.

# Recommendations for Future Development:

Firstly, we would like to fix the known bugs in our application. This would be to have a working login so the application can be a multi user system and also to have the passwords encrypted in the database.

The we would like to add features like graphs and categories. These graphs would be a visual representation of the users spending’s. The categories would split the transactions up in to suitable categories like social, groceries etc. and then the user could see where exactly they are spending most of their expenditure and where they could cut back.

# Conclusion:

Overall, throughout the development of this project, we both feel we have learned a lot with working with Spring Boot and the MySql database. However, we do feel that there is a lot more about these technologies, especially Spring that we could learn and gain experience in.