**Student Name: Tianle Shu**

**Supervisor: Kevin O’Brien**

**Shudent ID: G00353418**

**Program: Bachelor’s in Software Development Year 3**

**Module: Proffesional Skills in IT**

**Submission Date: 23th April 2018**

**GitHub Link:** [**https://github.com/Tianle97/IT-Project\_DiaryApplication**](https://github.com/Tianle97/IT-Project_DiaryApplication)

# 

# Table of Contents

[Table of Contents 2](#_Toc511815111)

[Introduction 3](#_Toc511815112)

[Technologies and Architecture 3](#_Toc511815113)

[Service Side Technologies 3](#_Toc511815114)

[Service Side Architecture 4](#_Toc511815115)

[Knowledge 8](#_Toc511815116)

[Recommendations for Future Development 11](#_Toc511815117)

[Debug 11](#_Toc511815118)

[Conclusions 11](#_Toc511815119)

# 

# Introduction

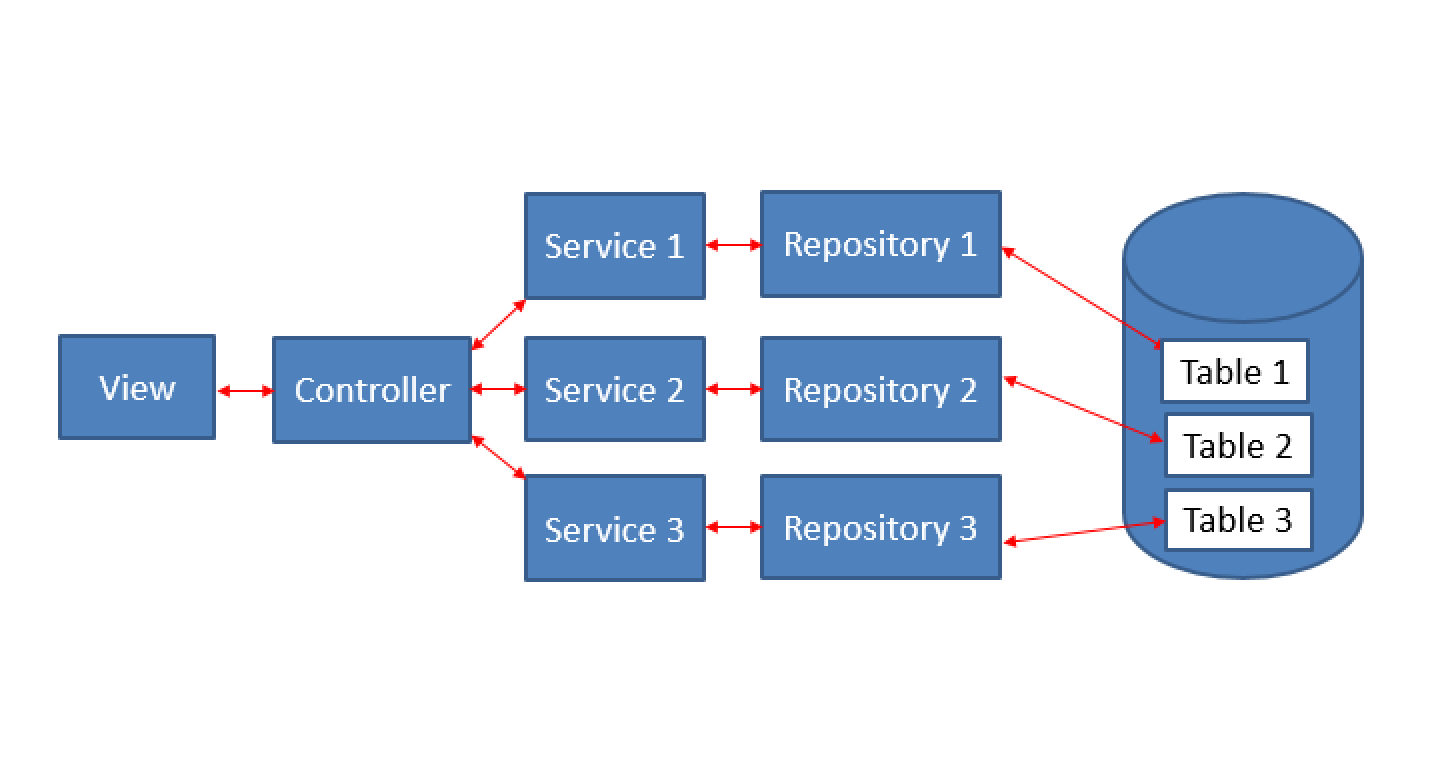
This project is the software development 3rd Year IT Practice Project.

Diary Application

There is a login page, registration page. When you regist success then will go to a ‘Welcome’ page for 3 seconds , then jump to diary page for user to write their diary and can list their diary title and can also delete and edit it.

GitHub Link: <https://github.com/Tianle97/IT-Project_DiaryApplication>

# Technologies and Architecture



### Service Side Technologies

#### *Spring Framework*

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications – on any kind of deployment platform. A key element of Spring is infrastructure support at the application level: Spring focuses on the “plumbing” of enterprise applications so that programmers can focus on application-level business logic, without unnecessary ties to specific deployment envitonments.

**Features**

* Dependency Injection and control flow
* Spring MVC web application
* Foundational support for JDBC, JPA
* And more…

All available features and modules are described in the [Modules section of the reference documentation](http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/overview.html#overview-modules) for Spring.

There are several other competing frameworks like Java EE(JSF) (I used it for different purpose this time) that can provide Service similar way.

During a last semester, I have developed applications using Java EE, so I picked Spring framework to expand our knowledge in modern technologies.

#### *MySql Database*

I am using MySql database to store user details for it’s convenient way of storing data in tabular fashion, security features it provides and its capabilities to expand into more complex database, as it supports relationships between data tables.

Currently I have one table to store registered users.

#### *Maven package builder*

To glue things up, I use Maven package builder to handle necessary dependencies required for the project and it also is an easy way to share the project among the team members, as it uses a common package structure and easly imported to a workspace of common IDEs.

All the dependencies are listed in a *pom.xml* file in a root folder of a project.

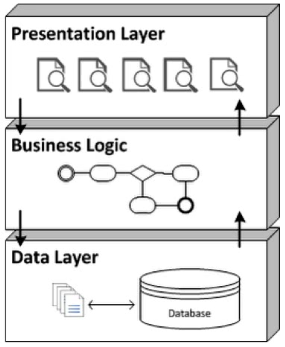
***Lombok Library***

Lombok is one of the tools that is used to avoid repetitive code, also know as boilerplate code.

For example, a Model class usually consists of many properties that require getter/setter methods. To make class tidier, with less lines of code, I can annotate parameters with @Getter @Setter annotations to ommit getter/setter method definition for a property.

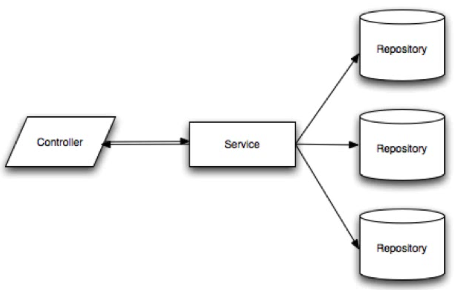
For more cool stuff from Lombok, visit their project [website](https://projectlombok.org/features/).

### Service Side Architecture



Benefits of N-Tier applications:

* Better Security
* Scalability
* Simpler Maintenance
* Easily Enhanced



* Controllers – presentation layer
* Services – business logic
* Repositories – data layer

**Key annotations and definitions in Spring**

Dependency Injection – also called Wiring, helps in gluing java classes together and at the same time keeping those classes as independent as possible to increase reusability and ease of testing.

Spring Beans – Java Object that is instantiated, assembled, and otherwise managed by a Spring IoC container. To become a Spring Bean, java class has to be either defined in XML file (in Spring versions prior to Spring Boot), marked with one of the following steriotype annotation

* @Component
* @Controller
* @Service
* @Repository

or defined in configuration class (marked with @Configuration annotation) as @Bean method.

@Autowired – annotation that performs dependancy injections of Spring Beans.

Autowired annotation can be used on properties, setter methods or class constructors with parameters to instantiate an object of a certain type. By default, Spring autowire objects by its type.

@Qualifier – in java, it is possible to have many objects of a same type. To use a specific object out of this group, autowiring can be done by name of a bean in combination with @Qualifier(“beanName”) annotation.

To give bean a name, you have to add *name = “beanName”* to a steriotype annotation mentioned above, for example:

@Component(*name = “beanName”*)

public class MyClass {

// class body

}

and autowiring will look like this:

@Qualifier(“beanName”)

@Autowired MyClass myClass;

**Controllers –** provide an URI end points to a Client applications to request data. Controllers “presenting” data to a client based on requested URI.

To make a Rest Controller in Spring Boot application and make it respond to requests, variouse Spring annotations are used:

@RequestMapping – annotation that links a URIs to a class or method inside the controller.

It is possible to mark entire class to have a specific URI and methods to extend from this URI to form a pattern to a resources.

For example, I can mark a class with @RequestMapping(“/chatroom/”) and a method with @RequestMapping (“/public”). The full URI to a public chat room would look like this: *…/chatroom/public*

Alternative to @RequestMapping annotation, you can use annotations like:

@GetMapping – is itself annotated with @RequestMapping(method = “GET”)

@PostMapping – @RequestMapping(method = “POST”)

to narrow down HTTP methods that end-point can process (GET, POST, DELETE, PUT, etc)

**Services –** indicates business logic.

@Service annotation makes the class autowireable. Now Spring treat Service class as Spring Bean and can inject it as a dependency into our controllers using @Autowired annotation.

In our Server, both *UserServices* (links to Sql Repository) contains the methods to call operations on databases.

**Repositories** – interfaces for quering databases

In our project, I am using two types of repository interfaces from Spring framework:

* JpaRepository

This allows us to use implemented methods from Spring framework to query database, as well as defining our custom methods to query data by a particular parameter. There is no need for a programmer to write MySql statements, although you can define a custom methods with custom statement implementation for more complex quering.

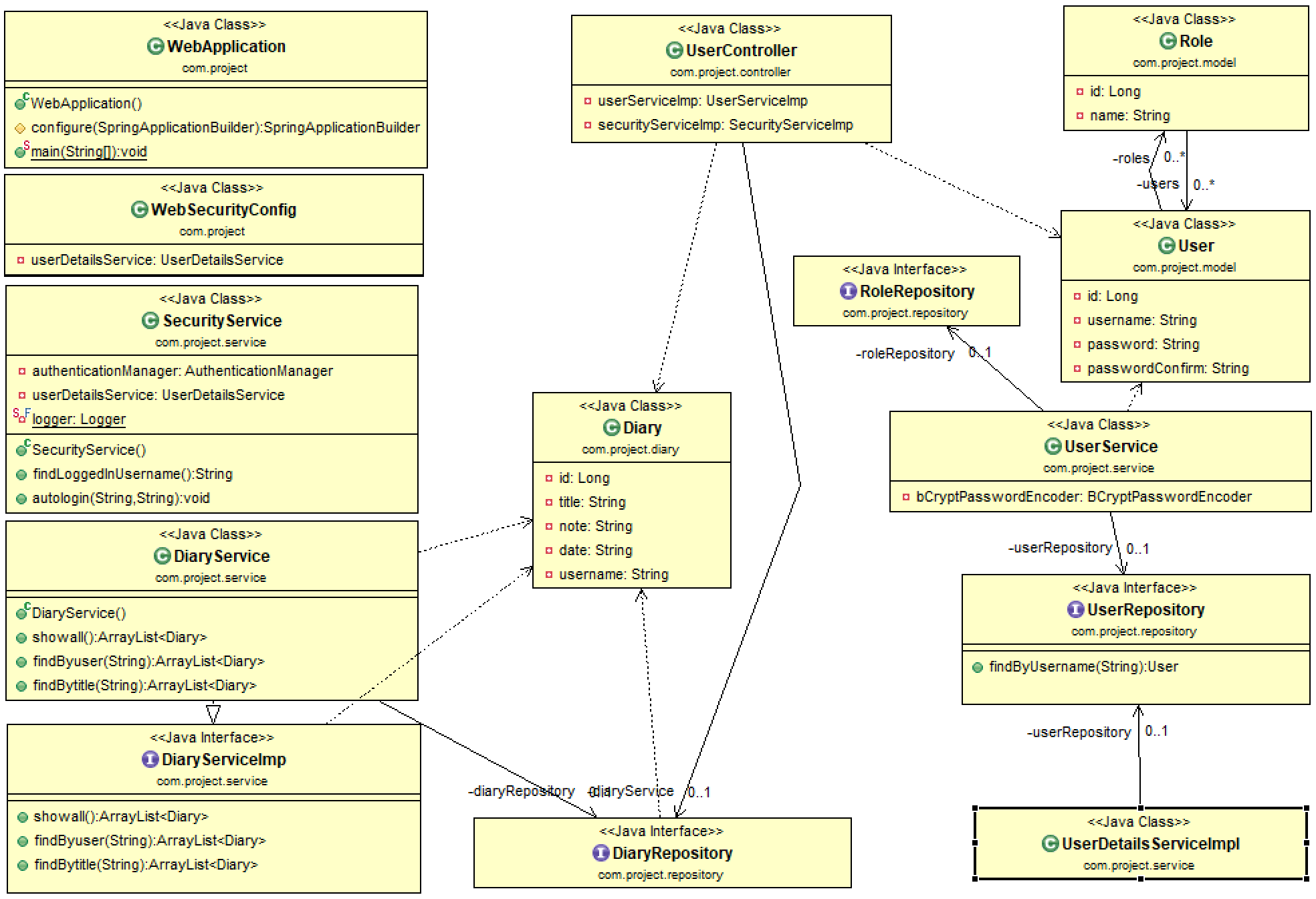
**JPA – Java Persistence API**

It is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational databases like MySql. It can be performed in XML or Java Annotations.

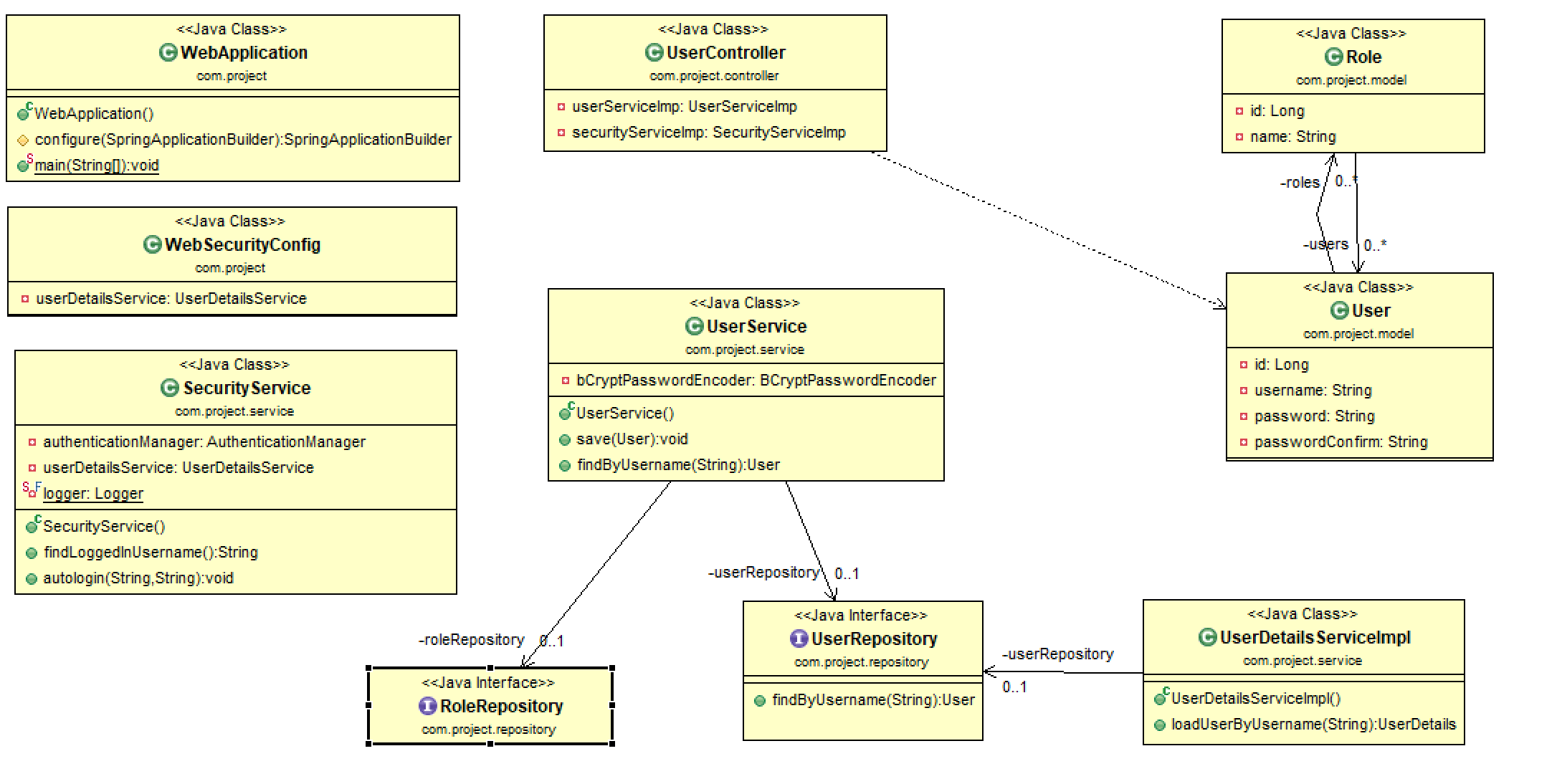
JPA Annotations for Models:

* @Entity - declares an object as an entity that the database should be aware of.
* @Table - describes more specific details about the entity e.g. table name.
* @Id – Identifier attribute for a Primary Key.
* @GeneratedValue – Used in conjunction with @Id to generate a unique value for a field.
* @Column – describes more specific details about the column e.g. column name.

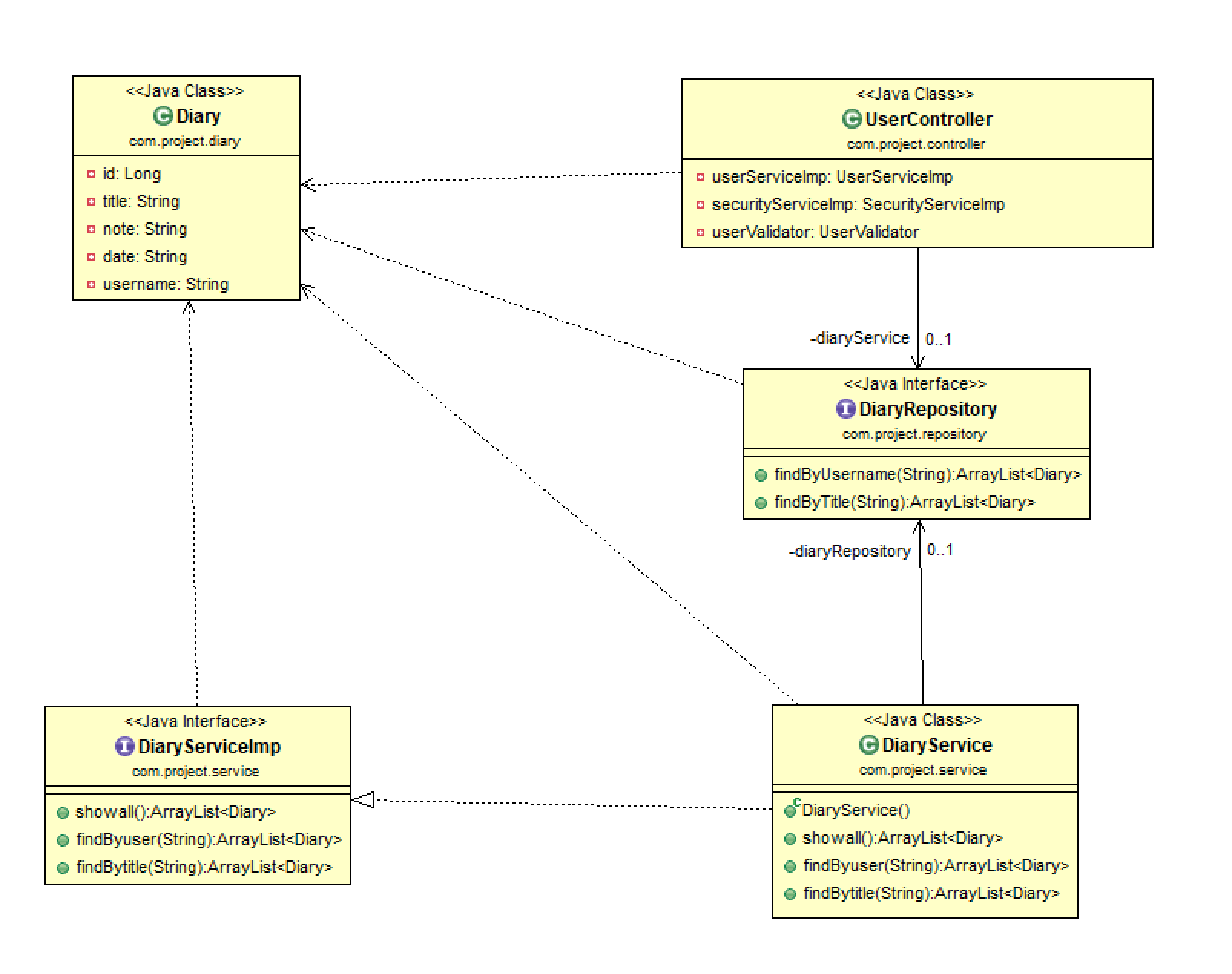
**Full Service Architecture UML Diagram**



**Log-in & register UML diagram**



**Write Diary UML diagram**

****

# Knowledge

#### *Validate*

Serverside validation is not only a way to prevent eventual attacks on a system, it also helps ensure data quality. In the Java environment [JSR 303 Bean Validation](http://docs.jboss.org/hibernate/beanvalidation/tck/2.0/reference/html_single/) and the javax.validation packages provide developers with a standardized way of doing so. Fields that have to fulfill certain criteria receive the corresponding annotations, e.g. @NotNull, and these are then evaluated by the framework. Naturally, for checking more specific conditions, there is the possibility of creating custom annotations and validators.

The Spring framework has a good Bean Validation integration. It is e.g. possible to validate an incoming request inside a RestController by adding the @Valid annotation to the request parameter.



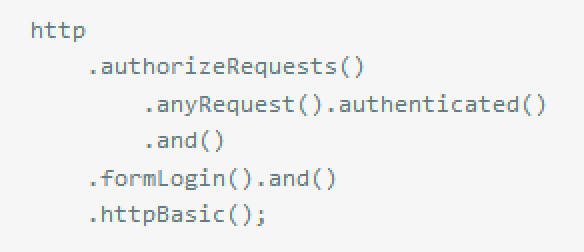
#### *SecurityConfig*



1. Turn on Spring Security's features with the @EnableWebSecurity annotation. Using the @EnableGlobalMethodSecurity(prePostEnabled = true) annotation, you can turn on annotations for security. We can use @PreAuthorize, @PreFilter annotations for methods that require permission control.

2.extends extends the WebSecurityConfigurerAdapter class and overrides its methods to set some web security details. We combine the @EnableWebSecurity annotation with the inherited WebSecurityConfigurerAdapter to add web-based security to our system.

3. In the configure(HttpSecurity http) method, the default authentication code is:



From the method name we can basically understand the function of these methods. The default login page above is the default login password authentication page for SpringBoot. Its source code is as follows:



# Recommendations for Future Development

#### *Better Design Plan and Scheduling*

I would definatly recommend investing time into design planning of a project and well thought out scheduling plan for tasks to be accomplished.

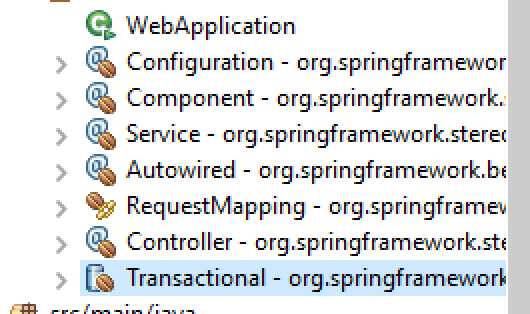
Important aspect is gathering of business requirements, what program should be able to do and what client want it to do. Than, from our experience of developing this project, it is much easier to pick a stack of technologies and plan design, and architecture for the project.

Following a schedule improves performance, saves time and money, because developers know exactly what should be done and by when. Development cycle can be broken up into milestones that indicates the state of development cycle.

#### *Desirable functionalities*

* Add the Research function for user can easier to find their diary.
* Add emai validate and if user forgot their password they can use their email address to find their password and also can reset their password.
* Deploy this project to the aws (cloud) so that can use it anywhere

# Debug

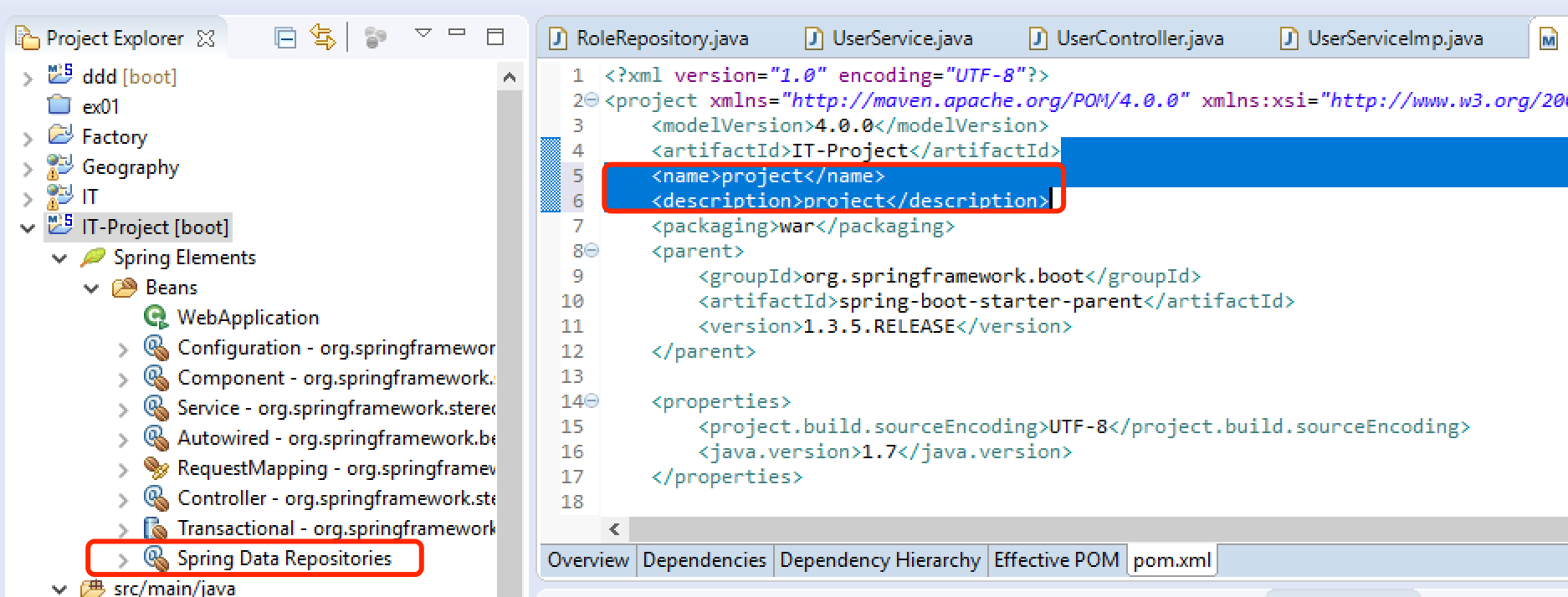


need Transactional - …

And Spring Data - …

Delete 2 lines

And then will have the Spring Data Repository of beans



# Conclusions

During this project development, I was able to find a use of our knowledge obtained from previous years modules, this semester modules, as well as learning new technologies not covered on a course.

From this experience, I have concluded that it is worth spending more time in researching Spring framework capabilities, as it can provide much more fundament for an application development.

Spring framework became my favorite Java Framework to develop with and I would definatly consider to develop other projects using it.

**Learning outcomes**

* Object Oriented Design Patterns – to design a robust, reusable, scalable, adaptable to a change applications.
* Spring Framework – a powerful Java framework that allows to write web Services.

Thank You for attention,

Tianle Shu.