<Assignment 2>

Analysis and Design Document

Student: Moldovan Anda

**Group:30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The application’s purpose is to track laboratory activity for the Software Design laboratory. It has two types of users, student and teacher, both of which can perform specific operations.

# Functional Requirements

All the data is stored in a MySql database, on the server side.

Both types of users have an account which they login into the system and are able to perform their operations.

The teacher can:

* Login with the username and password.
* Add, update or delete students. In order to add a student an email is required, after which a token is generated into the database, for that student.
* Add, update or delete laboratory classes.
* Add, update or delete attendances for each lab.
* Ass, update or delete assignments. An assignment is assigned to a laboratory, even though it has a deadline, which can be different from the laboratory. Also, more assignments can be assigned to the same laboratory.
* Grade the student assignments. It is possible to regrade the same assignment three times, after which it becomes impossible to modify the grade. The last grade that is added for an assignment, remains in the database as the actual grade for that assignment.
* Get a list of grades, for all students for a given assignment.

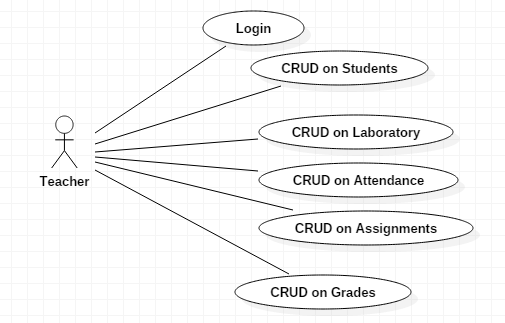
The student can:

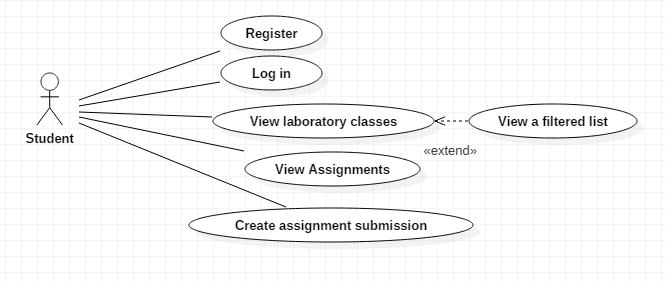
* Register on the platform using the email registered by the teacher and the token generated automatically.
* Login with the username and password.
* View a list of laboratory classes. Also view a filtered list of laboratory classes bases. The filtering is done in the laboratory curricula and description, based on a keyword the student provides.
* View assignments for a laboratory class.
* Create an assignment submission, where they provide a link to a github page and a note for the teacher.

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

2. Use-Case Model





Use case: Login

Level: user-goal level

Primary actor: Teacher, Student

Main success scenario: User is able to login into the system

Extensions: User is unable to login

Use case: CRUD on students

Level: user-goal level

Primary actor: Teacher

Main success scenario: operation on student table executed successfully

Extensions: failure to execute operation on student table

Use case: CRUD on laboratory

Level: user-goal lever

Primary actor: Teacher

Main success scenario: operation on laboratory table executed successfully

Extensions: failure to execute operation on laboratory table

Use case: CRUD on attendance

Level: user-goal lever

Primary actor: Teacher

Main success scenario: operation on attendance table executed successfully

Extensions: failure to execute operation on attendance table

Use case: CRUD on assignment

Level: user-goal lever

Primary actor: Teacher

Main success scenario: operation on assignment table executed successfully

Extensions: failure to execute operation on assignment table

Use case: CRUD on grades

Level: user-goal lever

Primary actor: Teacher

Main success scenario: operation on grade table executed successfully

Extensions: failure to execute operation on grade table

Use case: Register

Level: user-goal lever

Primary actor: Student

Main success scenario: student is able to register using the email and token generated by the teacher.

Extensions: failure to register

Use case: View laboratory classes

Level: user-goal lever

Primary actor: Student

Main success scenario: student is able to view the laboratory classes

Extensions: failure to view the laboratory classes

Use case: View a filtered laboratory list

Level: sub-function lever

Primary actor: Student

Main success scenario: student is able to view a filtered list of the laboratory classes

Extensions: failure to filter the laboratories

Use case: View assignments

Level: user-goal lever

Primary actor: Student

Main success scenario: student is able to view the assignments for a laboratory

Extensions: failure to view the assignments

Use case: Create assignment submission

Level: user-goal lever

Primary actor: Student

Main success scenario: student is able to create a submission for an assignment

Extensions: failure to create the submission

3. System Architectural Design

**3.1 Architectural Pattern Description**

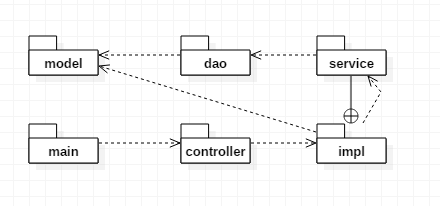
The architecture used is MVC – Model, View, Controller, although only the backend part was developed, consisting of Model, Controller and Repositories.

MVC pattern is used to separate the concern in an application:

* Model – represents an object or Java POJO carrying data.
* Repositories – represents the way to get the data from the database. In this application the repositories were simplified, because of the use of Hibernate ORM and Spring framework, which offer a way of getting the data, without explicitly writing the queries.
* Controller – acts on both the model and view. It controls the data flow into model and updates the view accordingly, whenever data changes. It keeps the view and model separate.
* View – represents the visualization of the data in the model. In this application, the view was implemented using the Swagger, to call the APIs made in the controllers

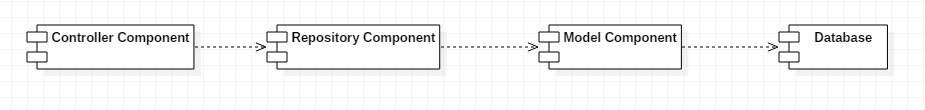
**3.2 Diagrams**

Package diagram

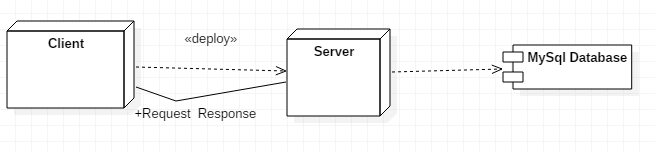


The division between model and controller can be seen in the package diagram. Each main component has classes in each package. The controllers sit at the top of the hierarchy. They are responsible to pass on requests to the corresponding models and receive the responses from them.

Component Diagram



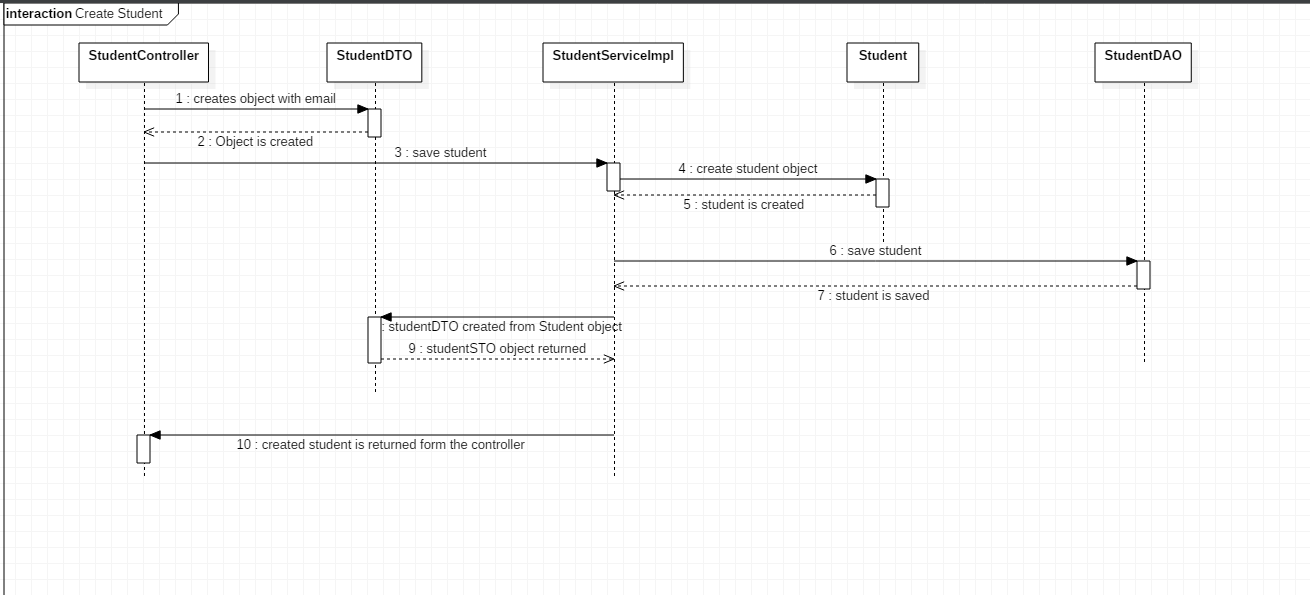
Deployment diagram



The server component is composed of multiple servers. MySql Server is used for the connection with the database.

Tomcat is used for the Spring Framework, and Swagger. This is why the swagger runs on localhost on port 8080 (tomcat).

4. UML Sequence Diagrams



Chosen scenario: Teacher creates a student.

The request comes form the student controller. The teacher creates a student with only the student email.

From the controller, the request goes to the student service implementation, where an object of type StudentDTO is created with the email and a token which is generated automatically.

The student service implementation uses studentDAO to save the student to the database. In order to sent the object to Student DAO, it is first transformed in a student object.

The id in the database is auto generated, so there is no need to worry about that.

5. Class Design

**5.1 Design Patterns Description**

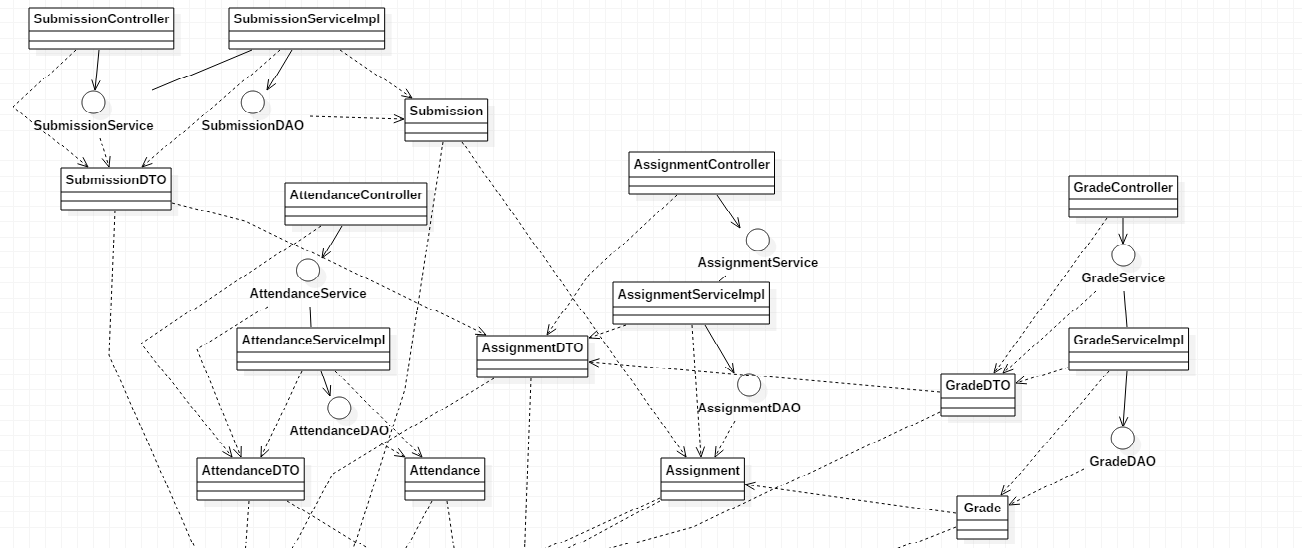
Design patterns used by Hibernate:

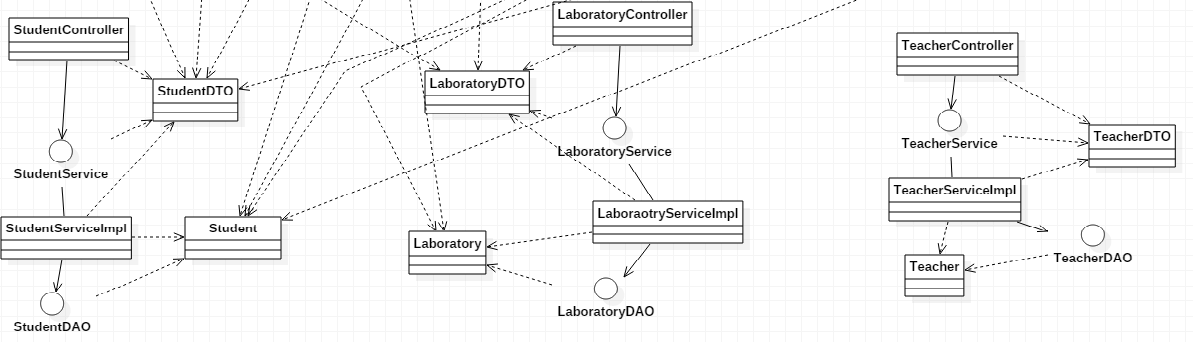
* Domain Model Pattern – An object model of the domain that incorporates both behavior and data.
* Proxy Pattern – used for lazy loading
* Data Access Object (DAO) – used to separate low level data accessing API or operations form high level business services. This design pattern contains the following participants:

1. Data Access Object Interface – defines the standard operations to be performed by model objects.
2. Data Access Object concrete class – implements the above interface. It is responsible to get data from a data source, which in this case is a database.
3. Model Object – this object is a simple POJO containing get/set methods to store, retrieve data using DAO class

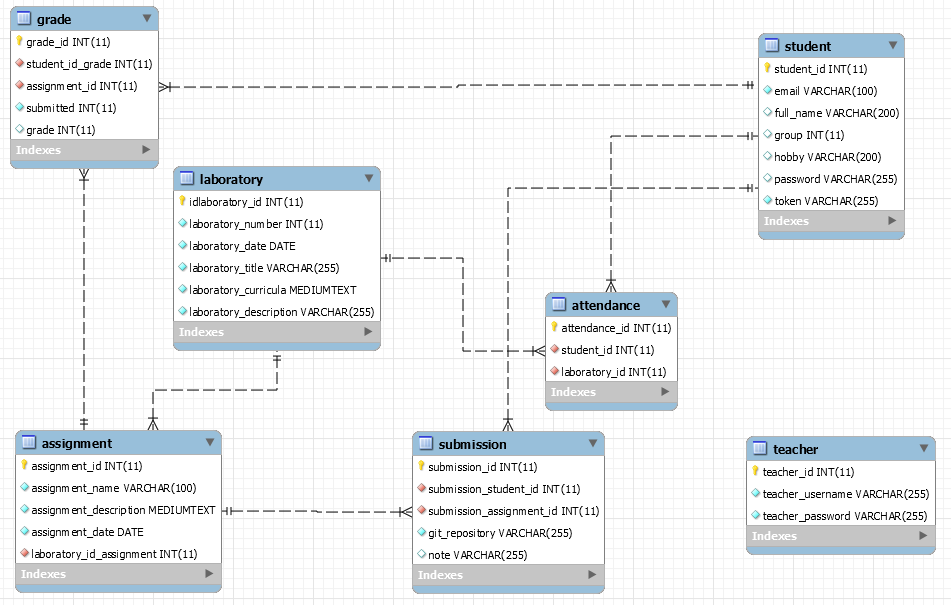
* ORM (Object Relational Mapping) – converts data between incompatible data types. (from a database object to oop object).

**5.2 UML Class Diagram**





6. Data Model



7. System Testing

Testing was done manually, for each component as they were written.

Testing is made easy with the use of hibernate, because the repository classes have basic crud operations already implemented, ad other basic queries are figured out by hibernate orm, if they are not too complicated.

8. Bibliography

* <https://spring.io/>
* <http://hibernate.org/>
* <https://swagger.io/>
* <https://github.com/g-nico/Spring_presentation/tree/master/src/main/java/com/spring/presentation>