<Assignment 2>

Student:Moldovan Ana Olivia

**Group:30231**

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

Use JAVA Spring/C# Web API to design and implement an application for the tracking the laboratory activity for the Software Design laboratory. The application should have two types of users (student and teacher) which must provide an email and a password to use the application.

# Functional Requirements

The teacher can perform the following operations:

- Login

- CRUD on students. For each student we should track: email address, full name, group (ex. 30434) and hobby – free field.

- Can add/edit/delete Laboratory classes (Subjects of lab classes- eg. Software design, Maths) is out of scope). For each class we should track: Laboratory number (#1-#14), date, title, objectives (string) and a long description with the laboratory text.

- CRUD on assignments. Some of the laboratory will have assignments: for each assignment we must track the name, deadline, and a long description with the assignment text.

- Grade the submitted assignments individually.

The student can perform the following operations:

- Login with the username and password.

- View a list of laboratory classes.

- View the assignments for a laboratory class.

- Create an assignment submission. Here, students should be able to insert a link to a git repository and a short comment (optional) for the teacher.

# Non-functional Requirements

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: use case goal*

*Level: user-goal level, sub-function*

*Primary actor: Teacher*

*Main success scenario: login->enter Assignment(AssignmentId, Name, Deadline, Description, LabId)->assignment is added to database*

*Extensions: login-> incorrect username or password*

*]*

3. System Architectural Design

**3.1 Architectural Pattern Description**

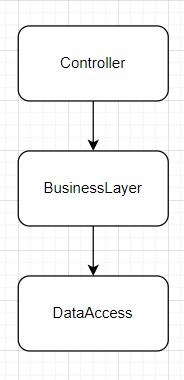
The **Model-View-Controller (MVC)** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects. MVC Components:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

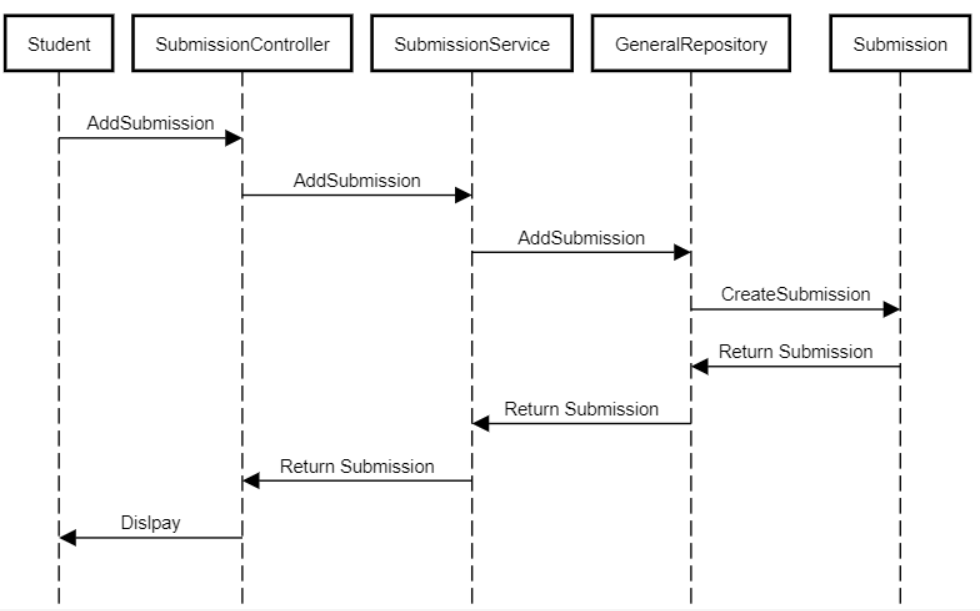
The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

**3.2 Diagrams**



4. UML Sequence Diagrams

**

5. Class Design

**5.1 Design Patterns Description**

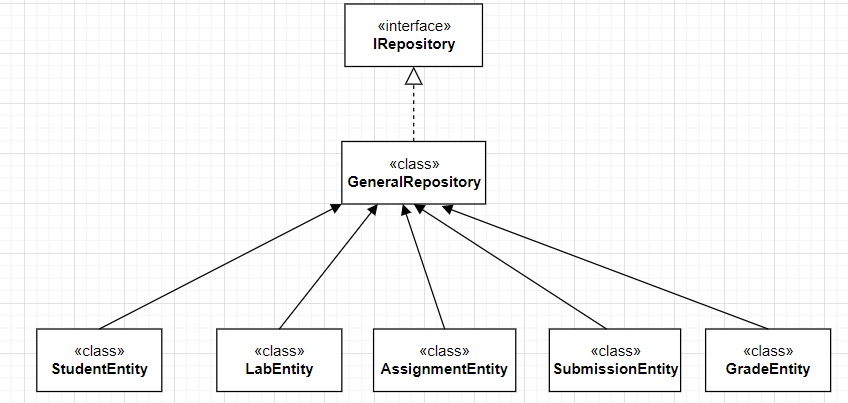
O imagine care conține text, interior

Descriere generată automat

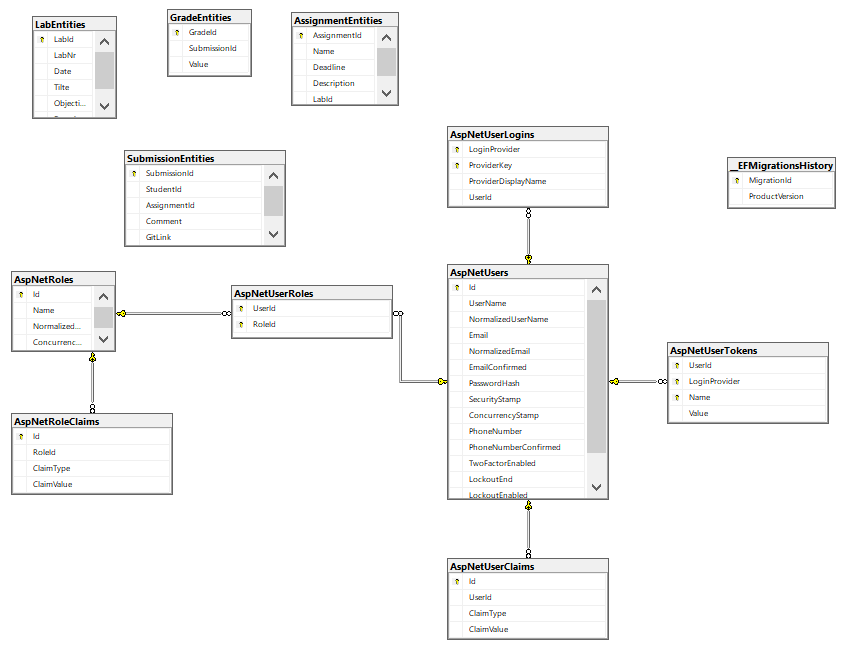
**5.2 UML Class Diagram**

O imagine care conține text, cer, interior

Descriere generată automat



6. Data Model

**

8. Bibliography

<https://app.pluralsight.com/library/courses/implementing-restful-aspdotnet-web-api/table-of-contents>

https://javabrains.thinkific.com/courses/take/springboot-quickstart/multimedia/2784009-adding-a-rest-controller