Laboratory Tracking API
Analysis and Design Document

Student: Avram Andrei-Alexandru
Group:30434

Table of Contents

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 4

3. System Architectural Design 5

3.1 Architectural Pattern Description 5

3.2 Diagrams 5

4. UML Sequence Diagrams 7

5. Class Design 7

5.1 Design Patterns Description 7

5.2 Entity Diagram **Error! Bookmark not defined.**

6. Data Model 9

7. System Testing 9

8. Bibliography 9

1. Requirements Analysis

1.1 Assignment Specification

The goal of this assignment was to implement a RESTful API for a Laboratory Tracking system for one discipline using MVC architectural pattern, services and repositories.

The application was created using the Spring Framework and implements the REST API which enables the possibility for this system to communicate with other systems using its endpoints and predetermined data transfer protocols.

1.2 Functional Requirements

The functional requirements this RESTful API system has are the following:

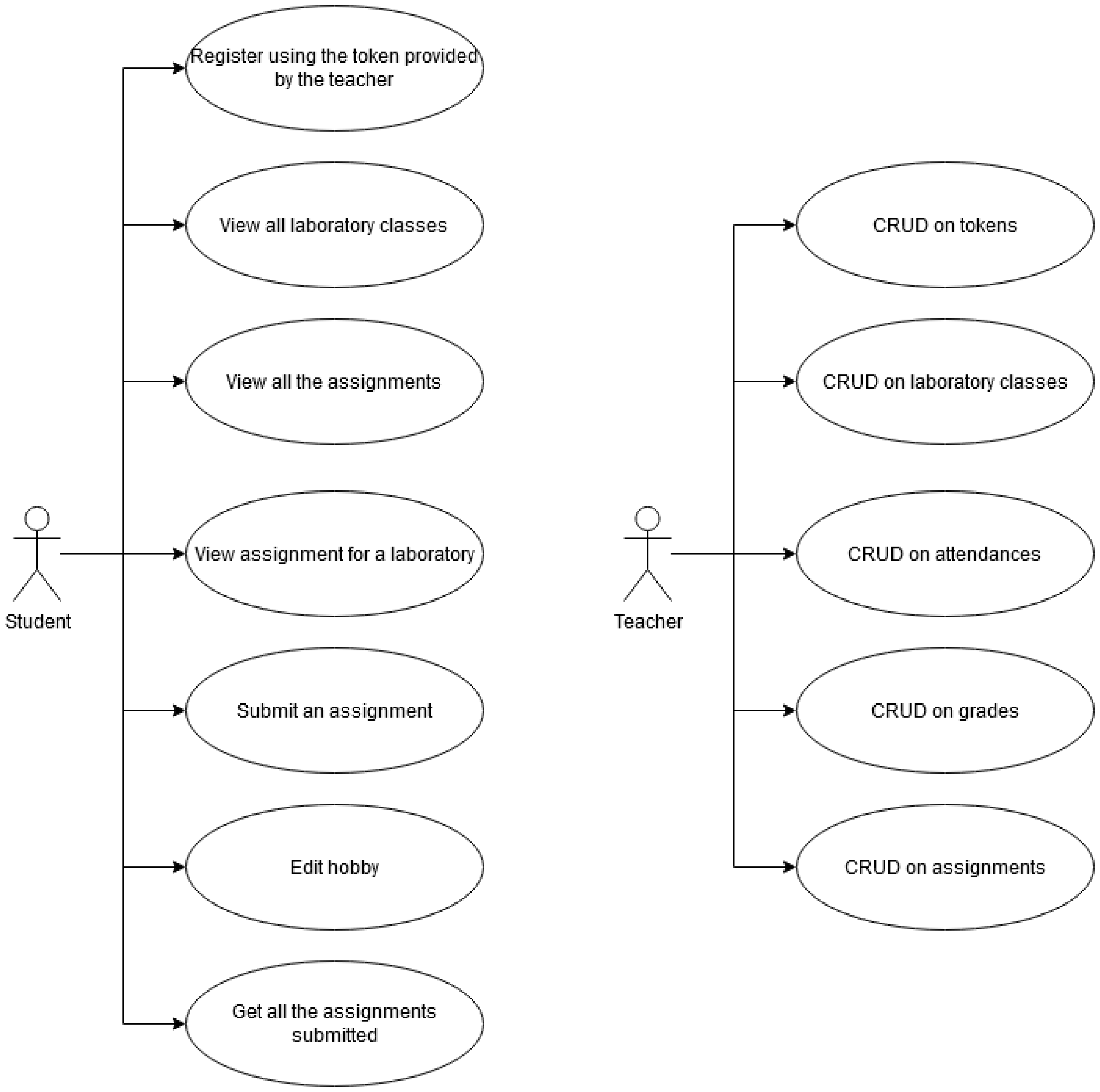
- Provide access to its endpoints using the common HTTP request methods (GET, POST, PUT, DELETE) to get data from the database, create new data, update existing data and delete data. In this context data has the following meanings: students, tokens, laboratory classes, assignments, submissions and grades.
- Prevent CRUD operations on invalid data or non-existent data. For each endpoint the corresponding services will provide validations and in case of invalid data the controller will send back an appropriate HTTP status code.
- The system supports two types of users: Students and Teachers. Every endpoint is secure and only authenticated users and users with correct authorization can access them. The only endpoint that is accessible without authentication and authorization is the one responsible for registering as a student.
- CRUD operations on students, laboratory classes, assignments, submitted assignments, attendance, grades.

1.3 Non-functional Requirements

The non-functional requirements this RESTful API system has are the following:

- **Security**: The system has protection against SQL injection. This is done using parameterized SQL queries. The user passwords are stored in the database using one-way encryption. The algorithm used for encryption is BCRYPT.
- **Availability**: The expected up-time for this system is 99.95%. This translates in a maximum downtime of roughly 22 minutes per month.
- **Performance**: Can work fine under medium load and expected response time is < 100ms.
- **Testability**: The authorization was tested by accessing the endpoints through postman using valid/invalid credentials and different user roles. The system is easily testable.
- **Usability**: This API is easy to use because each endpoint has a specific HTTP request method and an easy-to-use data transfer object model in the form of a JSON file.

2. Use-Case Model


Register using the token provided by the teacher
View all laboratory classes
View all the assignments
View assignment for a laboratory
Student
Submit an assignment
Edit hobby
Get all the assignments submitted
CRUD on tokens
CRUD on laboratory classes
CRUD on attendances
Teacher
CRUD on grades
CRUD on assignments

Use case: Create an assignment for a laboratory class
Level: user-goal level
Primary actor: Teacher
Main success scenario: The teacher enters all the required information for creating an assignment and provided that the laboratory class for which the assignments is created exists then the assignment is successfully created.
Extensions: The laboratory class does not exist or an assignment was previously created or invalid data detected then the user is notified with 406 (Bad request) HTTP status code.

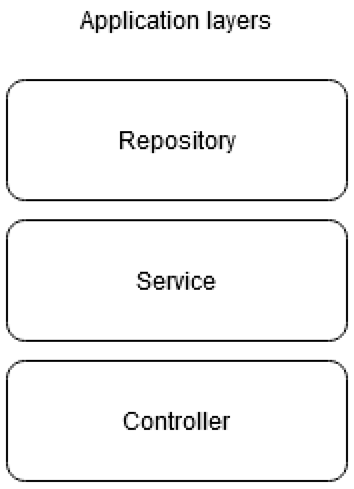
3. System Architectural Design

3.1 Architectural Pattern Description

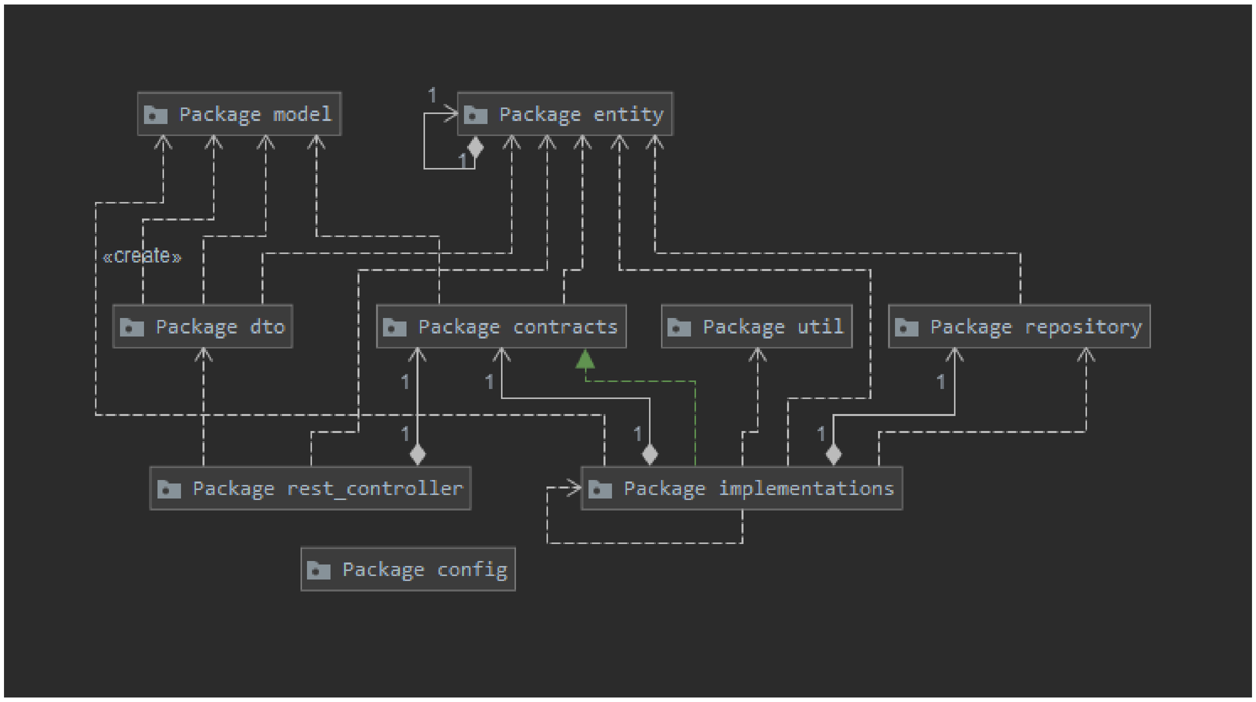
The application uses three-layer architectural design pattern. It is divided in three main layers: Repository, Service and Controller.

3.2 Diagrams

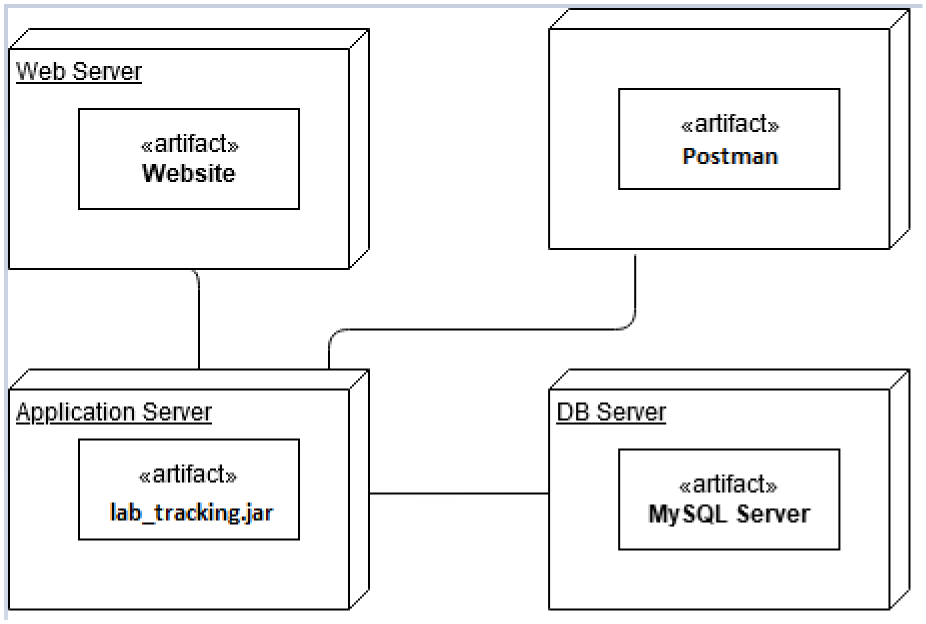
The repository layer is responsible for persisting data to the database and getting the data from the database. The service layer uses the repository layer and is responsible for data validation and supplies data to the controller. The controller layer has the endpoints and uses the service layer to get and send data. Through the controller the application communicates with the external world.



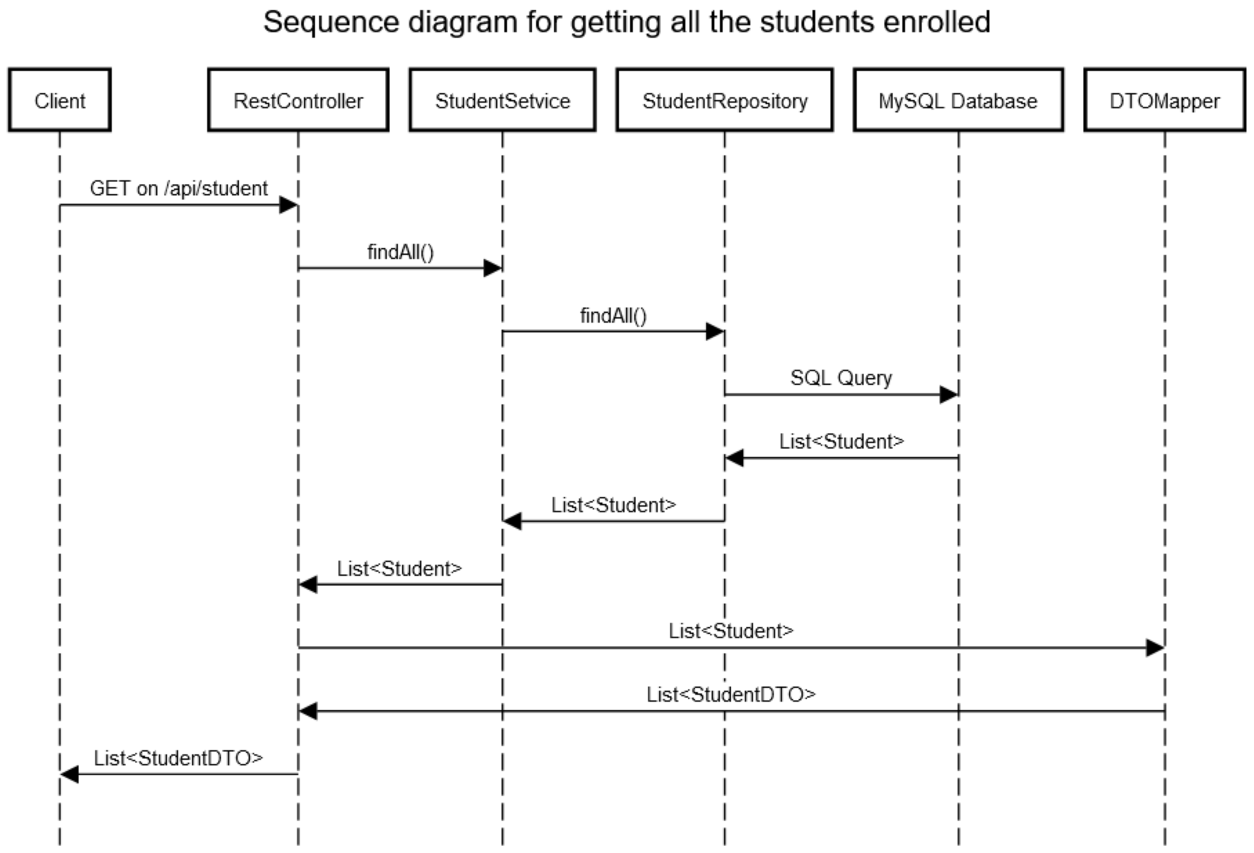
Package diagram:



Deployment diagram:



4. UML Sequence Diagrams

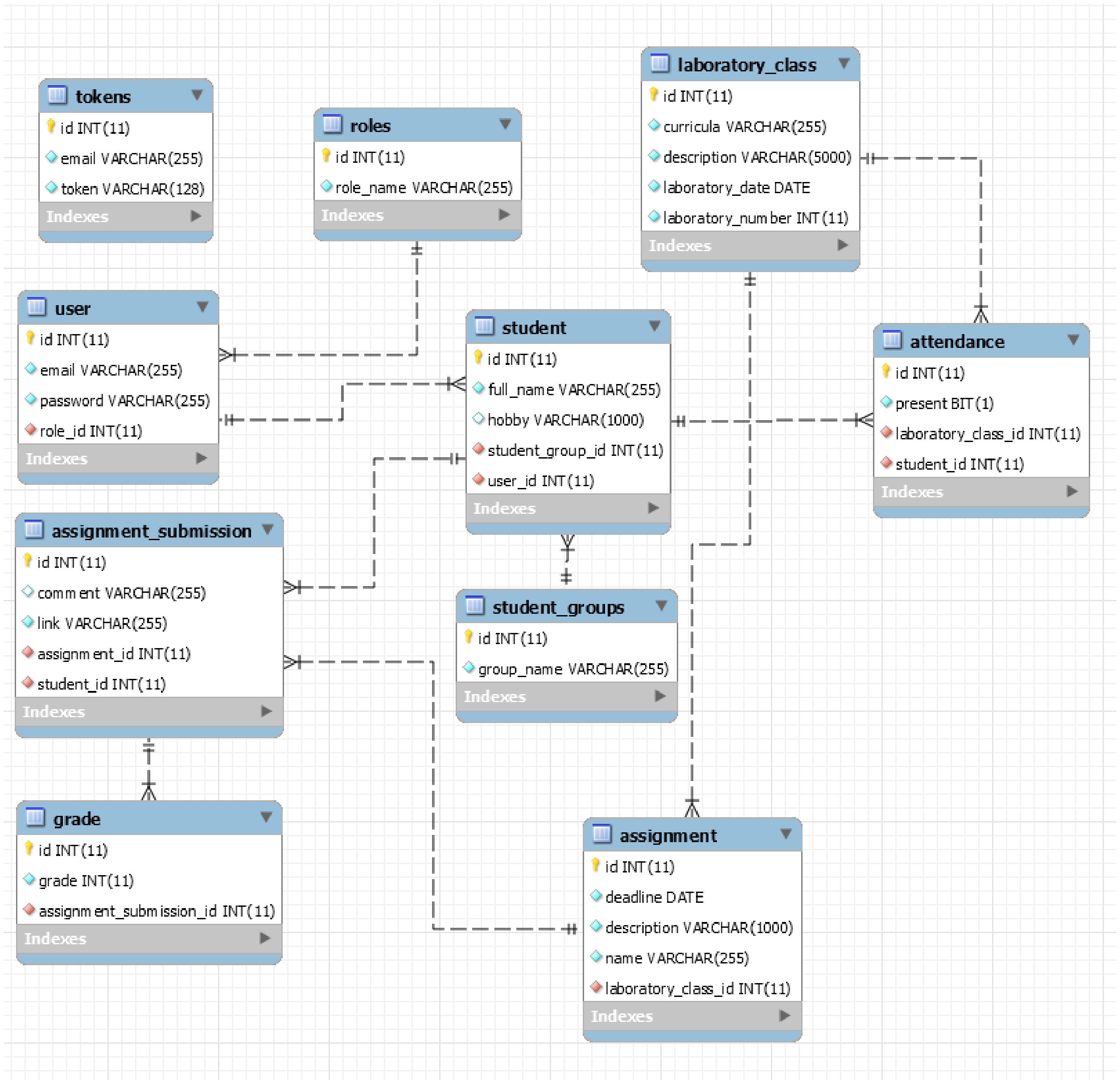


5. Class Design

5.1 Design Patterns Description

The application uses Façade design pattern. It was used to connect the User service and Student service together in a façade to use when a new student was created. In this façade the business logic for validation for creating a new user then creating a student that references that user was put.

5.2 Entity diagram



6. Data Model

Data models used in the application are:

- Lists – for storing objects

7. System Testing

Security was tested first by accessing an endpoint with the wrong authorization and the HTTP Forbidden status was sent. After that the test system tries to access the same endpoint with the correct credentials and authorization and the OK status was received.

8. Bibliography

The following sites were used to generate diagrams.
https://sequencediagram.org/
https://app.diagrams.net/