Gym Management System

Analysis and Design Document

Student: Ardelean Eugen-Richard

Group: 30433

Revision History

Date	Version	Description	Author
02/Apr/18	1.0	First update of documentation	Ardelean Eugen-Richard

Table of Contents

I. Project Specification

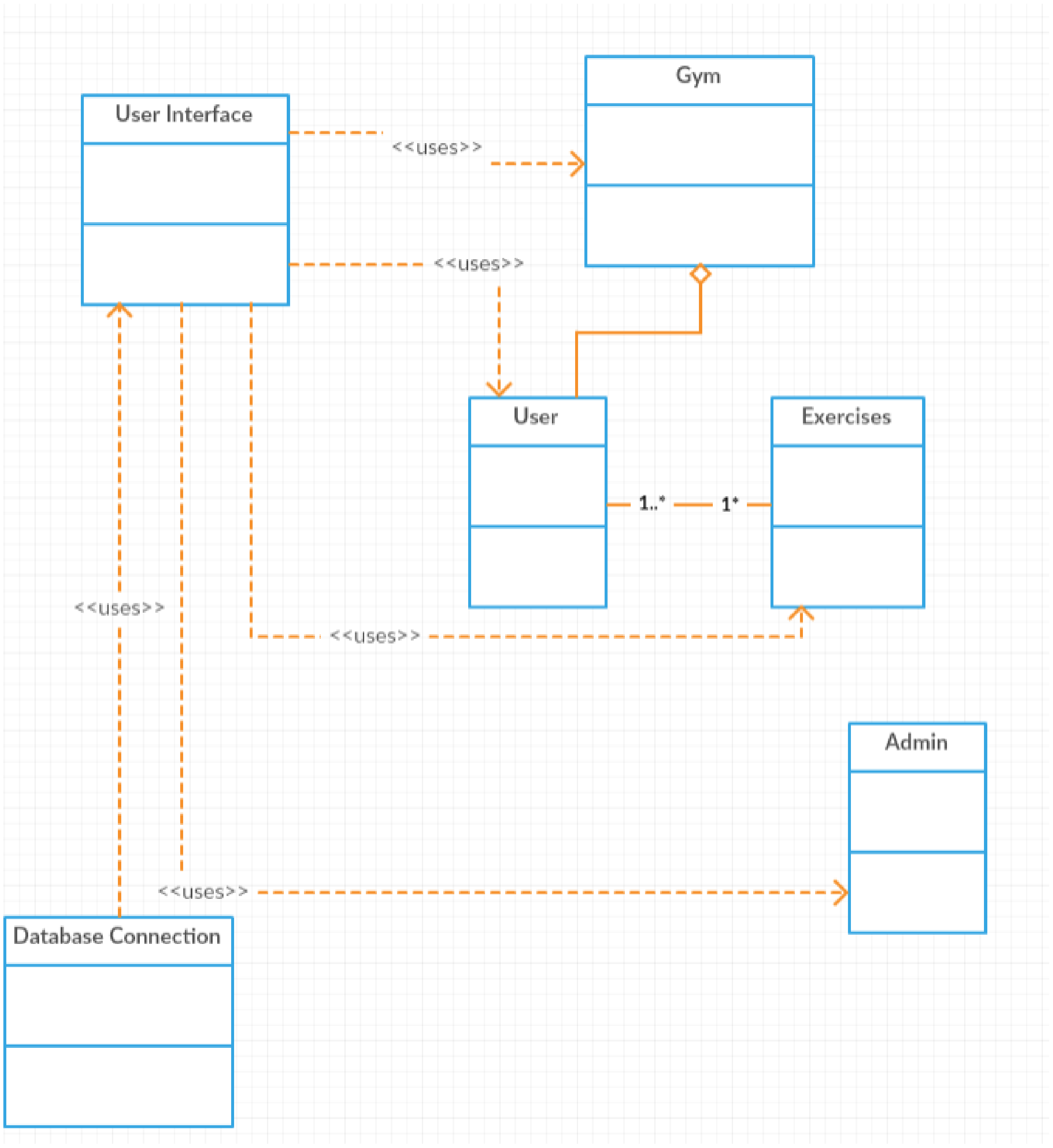
The Gym Management System offers beginners of bodybuilding help to develop a program of exercises personalized to their needs.

The application allows the users to login by entering their personal username and password, view lists of exercises, view their program or talk to a trainer for help.

II. Elaboration – Iteration 1.1

1. Domain Model

The Gym Management System will have multiple classes (possibly one for each table in database) to represent data from the database and other classes to make modification on the already existing data in the database. We will need the obvious classes like Admin, User, Exercises each representing what their names mean. There will be a graphical user interface with which the user will interact with the application.



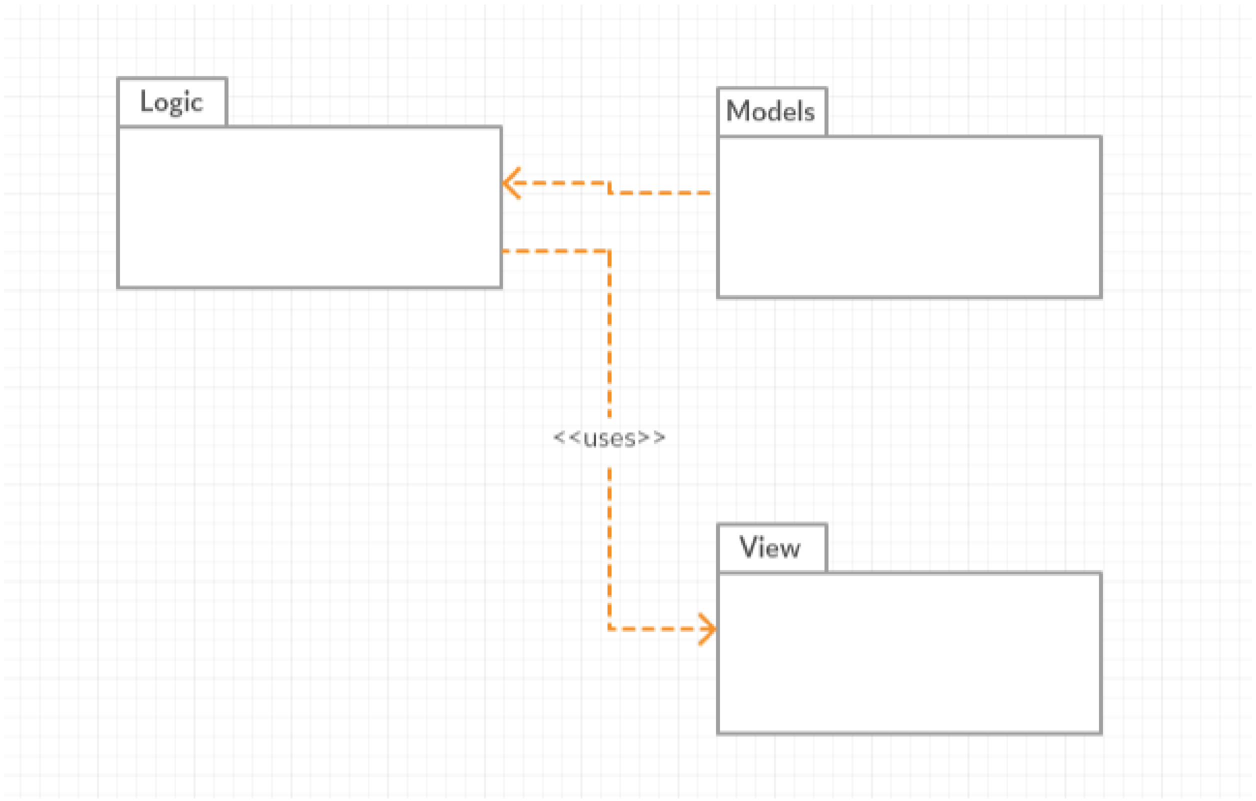
2. Architectural Design

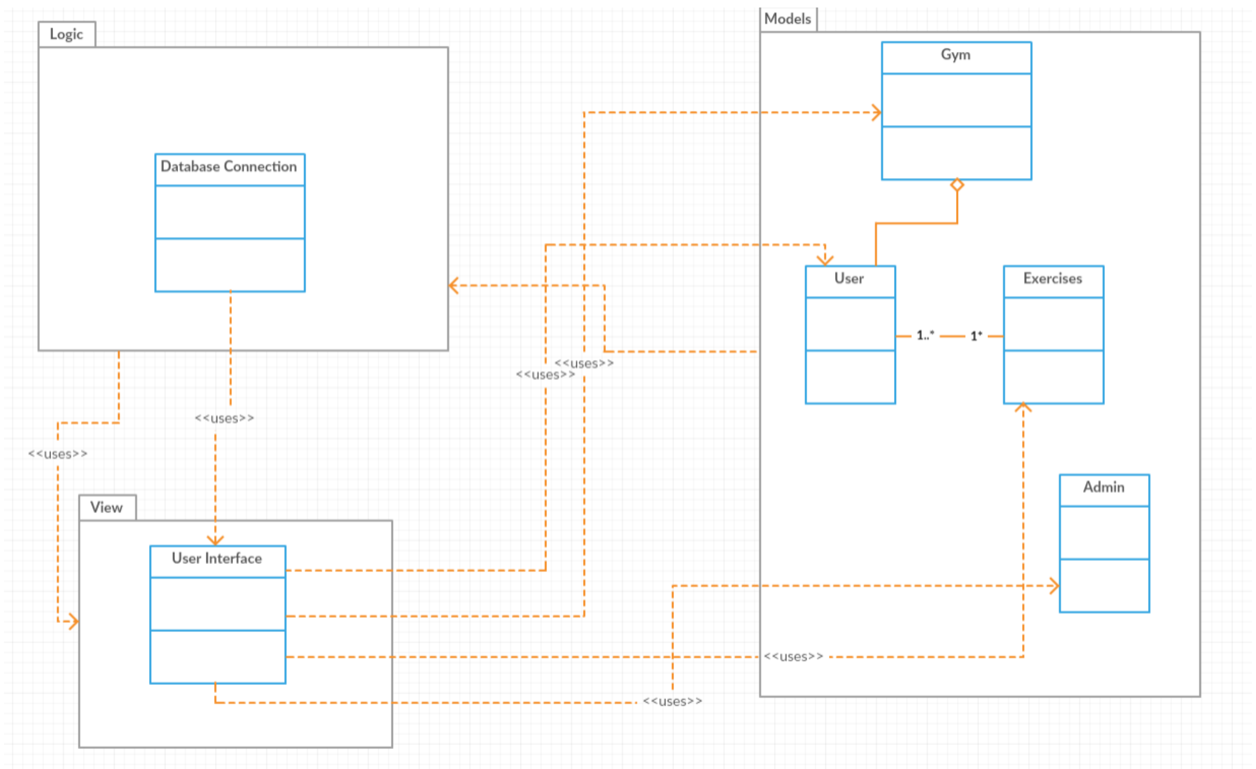
2.1 Conceptual Architecture

The system will use the Client-Server Design-Pattern. This architectural design is a a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function.

This being a web application, multiple users will connect at once, we can achieve using this pattern and putting the database on the server.

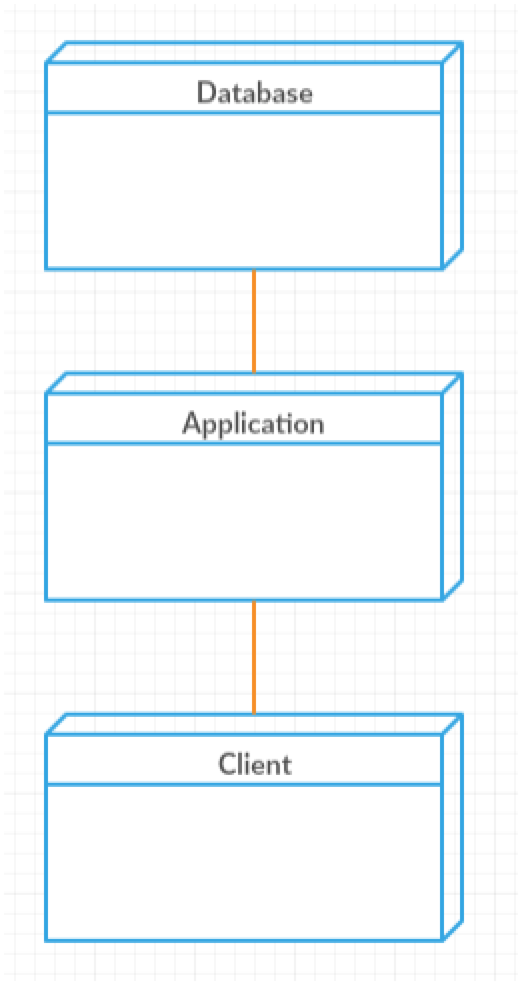
2.2 Package Design





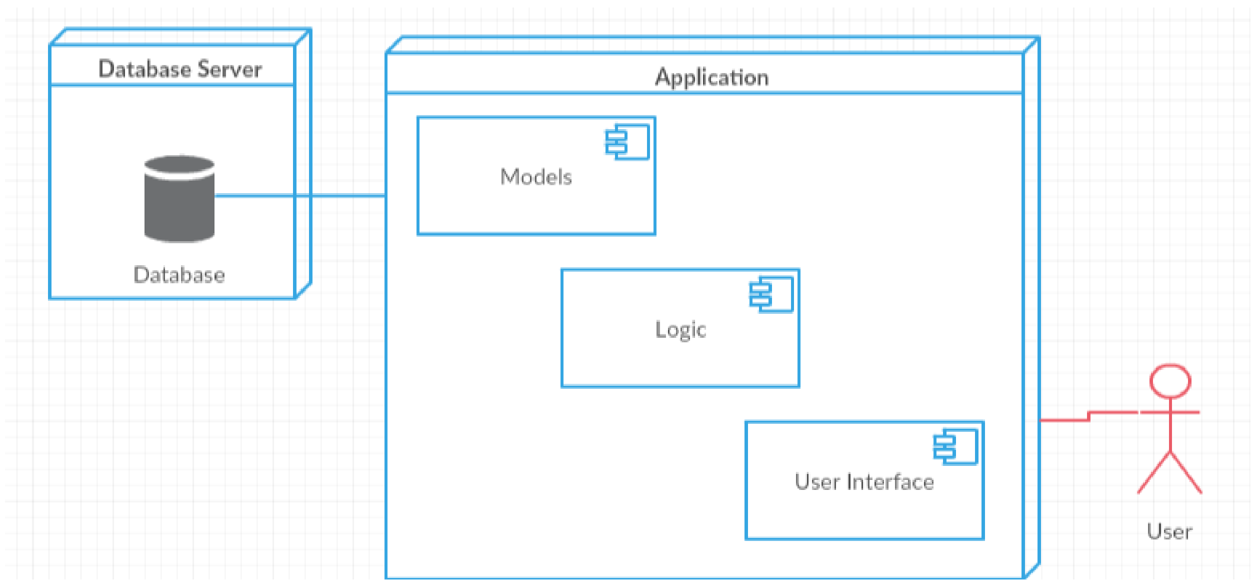
2.3 Component and Deployment Diagrams

Deployment diagram



Component Diagram

Gym Mangement System	Version: <1.0>
	Date: <02/Apr/18>
<document identifier>	

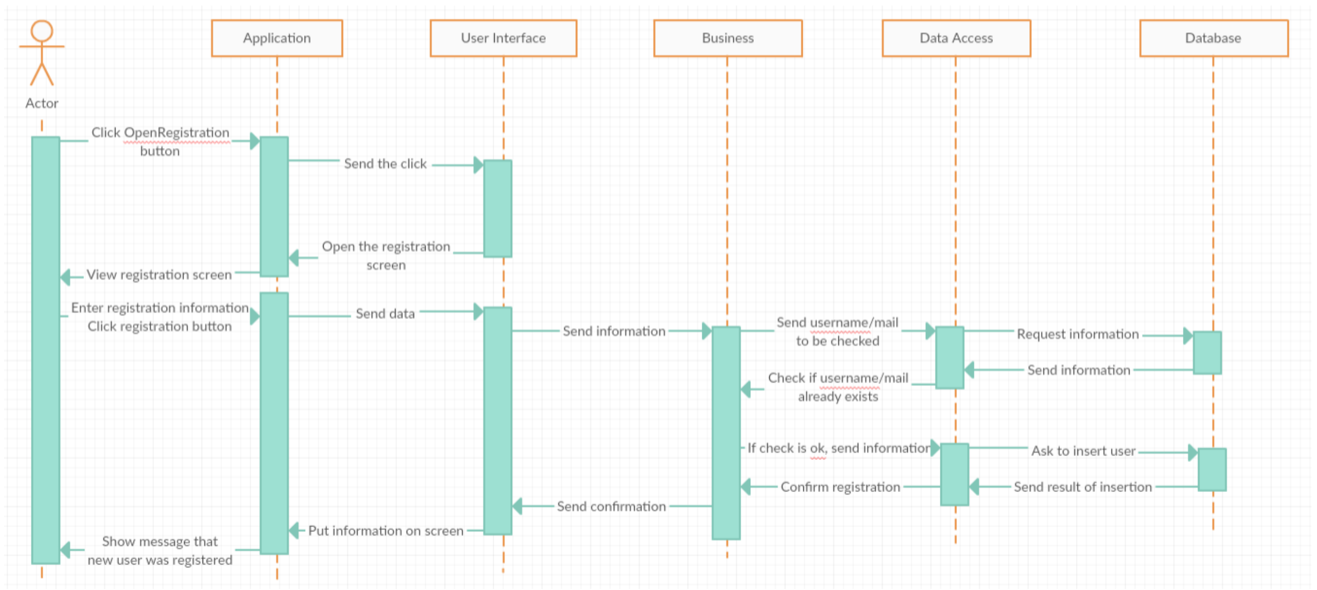


III. Elaboration – Iteration 1.2

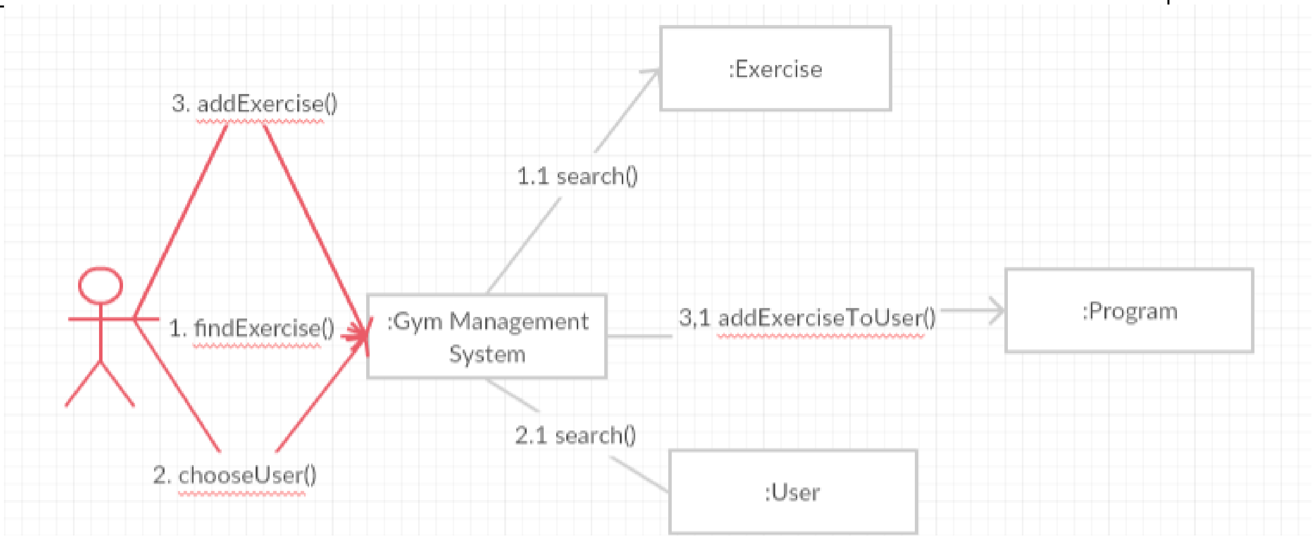
1. Design Model

1.1 Dynamic Behavior

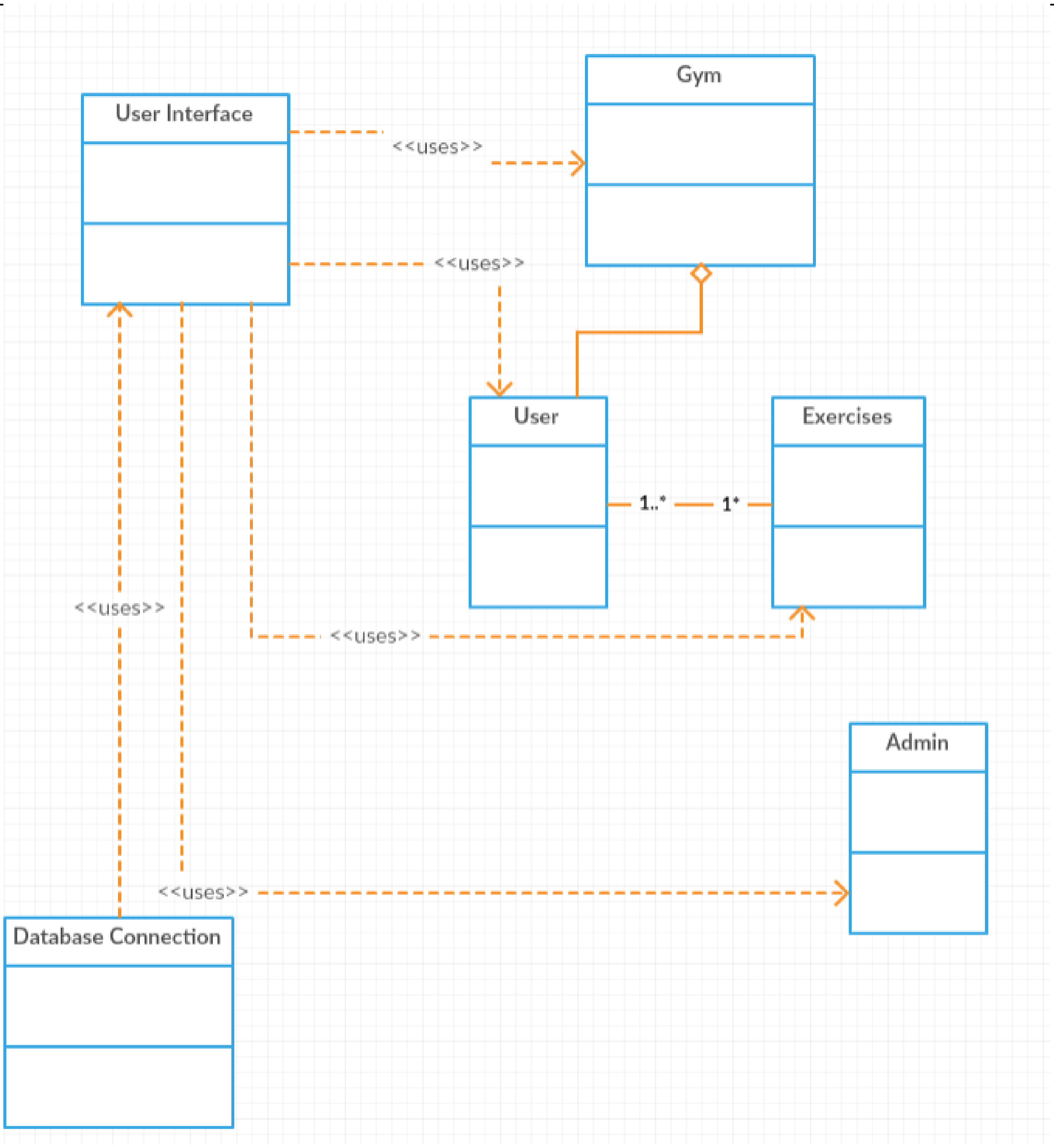
- **Sequence Diagram**


Actor
Application
User Interface
Business
Data Access
Database
Click OpenRegistration button
Send the click
Open the registration screen
View registration screen
Enter registration information
Click registration button
Send data
Send information
Send username/mail to be checked
Request information
Send information
Check if username/mail already exists
If check is ok, send information
Ask to insert user
Send result of insertion
Confirm registration
Send confirmation
Put information on screen
Show message that new user was registered

- **Communication Diagram**



1.2 Class Design



Facade

The facade pattern (also spelled as façade) is a software-design pattern commonly used with object-oriented programming. The name is an analogy to an architectural façade.[1]

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

- make a software library easier to use, understand, and test, since the facade has convenient methods for common tasks
- make the library more readable, for the same reason
- reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system

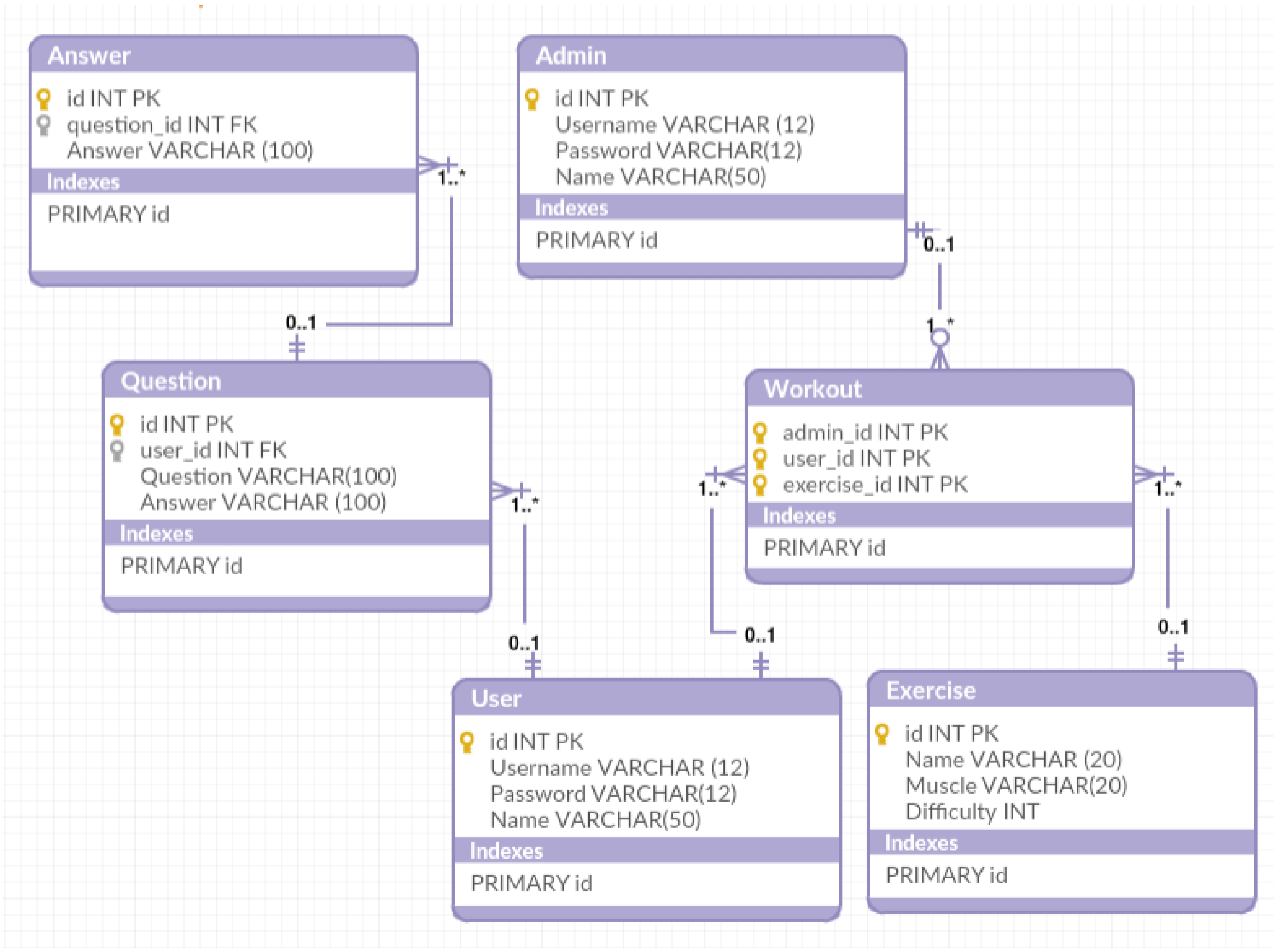
- wrap a poorly-designed collection of APIs with a single well-designed API

Factory method

In class-based programming, the factory method pattern is a creational pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created. This is done by creating objects by calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes—rather than by calling a constructor.[2]

2. Data Model

For this project, the data was modelled in the following way:
User: id, mail/username, password, name
Admin: id, mail/username, password, name
Exercise: id, name, muscle, difficulty
Workout: user_id, exercise_id



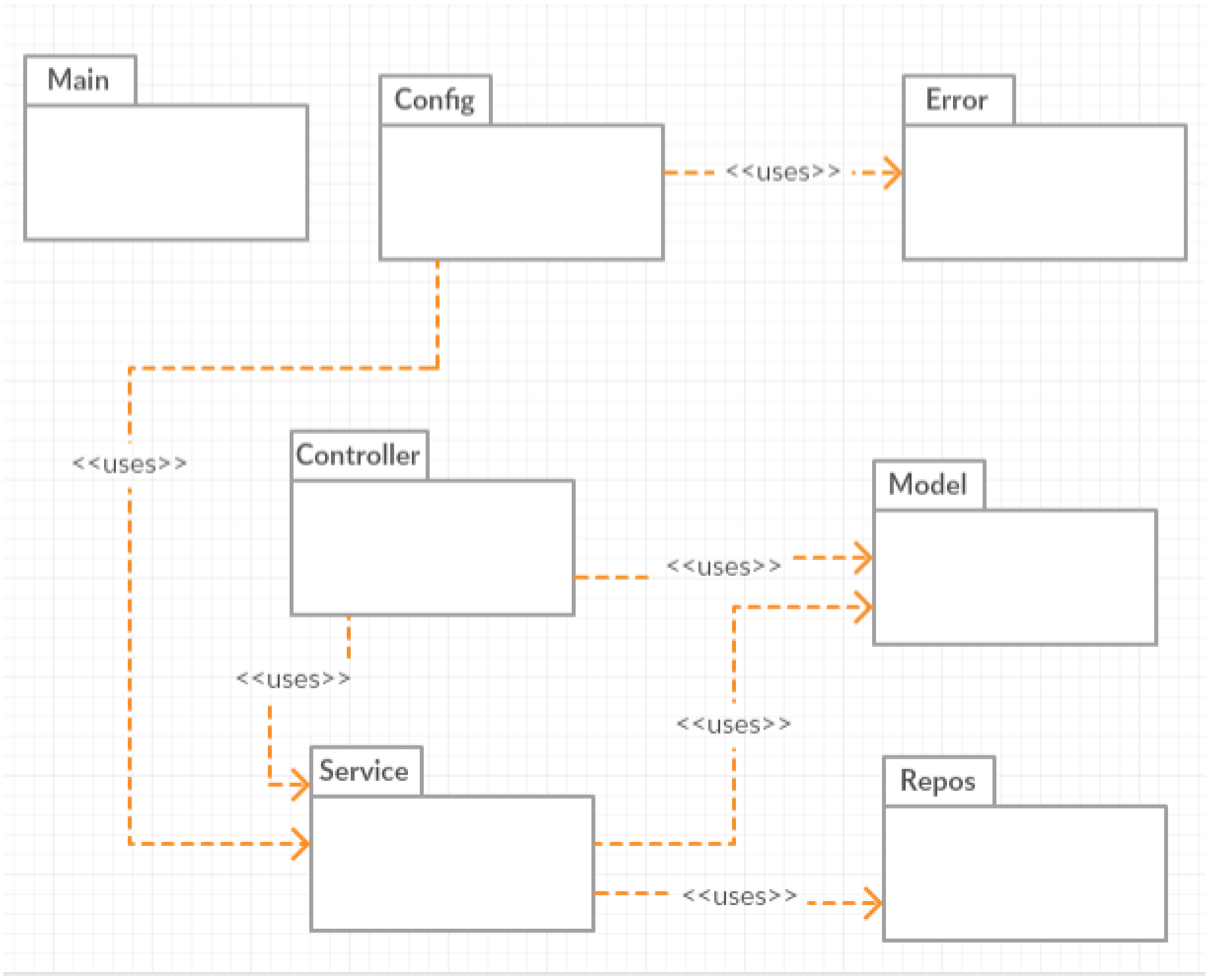
3. Unit Testing

At the beginning the testing was done using printing functions (System.out.println()) to see that the data sent and received was correct. Later the testing was done using unit testing (JUnit) which involves testing parts (units) of the code to verify that they work as expected.

IV. Elaboration – Iteration 2

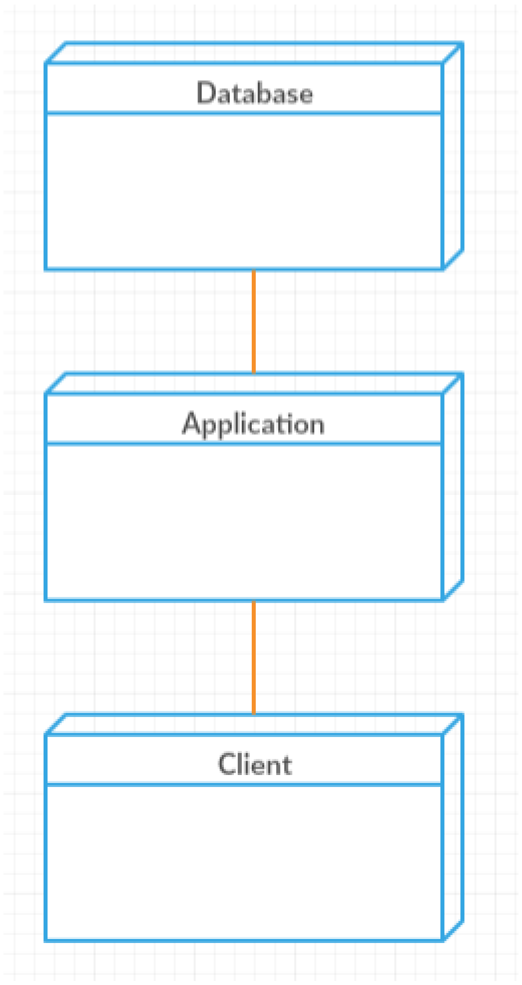
1. Architectural Design Refinement

Package Design



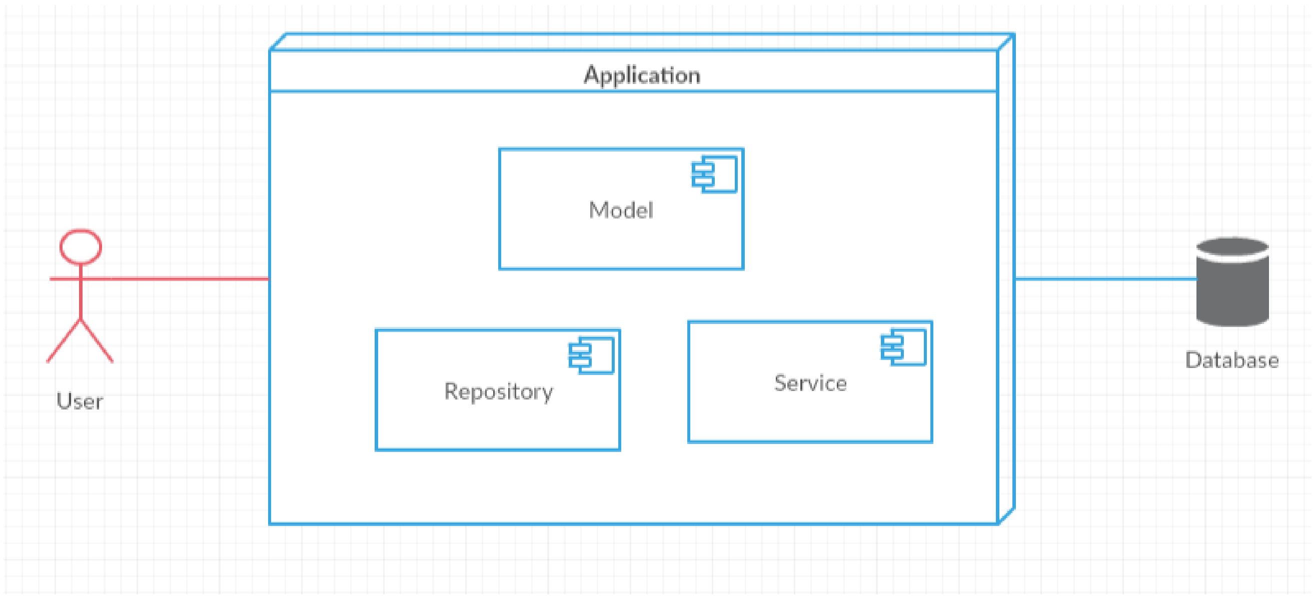
This is the updated version of the package diagram, changes were made due to the Spring framework and its functionalities, the service and repositories.

Deployment diagram



There have been no modifications to the deployment diagram.

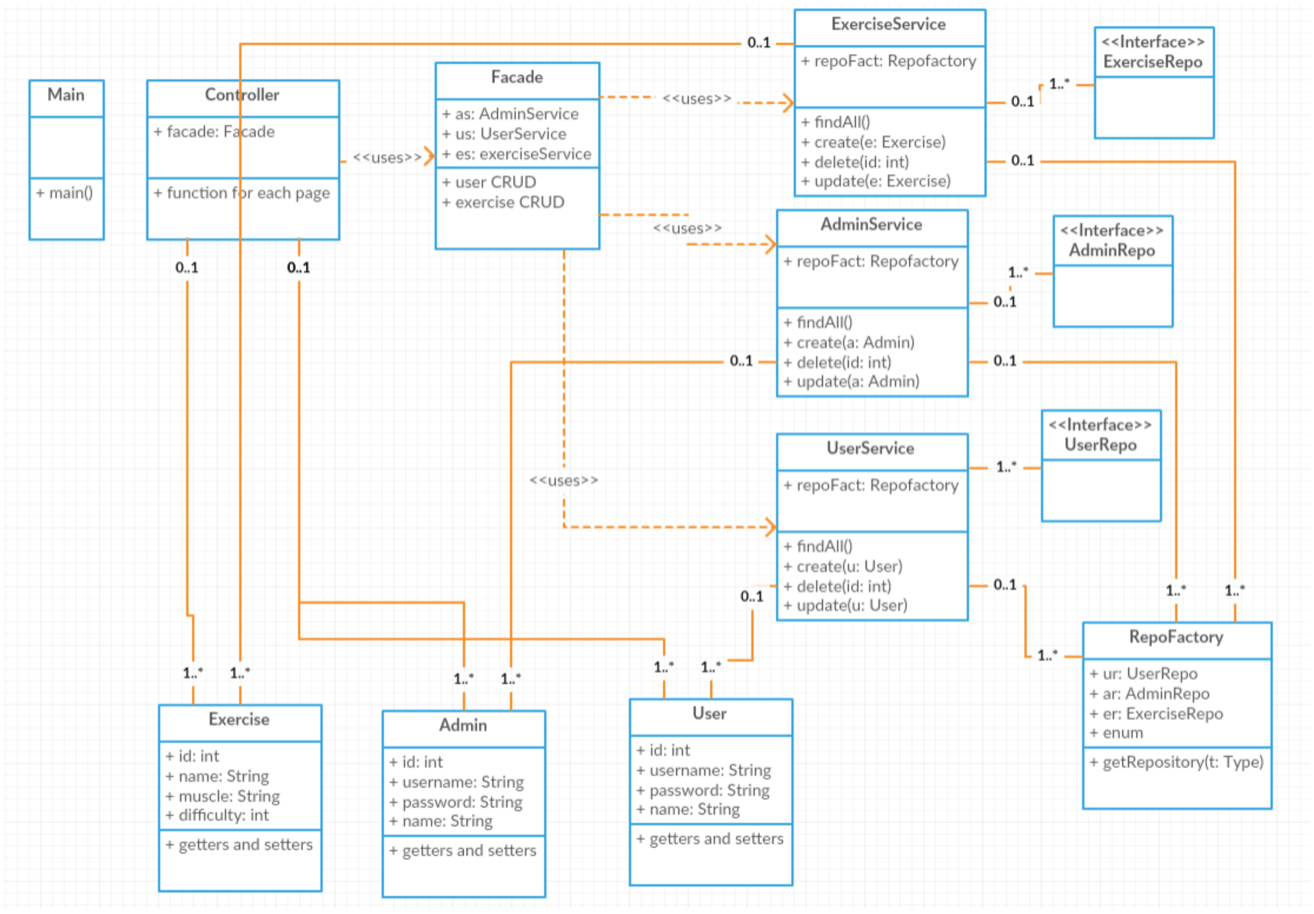
Component Diagram



Some modifications have been made to the component diagram to adjust to the spring framework.

2. Design Model Refinement

Class Diagram



The class diagram shows the main models in the application (admin, user, exercise) and their corresponding Repositories and Services, these are used by Spring.

The application contains 2 design patterns: Façade and Factory. The first is implemented in the services package and it is called Façade, it provides and interface through which to access the system, while the Factory is in the repository package called RepoFactory, this pattern provides the best way to create an object (without exposing the logic to the user and it allows the referral to the newly created object through a common interface).

V. Construction and Transition

1. System Testing

Registration

- user presses the register button and enter registration page.
- user enters his personal data to the according fields
- user presses the register account button.
- if the username has been taken, the application shows an error message and allows the user to choose another one
- if the username has not been taken, the account will be created

Create exercise

- admin presses the create exercise button and enters the exercise creation page
- he enters the information about the exercise (name, muscle group, difficulty)
- presses the create exercise button

2. Future improvements

Many improvements can be made to this system, a larger database of exercises and variations would be helpful, also there could be more information associated to each exercise like description, tips, images.

Some additions functionalities like, multiple programs for one user and being able to create a custom one for himself, selecting favorite exercises.

VI. Bibliography

[1] **https://en.wikipedia.org/wiki/Facade_pattern**
[2] **https://en.wikipedia.org/wiki/Factory_method_pattern**