ScheduleIT

Analysis and Design Document

Student: Coporîie Andreea

**Group: 30433**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2/Apr/18 | 1.0 | Iteration 1.1 added | Coporîie Andreea |
| 7/May/18 | 1.2 | Iteration 1.2 added | Coporîie Andreea |
| 28/May/18 | 2.0 | Iteration 2.0 added | Coporîie Andreea |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

The ScheduleIT application means to ease the creation of timetables in schools/universities by moving the responsibility of its creation to multiple registered users, instead of a single person, and encouraging the communication between professors and students.

The application’s interface should allow its users to log in, using either a username and password combination or just their own personal id, view their assigned timetables, attempt to modify them in case the user has the necessary rights to do so, and communicate with the system administrator in case of conflicts.

# Elaboration – Iteration 1.1

# Domain Model

The ScheduleIT application will have a class defined for each table in the database, to accurately represent the data model of the problem. The DAO pattern will be used to make connections to the database, and an administration (business) layer of classes will further link this to the user’s interface. The UI layer might consist of more than a class for ease of implementation and use.

# –https://i.gyazo.com/f0224a092cb3a477d177f01dfe429940.png

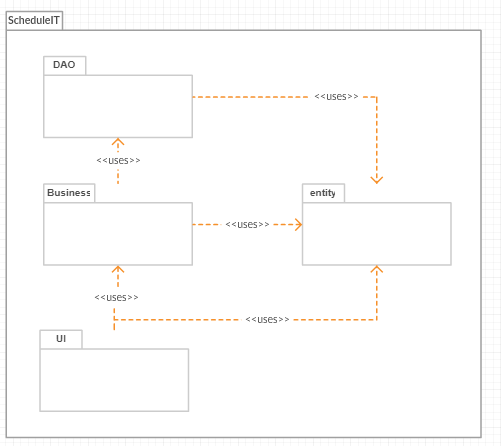
# Architectural Design

## Conceptual Architecture

The client–server model is 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. Clients therefore initiate communication sessions with servers which await incoming requests.[1]

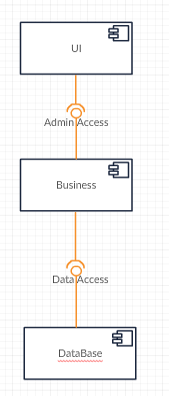
As the ScheduleIT app will support servicing several users at once, the client-server architecture is the best suited for the application’s needs.

## Package Design

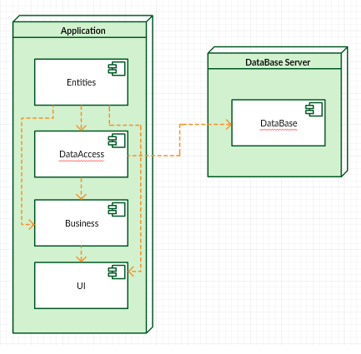


## Component and Deployment Diagrams

Component Diagram:



Deployment Diagram:

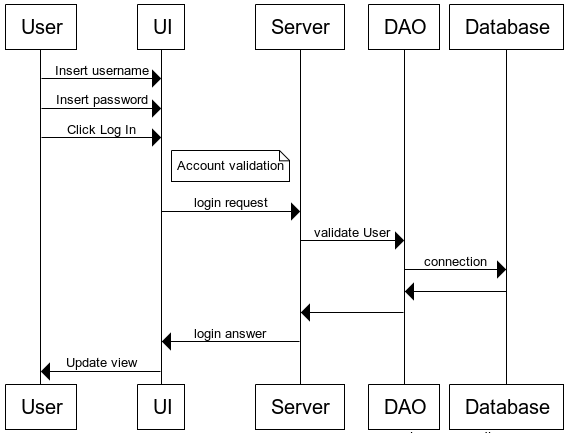


# Elaboration – Iteration 1.2

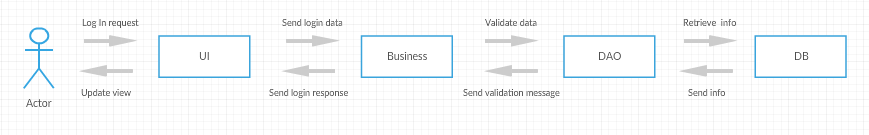
# Design Model

## Dynamic Behavior

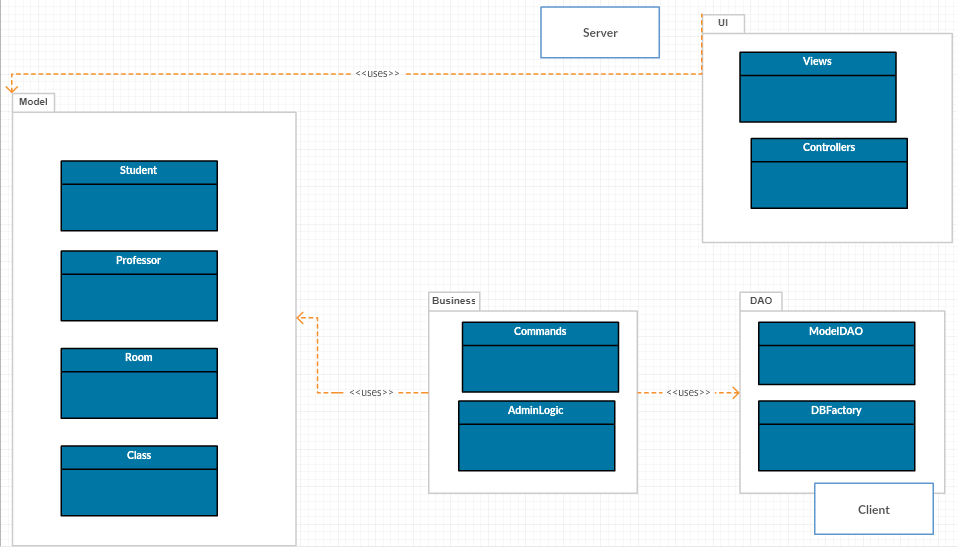
Log In Sequence diagram:



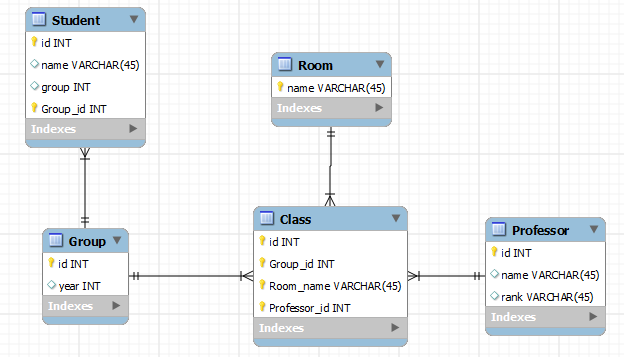
Communication diagram:



## Class Design



# Data Model



# Unit Testing

Login Scenario:

* Insert data
* Press Log In
* Receive Success message
* Alternatively receive Failure message and try again

View Table Scenario

* Enter group ID
* Press view schedule
* Check table

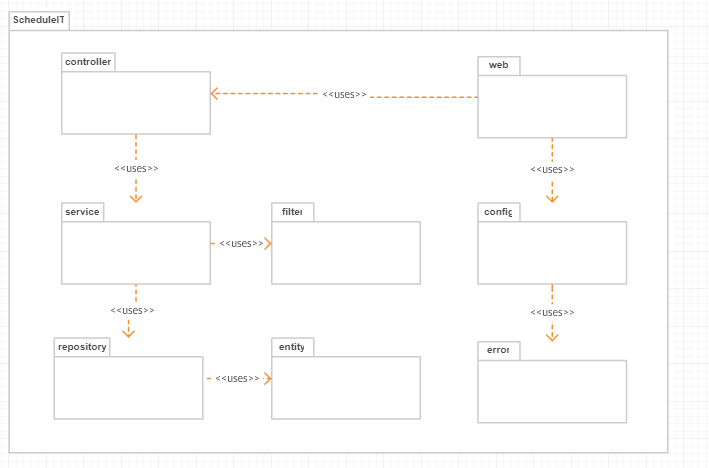
# Elaboration – Iteration 2

# Architectural Design Refinement

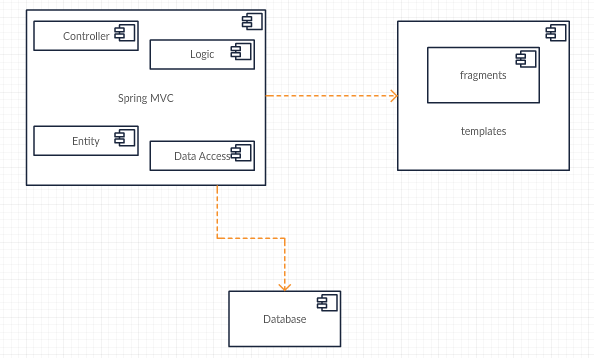
Refinement changes:

In addition to the Client-Server architecture of the project, an MVC model has been adopted and implemented using Spring Framework.

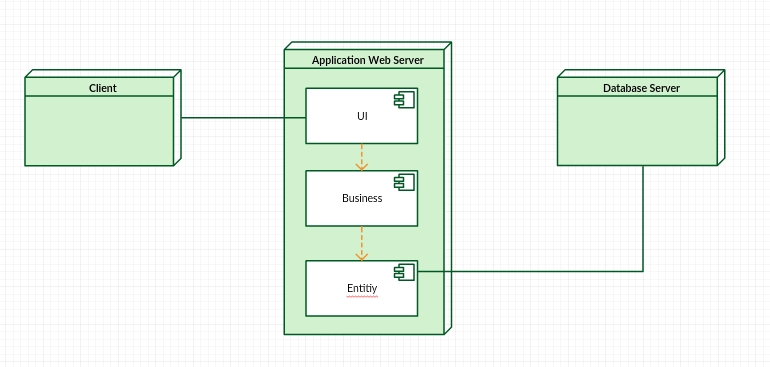
Package Diagram:



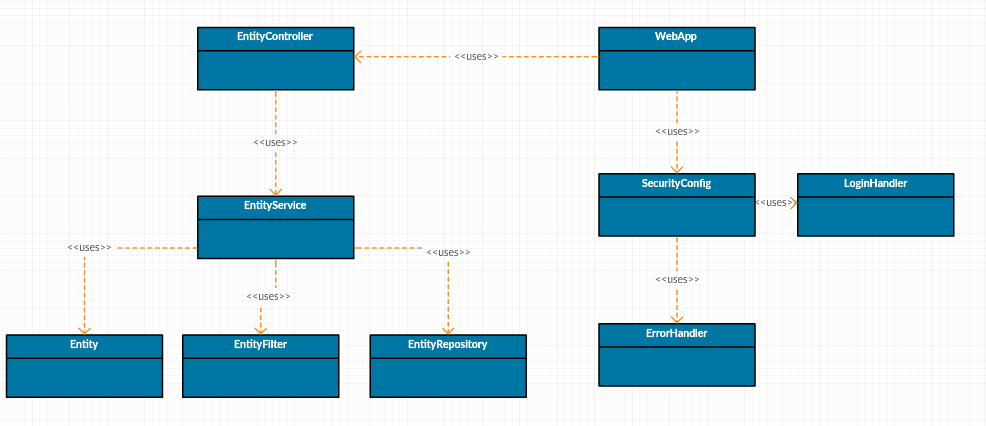
Component Diagram:



Deployment Diagram:



# Design Model Refinement



# Construction and Transition

# System Testing

The testing has been done unit-wise only for several packages of the project such as filter and service where the data underwent several processing stages, by using a Test class and the standard output to revise the units. Overall, integration testing has been done by trial and error until a satisfactory result has been met.

# Future improvements

The project represents only a prototype of the actual capabilities of the idea. To further improve it more filter options could be added, together with an extension of the UI to visualize them, as well as more options to perform operations on the available data, like CRUD operations. Furthermore, the data model itself could be improved to represent more closely the real life scenario, which would impact the projects’ complexity as well.

# Bibliography

[1] https://en.wikipedia.org/wiki/Client-server\_model

[2] <https://www.w3schools.com>

[3] <https://www.thymeleaf.org>

[4] <https://creately.com/app/>