Analysis and Design Document

Student: Bogdan Rogoz
Group: 30433

Revision History

Date	Version	Description	Author
01/04/2018	1.0	Initial documentation	Bogdan Rogoz

Table of Contents

I. Project Specification

The purpose of the Watch2Gether project is to provide its users a friendly environment where they could listen to a wide variety of music, in a synchronized manner, while being situated in totally different spaces.

II. Elaboration – Iteration 1.1

1. Domain Model

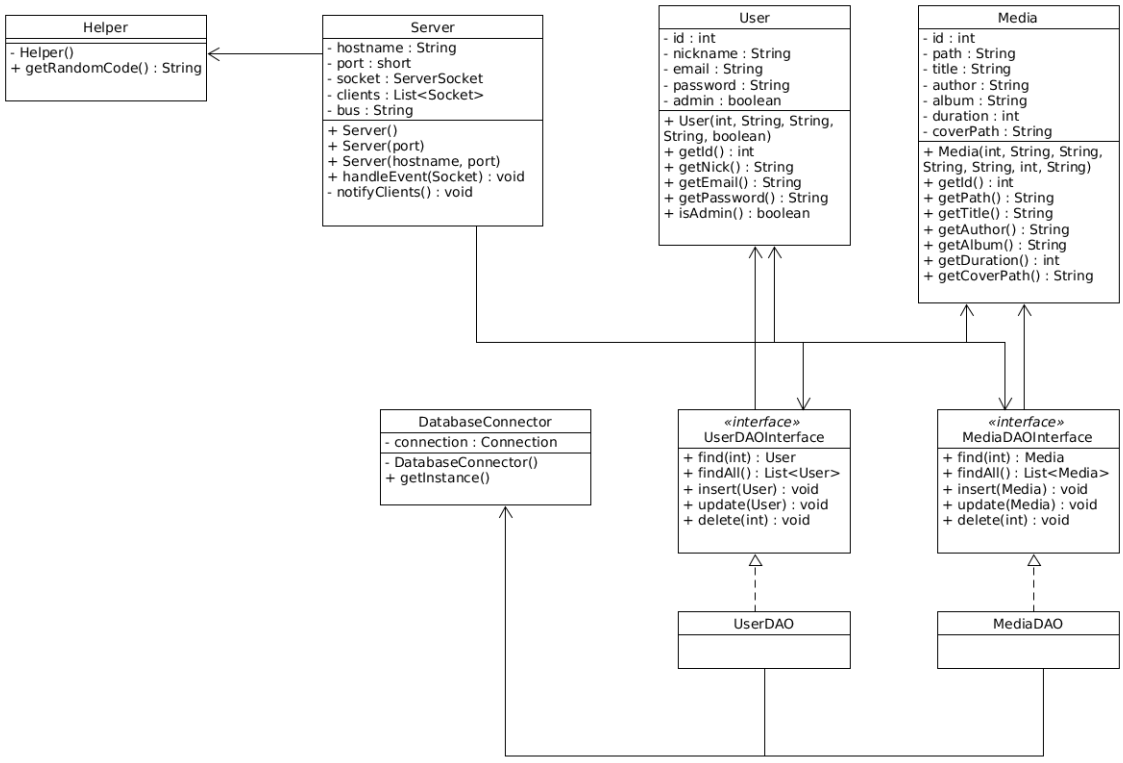
The Client will contain 2 models:

- *User – represents a user, either the active one or other users present in the room*
- *Media – holds the information about media files : title, URL address, cover image, duration etc.*

The Server consists of the following models:

- *User – same as above*
- *Media – same as above*

Although the Client will have a more declarative implementation, the server will have a more object – orietned design, as in the diagram below:



2. Architectural Design

2.1 Conceptual Architecture

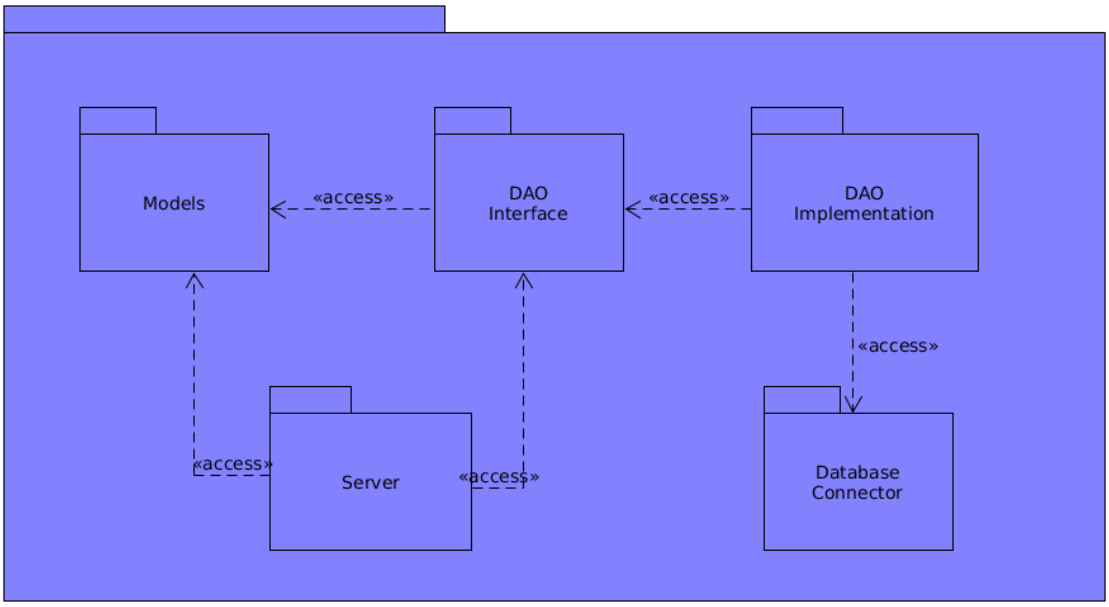
The system consists of two endpoints : the users and the server. The used architectural patterns are:

- Client – Server : The server sends request / response messages to each client at a given point in time and vice-versa. While communicating with all its clients, the server also listens for new connections.

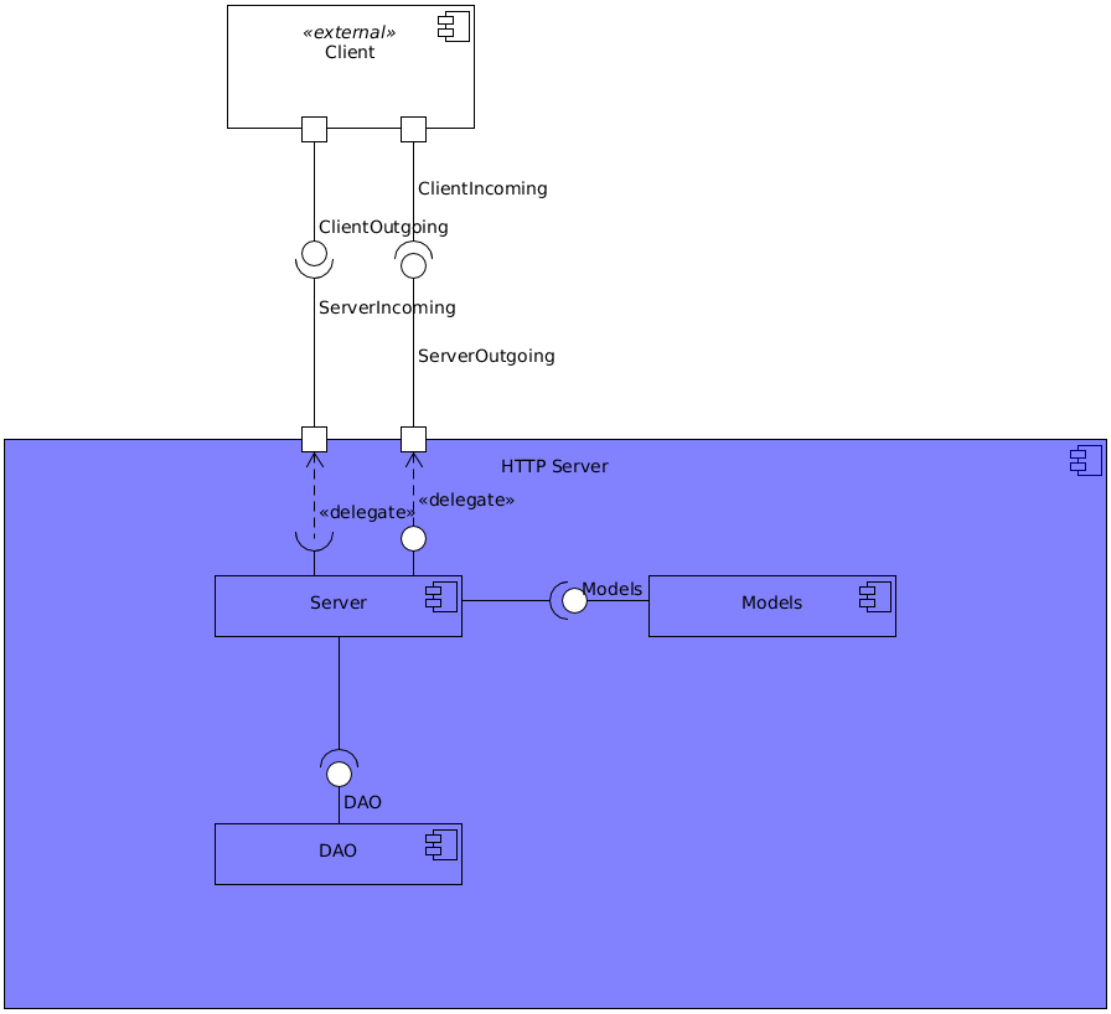
- Event Bus : The server acts as a virtual bus. Every client listens for events while performing playback. When a client wants to perform an action (eg. pause, stop), it sends a message to the server telling it to place the message on the bus, and all the other clients are then notified.

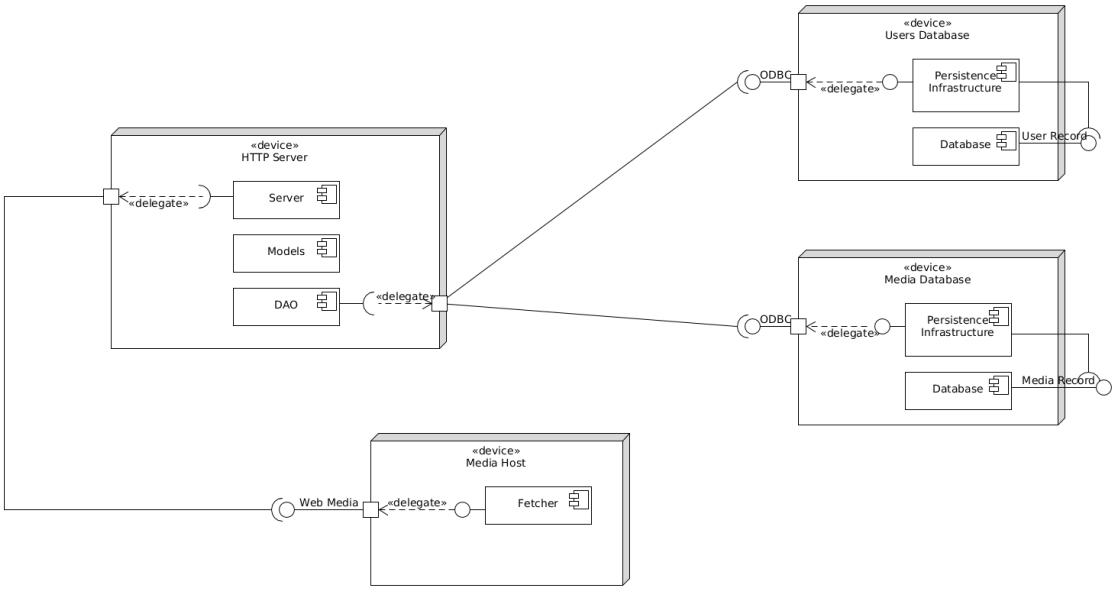
The described approach has been selected due to its simplicity and performance.

2.2 Package Design



2.3 Component and Deployment Diagrams

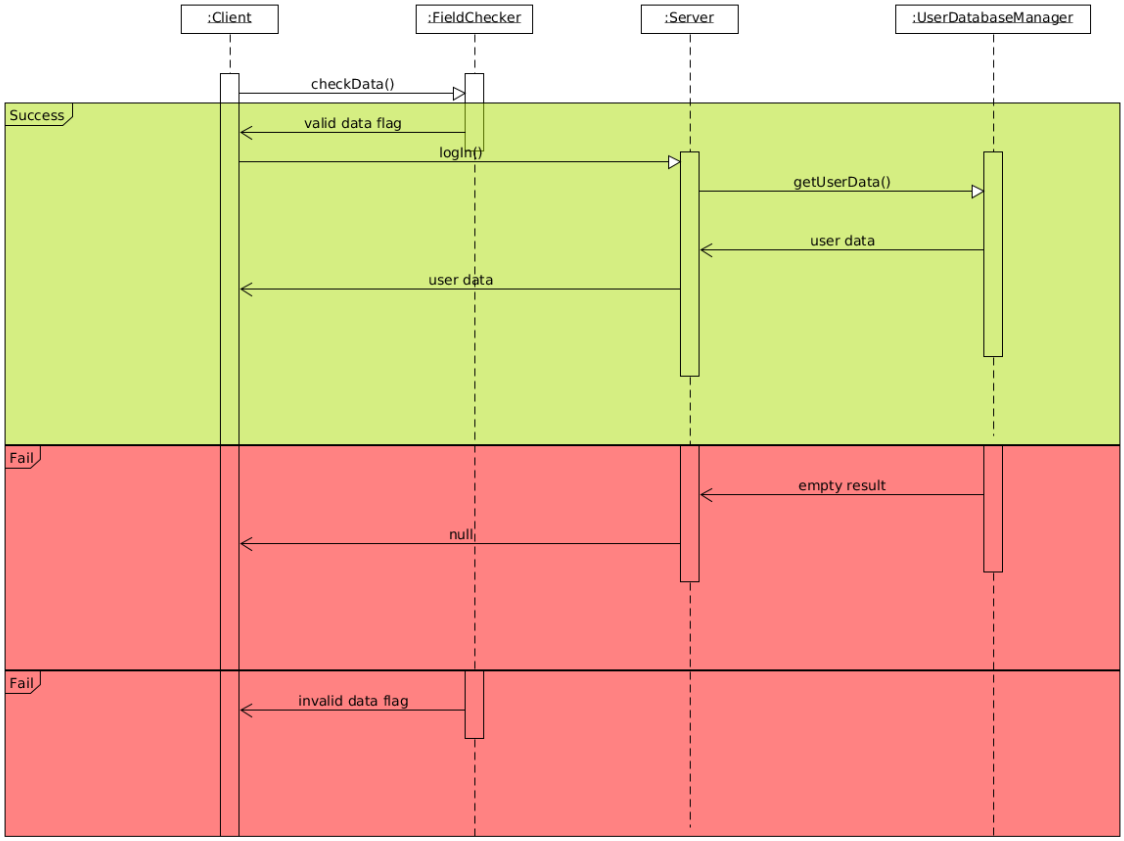



«device»
Users Database
ODBC
«delegate»
Persistence
Infrastructure
Database
User Record
«device»
HTTP Server
«delegate»
Server
Models
DAO
«delegate»
«device»
Media Database
ODBC
«delegate»
Persistence
Infrastructure
Database
Media Record
«device»
Media Host
Web Media
«delegate»
Fetcher

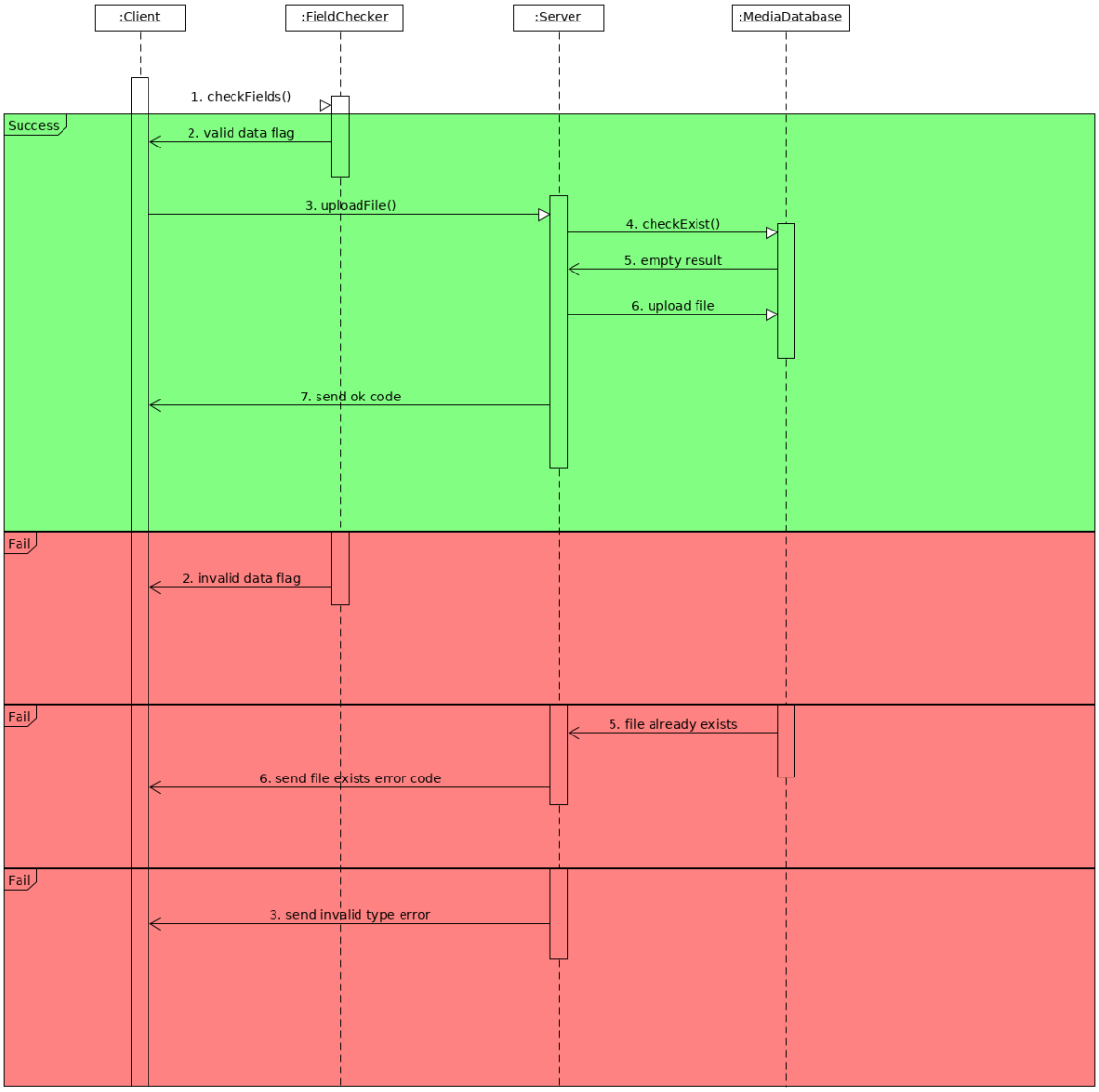
III. Elaboration – Iteration 1.2

1. Design Model

1.1 Dynamic Behavior

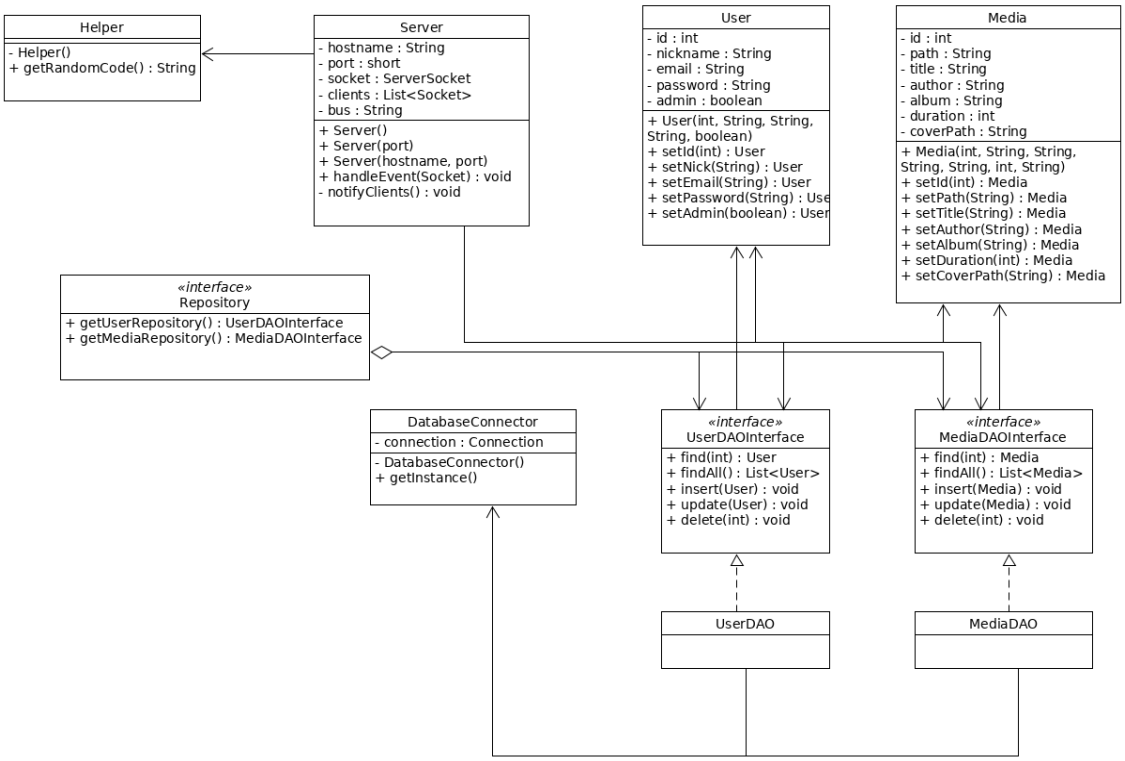


Log In Sequence Diagram

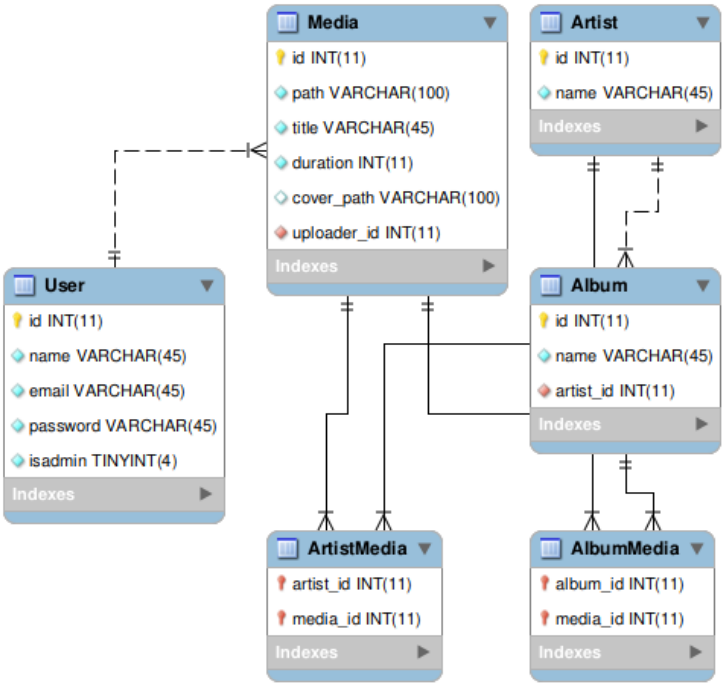


Upload Media communications diagram

1.2 Class Design



2. Data Model



3. Unit Testing

In order to ensure the good functionality of the application, we could perform multiple tests:

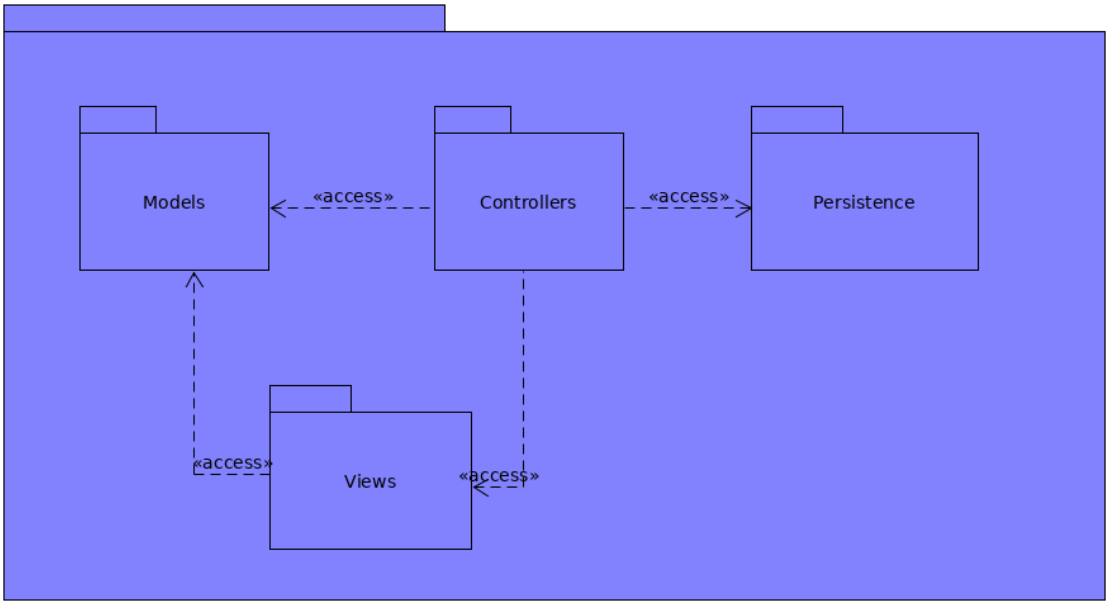
- provide invalid login data and see how the system reacts
- provide valid login data and see how the system reacts
- attempt to upload a non-media file (eg. text)

Although some features that require validation can be tested, there are some things that cannot be validated in any way, due to the high difficulty of the user to calculate the results ahead (eg. room code generation).

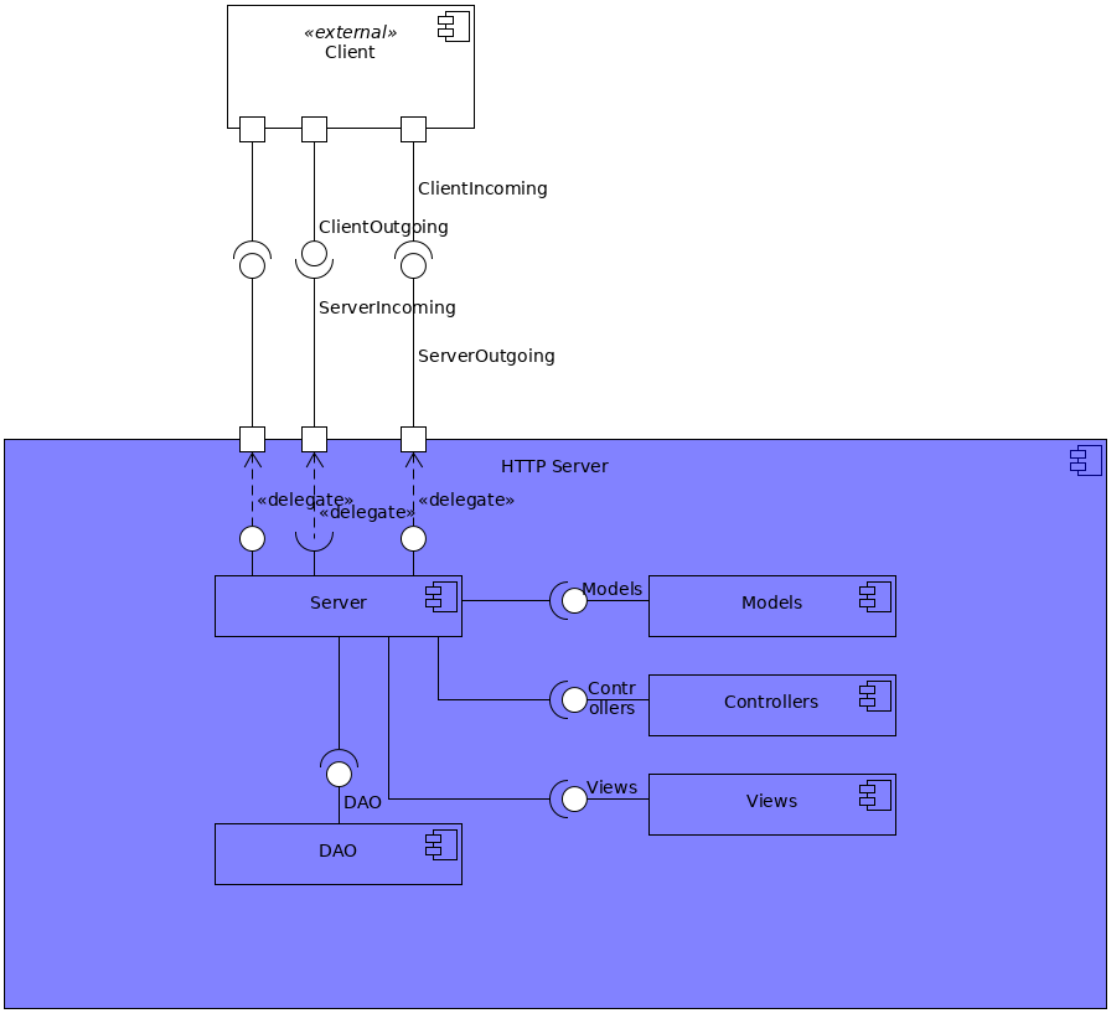
IV. Elaboration – Iteration 2

1. Architectural Design Refinement

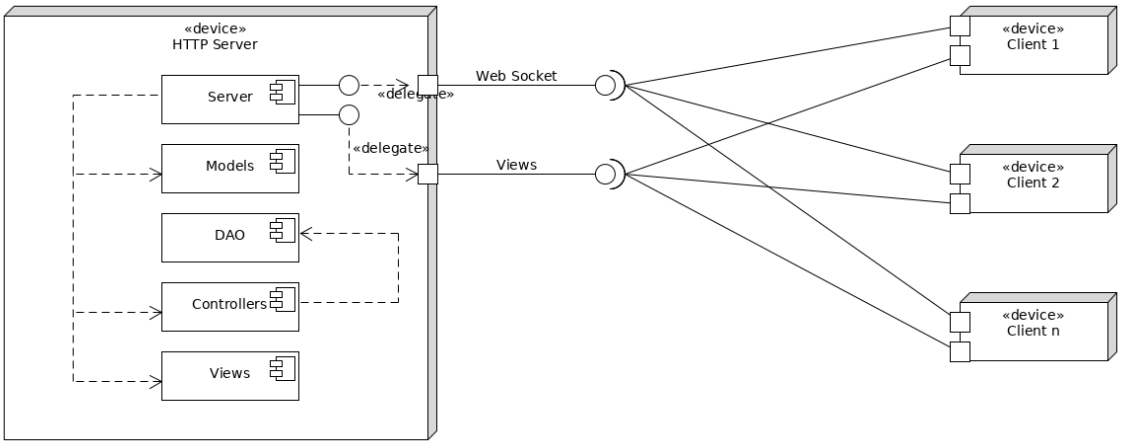
The application will be implemented using the Model – View – Controller architectural pattern. This way, it becomes easier to differentiate tasks performed by various components of the server, and also the views on the client side and their representation on the server.



Package diagram

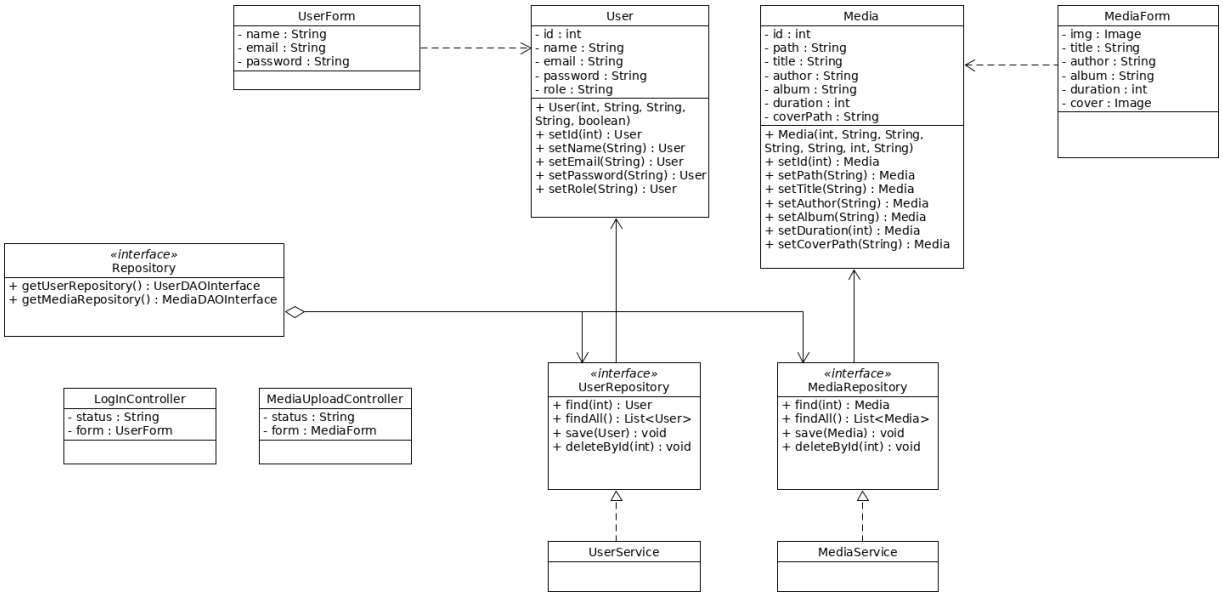


Component diagram



Deployment diagram

2. Design Model Refinement



V. Construction and Transition

1. System Testing

Each feature has been tested using a web browser. Some of the cases include:

1. Invalid login credentials
    1. Access the main server
    2. Enter wrong login credentials (eg. password with less than 8 characters)
    3. Get redirected to the login page
2. Invalid room id
    1. Access the create chat feature
    2. Enter an invalid room id
    3. Get redirected to the previous page

2. Future improvements

Some possible improvements include:

- View the number of people inside a room
- Room chat
- 3rd party service support (eg. YouTube, DailyMotion etc.)
- Better UI

VI. Bibliography

https://sourcemaking.com/
https://spring.io/guides/
http://mkyong.com/tutorials/spring-boot-tutorials/