ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Ingegneria e Architettura
Laurea Magistrale in Ingegneria e Scienze Informatiche

TITOLO

Elaborato in
MATERIA

Presentata da
GABRIELE GUERRINI

Anno Accademico 2018 – 2019

Table of contents

Chapter 1

Scoping

1.1 Conditions of satisfaction

- The project must end up within 50 days from the starting date.
- The project must be developed using Scala language.
- The developing process must be an agile one, in particular Scrum.

1.2 Analisys of requirements

The deliverable should consist of a library that simplifies the development of distributed client-server applications based upon a concept of room. Videogames largely use rooms in every feature (match, lobby, trading room etc.) but there are lots of potential different domains where such idea can be applied (chatrooms, for instance).
The main idea of the product is to provide a low level support that handles all aspects of communication, so that developers that use the library can focus on program logic itself, without concerning about issues due to networking, serialization, synchronization between clients, integration of clients and rooms, and so on.
Hence, the library provides two main notions of:

- **Room**: place where clients can gather to do something.
- **Client**: entity that might do operation on rooms, such as joining, sending messages etc.

1.2.1 Room

A room is something that clients can join and interact with. A room possesses a behavior and a state that combined embed the custom logic of the program. Obviously those vary from room to room, so the developer will be able to flexibly define its own type of room, namely a room that possesses its own state and behavior.

Rooms can be public or private. A public room can be joined by everyone; on the contrary, private rooms need a specific password to be joined.
It must be possible to make a public room private, and so to set a password in such room. Similarly, a private room could be made public; the existing password must be deleted so the room can be freely joined.

Rooms must expose a locking feature that permits to lock/unlock the room. While locked, the room can not be joined by any new client. By default room are unlocked.

The room state is made up of items that can be everything, from simple numbers to objects.
The state can be split in private and public one about visibility to clients. By default the state is private and never automatically communicated to clients in the room. Part (or the totality) of the state can be made public and periodically synchronized between all connected clients in the room.

The room behavior can be reactive and/or proactive both: the first specifies how the room should react as an event occurs, the latter instead defines how the room evolves as time passes.
Regarding reactive behavior, events that could occur, and that will be handled by the room, are:

- **Creation**: the room is created.
- **Closing**: the room is closed, intended as imminent deletion.
- **Joinin**: a client joins the room.
- **Leaving**: a client exits the room.
- **Receiving a message**: the room receives a message from a client.

In the end, the room can have properties, i.e. metadata values that describe room features and that can be used for several purposes, such as room filtering, joining constraints etc.

Those properties can be defined on room creation and are public to clients, either the ones in the room or not.
Room properties can be of four basic types: integer, double, string, boolean.
Few examples of room property may be max number of clients in the room, min/max Elo required to join, fiendly fire.

1.2.2 Client

A client should, in primis, display existing rooms and their properties.