Karelia
AMMATTIKORKEAKOULU

## Software requirements specification

# WormGame

**Version 2.1**

| | | |
|---|---|---|
| Karelia | TIKO | LTP7024 Testausmenetelmät |
| Author: Juhani Pirinen | | Printed: 3rd oct 2015 |
| Distribution: Teacher of the course, Public in GitHub | | |
| Status of the document: draft | | Edited: 3rd oct 2015 |

## VERSION HISTORY

| Version | Date | Authors | Description |
|---|---|---|---|
| 1.0 | 13th sep 2015 | The group 6 | http://qcheni.github.io/ola/reve/pirkkaplus/#/ (in finnish) |
| 2.0 | 2nd oct 2015 | Juhani Pirinen | Draft (in english) |
| 2.1 | 3rd oct 2015 | Juhani Pirinen | Updated use cases descriptions |

# 1. PREFACE

Purpose of this document is to be a very light weight systems requirements specification for a multi-player worm game made in network application development and software testing courses in university of applied sciences. Only those parts of software, that will be tested on testing course, are documented a little more in detail.

# 2. SPECIFICATION

## 1.1 Description

This is a multi-player online version of the classic worm game, that will run in web browser. There can be maximum 4 worms at the same time on game board. Each player controls his worm. Food for worms is randomly spawned to the game board. When worm eats food, its length grows, and thereby worm gets harder to control. Worm dies, if it collides game board walls or other worms. Each eaten food will increase player's score. Winner of the game is player, who has the highest score. Game is over, when there's no any worms alive.

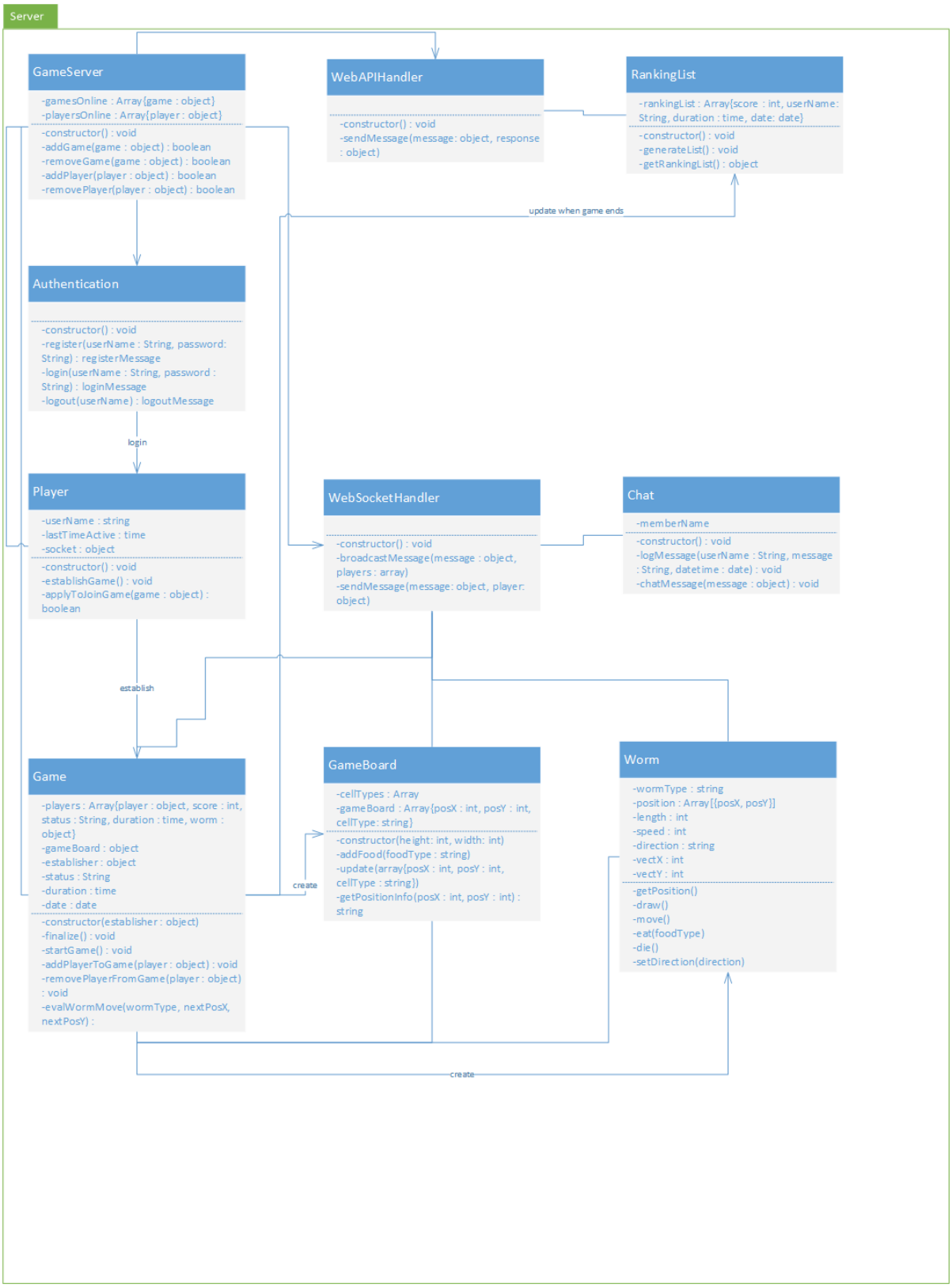
## 1.2 Architecture

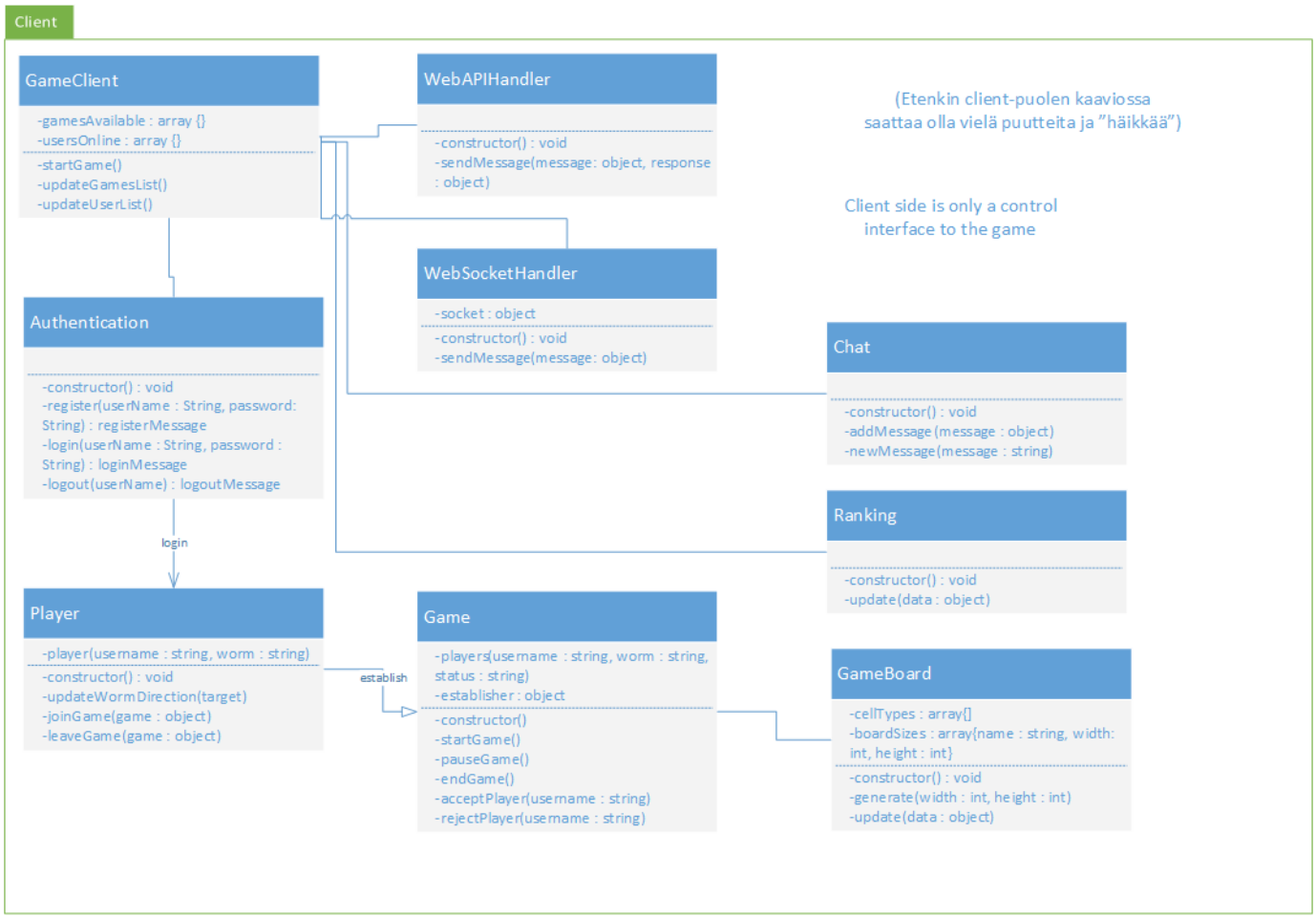
This will be a server-client application. Game server runs the game, when clients' role is to be only a controlling dashboard interface to it through web browser.

Server:
- Node.js framework
- Express.js framework
- MySQL database
- Socket.IO for utilizing web sockets

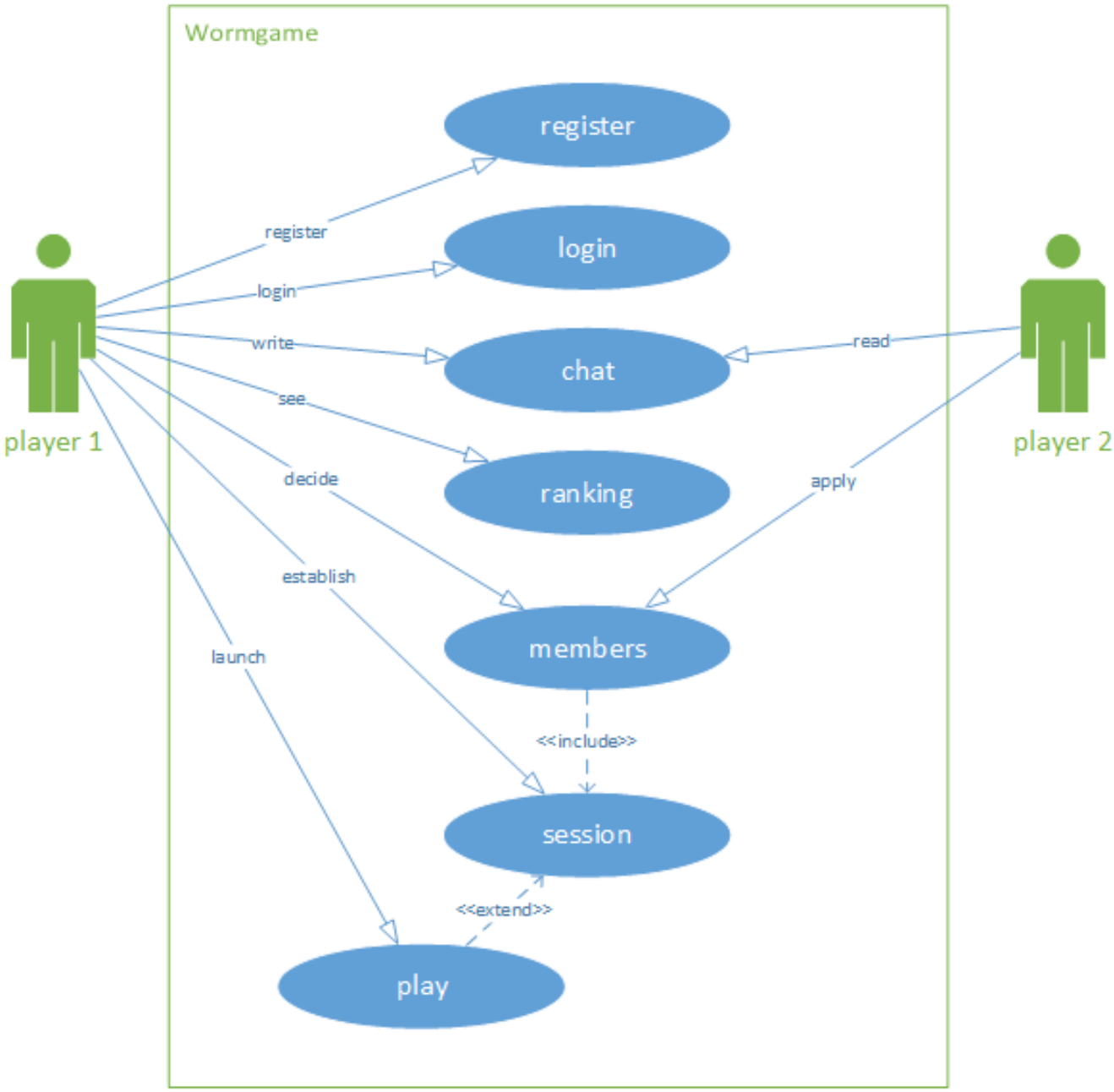
Client:
- HTML5 + CSS3
- Javascript (ecmascript 5)
- Socket.IO for utilizing web sockets

## 1.3 Concept models




Client
GameClient
-gamesAvailable : array {}
-usersOnline : array {}
-startGame()
-updateGamesList()
-updateUserList()
WebAPIHandler
-constructor() : void
-sendMessage(message: object, response : object)
(Etenkin client-puolen kaaviossa saattaa olla vielä puutteita ja ”häikkää”)
Client side is only a control interface to the game
WebSocketHandler
-socket : object
-constructor() : void
-sendMessage(message: object)
Authentication
-constructor() : void
-register(userName : String, password: String) : registerMessage
-login(userName : String, password : String) : loginMessage
-logout(userName) : logoutMessage
Chat
-constructor() : void
-addMessage(message : object)
-newMessage(message : string)
Ranking
-constructor() : void
-update(data : object)
login
Player
-player(username : string, worm : string)
-constructor() : void
-updateWormDirection(target)
-joinGame(game : object)
-leaveGame(game : object)
establish
Game
-players(username : string, worm : string, status : string)
-establisher : object
-constructor()
-startGame()
-pauseGame()
-endGame()
-acceptPlayer(username : string)
-rejectPlayer(username : string)
GameBoard
-cellTypes : array[]
-boardSizes : array{name : string, width: int, height : int}
-constructor() : void
-generate(width : int, height : int)
-update(data : object)

## 1.4 Use cases




*NOTE: not all use cases are described below; this document is incomplete*

**USE CASE:**

User registration

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

**POSTCONDITIONS:**

Username is unique and can be found from database

Password is encrypted and can be found from database

Password is encryptable

**SUCCESS SCENARIO OF EVENTS:**

1. User opens the registration form (if it's not on the front page)
2. User writes username. Letters are visible on screen.
3. User writes password. Letters are invisible on screen.
4. User presses registration button.
5. User sees from screen, that registration was successful.

**ALTERNATIVE EXTENSIONS OF EVENTS:**

5 a. User sees message on screen, that registration was unsuccessful, because username already exists

5 b. User sees message on screen, that registration was unsuccessful, because username or password was empty or had illegal characters

5 c. User sees message on screen, that registration was unsuccessful, because username or password was too short or long

5 d. User sees message on screen, that registration was unsuccessful, for some other (technical reason

**USE CASE:**

User login

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

User has registered

**POSTCONDITIONS:**

User stays logged in, until disconnected by page reload, closing browser window or network disconnection

**SUCCESS SCENARIO OF EVENTS:**

1. User opens the login form (if it's not on the front page)
2. User writes username. Letters are visible on screen.
3. User writes password. Letters are invisible on screen.
4. User presses login button.
5. User sees from screen, that login was successful.

**ALTERNATIVE EXTENSIONS OF EVENTS:**

5 a. User sees from screen, that login was successful, because username or password was incorrect

5 b. User sees message on screen, that registration was unsuccessful, for some other (technical) reason

**USE CASE:**

Chat

**ACTORS:**

User (1-n users)

**PRECONDITIONS:**

User has opened game's front page

**POSTCONDITIONS:**

Chat messages can be found from system log or database

**SUCCESS SCENARIO OF EVENTS:**

1. If user has not logged in, he/she has a dummy username; otherwise she has a registered username
2. User writes a chat message
3. User presses return or pushes button
4. All connected users (including user him-/herself) can read the message, with username of sending user and sending time

**ALTERNATIVE EXTENSIONS OF EVENTS:**

None

**USE CASE:**

Seeing ranking list

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

**POSTCONDITIONS:**

None

**SUCCESS SCENARIO OF EVENTS:**

1. User can see a ranking list of top 30 highest scores, with user names, on the screen
2. If some user gets a top 30 score in the game, user can see that ranking list becomes updated immediately after it

**ALTERNATIVE EXTENSIONS OF EVENTS:**

None

**USE CASE:**

Establish game session

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

User has registered

User has logged in

**POSTCONDITIONS:**

Game session stays established, until game is over, all members of it have became disconnected or some user has ended the session

**SUCCESS SCENARIO OF EVENTS:**

1. User presses establish-button
2. User sees that game session has became established, and establish-button disappears
3. User becomes automatically a member of established session

**ALTERNATIVE EXTENSIONS OF EVENTS:**

1 a. Establish-button doesn't exists, because someone else has established a game or max amount of game sessions is reached

2 a. User receives an error message, that game session couldn't be established

**USE CASE:**

Becoming member of a game session

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

User has registered

User has logged in

User or other user has established a game session

**POSTCONDITIONS:**

User stays member of a game session as long as session exists or until user exits the session

**SUCCESS SCENARIO OF EVENTS:**

1. User presses the join button

2. User becomes a member of a game session, that he/she can see from the screen

**ALTERNATIVE EXTENSIONS OF EVENTS:**

2 a. User couldn't become member of the game session, and reason will be shown on screen: session has already max amount of users, or user didn't become accepted by other members of the game session

**USE CASE:**

Playing the wormgame

**ACTORS:**

User

**PRECONDITIONS:**

User has opened game's front page

User has registered

User has logged in

User or other user has established a game session

User has became member of a game session

**POSTCONDITIONS:**

None

**SUCCESS SCENARIO OF EVENTS:**

1. Some of the users presses the start button
2. The game is running and user can play it as long as he/she is alive
3. User's personal part of the multiplayer game is over, but he/she stays online watching other users' game, until all players' game is over.
4. User can see all scores of each player in the game over screen.
5. User presses a button and the game session ends.

**ALTERNATIVE EXTENSIONS OF EVENTS:**

3 a. User's single-player game is over.

## 1.5 Functional requirements

- Single-player online-game for registered users
- Multi-player online-game for registered users; max 4 online players; each has unique worm colour; each player uses own browser windows for the game
- User registration (username and password) open to anyone; returns information if fails
- User authentication (username and password); returns information if fails
- Ranking list of highest scores in the game
- Text chat between registered and unregistered users
- Chat messages archived in the database (user, ip, datetime, message)

## 1.6 Non-functional requirements

- Playable; controlled by mouse or keyboard
- Modular and following OOP paradigm
- Installable to Red Hat OpenShift
- Passwords encryption in database
- Testable

## 1.7 Project plan

**Week 4**

- Specification
- GitHub repository
- Single player game (UI/client)
- User registration (UI/client, server, database)

**Week 5**

- Authentication (UI/client, server, database)
- Ranking list top 30 (UI/client, server, database)
- Chat (UI/client, server, database)
- Online players (UI/client, server)

**Week 6**

- Multi-player game (UI/client, Server, Database)

**Week 7**

- Finalizing the project
- Installable to OpenShift
- Software testing (partially)
- Debugging