**Peer-to-peer torpedo**

**Rendszerterv**

Webes kétszemélyes P2P játék   
 Szoftverarchitektúrák házi feladat

**Készítették:**

* **Földi Tamás János – HAOYZH**
* **Fábry Bálint –**

# Tartalom

[Tartalom 2](#_Toc436239186)

[A rendszer célja, funkciói és környezete 3](#_Toc436239187)

[Feladatkiírás 3](#_Toc436239188)

[A rendszer által biztosítandó tipikus funkciók 3](#_Toc436239189)

[Program környezete 3](#_Toc436239190)

[Megvalósítás 4](#_Toc436239191)

[Architektúra 4](#_Toc436239192)

[Adatbázis réteg 4](#_Toc436239193)

[Adatelérési réteg 4](#_Toc436239194)

[Nézetkezelő réteg 4](#_Toc436239195)

[Felhasználói felület 4](#_Toc436239196)

[Telepítési leírás 5](#_Toc436239197)

# ****A rendszer célja, funkciói és környezete****

## Feladatkiírás

Alkalmazás célja WebRTC használatával körökre oszott, kétszemélyes, webes peer-to-peer játék (Pl: tick-tack-toe, malom) megvalósítása. A játékosokról statisztikát kell tárolni valamilyen adatbázisban.

A feladat részletes specifikációja a követelmények specifikálására létrehozott dokumentumban található meg.

## A rendszer által biztosítandó tipikus funkciók

Vázlatosan az alábbi funkciók biztosítását várjuk el a rendszertől. (A funkciók részletes definíciója szintén a követelményspecifikáció dokumentumban olvasható.)

* Felületen történő regisztráció és bejelentkezés
* Felhasználó profil megtekintése és szerkesztése
* A profilon statisztikák megjelenítése
* Játékkezdési lehetőség bejelentkezett illetve vendég felhasználók számára
* Ellenfél megkeresése
* Peer-to-peer kapcsolat kiépítése a felek között
* Játék lejátszása

## Program környezete

Az elkészített szoftver egy Node.js környezetben futó webes alkalmazás, mely WebRTC technológia segítségével peer-to-peer kommunikációt tesz lehetővé a felek között. A felhasználói adatok MongoDB-ben tárolódnak. A frontend AngularJs, JavaScript, TypeScript, HTML, CSS nyelvek felhasználásával íródott, míg a backend JavaScriptben. A fent leírt tulajdonságok miatt a program használatához egy korszerű WebRTC-t támogató böngészőre van szükség, ennek pontos specifikációja szintén megtekinthető a követelményspecifikációban.

# Megvalósítás

Az alkalmazást többrétegű alkalmazásként készítettük el úgy, hogy az egyes rétegek jól elkülönülnek.

A fejezetben áttekintést adunk a program architektúrájáról, bemutatjuk az egyes komponensek feladatait és felelősségeit, továbbá részletesen ismertetjük a használt adatmodellt és a grafikus felhasználói felület felépítését

## Architektúra

A játék architektúrája 4 különálló részre bontható. Melyek a következők:

* adatbázis réteg
* peerkezelő réteg
* adatelérési réteg
* nézet kezelő réteg
* felhasználói felület

### Adatbázis réteg

**Célja:** Az adatbázis réteg felel az adatok perzisztálásáért.

Ennek megvalósítására MongoDB-t illetve a modellezéshez illetve validáláshoz pedig MongooseJs-t.

A választásunkat indokolja, hogy MongoDB egy korszerű technológia és mivel JSON objektumban tárolja az adatokat, ezért JavaScripttel való használata kényelmes. MongooseJs pedig egy MongoDB eléréséhez és használatához kényelmes felületet nyújtó Node.js tool, mely jól dokumentált és széleskörben használt ezért a közösségi háttere is nagy, ami nagyban segíti a gyors és hatékony fejlesztés.

**Kódban való megjelenés:** *./lib/modells/userSchema.js*

### Peer kezelő réteg

**Célja:** A peerkezelő réteg célja a peer-to-peer kapcsolat jelzésszerver adatainak kezelése és tárolása.

Mivel a peereket egymáshoz kell tudni rendelni, ezért lehetőséget kell biztosítani a peereknek egy egyedi azonosítóval történő beregisztrálására. A regisztrált peereket el kell tárolni, hogy később ebből az úgynevezett peer poolból ki tudjuk keresni a véletlenszerű vagy direkt peerhívások által kért peereket és egymáshoz tudjuk rendelni őket, hogy ezután már szerver nélkül egymás közt tudjanak adatot cserélni.

**Kódban való megjelnés:** ./lib/peerPool.js

### Adatelérési réteg

**Célja:** Authentikációs feltételek ellenőrzése után az adatbázisban tárolt adatok lekérése az adatbázis rétegtől vagy a peer pool használata.

Az authentikációhoz JSON Wen Token technológiát alkalmaztunk, előnye, hogy nem igényel session használatot illetve egy gyors biztonságos authentikáló protokoll JavaScript környezetben. Az adatok lekéréséhez a MongooseJs megfelelő interfészét használtuk.

A peerpool interfészeit is html hívások hatására lehet elérni, így tud a kliens kérést adni hogy bekerüljön a poolban vagy így tud hívást kezdeményezni.

**Kódban való megjelenés:** *./lib/config/user-routes.js,* ./lib/config/peer-routes.js

### Nézetkezelő réteg

**Célja:** A felhasználói felület mögé egy olyan réteg megvalósítása mely elvégzi a szerverrel való kommunikációt, illetve a lokális műveleteket végzi el.

A megvalósításhoz AngularJs illetve JavaScriptet használtunk. Előnye hogy lehetővé teszi a kétirányú adatkötést, jól struktúrált programot kíván meg a fejlesztőtől, illetve lehetőség van benne előre definiált interfészeken keresztül kommunikálni az alkalmazásszerverrel.

**Kódban való megjelnés:** *./app/scripts/\**

### Felhasználói felület

**Célja:** Ez a réteg szolgáltatja a felhasználónak azt az interfészt melyen könnyedén kényelmesen tudja kezelni az alkalmazást, azaz képes regisztrálni játszani egy emberi agy számára könnyen érthető felületen keresztül.

**Kódban való megjelenés:** *./app/views/\**

Adatbázis réteg

Peer kezelő réteg

Adatelérési réteg

Nézetkezelő réteg

Felhasználói felület

# Telepítési leírás

Az alkalmazás elérhető egy Google Cloud szerveren a <http://173.255.124.206> címen. A felhasználónak ezt a címet kell beírni a böngészőjének URL sávjába és miután betöltött az alkalmazás lehetősége van annak használatára, amennyiben a böngészője teljesíti a követelményspecifikációban leírt feltételeket.