

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM

SimultanQ - egy interaktív kvízzjáték tervezése és fejlesztése

DIPLOMADOLGOZAT

Témavezető:
Dr. Antal Margit,
Egyetemi docens

Végzős hallgató:
Veres Napsugár Anna

2023

UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ



UNIVERSITATEA
SAPIENTIA

SimultanQ - proiectarea și dezvoltarea unui joc interactiv de tip
quiz

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Antal Margit,
Conferențiar universitar

Absolvent:
Veres Napsugár Anna

2023

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION**



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA


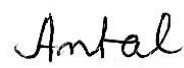
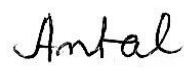
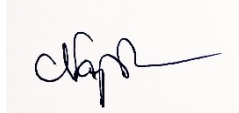
SimultanQ - the Design and Development of an Interactive Quiz
Game

BACHELOR THESIS

Scientific advisor:
Dr. Antal Margit,
Associate Professor

Student:
Veres Napsugár Anna

2023


UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA Facultatea de Științe Tehnice și Umaniste din Târgu Mureș Programul de studii: Informatică		Viza facultății:
LUCRARE DE DIPLOMĂ		
Coordonator științific: Conf. dr. ANTAL Margit		Candidat: Veress Napsugár Anna Anul absolvirii: 2023
<p>a) Tema lucrării de licență: SimultanQ - proiectarea și dezvoltarea unui joc interactiv de tip quiz</p> <p>b) Problemele principale tratate: - Studiu bibliografic privind jocurile interactive de tip quiz - Tehnologii alese pentru implementarea aplicației: protocolul WebSocket, Spring Boot pentru backend și Angular pentru frontend</p> <p>c) Desene obligatorii: - Schema bloc a aplicației - Diagrame de proiectare pentru aplicația software realizată.</p> <p>d) Softuri obligatorii: - backend implementat în Spring Boot - frontend implementat în Angular</p> <p>e) Bibliografia recomandată:</p> <ul style="list-style-type: none"> • https://spring.io/ • https://spring.io/projects/spring-security • https://angular.io/ 		
<p>f) Termene obligatorii de consultații: săptămânal</p> <p>g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, laboratorul 417</p> <p>Primit tema la data de: 01.06.2022. Termen de predare: 02.07.2023.</p>		
Semnătura Director Departament 		Semnătura coordonatorului 
Semnătura responsabilului programului de studiu 		Semnătura candidatului 

Declarație

Subsemnatul/a VERES NAPSUGĂR ANNA, absolvent(ă) al/a specializării
INFORMATICĂ, promoția 2020-2023 cunoscând
prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a
Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta
lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală,
cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de
specialitate sunt citate în mod corespunzător.

Localitatea, TÂRGU-MUREȘ,
Data: 13.06.2023.

Absolvent

Semnătura.....

Kivonat

Dolgozatom témája egy interaktív kvízzjáték kivitelezése a résztvevő játékosok számának korlátozása nélkül egy webes alkalmazás segítségével. A kvízzjátékok rendkívül népszerűek napjainkban, és számos előnyt kínálnak. Az emberek élvezik a kihívásokat, főleg, ha ezek játékos formában kerülnek eléjük. Az élő kvízzjátékok izgalmas és interaktív élményt nyújtanak, mivel lehetővé teszik a játékosok számára, hogy versenyezzenek egymással és tesztelhessék tudásukat. A többjátékos mód pedig mindezt biztosítja.

Az, hogy a játékosok egyszerre vesznek részt a kvízzjátékban, és lehetőség van az eredmények valós idejű követésére, a játékot még izgalmasabbá teszi, sőt, mindez közösségépítő erővel is bírhat.

Az általam tervezett alkalmazás neve, a "SimultanQ", ezt a koncepciót hivatott képviselni. A "Simultan" a közös, egyidejű tevékenységre utal, míg a "Q" végződés a "Quiz" (kvíz vagy teszt) angol szót takarja.

A már létező, online és ingyenesen elérhető hasonló alkalmazásokat vizsgálva kiderül, hogy a játék során korlátozva van a résztvevők száma. A projekt keretein belül egy olyan webes alkalmazás került megvalósításra, amelynek nincsenek ilyen korlátozásai. A felhasználók könnyedén regisztrálhatnak, bejelentkezés után létrehozhatnak kvízeket és egyszerű módon elindíthatják a játékot, bekapcsolhatják a kvíz megoldásába a játékosokat egy PIN-kód megosztásával, a játékosok pedig ennek ismeretében hozzáférhetnek a kérdésekhez.

Az alkalmazás hozzájárul az emberek által kedvelt kihívások megvalósításához, különböző csoportok különböző témában való érdekesítő játékainak kivitelezéséhez, tudásuk teszteléséhez, a versenyszerű megközelítés pedig igazi csapatformáló élményt nyújthat.

Kulcsszavak: kvízzjáték, interaktív, kihívások

Rezumat

Tema lucrării mele este dezvoltarea unui joc interactiv de tip quiz fără restricții legate de numărul de participanți, prin intermediul unei aplicații web.

Jocurile de tip quiz sunt extrem de populare în zilele noastre și oferă numeroase avantaje. Oamenii se bucură de provocări, mai ales când acestea sunt prezentate sub formă de joc. Jocurile de tip quiz oferă o experiență palpitantă și interactivă, deoarece permit jucătorilor să concureze între ei și să-și testeze cunoștințele. Modul multiplayer facilitează toate acestea.

Faptul că jucătorii participă simultan la jocul de tip quiz și există posibilitatea urmăririi în timp real a rezultatelor face jocul și mai captivant și, în plus, poate avea o putere de formare a echipei.

Numele aplicației pe care am dezvoltat este "SimultanQ" și urmărește să reprezinte acest concept. "Simultan" face referire la activitățile comune și simultane, în timp ce sufixul "Q" provine de la "Quiz" (quiz sau test) în limba engleză.

Analizând aplicațiile similare existente, accesibile online și gratuite, se constată că numărul de participanți este restricționat în timpul jocului. În cadrul proiectului meu, am dezvoltat o aplicație web fără astfel de restricții. Utilizatorii pot să se înregistreze ușor, după autentificare, pot crea quiz-uri și pot lansa jocul într-un mod simplu, permitând jucătorilor să se alăture rezolvării quizului prin partajarea unui cod PIN, iar jucătorii pot accesa întrebările pe baza acestuia.

Aplicația contribuie la realizarea provocărilor pe care oamenii le apreciază, dezvoltarea unor jocuri captivante în diferite tematici pentru diverse grupuri, testarea cunoștințelor și abordarea competitivă poate oferi o adevărată experiență de formare a echipei.

Cuvinte de cheie: joc de tip quiz, interactiv, provocări

Abstract

The topic of my thesis is to implement an interactive quiz game without restricting the number of participating players using a web application.

Quiz games are extremely popular nowadays and offer numerous advantages. People enjoy challenges, especially when presented in a playful form. Live quiz games provide an exciting and interactive experience as they allow players to compete with each other and test their knowledge. The multiplayer mode facilitates all of this.

The fact that players participate simultaneously in the quiz game and there is the possibility of real-time tracking of results makes the game even more thrilling and can have a community-building impact.

The name of my application is "SimultanQ," which aims to represent this concept. "Simultan" refers to shared, simultaneous activities, while the suffix "Q" stands for "Quiz."

Analyzing existing online and freely accessible similar applications reveals that the number of participants is limited during the game. Within the scope of this project, a web application has been developed without such restrictions. Users can easily register, create quizzes after logging in, and start the game in a straightforward manner. They can enable players to join the quiz by sharing a PIN code, and the players can access the questions based on this information.

The application contributes to the realization of beloved challenges, the implementation of engaging games in various topics for different groups, testing knowledge, and the competitive approach can provide a genuine team-building experience.

Keywords: quizgame, interactive, challenges

Tartalomjegyzék

1. Bevezető	11
2. Célkitűzések	12
3. Elméleti megalapozás és szakirodalmi tanulmány	13
3.1. Hasonló alkalmazások	13
3.1.1. Kahoot!	13
3.1.2. Gimkit	14
3.1.3. Quizlet	15
3.1.4. Mentimeter	16
3.1.5. Poll Everywhere	17
3.1.6. Összegzés	18
3.2. Felhasznált technológiák	18
3.2.1. Spring Boot	18
3.2.2. WebSocket	20
3.2.3. Angular	22
4. A rendszer specifikációja	24
4.1. Felhasználói követelmények	24
4.1.1. Szerepkörök	24
4.2. Fontosabb használati esetek	25
4.2.1. Bejelentkezett felhasználó használati esetei	25
4.2.2. A játékosok használati esetei	27
4.3. Rendszerkövetelmények	28
4.3.1. Funkcionális követelmények	28
4.3.2. Nem funkcionális követelmények	29
5. Tervezés	30
5.1. A rendszer architektúrája	30
5.2. Adatbázis	31
5.3. Szerveroldali architektúra	31
5.3.1. REST API kérések	31
5.3.2. WebSocket	32
5.4. Kliensoldal	33
5.4.1. Modulok, komponensek és szolgáltatások	33
5.4.2. Felhasználói felület tervezése	34

6. Kivitelezés	36
6.1. Szerveroldali megvalósítás	36
6.1.1. Entity réteg	36
6.1.2. Repository réteg	38
6.1.3. Service réteg	39
6.1.4. Controller réteg	39
6.1.5. Websocket implementálása	40
6.2. Kliensoldali megvalósítás	42
6.2.1. Modellek	43
6.2.2. Komponensek	44
6.2.3. Szolgáltatások	46
7. Eszközök és munkamódszerek	49
7.1. Segédeszközök	49
7.1.1. Integrált fejlesztői környezet (IDE)	49
7.1.2. Adatbázis	49
7.1.3. Verziókövetés eszközök	49
7.1.4. Api tesztelés	49
7.1.5. Design eszközök	49
7.2. Tesztelés	50
Összefoglaló	51
Köszönetnyilvánítás	52
Ábrák jegyzéke	53
Irodalomjegyzék	55

1. fejezet

Bevezető

Ebben a fejezetben röviden bemutatásra kerül a dolgozatom témája, illetve ismertetem a folyamatot, amely során létrejött az alkalmazás gondolata és kialakult a témaválasztás.

A kvízzjátékok rendkívül népszerűek, és számos előnyt kínálnak, mivel az emberek szeretnek kihívásokkal szembesülni, főleg, ha ez játékosan van tálalva. Az élő kvízzjátékok izgalmas és interaktív élményt nyújtanak, mert biztosítják a játékosok számára, hogy versenyezzenek egymással és ellenőrizhessék tudásukat. A többjátékos mód mindezt lehetővé teszi.

Az, hogy a játékosok egyszerre vesznek részt a kvízzjátékban, és lehetőség van az eredmények valós idejű követésére, a játékot még izgalmasabbá teszi, sőt, mindez közösségépítő erővel is bírhat.

Az alkalmazásom elnevezése, a "SimultanQ" is ezt hivatott képviselni. A "Simultan" az egyszerre történő tevékenységre utal, a "Q" végződés pedig a "Quiz" (kvíz vagy teszt) angol kifejezés rövidítése.

Az alkalmazás alapötlete onnan származik, hogy megvizsgálva a már létező, online és ingyenesen elérhető hasonló alkalmazásokat, az ember a játékosok létszámának korlátozásával szembesül. Azaz, a kvízzjátékok ingyenes verziójában a játékosok létszáma korlátozva van 5, 10, 20, 40 főre, ami például egy egyetemi évfolyam csoportja vagy egy közepes vagy nagyobb rendezvény során hátrányt jelent. Így a saját alkalmazásom tervezése és fejlesztése létjogosultságot nyert.

A továbbiakban ismertetni szeretném röviden a fejezetek tartalmát. A bevezetőt követően a célkitűzések megfogalmazása következik, ahol felsorolom a technológiai célokat, amelyeket a tervezés és fejlesztés során szem előtt tartottam. Ezután következik az elméleti megalapozás és a szakirodalmi tanulmány fejezet. Ebben a fejezetben bemutatásra kerülnek azok a népszerű kvízzjátékok, amelyek inspirációt jelentettek a saját applikációm, a SimultanQ létrehozásában: Kahoot!, Gimkit, Quizlet, Mentimeter és Poll Everywhere. Ugyanebben a fejezetben bemutatásra kerültek az alkalmazás implementálása során felhasznált technológiák. A következő fejezetben foglalkozom a rendszer specifikációival, majd a tervezési részletek és architektúrák felépítése következik. A hatodik fejezetben a kivitelezés kerül bemutatásra, azután pedig felsorolom azokat az eszközöket és munkamódszereket, amelyeket használtam a fejlesztés során. Végül a dolgozat összefoglalója kerül sorra, ahol megfogalmazom a továbbfejlesztési lehetőségeket és összegzem a végső következtetéseket.

2. fejezet

Célkitűzések

Ebben a fejezetben a projekt megvalósításával kapcsolatos kitűzött célok kerülnek bemutatásra, tartalmazva az applikáció implementációjával kapcsolatos célkitűzéseket.

Az applikáció fejlesztésével kapcsolatos célkitűzések a következők:

- az applikáció kinézetének megtervezése,
- az applikáció különböző összetevőinek megtervezése (adatbázis, backend, frontend),
- az applikáció részei közötti kapcsolatok megtervezése,
- különböző funkciók integrációja: regisztráció, bejelentkezés, kvízek létrehozása, kvízek listázása, kvízzjáték indítása, játékhoz való kapcsolódás.

3. fejezet

Elméleti megalapozás és szakirodalmi tanulmány

3.1. Hasonló alkalmazások

A következőkben bemutatásra kerül néhány népszerű kvízzjáték, amelyek inspirációt jelentettek a saját applikációm létrehozásában[1]: Kahoot!, Gimkit, Quizlet, Mentimeter, Poll Everywhere.

3.1.1. Kahoot!



A Kahoot! egy interaktív online kvízzjáték platform, amely lehetővé teszi a felhasználók számára, hogy létrehozzanak, megosszanak és együtt játsszanak különböző kvízzjátékokat. Az alkalmazás elérhető webes böngészőben és mobil eszközökön is.

- **Kvízkészítés**

- A Kahoot! Quiz Creator segítségével elkészíthetők a kvízek. Egyszerűen hozzáadhatóak több választos kérdések, képek, hangfájlok és videók az adott játékhoz.

- **Élő játék**

- A kvízzjáték egyik sajátossága, hogy élő játék formájában bonyolítható le, azaz a játék során a résztvevők csatlakozhatnak a kvízhez saját eszközeiken, és válaszolhatnak a kérdésekre a meghatározott időn belül - a Kahoot! Live game opció segítségével. A válaszok gyorsasága és pontossága alapján pontokat szerezhetnek, ami versenyszerű és interaktív élményt nyújt.

- **Kihívások**

- Kahoot! Challenge: ez az opció lehetővé teszi, hogy a játékosok önállóan, a saját tempójukban játsszák végig a kvízt.

- **Pontozás**

- A gyorsabb válaszok több pontot érnek. A résztvevők pontszáma a játék során is nyomon követhető.

- **Statisztikák**

- A játék végeztével az azzal kapcsolatos jelentések és statisztikák elérhetővé válnak, úgy, mint a játék résztvevői teljesítményének elemzése, a helyes/hibás válaszok, pontszámok.

- **Közösség**

- A kvízek téma szerint is könnyen testreszabhatóak, lehetőség van a publikálásra és megosztásra más felhasználókkal annak érdekében, hogy ők is csatlakozhassanak a játékhoz. Más felhasználók által összeállított kvízek között is böngészhetünk.

- **Elérhetőség**

- A Kahoot! játék ingyenes verziója egyszerre 10 játékos számára biztosítja a közös játékot.

3.1.2. Gimkit



A Gimkit - interaktív online tanulási platform és kvízzjáték szoftver, amelyet kifejezetten az oktatási környezetben való használatra hoztak létre.

- **Kvízkészítés**

- Saját kvízeket lehet létrehozni, kérdések, válaszok hozzáadásával, képekkel, videókkal, hangfájlokkal.

- **Többféle játékmód**

- Az alkalmazáson belül elérhető többféle játékmód. Például a "Classic" módban a játékosok a helyes válaszokért pontokat kapnak, a "Team" módban pedig a résztvevők csapatokat alkothatnak és egymással is versenyezhetnek.

- **Virtuális valuta**

- A Gimkit különböző opciókkal ösztönzi a játékosokat, ezek közé tartozik egy-fajta virtuális valuta kiosztása, gyűjtése, amelynek segítségével a játékosok különböző előnyökhöz juthatnak, például gyorsabb válaszadás vagy több pont megszerzése.
- **Statisztikák**
 - Az alkalmazás jelentéseket generál a válaszadások után, amelyek segítenek megérteni a tanároknak és a diákoknak az eredményeket, lehetővé teszi a diákok teljesítményének elemzését.
- **Közösség**
 - A Gimkit-nek van egy kérdésszerkesztő közössége, ahol a felhasználók megoszthatják egymással a saját kvízkérdéseiket, válaszlehetőségeiket, illetve elérhetővé tehetik a teljes kvízzjátékaikat is mások számára.
- **Elérhetőség**
 - A Gimkit játék ingyenes verziója egyszerre 5 játékos számára biztosítja a közös játékot.

3.1.3. Quizlet

Quizlet

A Quizlet ugyancsak egy kiterjedt tanulási platform, amelyet tanuló anyagok létrehozása, megosztása érdekében fejlesztettek.

- **Kvízkészítés**
 - A felhasználók saját digitális tanuló anyagokat hozhatnak létre, köztük kvízeket is. Lehetséges a kártyák, kifejezések és definíciók párosítása is. Egyedi célokhoz egyedi tanulói anyagok.
- **Közösség**
 - A Quizlet segítségével a tanulók közös projekteken is dolgozhatnak, közös tanulói csoportokat hozva létre, segítve egymást.
- **Tanulás és gyakorlás**
 - A létrehozott tanulói anyagok megosztása segít a tananyag elsajátításában, gyakorlásában. A tanulók a saját tempójukban és preferenciáik szerint gyakorolhatnak, az alkalmazás igazodik a felhasználók egyedi tempójához.
- **Élő játék**

- A Quizlet Live segítségével élő játékokra is van lehetőség a felhasználók között számítógépen és mobil eszközökön is.
- **Támogatás több nyelven**
 - A Quizlet több nyelvet is támogat, köztük a német, spanyol, japán koreai, lengyel és orosz nyelveket is.
- **Elérhetőség**
 - A Quizlet ingyenes verziója egyszerre maximum 30 játékos számára biztosítja a közös játékot.

3.1.4. Mentimeter



A Mentimeter egy interaktív alkalmazás, amelyet elsősorban prezentációkra és felmérésekre fejlesztettek. Az applikáció segítségével résztvevő-orientált prezentációkat lehet létrehozni.

- **Kvízek létrehozása**
 - A Mentimeter is lehetővé teszi különböző interaktív kvízek létrehozását: felletválasztós, szöveges válaszos típusúak vagy prioritási sorrend típusúak.
- **Élő játék**
 - Az applikációban lehetséges a valós idejű játék, a válaszadóktól való véleménygyűjtés, reakciók rögzítése. Ezen kívül elérhetőek különböző vizualizációs eszközök, melyek segítségével megvalósíthatóak diagramok, szófelhők vagy sorozatos képek az eredmények könnyebb értelmezése érdekében.
- **Moderáció és ellenőrzés**
 - A Mentimeter applikációban a kérdéseket és a válaszokat lehet moderálni, így létrejöhét a tartalomszűrés, ellenőrzés, sőt, a módosítás is, mielőtt a széles közönség elé lenne terjesztve. Ha nagyobb eseményre vagy rendezvényre gondolunk, ez különösen hasznos lehet.
- **Egyedi design**
 - Az alkalmazáson belül a felhasználói felület testreszabható, amely egy saját márka kiépítése során hasznos lehet, mivel egyedi megjelenést biztosít a prezentációknak.

- **Integráció**

- A Mentimeter integrálható prezentációs szoftverekbe, mint például a PowerPoint vagy Keynote, ezáltal interaktív elemek adhatóak hozzá a bemutatókhoz.

- **Elérhetőség**

- A Mentimeter alkalmazás ingyenes verziója egyszerre legfeljebb 20 személy számára biztosítja a közös játékot.

3.1.5. Poll Everywhere



A Poll Everywhere alkalmazás erőssége a szavazás lehetősége és a közösség bevonása. Segítségével interaktív kvízeket, szavazásokat lehet létrehozni.

- **Kvízek létrehozása**

- Különböző típusú kvízeket lehet létrehozni: feleletválasztós, skála, szöveges válaszos és mint a Mentimeter esetében, prioritási sorrend típusút.

- **Valós idejű játék**

- A Poll Everywhere alkalmazásnak is erőssége a valós idejű játék. Csatlakozni lehet számítógépről és mobil eszközökről is. Az eredmények azonnal megjelennek a képernyőn, és a vizualizációs eszközök segítségével a közönség teljesen bevonható a folyamatokba. Diagramok, szavazógombok segítenek az adatok megjelenítésében.

- **Statisztikák**

- Az applikációval megvalósítható a résztvevők véleményeinek begyűjtése, interakciók, reakciók rögzítése, eredmények összegzése.

- **Részletes elemzések**






- A Poll Everywhere segítségével részletes elemzések és beszámolók is létrehozhatóak. Ezáltal az események hatékonysága, a közönség válaszainak elemzése részletes és teljesebb képet ad a különböző vélemények, állásfoglalások kielemezésében.

- **Elérhetőség**

- A Poll Everywhere ingyenes verziója egyszerre 40 játékos számára teszi lehetővé a közös játékot.

3.1.6. Összegzés

A felsorolt alkalmazások számos funkcionalitást tartalmaznak és segítségükkel létrehozhatóak különböző érdekes kvízzjátékok, melyek sajátossága az élő játék, az interaktivitás, az eredmények azonnali nyomkövethetősége. Egyetlen hátrányuk, hogy az ingyenes, mindenki számára elérhető online verziók csak limitált létszámú csoportok közös játékát teszik lehetővé. Az általam fejlesztett alkalmazás ezt az űrt igyekszik betölteni.

Inspirációt jelentő alkalmazások					
					
Főbb jellemzők	Átlátható, felhasználóbarát	Bonyolultabb feladatok is megvalósíthatóak	Tanulási, oktatási anyagokban gazdag	Versenyek szervezésére kiváló	Erőssége a szavazások megoldása
Élő játék lehetősége	igen	igen	igen	igen	igen
Statisztikák, elemzések	Egyszerű	Az elemzések következtetéseként továbbfejlesztési tervet is javasol	A tanulási folyamat dokumentálható	Egyszerű pontozási lista	Részletes analitika is elérhető
Ingyenes verzió elérhető	Max. 10 személy számára	Max. 5 személy számára	Max. 30 személy számára	Max. 20 személy számára	Max. 40 személy számára
Link	Kahoot! ^[2]	Gimkit ^[3]	Quizlet ^[4]	Mentimeter ^[5]	Poll Everywhere ^[6]

3.1. táblázat. Az inspirációt jelentő alkalmazások főbb jellemzői

3.2. Felhasznált technológiák

Az applikációm kivitelezésében a backend részhez Spring Boot, a frontend részhez pedig Angular keretrendszert használtam. Az élő játék megvalósításához a backend elengedhetetlen alappillére a WebSocket végpontok használata volt.

3.2.1. Spring Boot

A Java ökoszisztémában a Spring ^[7] messzemenően a legnépszerűbb keretrendszer, ami az alkalmazásfejlesztéseket illeti. Jelentős előnyökkel rendelkezik más technológiákhoz képest. Rengeteg lehetőség rejlik benne, és minden kiadással egyre jobbra válik már több,

mint egy évtizede. A Spring Boot-ot a Pivotal Software (jelenleg a VMware része) fejleszti, egy csiszolt, modern és innovatív keretrendszer.

- **Inicializás**

A Spring Boot a Spring keretrendszer kiterjeszése, amely egyszerűsíti a Spring alkalmazások kezdeti konfigurálását. Lehetővé teszi egy működő, önálló Spring alkalmazás létrehozását minimális alapértelmezett konfigurációval. A Spring Initializr webalkalmazás segítségével a Spring Boot projekt könnyedén generálható. Kiválasztható a szükséges konfiguráció, beleértve a fordítási eszközt (build tool), a nyelvet, a Spring Boot keretrendszer verzióját és a projekt összes függőségét. [8] Így a fejlesztőknek kevesebb időt kell tölteni az infrastruktúra beállításával, és inkább az üzleti logika írására koncentrálhatnak.

A Spring Boot széleskörű eszközöket és funkciókat biztosít a fejlesztőknek azáltal, hogy gyorsan és hatékonyan építhetnek robusztus és skálázható alkalmazásokat.

- **Beágyazott szerver**

A Spring Boot egyik kulcsfontosságú funkciója a beágyazott szerver. Beépített servetet tartalmaz, mint például a Tomcat, Jetty vagy Undertow, így nincs szükség az alkalmazás külső szerverre való telepítésére. Ez rendkívül könnyűvé teszi az alkalmazás fejlesztését, tesztelését és telepítését, mivel minden Spring Boot alkalmazásban önmagában megtalálható.

- **Függőségek**

A keretrendszer erős függőségkezelő rendszert is kínál. A kezdeti függőségek koncepciójára épít, amely az előre beállított függőségek és adott funkciók készleteit biztosítja specifikus használati esetekhez. Ezeket a kezdeti függőségeket könnyen be lehet állítani a projekt build funkciójában és automatikusan konfigurálják a szükséges függőségeket és ésszerű alapértékeket adnak. Ez jelentősen csökkenti a projekt beállításához szükséges erőfeszítést, és biztosítja, hogy a legjobb gyakorlati megoldások legyenek alkalmazva.

- **Automatikus konfiguráció**

A Spring Boot másik fontos funkcionalitása az automatikus konfigurációra való képesség. Ez a funkcionalitás elemzi az osztályelérési útvonalat, és automatikusan konfigurálja az alkalmazást a megtalált könyvtárak és függőségek alapján. Ez a funkció könnyű testreszabást és felülírást tesz lehetővé, amely biztosítja a fejlesztőknek az alkalmazás viselkedésének saját igényeik szerinti testreszabását.

- **Annotációk**

A Spring Boot előnyben részesíti a különböző feladatokhoz szükséges annotációk használatát, például a RESTful végpontok meghatározását, a kérések kezelését. A `@RestController`, `@RequestMapping` és `@Autowired` annotációk egyszerűsítik a fejlesztési folyamatot, az ismétlődő, sablonkódot csökkentve, a komponensek rövid és kifejező módon történő azonosításával, meghatározásával. Ez a megközelítés javítja a kód olvashatóságát és karbantarthatóságát.

- **Spring Data JPA**

A Spring Boot automatikusan konfigurálja a Spring Data JDBC repository-kat, amikor a szükséges függőségek megtalálhatóak az osztályelérési útvonalon. A spring-boot-starter-data-jpa függőség egységes és következetes API-t biztosít különböző adattárolási technológiákhoz, mint például a relációs adatbázisok, NoSQL adatbázisok.

- **Spring Security**

A Spring Boot egyik fontos függősége a spring-boot-starter-security, amely a biztonsági funkciók integrálását teszi lehetővé, ideértve az autentikációt, az autorizációt és a munkamenet kezelését. Az autentikáció révén lehetségessé válik a felhasználók azonosítása és hitelesítése, például felhasználónév/jelszó kombinációval vagy esetleg külső hitelesítő szolgáltatásokkal, mint például OAuth vagy SAML. Az autorizáció lehetővé teszi a jogosultságok kezelését és a felhasználókhoz való hozzáférés korlátozását a rendszerben. A munkamenetkezelés lehetőséget ad a felhasználói munkamenetek nyomon követésére és kezelésére.

Összességében a Spring Boot egy jelentős keretrendszer a háttéralkalmazások készítéséhez. Különböző funkcionalitásai hatékony és fejlesztőbarát környezetet biztosítanak lehetővé téve a skálázható, biztonságos alkalmazások fejlesztését. A Spring Boot növekvő és aktív közössége biztosítja, hogy naprakész maradjon a legjobb gyakorlatok tekintetében.

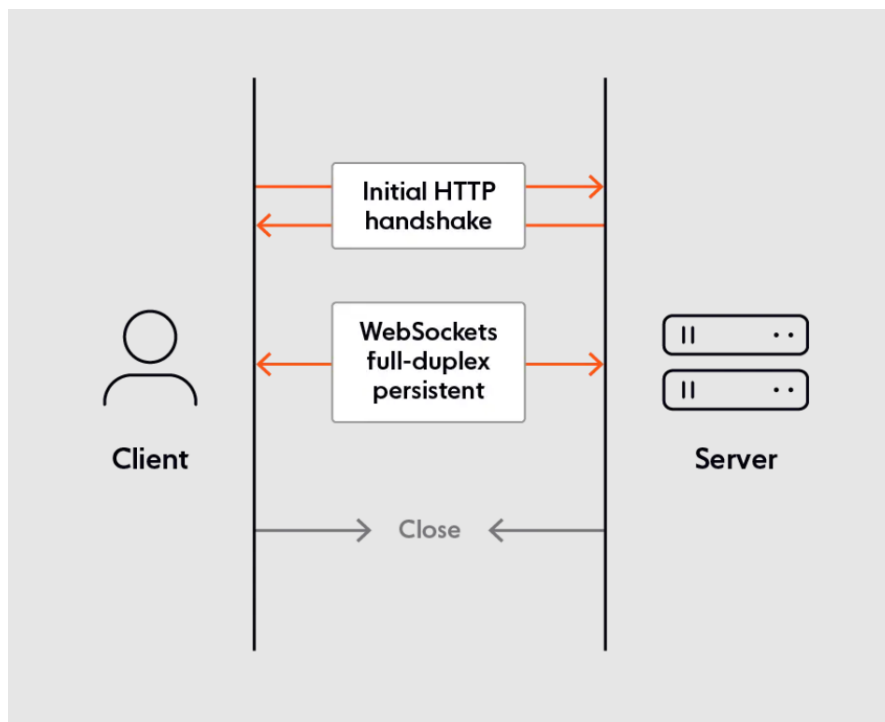
3.2.2. WebSocket

A Spring WebSocket a valós idejű kommunikációhoz nyújt támogatást a Spring alkalmazásokban. A WebSocket protokollt használva lehetővé válik a kétirányú, aszinkron kommunikáció a kliens és a szerver között.

A Spring WebSocket lehetőséget nyújt a valós idejű események és üzenetek gyors és hatékony átvitelére. A WebSocket protokoll a hagyományos HTTP kommunikációtól eltérően nyitva tartja a kapcsolatot a szerver és a kliens között, ezáltal lehetővé teszi az állandó és folyamatos kommunikációt. Ez különösen hasznos azoknál az alkalmazásoknál, ahol azonnali frissítésre és valós idejű információkra van szükség, vagy ahol az időfaktor jelentős fontossággal bír, mint például versenyszerű játékok, chat szolgáltatások vagy valós idejű monitoring rendszerek esetén.

A WebSocket felépíti a kapcsolatot a kliens és a szerver között úgy, hogy a kliens kezdeményezi a kapcsolatot a szerverrel egy speciális HTTP kérelem formájában, amely tartalmazza az "Upgrade" és a "Connection" fejléceket. Ha a szerver támogatja a WebSocket protokollt, akkor válaszként egy "101 Switching Protocols" választ küld, és a protokoll vált a WebSocket-re. Ez a kapcsolat egyszer felépül és fennmarad, amíg egyik fél le nem zárja. Létrejön az üzenetek közlekedése. Az egyszerű szöveges üzenetek szöveggént kerülnek átvitelre, a bináris üzeneteket speciális adatkeretekbe csomagolják.

A Spring Boot a WebSocket protokollt támogató beépített csomagokat és osztályokat biztosít. Használatához először be kell konfigurálni a szerveroldalt. A Spring keretrendszer biztosítja a WebSocketHandlerAdapter és a WebSocketHandler osztályokat, amelyeket a szerver oldalon implementálni lehet. A WebSocketHandler felelős az üzenetek fogadásáért



3.1. ábra. WebSocket kapcsolat [9]

és kezeléséért, míg a `WebSocketHandlerAdapter` az üzenetek kezelését integrálja a Spring alkalmazásba.

A Spring Boot WebSocket anotáció alapú programozási modellt használ, ez megkönnyíti a végpontok definiálását és kezelését. A Spring Boot WebSocket beépített skálázhatósági és teljesítményjavításokat is kínál, amelyek lehetővé teszik az alkalmazások számára, hogy hatékonyan kezeljék a nagy terhelést és a többfelhasználós környezetet.

A kliens oldalon a WebSocket kommunikációt a JavaScript vagy TypeScript segítségével is lehet kezelni a böngészőben. A WebSocket API lehetővé teszi a WebSocket kapcsolat inicializálását, az üzenetek küldését és fogadását, valamint ugyanúgy, az események kezelését. A Spring keretrendszer további támogatást nyújt a böngésző-kliens és a szerver közötti kommunikáció egyszerűsítéséhez, például a SockJs és a STOMP protokollok segítségével.

A Spring WebSocket kiterjeszthetőséget is biztosít a fejlesztők számára. A keretrendszer lehetővé teszi a saját `WebSocketHandler` implementálását és testreszabását is, az alkalmazás speciális igényei szerint. Ugyanakkor beépített skálázhatósági és teljesítményjavításokat is megvalósít, amely lehetővé teszi az alkalmazások számára, hogy hatékonyan kezeljék a nagy terhelést és a több felhasználós környezetet. Elméletileg egy szerver akár 65 536 socketet is kezelhet IP címenként. Természetesen mindez a közlekedő adatok méretétől is függ [10].

Ezenkívül a Websocket komponensek könnyen összekapcsolhatóak más Spring projektekkel, ami tovább egyszerűsíti a fejlesztést és jobb produktivitást biztosít.

3.2.3. Angular

Az Angular egy elterjedt keretrendszer a modern webalkalmazások fejlesztésében [11].

- **Komponensek**

Az Angular alkalmazások architektúrája komponens-alapú. Az alkalmazások különböző részekre, komponensekre bomlanak, amelyek egymással kommunikálnak. Ezek az Angular alapegységei, és a felhasználói felület különböző részeit reprezentálják. A komponensek lehetnek a fejléc, lábléc, a navigációs sáv vagy a konkrét tartalom megjelenítéséért felelős részek.

- **Nyelv**

Az Angular fejlesztéséhez a használt programozói nyelv a TypeScript. A TypeScript a JavaScriptre épül, kibővíti azt statikus típusozással (a fejlesztő előre meghatározhatja a változók és függvények típusát). A TypeScript végrehajt típusellenőrzéseket is a kód fordítási időszakában, támogatja az objektumorientált programozás alapvető elemeit, mint az osztályok és interfészek, a JavaScript legújabb nyelvi funkcióit (pl. lambda függvények).

- **Szerkezet**

A komponens-alapú szerkezet lehetővé teszi a kódbázis modularitását és újrafelhasználhatóságát. A komponensek hierarchikusan rendezhetők, így lehetőség van a komponensek közötti adat- és eseményáramlás kezelésére.

- **Függőségek**

Az Angular függőségek kezelése az Injektort és a Service-eket (szolgáltatásokat) használja. Az injektor felelős az objektumok létrehozásáért és a függőségeik feloldásáért. Az injektor alapvetően egy hierarchikus konténert képez, amely megoldja a függőségek beágyazását és kezelését. A szolgáltatások olyan osztályok, amelyek specifikus feladatokat látnak el, például adatlekérdezések, hálózati kommunikáció vagy adatfeldolgozás. A szolgáltatások használata révén az alkalmazások könnyen skálázhatóvá és karbantarthatóvá válnak.

- **Modulok**

Az Angular modulok segítségével logikailag összekapcsolhatjuk az alkalmazás különböző részeit. A különböző modulok használatával lehetőség van az alkalmazás egészének gyorsabb betöltésére és a fejlesztés könnyebb skálázására.

- **Direktívák**

Az Angular direktívák lehetővé teszik az alkalmazásokban a DOM manipulációját és az alkalmazás viselkedésének módosításait. Ezek közé tartoznak a beépített direktívák, mint például az ngIf, ngFor, ngClass, amelyek segítenek az adott elemek megjelenítésében és viselkedésének vezérlésében. Továbbá, az Angular lehetővé teszi saját direktívák elkészítését is.

- **Routing és navigáció**

Az Angular keretrendszer további fontos eleme a Routing és a moduláris felépítés. Az Angular Routing lehetővé teszi az alkalmazásban a különböző útvonalak kezelését. Ezáltal az alkalmazásnak több oldalból való felépítése lehet, ahol az egyes útvonalakhoz különböző komponensek és funkcionálisok vannak rendelve.

- **Szerverrel való kapcsolat**

A szerverrel való szinkron és aszinkron kapcsolatok az Angular alkalmazásokban fontos szerepet játszanak. Az Angular HTTP modulja megteremti a szerverrel való kommunikációt HTTP protokoll segítségével. Ez a modul számos beépített funkciót és metódust biztosít az adatlekérdezések és adatküldések kezelésére. Az Angular Observable osztálya az aszinkron programozást támogatja, amellyel megvalósítható az események követése és az aszinkron lekérdezések kezelése.

- **Beépített könyvtárak és eszközök**

Az Angular keretrendszer sok beépített könyvtárral és eszközzel rendelkezik, amelyek megkönnyítik a fejlesztést. Például az Angular Forms könyvtár, amely a felhasználói űrlapok megvalósításában segít. Vagy az Angular Animations, amely a felhasználói felület animációit támogatja. Továbbá az Angular CLI (Command Line Interface) több fejlesztési feladatot automatizál. Ez az eszköz lehetővé teszi az új projekt inicializálását, új komponensek, új szolgáltatások, új modulok létrehozását. Biztosítja a kódgenerálást, a fejlesztői szerver indítását és a fordítási folyamatokat.

- **Single Page Application**

A keretrendszer támogatja az SPA (Single Page Application) fejlesztést, azaz a felhasználói élmény gazdagítását, a gyors reakcióidő elérését. Az SPA applikációknál a teljes alkalmazás egyetlen HTML oldalon jelenik meg, és a navigáció, az adatlekérdezések JavaScript segítségével történnek, anélkül, hogy a teljes oldal újratöltődne.

Az Angular keretrendszer kiterjedt eszközkészlete lehetővé teszi a fejlesztők számára a hatékony és rugalmas webalkalmazások létrehozását, támogatja a legújabb webes technológiákat és fejlesztői elveket. A keretrendszer népszerűségét és a fejlesztői közösség támogatását tekintve az Angular egy ideális választás lehet a webalkalmazások fejlesztésében.

4. fejezet

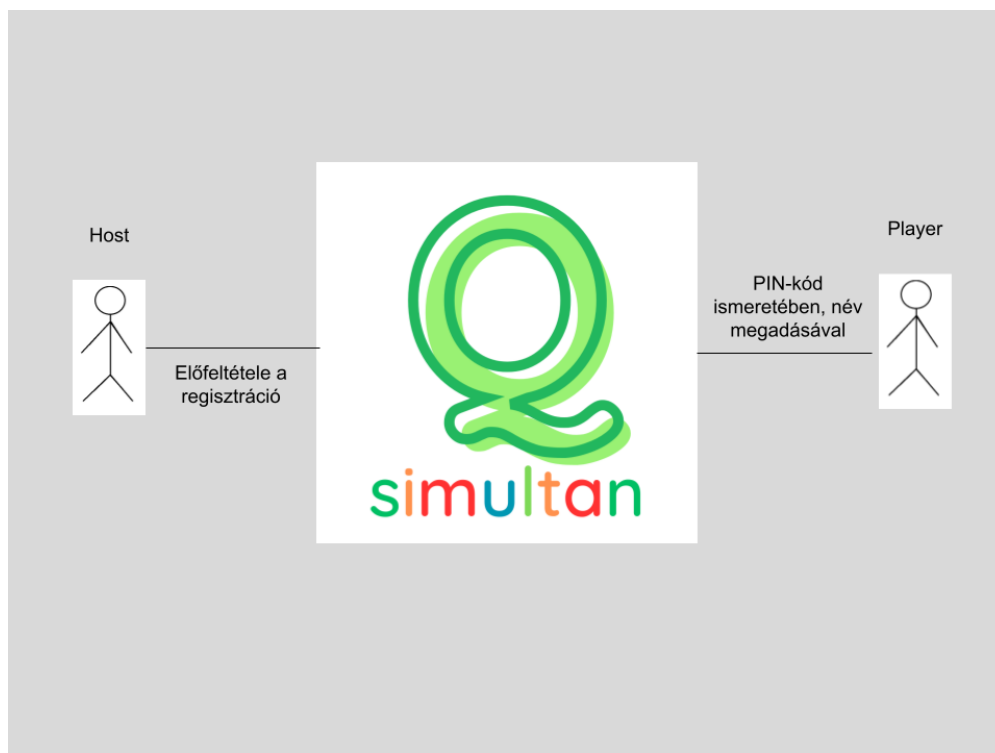
A rendszer specifikációja

4.1. Felhasználói követelmények

Ebben az alfejezetben találhatóak azok a fontosabb használati esetek és azok leírásai, amelyekkel a felhasználó az applikáció használata közben találkozhat. Első körben bemutatom a szerepköröket, majd a már bejelentkezett felhasználó használati eseteit sorakoztatom fel, azután pedig a játékos használati eseteit.

Az applikáció elindításakor kétféle aktorra számítunk: házigazdára (Host) - ennek előfeltétele a regisztráció - és játékosra (Player).

4.1.1. Szerepkörök



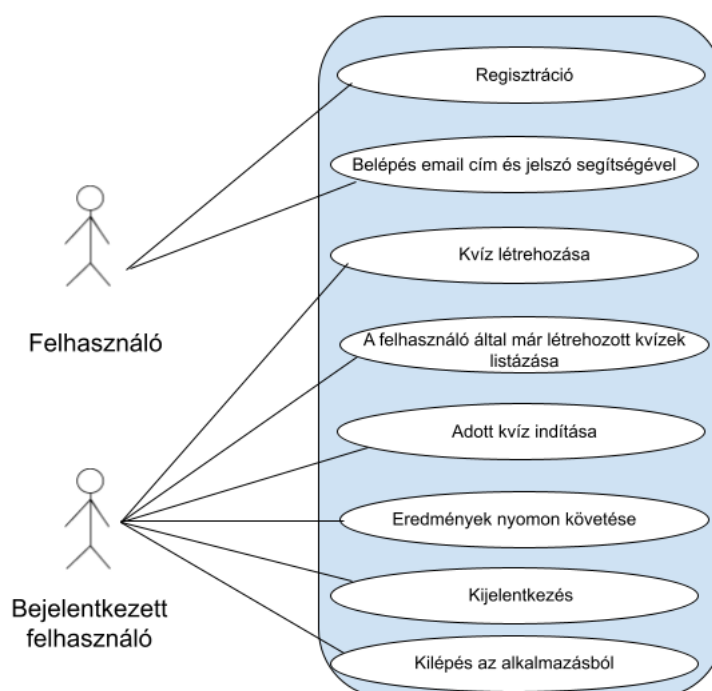
4.1. ábra. Szerepkörök az applikációban

A SimultanQ alkalmazásnak alapvetően két szerepköre van. Az egyik a Host, azaz házigazda, aki létrehozhat illetve elindíthatja a kvizeket - ennek előfeltétele a regisztráció, és a másik a Player, azaz játékos, aki név megadásával és PIN-kód alapján csatlakozik a játékhoz - neki nincs szüksége regisztrációra.

4.2. Fontosabb használati esetek

A következőkben bemutatom a kétféle szerepkörhöz tartozó fontosabb használati eseteket.

4.2.1. Bejelentkezett felhasználó használati esetei



4.2. ábra. A Host használati esetei

- kvíz létrehozása

- előfeltételek:

- * PR1: a felhasználó be van jelentkezve

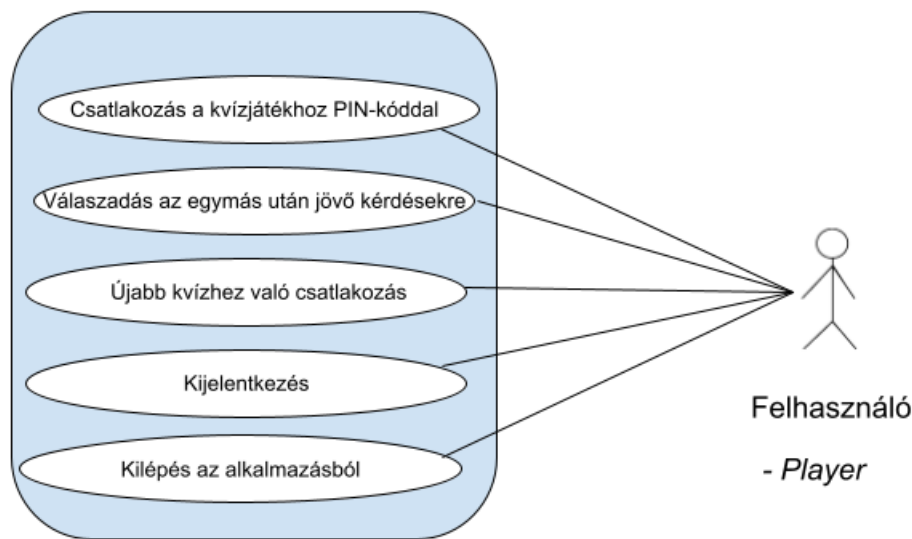
- folyamat:

- * F1: a felhasználó bejelentkezik

- * F2: a felhasználó megnyomja a kvíz létrehozása gombot

- * F3: a felhasználó a megjelenő űrlap segítségével megadja a kvíz címét, hozzáadja a tesztöleget számú kérdéseket, válaszokat, megjelöli a helyes választ

- *alternatív folyamat:*
 - * AF1: ha a felhasználó már be van jelentkezve, akkor a folyamat az F2-től kezdődik
- *utófeltételek:*
 - * PO1: a saját kvizek listájában megjelenik az újonnan létrehozott kvíz, egyedi PIN-kóddal ellátva
- **kvizek listázása**
 - *előfeltételek:*
 - * PR1: a felhasználó be van jelentkezve
 - *folyamat:*
 - * F1: a felhasználó bejelentkezik
 - * F2: a felhasználó megnyomja a kvizek listázása gombot
 - *alternatív folyamat:*
 - * AF1: ha a felhasználó már be van jelentkezve, akkor a folyamat az F2-től kezdődik
 - *utófeltételek:*
 - * PO1: a bejelentkezett felhasználó által korábban létrehozott kvizek listája megjelenik
- **kvíz elindítása**
 - *előfeltételek:*
 - * PR1: a felhasználó be van jelentkezve
 - *folyamat:*
 - * F1: a felhasználó bejelentkezik
 - * F2: a felhasználó megnyomja a kvizek listázása gombot
 - * F3: a felhasználó a megjelenő listából kiválasztja az elindítandó kvízt
 - * F4: a felhasználó megosztja a kiválasztott kvíz PIN-kódját a potenciális játékosokkal
 - * F5: a felhasználó elindítja a játékot
 - *alternatív folyamat:*
 - * AF1: ha a felhasználó már be van jelentkezve, akkor a folyamat az F2-től kezdődik
 - *utófeltételek:*
 - * PO1: a bejelentkezett felhasználó, a játékot indító házigazda dashboardján nyomon követhető a játékosok pontszáma, rangsora



4.3. ábra. A Player használati esetei

4.2.2. A játékosok használati esetei

- csatlakozás a kvízzjátékhoz

- *előfeltételek:*

- * PR1: a játékos birtokában van a kvíz PIN-kódjának

- *folyamat:*

- * F1: a játékos a PLAY gombot megnyomva hozzákezd a játékhoz való csatlakozásnak
- * F2: a játékos megadja a nevét és a kvíz PIN-kódját
- * F3: a játékos csatlakozik a játékhoz és válaszol az egymás után következő kérdésekre

- *utófeltételek:*

- * PO1: a játékos pontszáma és rangsora nyomon követhető a játékot indító házigazda (Host) oldalán

- *hibák:*

- * E1: "Invalid Quiz PIN!" toast megjelenítése, ha a játékos érvénytelen PIN-kódot adott meg

4.3. Rendszerkövetelmények

4.3.1. Funkcionális követelmények

Ebben az alfejezetben bemutatásra kerül az alkalmazás néhány funkcionalitása.

- **Szerepkörök elkülönülése:** az applikáció megnyitásakor a felhasználót a kezdő oldal fogadja, ahol a szerepkörök egyből elkülönülnek. A "Host a quiz" gombra kattintva a felhasználó intenciója a kvízzjáték indítása, a házigazda szerepének felvétele, amely során kvízek létrehozására vagy előzőleg létrehozott kvíz elindítására kerülhet sor. A "Play" gombra kattintva viszont a felhasználó a Player, azaz a játékos szerepét veszi fel, azzal az intencióval, hogy részt vegyen egy valaki más által indítandó játékban.
- **Bejelentkezés:** a felhasználó a "Login" gombra kattintva megadja az emailcímét és a jelszavát, hogy létrehozasson kvízeket, vagy előzőleg létrehozottakat listázzon vagy kvízt indítson.
- **Regisztráció:** amennyiben a felhasználó először szeretné felvenni a házigazda szerepet, akkor szükséges, hogy regisztráljon, megadva a felhasználói nevet, emailcímet és jelszót. Sikeres regisztráció esetén erről toast értesítést kap, illetve az alkalmazás átnavigál a következő oldalra, ahol a bejelentkezett felhasználó két gomb közül választhat: "Create a quiz" és "List my quizzes". Létrehozhat egy új kvízt vagy listázhatja a már meglévőket.
- **Új kvíz létrehozása:** a bejelentkezett felhasználó új kvízt hozhat létre a "Create a quiz" gombra kattintva. Meg kell adnia a kvíz címét, a kérdések számát, illetve a kérdés szövegének megadása után a válaszokat is meg kell adnia, azt is megjelölve, hogy melyik a helyes válasz.
- **Kvízek listázása:** a "List my quizzes" gombra kattintva kilistázódik a felhasználó által már létrehozott, meglévő kvízek listája.
- **Kvízek indítása:** a kvízek listájában az "Activate quiz" gombra kattintva a kiválasztott kvíz aktiválódik, majd a "Start quiz" gombbal el lehet indítani a kvízt és a játékosok a PIN-kód ismeretében csatlakozhatnak a játékhoz. Megjelenik a kvíz PIN-kódja mellett a kvízhez csatlakozott játékosok névsora is pontszámaikkal együtt, ami valós időben frissül.
- **Kvízzjátékhoz való csatlakozás:** az applikáció megnyitásakor a "Play" gombra kattintva két mező kitöltésével a felhasználó azonnal, regisztráció nélkül csatlakozhat egy aktív kvízhez. Meg kell adnia egy tetszőlegesen választott nevet, illetve a kvíz PIN-kódját, melyet a kvíz házigazdája megadott számára. Nem létező PIN-kód beütése esetén erről toast értesítést kap.
- **A játék folyamata:** ha a kvíz létezik és aktív, akkor máris megjelenik az első kérdés, amelyre a felhasználónak válaszolnia kell ahhoz, hogy elérhetővé váljék a következő kérdés. Helyes válasz esetén a házigazda oldalán nyomon követhető a játékos pontszámának növekedése. Amikor az idő lejár, a játék lezárul.

4.3.2. Nem funkcionális követelmények

- **Termék követelmények**

- *Felhasználhatóság*: a termék a legnépszerűbb böngészőkben futtatható, mint a Google Chrome, Mozilla FireFox, Microsoft Edge
- *Hatékony*: a termék megfelelő internetkapcsolat mellett kiszolgálja a felhasználót a másodperc tört része alatt
- *Megbízhatóság*:
 - * fejlesztés alatt sok hibalehetőség tesztelve volt
 - * szabványos használat mellett nem fordul elő hibajelenség, nem jelenik meg hibaüzenet
- *Tárhely takarékos*: az applikáció kevesebb mint 50 MB tárhelyt igényel
- *Biztonság*:
 - * az alkalmazás beépített biztonságos rendszerű bejelentkezést használ
- *Hordozhatóság*: az applikáció előzetes telepítést igényel

- **Folyamat követelmények**

- *Kódolási standard*
 - * CamelCase standard követése a változó- és függvénynevek esetén
 - * tabulátorral való tördelés
- *Verziókövetés*
 - * Git [12] verziókövető rendszer használata
 - * minden fontosabb funkció leimplementálása után commit parancs használata

- **Külső követelmények**

- *Összeférhetőség*: az alkalmazás semmilyen más befolyással nem bír más alkalmazásokra tekintve, teljesen elkülöníthető a már meglévő alkalmazásoktól.
- *Etikai*: a termék semmilyen adatot nem szolgáltat ki külső forrásoknak

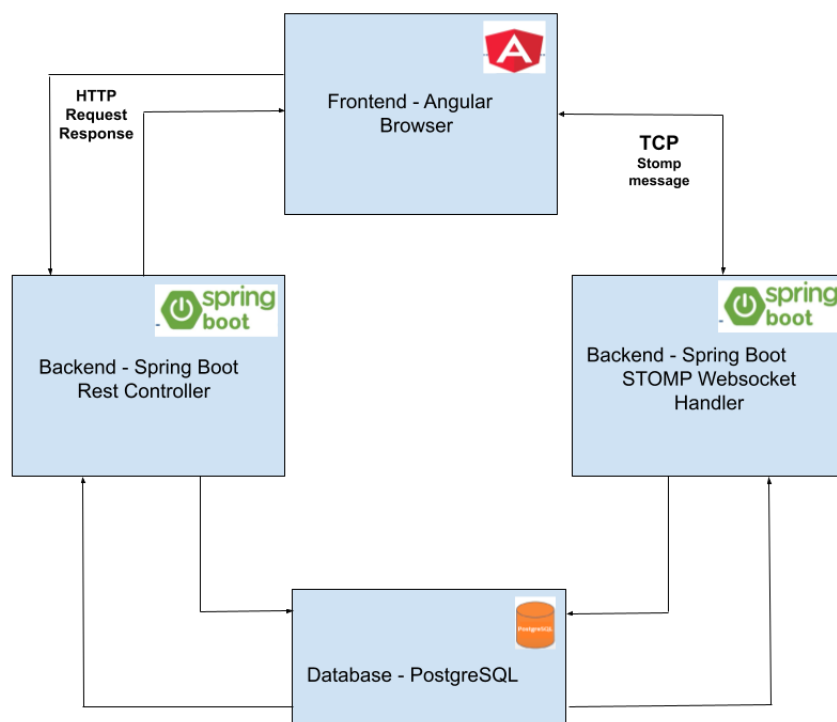
5. fejezet

Tervezés

Ebben a részben vázolom a teljes projekt rendszertervét és részletezem a kliens oldali és a szerver oldali applikáció architektúráját.

5.1. A rendszer architektúrája

A rendszer három fő komponensből tevődik össze: kliensoldal (frontend), szerveroldal (backend) és adatbázis.



5.1. ábra. Blokkdiagram a rendszer architektúrájáról

- *adatbázis*: a rendszer legalsó rétege, amely a tartós adattárolást végzi és közvetlen a szerver oldallal kommunikál.

- *szerveroldal*: egyik fő feladata a kientől érkezett REST API kérések kiszolgálása, adatok lekérése az adatbázisból és azok rögzítése az adatbázisban. A másik fő feladata a kliens jelzésére megnyitni a websocket kapcsolatokat, amely valós idejű, kétoldali kommunikációt tesz lehetővé.
- *kliensoldal*:
 - **webes felület**: az alkalmazás webes interfésze a felhasználó számára, amely a modulok, komponensek és szolgáltatások összességére épül.

5.2. Adatbázis

Az adatok tárolásának megtervezésekor a relációs adatbázisra esett a választásom. A relációs adatbázisok táblákban tárolják az adatokat, amelyek struktúrált formában vannak. Ez lehetővé teszi a logikai kapcsolatok kifejezését az adatok között, valamint a konzisztencia és az adatintegritás fenntartását. Az adatok oszlopok és sorok formájában vannak elrendezve, amelyek egyszerű és hatékony hozzáférést biztosítanak kihasználva az elsődleges és az idegen, vagy külső kulcs lehetőségét. A relációs adatbázisok SQL nyelve kézenfekvő az adatok beszúrásához, módosításához, törléséhez. Nem utolsósorban a relációs adatbázisok nagyon jól skálázhatóak, ami lehetővé teszi az adatbázis méretének és kapacitásának növelését az adatok növelésekor.

5.3. Szerveroldali architektúra

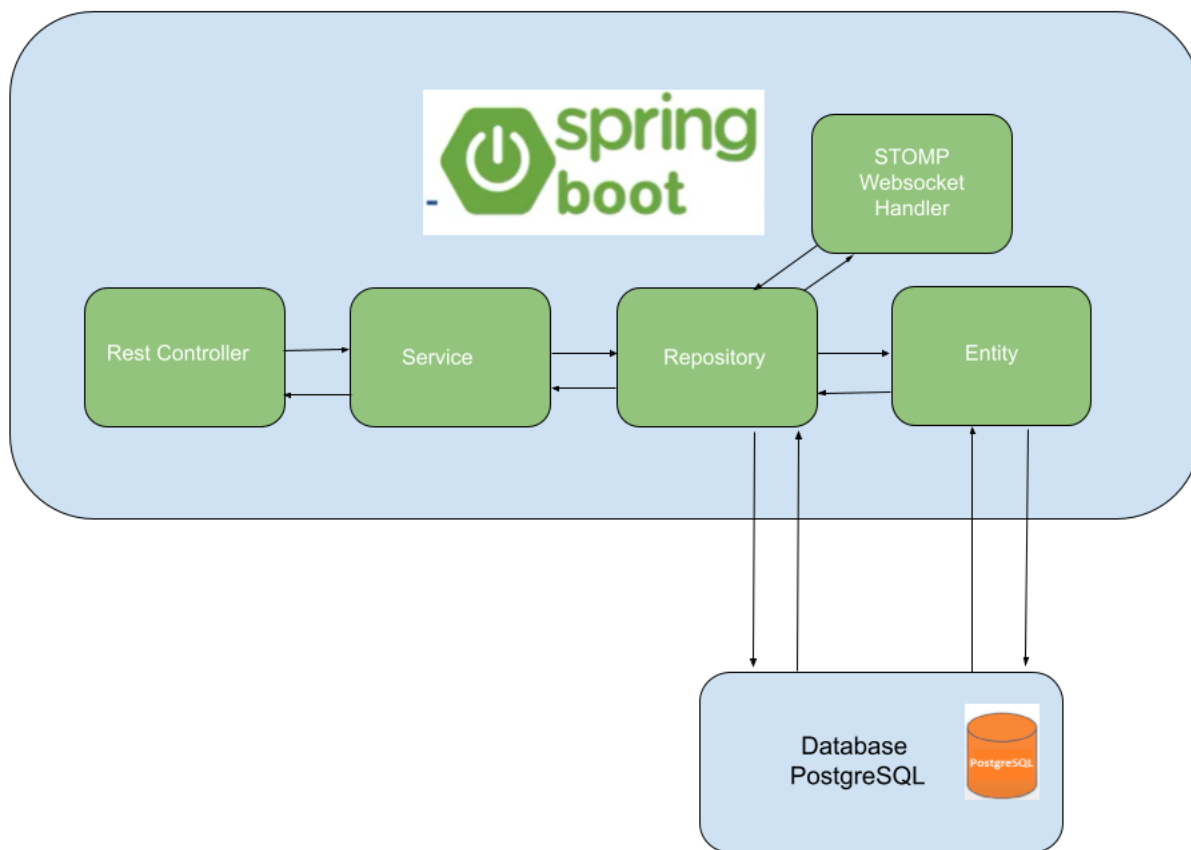
Az alkalmazás fő része itt helyezkedik el. A szerveroldali architektúra a SimultanQ esetében alapjában véve két fő pilléren nyugszik. Az egyik a REST API kérések kezelése, a másik pedig a WebSocket kapcsolatok létrehozása és működésük biztosítása.

5.3.1. REST API kérések

A feldolgozás, az adatkezelés és logika nagy része a szerveren történik, a szerver válaszol a kliensoldali kérésekre, szolgáltatásokat nyújt a felhasználók számára.

A REST (Representational State Transfer) egy architektúrális stílus, amelyet webes szolgáltatások fejlesztésére használnak. A REST alapelvei közé tartozik az erőforrásokra történő hivatkozások használata és a kliens-szerver szétválasztása. A REST API (Application Programming Interface) egy olyan interfész, amely lehetővé teszi a különböző alkalmazásoknak és a szoftverek komponenseinek, hogy kommunikáljanak HTTP protokollon keresztül. A REST API-t használva az alkalmazások képesek erőforrásokat (pl. az adatbázisban található információkat) manipulálni és műveleteket elvégezni.

A szerveroldali architektúra rétegzett struktúrában jelenik meg, a különböző rétegek felelősek az egyes feladatokért. Az Entity, vagy model réteg tartalmazza az adatbázisobjektumokat, amelyeket az alkalmazás az adatbázisban tárol. Az objektumokban találhatóak az attribútumok és azok viselkedése. Az Entity objektumok a Repository rétegben és a Service rétegben is használatosak. A Repository réteg felelős az adatelérésért és az adatbáziskezelésért. Ez a réteg tartalmazza a kódokat és metódusokat az adatbázisobjektumok létrehozására, lekérdezésére, frissítésére és törlésére. Általában interfészek



5.2. ábra. Szerveroldali architektúra Java Spring Boot-ban

formájában vannak jelen és implementálhatók különböző adatbázis-kezelő technológiák, például JPA, Hibernate vagy JDBC segítségével. A Service réteg az a központi hely, ahol az alkalmazás üzleti logikája helyezkedik el. Ez a réteg tartalmazza azokat a szolgáltatásokat és metódusokat, amelyek megvalósítják a különböző üzleti folyamatokat és funkciókat. A Service réteg általában interakcióba lép a Repository réteggel az adatelérés érdekében. A Controller réteg felelős a felhasználó részéről érkezett kérések fogadásáért és kezeléséért. Fogadja a kéréseket és továbbítja őket a megfelelő Service rétegbe.

5.3.2. WebSocket

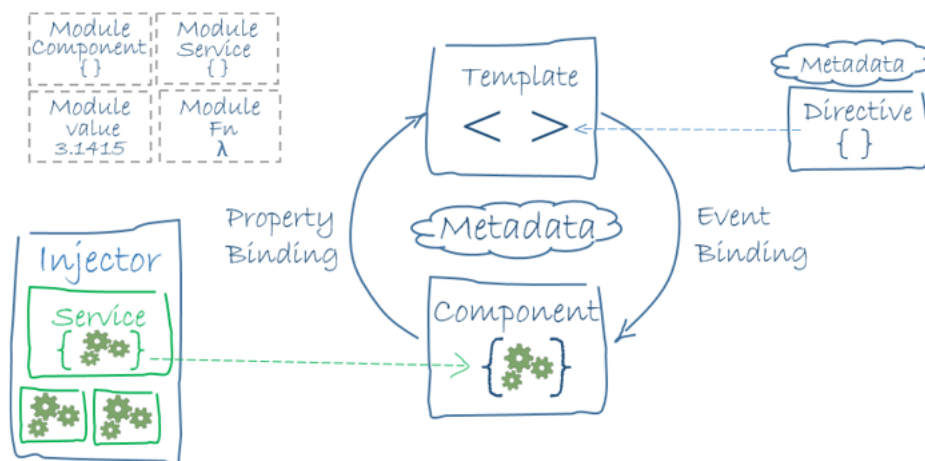
WebSocket egy kommunikációs protokoll, amelyet a webalkalmazások fejlesztésére használnak valós idejű, kétirányú kommunikáció megvalósítására a szerver és a kliens között. A hagyományos HTTP protokollhoz képest a WebSocket folyamatos kapcsolatot tart fenn a szerver és a kliens között, ami lehetővé teszi az adatok valós idejű továbbítását anélkül, hogy az alkalmazásnak folyamatosan kérnie kellene azokat.

5.4. Kliensoldal

5.4.1. Modulok, komponensek és szolgáltatások

A **Modulok, komponensek és szolgáltatások** egy szoftvertervezési architektúrális minta, amely az Angular keretrendszer egyik sajátossága.

A modulok olyan konténerek, amelyekben a komponensek, szolgáltatások, direktívák és más fájlok logikailag össze vannak csoportosítva. Ezek az összetartozó elemeket együtt kezelik, és teszik elérhetővé egymás számára. A modulok segítségével az alkalmazást logikai egységekre, funkcionális részekre bonthatjuk. Ez lehetővé teszi a fejlesztők számára a kódbázis könnyebb kezelését, tesztelhetőségét és újrafelhasználhatóságát. A modulok révén az alkalmazást logikai szempontból több részre osztva könnyebben megérthetővé válik a kód struktúrája, és könnyebben bővíthetővé válnak az új funkciók. Egy alkalmazásnak mindig legalább egy gyökermodulja van, amely lehetővé teszi az inicializálást, és általában több további modulja van. A komponensek az Angular alkalmazások



5.3. ábra. Kliensoldali architektúra Angular-ben [13]

építőkövei. Minden komponenshez tartozik egy különálló HTML-sablon, ami meghatározza a felhasználói felület szerkezetét és stílusát. Ezen kívül a komponensek logikai részt is tartalmaznak, amely a szükséges adatokat és műveleteket kezeli. A komponensek a modulokon belül helyezkednek el, és könnyen hasznosíthatók, egymással kombinálhatók a nagyobb funkcionalitás eléréséhez. A nézet-sablonok olyan deklaratív sablonokat kínálnak, amelyek segítségével definiálhatjuk az alkalmazásunk felhasználói felületét. Ez a sablonnyelv tartalmazza az adatkötéseket, a ciklusokat és az eseménykezelőket. Az Angular sablonok HTML-alapúak, de kiegészülnek különleges direktívákkal, amelyek lehetővé teszik a dinamikus adatok beillesztését és az eseményekre történő reagálást.

A szolgáltatások olyan osztályok vagy objektumok, amelyeket a komponensek használnak, hogy közös logikát és műveleteket valósítsanak meg. A szolgáltatások különböző funkciókat látnak el: hálózati kommunikációt, adatkezelést vagy harmadik féltől származó API-k integrációját. A szolgáltatások injektálhatóak, tehát az alkalmazás más részei számára hozzáférhetőek és könnyen cserélhetőek.

A modulok, komponensek és szolgáltatások osztályok, amelyek dekorátorokat használnak. Ezek a dekorátorok jelzik a típusukat és metaadatot biztosítanak, amelyek elmondják az Angularnak, hogyan kell használni őket.

A modulok segítségével csoportosíthatjuk az alkalmazásunk különböző részeit, a komponensek kialakítják a felhasználói felületet, míg a szolgáltatások biztosítják a közös logikát és a háttér műveleteit. Ez a három pillér lehetővé teszi az Angular alkalmazások skálázhatóságát, újrahasznosíthatóságát és karbantarthatóságát. A különböző rétegek elkülönülése lehetővé teszi a fejlesztők számára a párhuzamos munkát és a felelőségek jól meghatározott elosztását, ami hozzájárul a hatékony és strukturált fejlesztési folyamat-hoz.

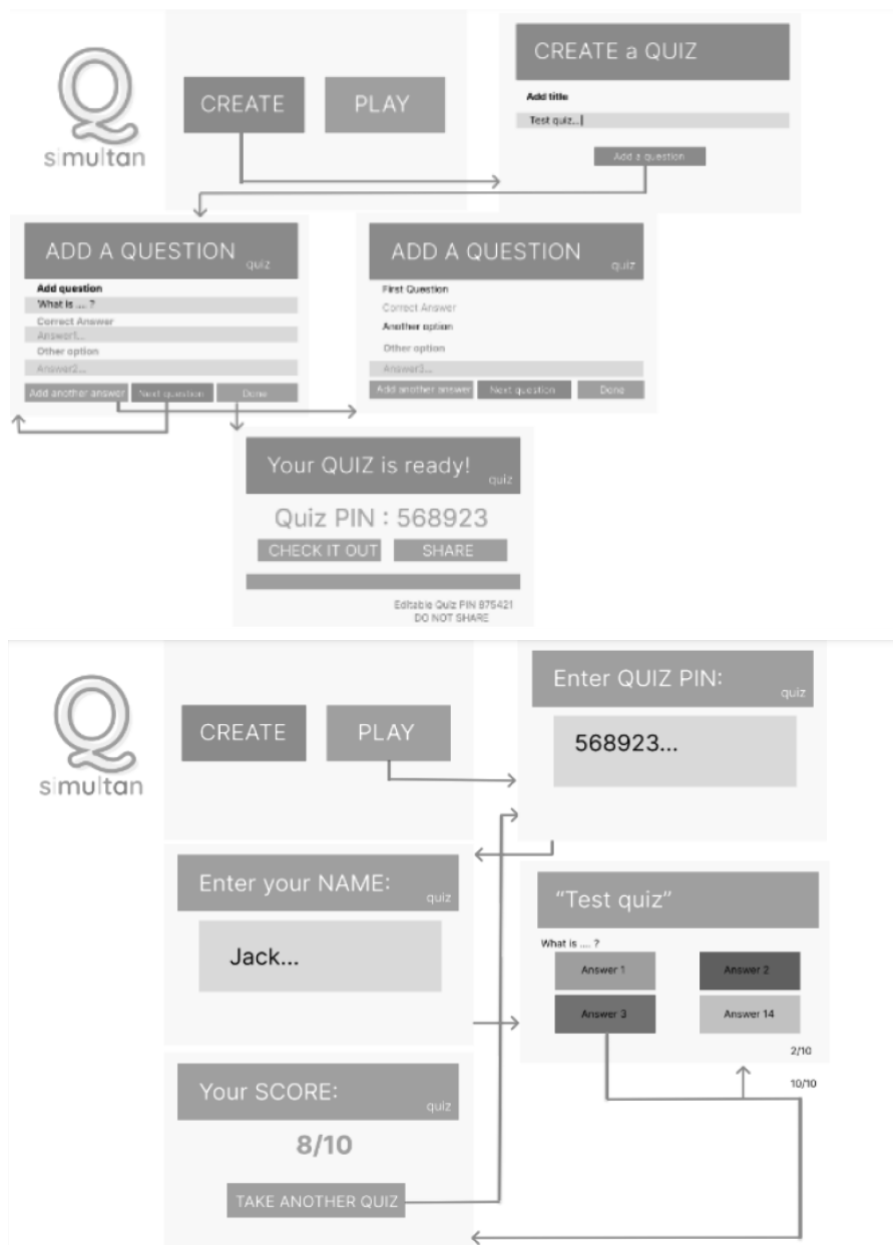
5.4.2. Felhasználói felület tervezése

A felhasználói felület tervezésekor először elkészítettem a drótváztervet. A drótvázterv (wireframe) egy legyszerűsített szerkezete a megjelenítendő elemeknek, bemutatja az alkalmazás alapvető elrendezéseit és funkcióit, melyek segítenek megérteni, hogy fog kinézni és működni a felhasználói felület. Az egész egy folyamat volt, és a fejlesztés során alkalomadtán módosításokra, igazításokra is sor került.

A drótvázterv tartalmazza az információk fő elemeit rendszerbe szervezve, mint például a nyomógombok, szöveges bementek, a képek helyét és méretét. A drótvázterv segítségével alapvetően egy absztrakt térképet tudunk vizuális formában kialakítani, ezáltal könnyítve az elképzelés megértését és áttekinthetőségét (5.4 ábra).

Mindez a navigáció megtervezésében is elengedhetetlen volt. A különböző oldalak közötti váltás előre projektálása külön odafigyelést igényelt.

A drótváztervre már az elején rákerült a SimultanQ egyedi logója is, amelyet az alkalmazáshoz terveztem, és amely az alkalmazás közösségi és játékelmény voltát hivatott tükrözni.



5.4. ábra. A felhasználói felület drótváza

6. fejezet

Kivitelezés

Ebben a fejezetben bemutatom az applikáció elkészítésének kivitelezését, az applikáció működését különböző felhasználói szerepkörök szempontjából. Összességében két felhasználó típust különböztetünk meg, regisztrált felhasználót, más néven Host-ot, aki létrehozza és majd tetszőleges időben elindítja a kvizeket, és általános felhasználót, játékost vagy Player-t, aki regisztráció nélkül, a PIN-kód ismeretében csatlakozhat a játékhoz.

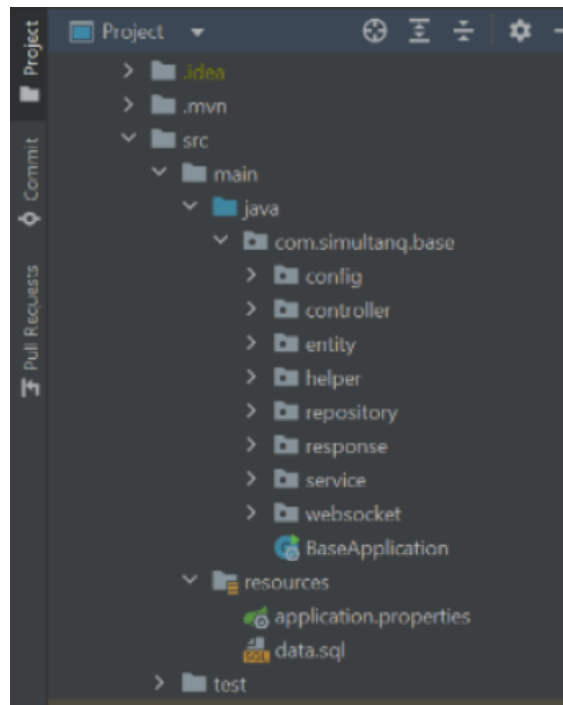
A már regisztrált felhasználó bejelentkezés után kvizeket hozhat létre, megadva a kérdéseket, válaszokat, megjelölve a helyes választ. A játékos a házigazda által megadott PIN-kód alapján csatlakozhat a játékhoz, regisztráció nélkül, szabadon választott saját névvel. Amikor a játék elindul, a játékosoknak a felsorolt válaszok közül ki kell választaniuk a szerintük helyes választ. A kérdésre megfelelő válasz esetén pontot kapnak. A játékosok pontszámainak változása és a rangsor valós időben követhető a házigazda oldalán, teljessé téve a közösségi élményt. A játékidő leteltével kialakul a játékosok végleges rangsora. A házigazda újabb kvízt indíthat, közzé téve annak PIN-kódját.

6.1. Szerveroldali megvalósítás

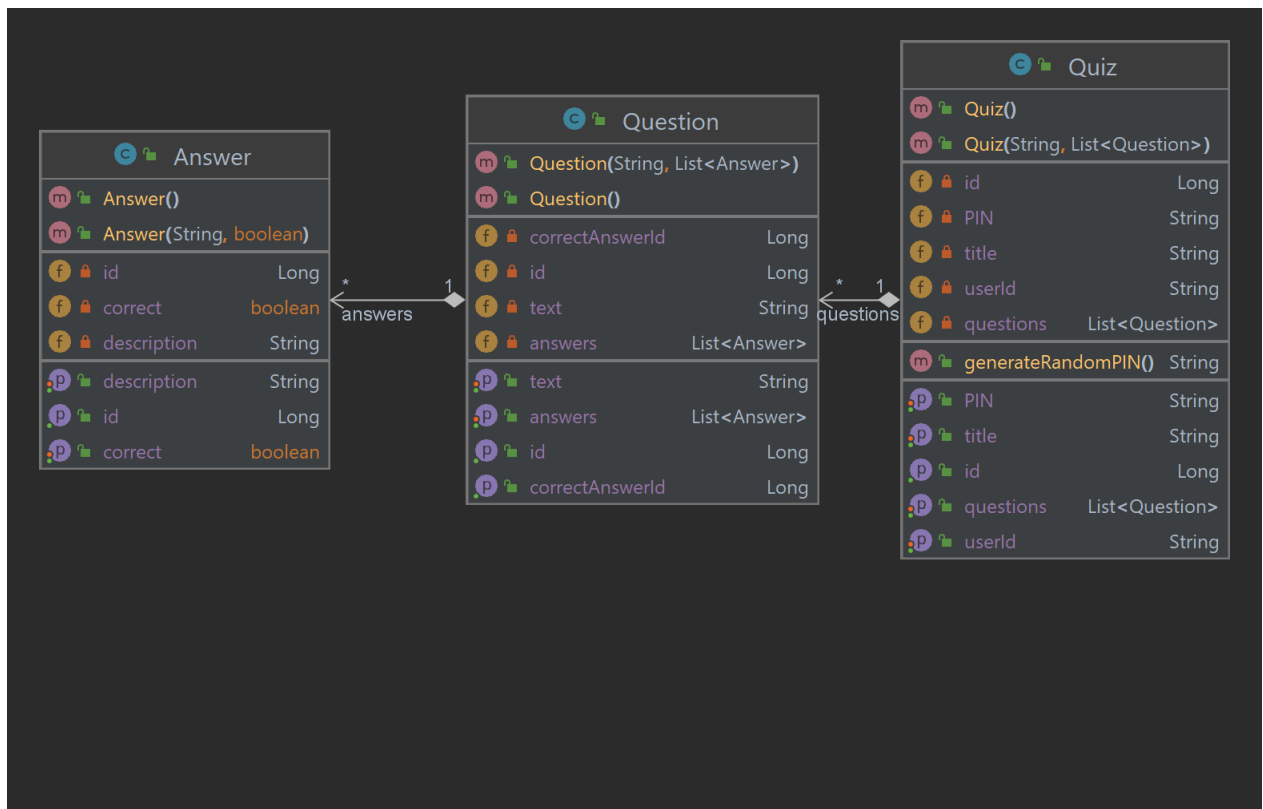
A SimultanQ kvízzjáték szerveroldali részét a Spring Boot keretrendszer segítségével valósítottam meg. A regisztrációhoz, bejelentkezéshez, kvizek létrehozásához REST API-t használtam, míg a tényleges játék lebonyolításához WebSocketet. A Spring Boot keretrendszert használva a szerveroldali implementációt rétegekre bontottam: Controller, Service, Repository és Entity rétegekre.

6.1.1. Entity réteg

Az Entity réteg tartalmazza az entitásokat, amelyek az alkalmazás adatainak reprezentációját képviseli. Az Entity rétegben az entitásokat osztályokként definiáltam. Az entitások tábláknak felelnek meg az adatbázisban, és tartalmazzák ezen felül az adatok struktúráját és viselkedését. A User entitás tartalmazza a felhasználó nevét, e-mail címét és jelszavát. A User entitás példányai reprezentálják a regisztrált felhasználókat az alkalmazásban, és ezek az objektumok kerülnek mentésre az adatbázisba. Hasonlóképpen a Quiz, Question és Answer entitások, amelyek a kvizek címeit, PIN-kódjait, kérdéseit, válaszait tartalmazzák. A Quiz és Question, valamint a Question és Answer osztályok között one-to-many kapcsolat van.



6.1. ábra. Mappaszerkezet Spring Boot-ban



6.2. ábra. Quiz, Question és Answer osztályok és kapcsolataik

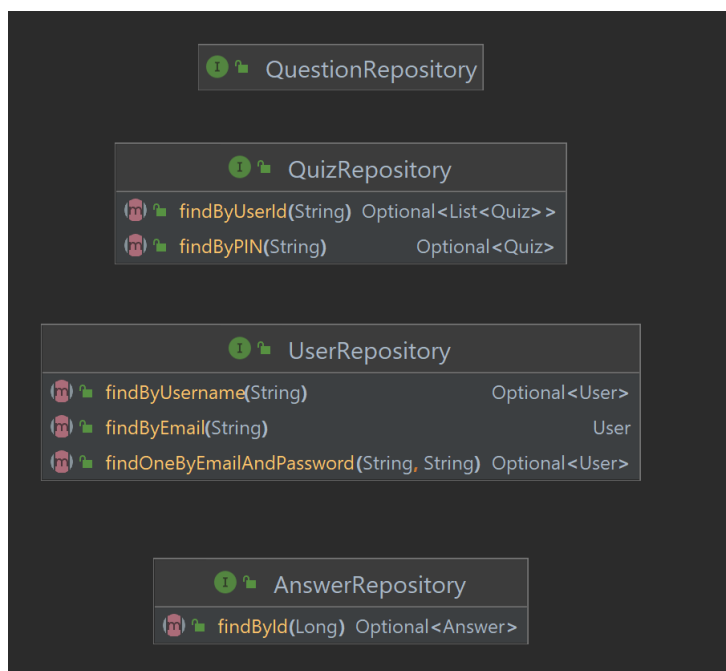
Az **Entity Framework** egy keretrendszer, amely lehetővé teszi az entitások és az adatbázis közötti kölcsönhatást, és automatikusan generálja az adatbázis lekérdezéseket

és műveleteket az entitások alapján. Ennek a keretrendszernek a segítségével az entitások könnyen mappelhetőek az adatbázistáblákhoz, így létrejön az adatbázis-összeköttetés és az adatbázisműveletek kezelése az entitások és az adatbázis között. Az entitásokat **@Entity** annotációval jelöltem meg, így az Entity Framework számára entitásként felismerhető és kezelhető. A keretrendszer figyeli az entitások változásait és automatikus lekérdezéseket és műveleteket generál a változások végrehajtásához az adatbázisban.

Az Entity Framework támogatja a különböző adatbázis-motorokat, így a PostgreSQL-t is, amit az alkalmazáshoz használtam. Ezzel az adatbázishoz való kapcsolódás egyszerűvé és hatékonyá vált.

6.1.2. Repository réteg

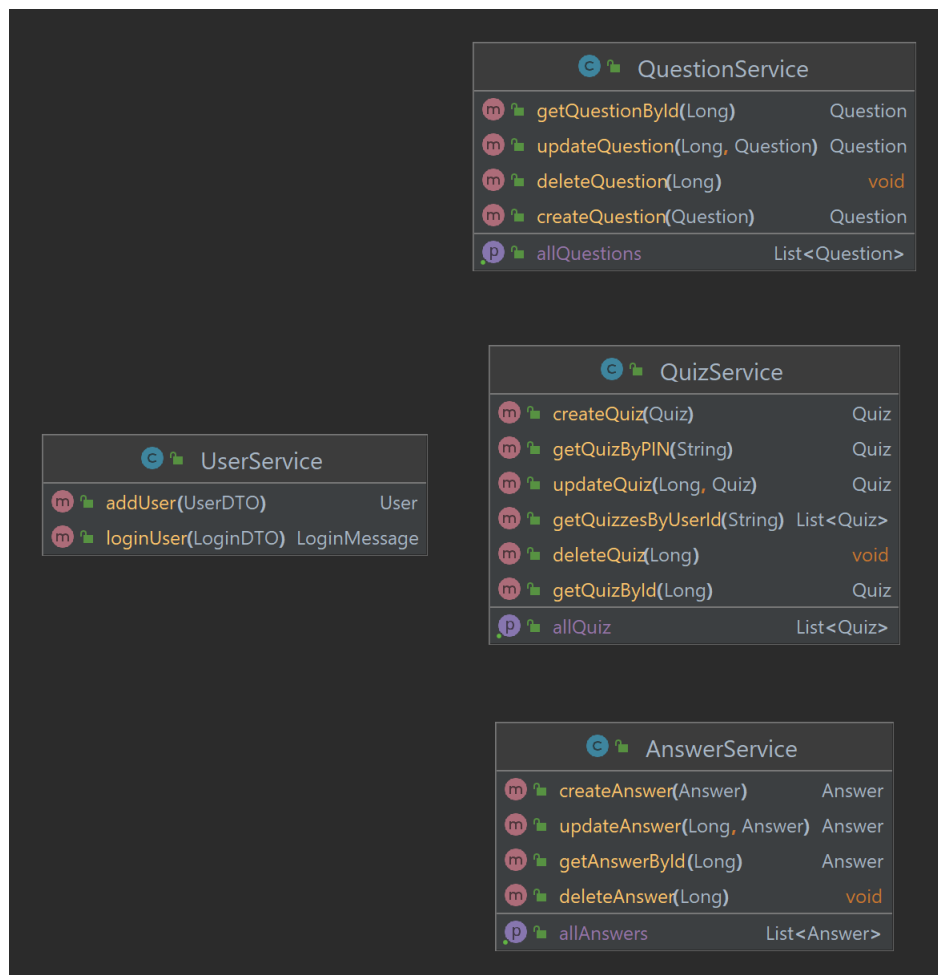
A Repository réteg felelős az adatbázisműveletek végrehajtásáért és az entitásokhoz kapcsolódó adatok kezeléséért. A SimultanQ applikációban ennek érdekében létrehoztam a UserRepository, QuizRepository, QuestionRepository és AnswerRepository interfészeket. Ezek az interfészek kiterjesztik a keretrendszer által biztosított JpaRepository interfészt. Ez egy interfész a Spring Data JPA projektben, amely már tartalmazza az adatbázisműveletek alapjait az entitásokkal való munkához: definiálja, automatikusan generálja az adatbázisműveleteket az entitások alapján, mint például az adatok lekérdezése, mentése, frissítése és törlése. Emellett a QuizRepository-t kiegészítettem úgy, hogy az alaplekérdezéseken túl tartalmazza a findByPIN() metódust is, amely a kvízek közül megkeresi az adott PIN-kóddal rendelkező kvízt. A leggyakrabban használt metódusok azok, amelyek azonosító, azaz Id alapján adnak vissza adatokat.



6.3. ábra. Repository réteg

6.1.3. Service réteg

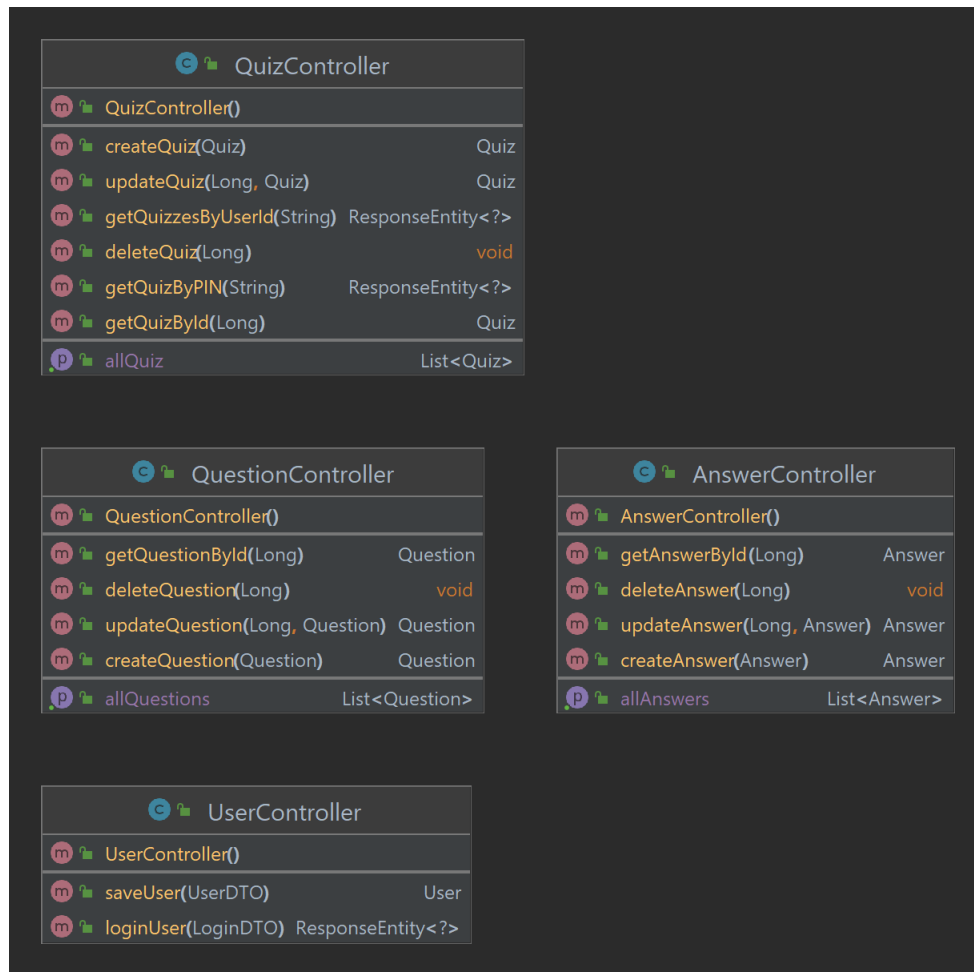
A Service réteg az üzleti logikáért felelős, összekapcsolja az Entity és a Repository rétegeket. Létrehoztam a UserService, QuizService, QuestionService és AnswerService osztályokat, amelyek validációkat, adatmanipulációkat és egyéb üzleti szabályokat tartalmaznak. A Service osztályok használják a Repository osztályokat az adatok kezelése érdekében. Így QuizService felelős a kvízek létrehozásáért a createQuiz() metódus használatával, ugyanakkor a getQuizByPIN() visszaadja az adott PIN-kóddal rendelkező kvízt.



6.4. ábra. Service réteg

6.1.4. Controller réteg

A Controller rétegben létrehoztam az egyedi végpontokat. Ezek az API végpontok kapcsolatot teremtenek a felhasználói kérések és a Service réteg között. A Controller osztályok tartalmazzák a végpontok elérési útvonalát, a kérések típusát és az ezekre adott válaszokat. Létrehoztam a UserController, QuizController, QuestionController és AnswerController osztályokat, ezeket **@RestController** annotációval jelöltem meg. A Controller osztályok használják a Service osztályokat az üzleti logika hívásához és az eredmények visszaküldéséhez a kliensek felé.

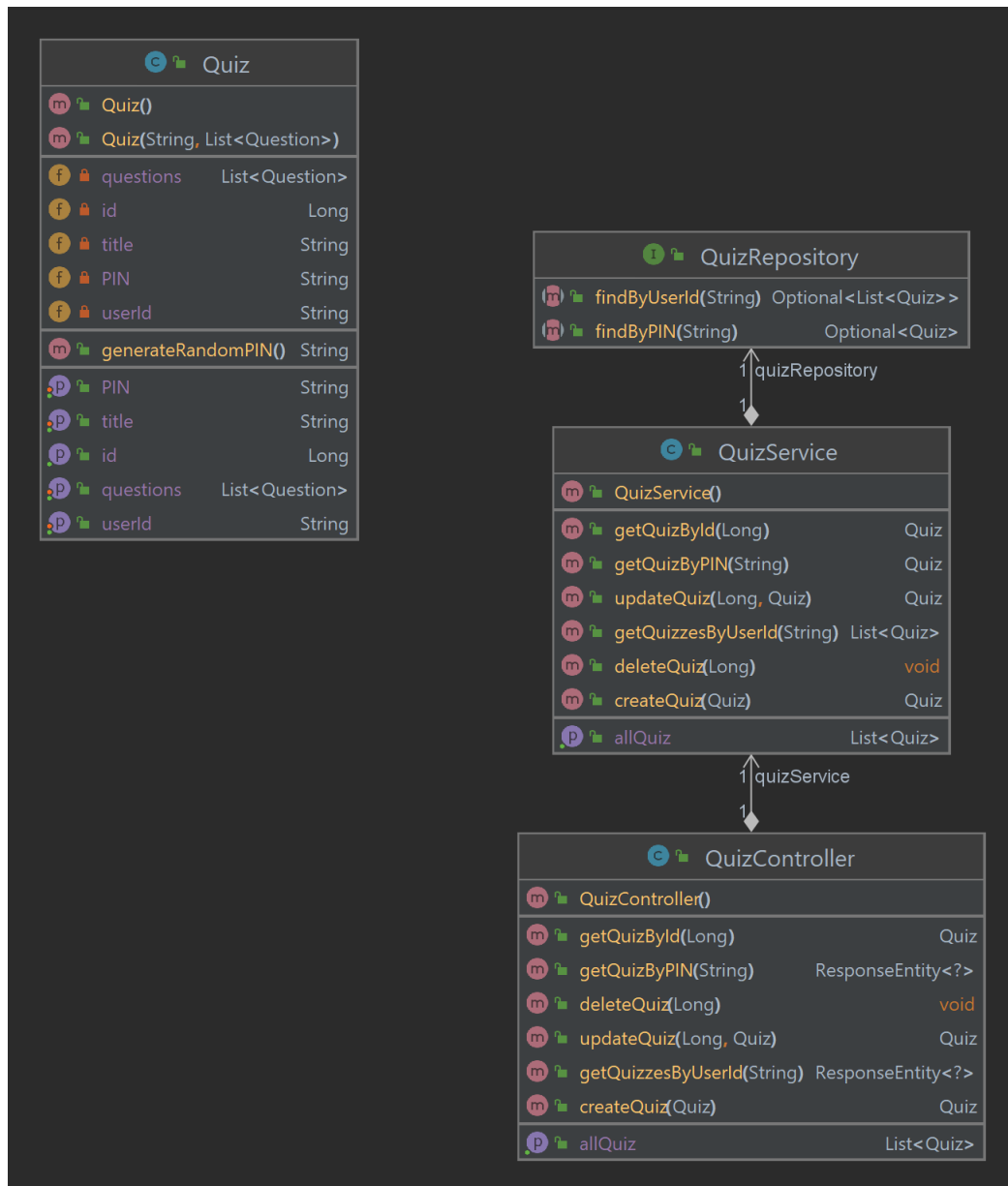


6.5. ábra. Controller réteg

6.1.5. Websocket implementálása

A WebSocket protokoll lehetővé teszi a valós idejű, kétirányú kommunikációt a kliensek és a szerver között. A kliensek képesek küldeni és fogadni üzeneteket a szerverrel, és a szerver ezeket az üzeneteket feldolgozza és válaszokat küld a klienseknek. Ez lehetővé teszi a valós idejű frissítéseket a játék állapotáról és az eredményekről a résztvevők számára.

A WebSocketQuizConfiguration osztály a WebSocket konfigurációját végzi. A **@Configuration** annotáció jelzi, hogy ez a osztály a Spring konfigurációjában részt vesz. Az **@EnableWebSocketMessageBroker** annotáció engedélyezi a WebSocket üzenetek kezelését a Spring Message Broker használatával. Az osztály implementálja a WebSocketMessageBrokerConfigurer interfészt, amely lehetővé teszi a konfiguráció beállításait. A configureMessageBroker metódusban a MessageBrokerRegistry objektumon keresztül beállítjuk a szerver oldali üzenetek kezelésének konfigurációját. A config.enableSimpleBroker("/quiz") meghatározza a témákat (topics), amelyekre fel lehet iratkozni a klienseknek. Az üzenetek ezeken a témákon keresztül továbbítódnak. A config.setApplicationDestinationPrefixes("/game") megadja a célútvonal előtagját a kliensek által küldött üzenetekhez. A registerStompEndpoints metódusban a Stom-

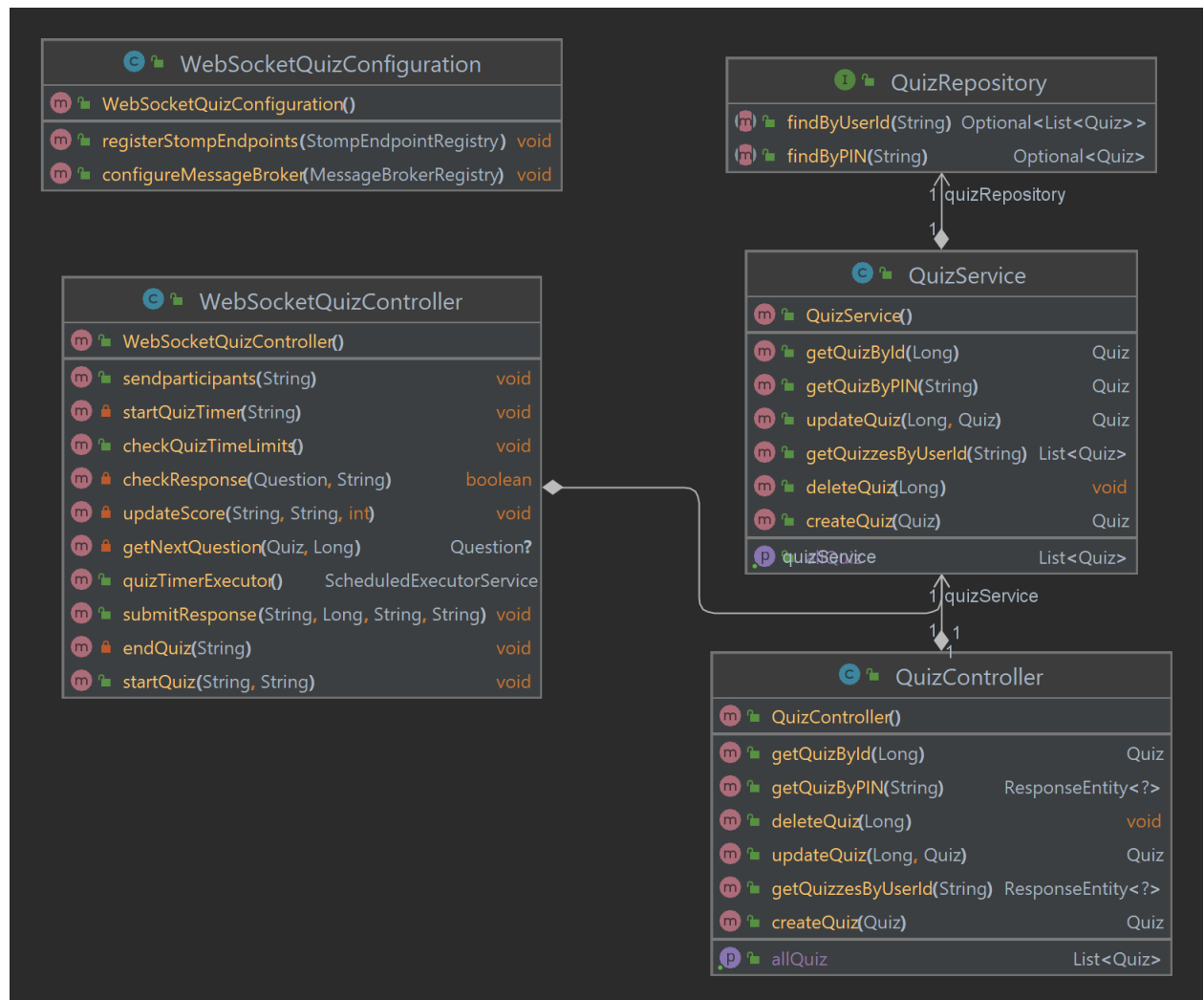


6.6. ábra. A Quiz Entity, Repository, Service és Controller

pEndpointRegistry objektumon keresztül regisztráljuk a Stomp végpontot, amely a WebSocket kapcsolatot kezeli. Az .addEndpoint("/quiz") hozzáadja a "/quiz" végpontot, amelyen keresztül a kliensek csatlakozhatnak a WebSocket-hez. A .setAllowedOrigins("http://localhost:4200") meghatározza, hogy mely eredetektől (origins) érkező kérések engedélyezettek. Az .withSockJS() hozzáadja a SockJS támogatást, amely lehetővé teszi a WebSocket alternatívát olyan böngészők számára, amelyek nem támogatják közvetlenül a WebSocket protokollt.

A WebSocketQuizController osztály a WebSocket kérések kezelését végzi. A **@Controller** annotáció jelzi, hogy ez az osztály a Spring MVC kontrollerek része. A **@CrossOrigin** annotáció meghatározza, hogy milyen eredetektől érkező kérések engedélyezettek a kontrollerekben.

Az osztályban található metódusok **@MessageMapping** annotációval vannak ellátva, amelyek meghatározzák, hogy a kliensek milyen WebSocket üzeneteket küldhetnek a megfelelő végpontokra. Például az **@MessageMapping("/startQuiz/quizPin/playerId")** metódus a **"/startQuiz/quizPin/playerId"** végponton érkező üzeneteket kezeli. Az üzenetek tartalmát, a célvégpontokat és a kliens azonosítókat a metódus paramétereiben kapjuk meg. Az osztály tartalmaz továbbá néhány segédfüggvényt, például az **updateScore**, **getNextQuestion**, **checkResponse**, stb., amelyek a kvízzjáték logikáját végzik. Ezek a függvények manipulálják a **participants** és **quizTimers** változókat, valamint az üzeneteket küldik a klienseknek a **messagingTemplate** segítségével.



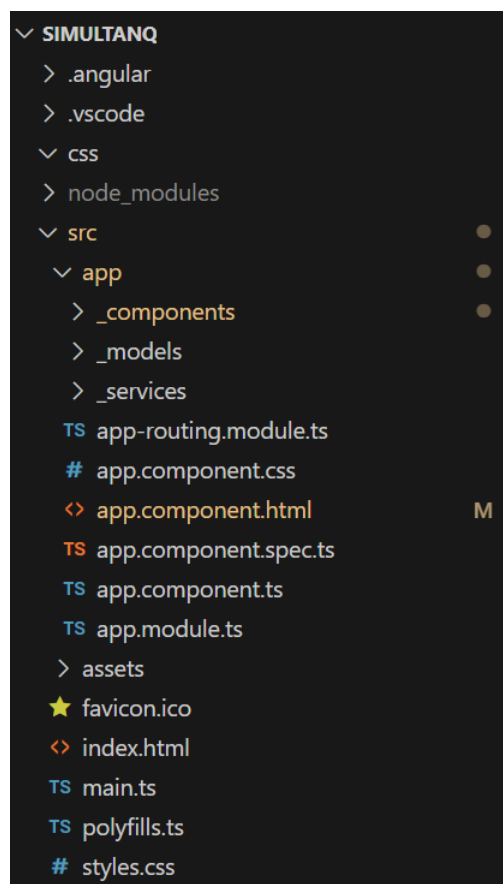
6.7. ábra. A **WebSocketQuizConfiguration** és a **WebSocketQuizController** osztályok

6.2. Kliensoldali megvalósítás

A kliens oldal egy webes felület, amely az Angular keretrendszert használva lett implementálva. A kliens oldal a kvízek létrehozásakor REST API hívásokon keresztül

kommunikál a szerver oldallal, amikor pedig a játéklétrehozó felhasználó elindítja a kvíz-játékot, akkor WebSocketen keresztül jön létre a kapcsolat a játékosokkal és a szerver oldallal. Az alkalmazás használatához előzetes telepítés szükséges, a futtatáshoz pedig egy webes böngésző.

Az Angular egy nyílt forráskódú webes alkalmazásfejlesztési keretrendszer, amelyet a Google fejlesztett ki. Az Angular célja, hogy hatékonyan és struktúráltan lehessen építeni összetett és kifinomult webes alkalmazásokat. Az Angular alapvető építőköve a komponensek. A komponensek önállóan működő, újrafelhasználható és tesztelhető egységek, amelyek a felhasználói felületek részleteit definiálják. A komponens alapú architektúra tiszta elkülönítést biztosít a különböző feladatok között, ami könnyebb fejleszthetőséget és karbantarthatóságot eredményez.



6.8. ábra. Mappaszerkezet Angular-ben

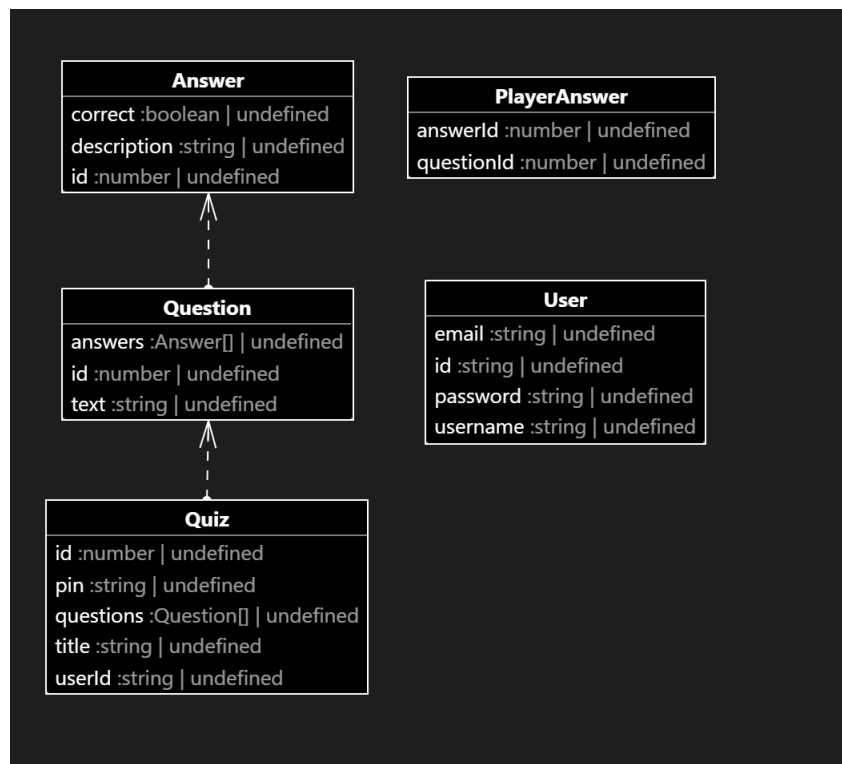
Az Angular keretrendszer másik nagy előnye, hogy lehetővé teszi a kétirányú adat-kötést, amely az adatok automatikus frissítését oldja meg a felhasználói felületen és a komponensben is. Így az alkalmazások interaktívan, dinamikusan működhetnek.

6.2.1. Modellek

A kliensoldali applikációban létrehoztam különböző adatmodelleket, amelyek a szer-veroldalról érkező adatok leképezésében és a szer-veroldalra való adatok továbbításában

voltak segítségemre. Ezek az osztályok tartalmazzák az adatok struktúráját és attribútumait, amelyekre az alkalmazásnak szüksége van.

Így használtam a User, Quiz, Answer, PlayerAnswer modelleket. A User modell osztály tartalmazza a felhasználónevet, egyedi azonosítót, jelszót és emailcímet. A modell osztályok segítenek az adatok átvitelében és kommunikálásában a szerverrel, biztosítják az adatok konzisztenciáját a kérések során.



6.9. ábra. Modellek Angular-ben

6.2.2. Komponensek

A komponensek felelősek a felhasználói felület létrehozásáért és vezérléséért. A komponensek meghatározzák a felhasználói felület részleteit, mint például a HTML sablonok és a stílusok: hogy hogyan jelennek meg az adatok és az interakciók a felhasználó számára. A komponensek HTML kódot tartalmaznak, amely beágyazza a dinamikusan változó adatokat, és lehetővé teszi az események kezelését. Tartalmazhatnak üzleti logikát, amely felelős az adatok feldolgozásáért és manipulálásáért. Ez magában foglalja az adatok betöltését vagy mentését a háttérben, az adatok formázását, valamint az alkalmazás általános működését befolyásoló döntéseket. A komponensek lehetővé teszik az adatok és a felhasználói interakciók közötti kapcsolatok kezelését. Rendelkeznek életciklus eseményekkel, amelyek azokat a pontokat jelölik, amikor egy komponens létrejön, frissül vagy megszűnik. Ezek az események lehetővé teszik a fejlesztők számára, hogy különböző tevékenységeket hajtsanak végre a komponens életciklusa során, például adatok inicializálása, erőforrások felszabadítása vagy külső szolgáltatásokkal történő kommunikáció. Lehetővé teszik a kétirányú adatkötést az adatok és a felhasználói felület között. A komponens

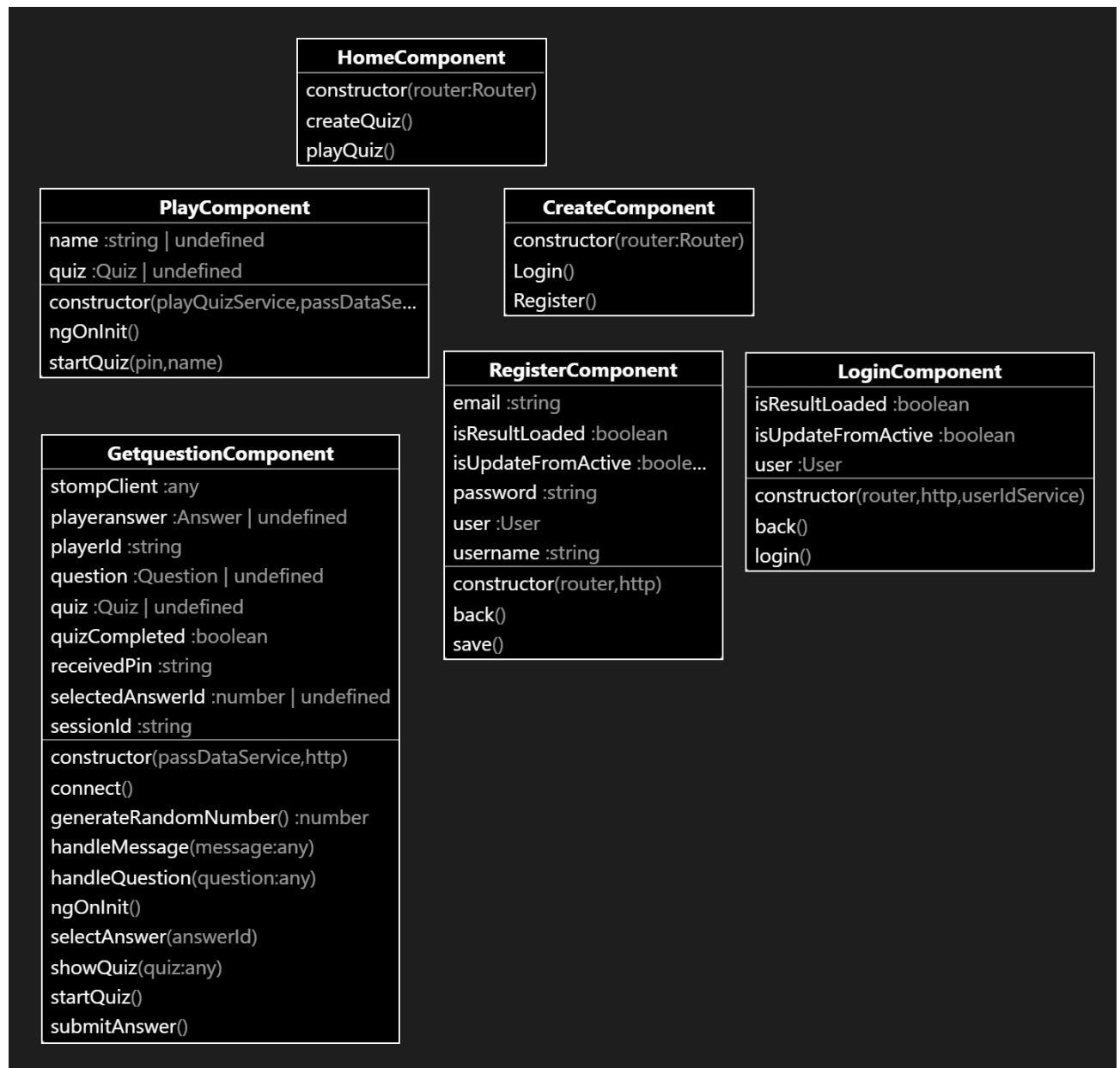
képes az adatok automatikus frissítésére mind a felhasználói felületen, mind a komponensben történő változások során. Az adatkötés segítségével a felhasználói felület és az adatok szinkronizáltak maradnak. A komponensek modulárisak és újrafelhasználhatóak. A komponenseket más komponensekbe is be lehet ágyazni vagy újrahaznosítani a kód-bázisban. Így hatékonyan és strukturáltan lehet felületeket építeni és egyszerűen kezelni az alkalmazást.

A SimultanQ alkalmazás fejlesztése során több komponensre is szükség volt. A legelső komponens, amellyel a felhasználó találkozik, az a Home komponens. Itt a felhasználói felületen két nyomógomb található, melyek kifejezik, hogy már az elején kétféle felhasználót különböztetünk meg. A Host a Quiz nyomógombra kattintva lehetőség nyílik regisztrálni vagy bejelentkezni. A regisztráció során szükséges megadni a felhasználónevet, emailcímet és jelszót. A bejelentkezés során email címre és jelszóra van szükség. A bejelentkezett felhasználó létrehozhat kvizeket a CreateQuizComponent segítségével, illetve megtekintheti a már létrehozott kvizeit a ListQuizzesComponent használatával. A QuizLeaderBoard komponens tartalmazza a kvíz elindításához, a STOMP WebSocketkel létrehozandó kapcsolathoz szükséges kódokat. Ennek a komponensnek a felhasználói felülete vizualizálja a kvízhez csatlakozott felhasználókat, elért pontszámaikat és rangsorukat.



6.10. ábra. A bejelentkezés után elérhető komponensek

A regisztrációhoz nem kötött komponensek közé tartozik a PlayComponent és a GetquestionComponent. Mindkettő a játékosok kvízhez való PIN-kóddal történő csatlakozását, a kérdések elérését és válaszaik elküldését szolgálja.



6.11. ábra. Az alkalmazás mindenki számára hozzáférhető komponensei

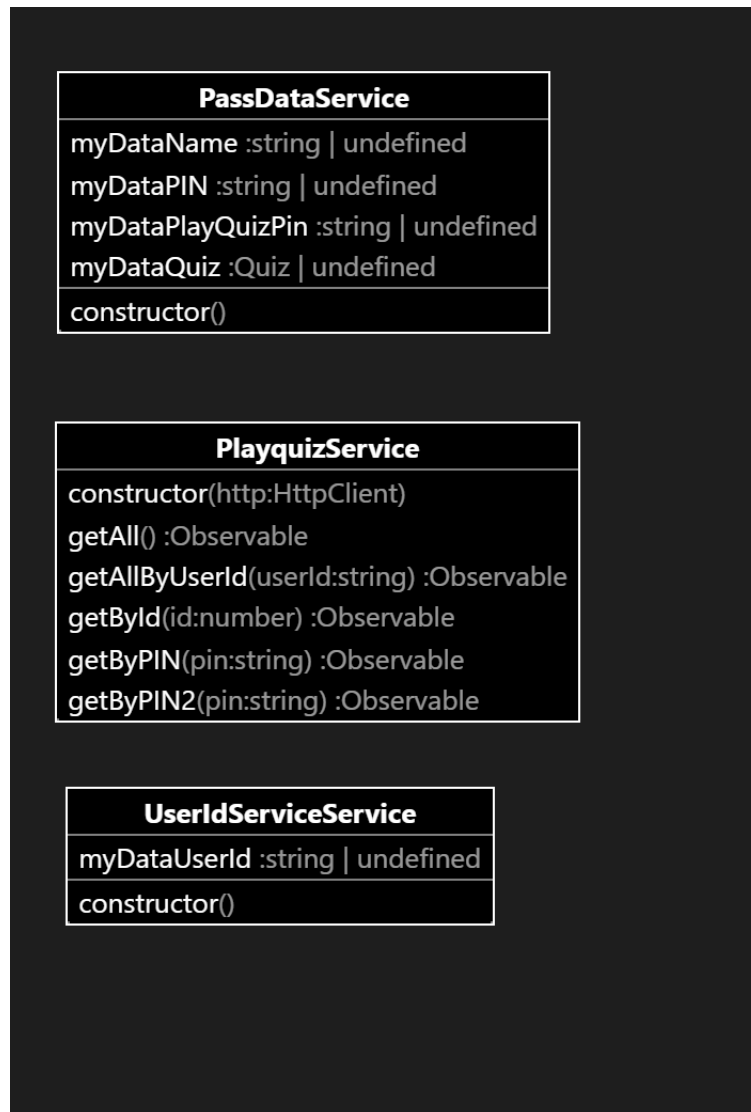
6.2.3. Szolgáltatások

Itt található az alkalmazás üzleti logikájának egy része, itt folyik az adatkezelés. A Service osztályok lehetővé teszik a kommunikációt külső szolgáltatásokkal, például HTTP hívásokkal, WebSocketekkel vagy más API-kkal. Ezek az osztályok felelősek a szerverrel

való adatkérések és válaszok kezeléséért, az adatok feldolgozásáért és az alkalmazás számára releváns információk visszaadásáért. A Service osztályok segítenek a kódrendszer tisztán tartásában és a felelősségek szétválasztásában. A Service osztályok ugyanakkor lehetőséget adnak az adatmegosztásra és a komponensek közötti kommunikációra. Ezek az osztályok tárolnak adatokat, amelyeket a komponensek között megosztanak, így biztosítva a konzisztens adatokat és állapotokat az alkalmazásban. A Service osztályok közvetítőként működnek a komponensek között, ami lehetővé teszi a lazább csatolást és az alkalmazás skálázhatóságát.

A Service osztályokat gyakran használtam adatmegosztásra a különböző komponensek között. Például a komponensek közötti navigáció során sokszor szükséges volt egyes adatokat, mint a felhasználó neve és azonosítója, a kvíz PIN-kódja vagy éppen egy teljes Quiz objektumot továbbadni vagy átvenni. Ehhez elengedhetetlen volt a Service osztályok használata.

Ugyanakkor az egyes felhasználó kvízeinek listázása során a HTTP hívásokat a PlayQuizService által valósítottam meg, ahol implementáltam az ehhez szükséges kódokat.



6.12. ábra. Angular Service osztályok

7. fejezet

Eszközök és munkamódszerek

7.1. Segédeszközök

7.1.1. Integrált fejlesztői környezet (IDE)

Fejlesztői környezetnek backenden az **IntelliJ IDEA Ultimate**[\[14\]](#)-t használtam. Nagymértékben megkönnyítette a fejlesztést, ugyanakkor szintaxis kiemelést, kódkiegészítést, kódformázást, git integráció mellett több kiegészítőt is biztosít.

Frontenden pedig **Visual Studio Code**[\[15\]](#)-ot, ami szintén tartalmazza az előbb megnevezett funkciókat.

7.1.2. Adatbázis

A PostgreSQL adatbázis menedzselését, az elsődleges és külső kulcsok helyes implementálásának ellenőrzését a pgAdmin4 [\[16\]](#)-nal oldottam meg.

7.1.3. Verziókövetés eszközök

A **Github**-ot használtam a verziókövetésre, a projekt különböző változatainak tárolására. A fejlesztés során többször olyan irányba indultam el, amelyből később átgondolva vissza kellett térnem a projekt egy korábbi verziójához, és a Github használata ezt nagyon megkönnyítette. Külön repository-t hoztam létre a backend és a frontend részeknek.

7.1.4. Api tesztelés

A REST API tesztelésére **Postman** [\[17\]](#)-t használtam, amely egy grafikus felületet biztosít a HTTP kérések küldésére. Segítségével a szerver oldali munka tesztelését hatékonyan végezhettem, a hibákat is sokkal könnyebb és gyorsabb volt észrevenni és kijavítani.

7.1.5. Design eszközök

A design elkészítéséhez **Google Drawings** [\[18\]](#)-t használtam, amely egy webes alapú vektorgrafikus szerkesztő- és diagramkészítő eszköz, amelyet a Google fejlesztett ki és kínál felhőalapú szolgáltatásként. A Google Drive alkalmazáscsomag része. Az eszközön

belül számos rajzolósi eszköz, alakzat, szín és szövegformázási lehetőség áll rendelkezésre, amelyek segítségével könnyedén létrehozhatók és testre szabhatók a grafikák.

7.2. Tesztelés

Az alkalmazásfejlesztési folyamat szerves része a tesztelés. Következetes tesztek futtatásával célzottan ellenőrizhetjük az alkalmazás helyességét, a funkcionálisok viselkedését és a használhatóságot. A tesztelés előnyei közé tartozik a gyors visszajelzés a hibákról, a hibák korai felismerése a fejlesztési ciklusban és a biztonságosabb kódújrafaktorálás.

Az alkalmazás teszteléséhez **kézi tesztelést** alkalmaztam, amely során végignavigáltam a különböző funkcionálisokon és igyekeztem kiszűrni a hibákat.

Összefoglaló

Összeségében megállapíthatom, hogy sikerült elérni a kitűzött célokat. A hasonló alkalmazások, mint a Kahoot!, Gimkit, Quizlet, Mentimeter, Poll Everywhere tanulmányozása során nagyon sok funkcionalitást megfigyelhettem és mindezek inspirációt jelentettek a saját alkalmazásom, a SimultanQ tervezésében és fejlesztésében, de fény derült arra is, hogy ezeknek az alkalmazásoknak az ingyenes verziói korlátozzák a kvízzjátékban egyszerre részt vevő játékosok számát. Ez pedig hátrányt jelent akkor, ha például egy egész évfolyam diákjait szeretnénk bevonni egy előadás utáni kvízzjátékba, vagy egy közepes/nagy méretű rendezvény során igyekeznénk meghívni a résztvevőket egy közös online játékra.

Az applikációm tervezése és fejlesztése során felhasznált technológiák, a Spring Boot és Angular keretrendszerek, a PostgreSQL adatbáziskezelő rendszer mind alkalmasnak és rendkívül hasznosnak bizonyultak a kitűzött célok megvalósításában.

A SimultanQ alkalmazás lehetővé teszi a két jól elhatárolt szerepkörben való tevékenységeket, mint a Host és a Player, azaz a kvízzjátékok létrehozásáért és elindításáért felelős házigazda és a játékos tevékenységei. A felhasználónak, mint Host-nak, lehetősége van a regisztrációra, bejelentkezésre, majd kvízek létrehozására, illetve azok elindítására, az élő játék során az eredmények nyomon követésére, megjelenítésére. A Player-nek, azaz a játékosnak pedig egyszerűvé van téve a folyamat, hiszen regisztráció nélkül, csupán a PIN-kód birtokában lehetősége van a játékhoz való csatlakozásra, a kvízzjátékban való részvételre.

Az alkalmazás megfelelő használat mellett ideálisan működik, azonban nem tökéletes. Az applikáció fejlesztése és a tesztelése során további fejlesztési ötletek, új funkcionalitások, pontosítások és módosítások fogalmazódtak meg, amelyek megvalósítása a továbbiakban fog megtörténni. Ilyenek például a más típusú kvízkérdések létrehozásának implementálása a már meglévő többválaszos mellett, a kvízek időzítésének testreszabása, multimédiás fájlok használatának lehetősége, a statisztikák, eredmények kielemezésének részletezése, bővítése, ezek dokumentálása.

Publikus GitHub repository-k linkjei:

- Backend: https://github.com/napsugarinf/simultanQ_backend
- Frontend: https://github.com/napsugarinf/simultanQ_frontend

Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, Dr. Antal Margit tanárnőnek, aki az alkalmazás tervezése és fejlesztése során végig nagyban segítette és támogatta munkámat.

Ábrák jegyzéke

3.1. WebSocket kapcsolat [9]	21
4.1. Szerepkörök az applikációban	24
4.2. A Host használati esetei	25
4.3. A Player használati esetei	27
5.1. Blokkdiagram a rendszer architektúrájáról	30
5.2. Szerveroldali architektúra Java Spring Boot-ban	32
5.3. Kliensoldali architektúra Angular-ben [13]	33
5.4. A felhasználói felület drótváza	35
6.1. Mappaszerkezet Spring Boot-ban	37
6.2. Quiz, Question és Answer osztályok és kapcsolataik	37
6.3. Repository réteg	38
6.4. Service réteg	39
6.5. Controller réteg	40
6.6. A Quiz Entity, Repository, Service és Controller	41
6.7. A WebSocketQuizConfiguration és a WebSocketQuizController osztályok	42
6.8. Mappaszerkezet Angular-ben	43
6.9. Modellek Angular-ben	44
6.10. A bejelentkezés után elérhető komponensek	45
6.11. Az alkalmazás mindenki számára hozzáférhető komponensei	46
6.12. Angular Service osztályok	48

Irodalomjegyzék

- [1] „G2 -Where you go for software.” <https://www.g2.com/products/kahoot/competitors/alternatives>. (elérés dátuma: 2023. jún. 24.).
- [2] „Kahoot! -Learning games.” <https://kahoot.com/>. (elérés dátuma: 2023. jún. 25.).
- [3] „Gimkit -Live learning game show.” <https://www.gimkit.com/>. (elérés dátuma: 2023. jún. 25.).
- [4] „Quizlet -Learning tools, flashcards and textbook solutions.” <https://quizlet.com/>. (elérés dátuma: 2023. jún. 25.).
- [5] „Mentimeter -Interactive presentation software.” <https://www.mentimeter.com/>. (elérés dátuma: 2023. jún. 25.).
- [6] „Poll everywhere -Interactive and fun polls.” <https://www.polleverywhere.com/>. (elérés dátuma: 2023. jún. 25.).
- [7] „Spring -What Spring can do.” <https://spring.io/>. (elérés dátuma: 2023. júl. 1.).
- [8] „Intellij -Spring Boot Documentation.” <https://www.jetbrains.com/help/idea/spring-boot.html>. (elérés dátuma: 2023. jún. 25.).
- [9] „Abyl -Websockets.” <https://ably.com/topic/the-challenge-of-scaling-websockets>. (elérés dátuma: 2023. jún. 25.).
- [10] „Abyl -Scaling Websockets.” <https://ably.com/topic/the-challenge-of-scaling-websockets>. (elérés dátuma: 2023. jún. 25.).
- [11] „Angular -What is Angular.” <https://angular.io/guide/what-is-angular>. (elérés dátuma: 2023. jún. 25.).
- [12] „Git hivatalos dokumentáció.” <https://git-scm.com/doc>. (elérés dátuma: 2023. júl. 2.).
- [13] „Introduction to angular concepts -Architecture.” <https://angular.io/guide/architecture/>. (elérés dátuma: 2023. jún. 20.).
- [14] „Download intelij idea ultimate -The Leading Java and Kotlin IDE.” <https://jetbrains.com/idea/download/>. (elérés dátuma: 2023. jún. 18.).
- [15] „Download visual studio code -Free and built on open source.” <https://code.visualstudio.com/download>. (elérés dátuma: 2023. jún. 18.).

- [16] „Download pgadmin 4 - *Management Tools for PostgreSQL*.” <https://www.pgadmin.org/download/>. (elérés dátuma: 2023. jún. 19.).
- [17] „Introduction | postman learning center. - *Postman Learning Center*.” <https://learning.postman.com/docs/getting-started/introduction/>. (elérés dátuma: 2023. jún. 18.).
- [18] „Google drawings - *Easily create diagrams and charts*.” <https://docs.google.com/drawings>. (elérés dátuma: 2023. jún. 18.).