Eötvös Loránd Tudományegyetem

Informatikai Kar

Információs Rendszerek Tanszék

Tanulást támogató webalkalmazás valós idejű kvízjátékokkal

Témavezető:
Kotroczó Roland
egyetemi tanársegéd

Szerző:
Takács Simon
programtervező informatikus BSc

Budapest, 2025

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Figyelem! Nyári záróvizsga esetén a módosítás határideje <u>február 1.</u>, őszi záróvizsga esetén <u>augusztus 31.</u>

Hallgató adatai:

Név: Takács Simon

Neptun kód: F8BTO8

Képzési adatok:

Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)

Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: Kotroczó Roland

<u>munkahelyének neve, tanszéke:</u> **ELTE IK, Információs rendszerek Tanszék**

<u>munkahelyének címe:</u> **1117, Budapest, Pázmány Péter sétány 1/C.**

<u>beosztás és iskolai végzettsége:</u> **Egyetemi tanársegéd, programtervező informatikus MSc**

A szakdolgozat címe: Tanulást támogató webalkalmazás valós idejű kvízjátékokkal

A szakdolgozat témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

Szakdolgozatomban egy modern, tanulást támogató webalkalmazást tervezek. Az alkalmazás lehetőséget ad interaktív kvízek létrehozására és közös lejátszására. Célja egy sokoldalú digitális platform megvalósítása, amely a hagyományos oktatáson túl közösségi eseményeket is interaktív élménnyé alakít. Így egyaránt használható tanórákon, céges tréningeken vagy baráti összejöveteleken.

A kérdéseket regisztrált felhasználók készíthetik, amelyek egy központi adatbázisban tárolódnak a hosszú távú megőrzés érdekében. A kvízek összeállítását mesterséges intelligencia funkciók is segítik, hogy a kérdések létrehozása gyorsabb és hatékonyabb legyen. A játékosoknak nem kötelező a regisztráció: elég egy tetszőleges becenév és a játékhoz tartozó hatjegyű kód megadása.

Minden kérdés megválaszolására előre meghatározott idő áll rendelkezésre. A kérdések után a játékosok azonnali visszajelzést kapnak a helyes válaszról, valamint az aktuális helyezésükről a ranglistán. A sorrendet a válaszok pontossága és gyorsasága együtt határozza meg. A kvíz befejezésekor a játékot indító felhasználó részletes, letölthető elemzést kap a résztvevők teljesítményéről.

Az alkalmazás támogatja a kérdésbankok folyamatos bővítését a játékosok visszajelzései alapján. Minden játék után a résztvevők egy ötfokozatú skálán értékelhetik az adott kvízt. A regisztrált felhasználók számára különböző előfizetői csomagok is elérhetők, amelyek aktiválását a rendszer adminisztrátorai végzik. Az ő munkájukat egy adminisztrátori felület segíti, ahol áttekinthetik a felhasználói fiókokat és a létrehozott tartalmakat.

Budapest, 2025. 08. 31.

Tartalomjegyzék

1. fejezet

Bevezetés

Egyre több tanulást elősegítő webalkalmazást találhatunk az interneten. Van, amelyik kifejezetten nyelvek tanulására van optimalizálva, és segítségével hatékonyan bővíthetjük a szókincsünket. Rengeteg webalkalmazás készült már egyetemek számára is, amelyek különböző biztonsági technológiákat alkalmaznak. Ezek tökéletesen alkalmasak zárthelyi dolgozatok lebonyolítására, ahol a diákok egy előre kiosztott feladatsort kapnak, és meghatározott idő áll rendelkezésükre a megoldásra.

Én egy olyan webalkalmazást szerettem volna létrehozni, amely valós időben működik. Ebben a rendszerben a játékosok egyidejűleg válaszolják meg a kérdéseket, és minden résztvevő pontosan ugyanabban a pillanatban kapja meg az egyes kérdéseket. Ez a megközelítés több szempontból is előnyös.

A valós idejű működés biztonsági szempontból is hasznos lehet. Ha rövid időt adunk a játékosoknak egy-egy kérdés megválaszolására, akkor csak magára a kérdésre tudnak koncentrálni. Nincs idejük arra, hogy megnézzék, a szomszédjuk milyen válaszlehetőséget tart helyesnek, vagy hogy külső forrásokat keressenek. Ráadásul mivel mindenki számára ugyanabban a pillanatban érkeznek a kérdések, senki sem szerezhet előnyt azzal, hogy korábban látja a feladatokat.

Természetesen már számos hasonló webalkalmazás megtalálható az interneten, amelyek ezt az alapötletet alkalmazzák. Én mégis szerettem volna létrehozni egy sajátos megoldást, amely egyszerűen áttekinthető, ugyanakkor széles körű lehetőségeket biztosít a kvízmesterek számára a kérdések létrehozásához. Célom volt, hogy a felhasználói élmény intuitív legyen, miközben a funkciók gazdagsága is megmarad.

Remek lehetőségnek tűnt, hogy ezt az ötletemet szakdolgozati munkámként valósítom meg. Rengeteg olyan technológiával szerettem volna dolgozni, amelyekkel ko-

rábban csak felületesen ismerkedtem meg, de nem használtam őket hosszabb ideig, és nem merültem el mélyebben a lehetőségeikben. A fejlesztés során olyan technológiákkal is megismerkedtem, amelyekről nem is gondoltam volna, hogy egy full-stack alkalmazás fejlesztése közben szükség lesz rájuk. Ez a projekt így nemcsak a tervezés és megvalósítás terén bővítette a tudásomat, hanem olyan váratlan kihívásokkal is szembesített, amelyek tovább gazdagították a szakmai tapasztalatomat.

2. fejezet

Felhasználói dokumentáció

2.1. Rövid ismertetés

A webalkalmazás lehetővé teszi a regisztrált felhasználóknak, hogy kvízeket hozzanak létre, amelyek segítségével valós idejű, többjátékos játékmenetben tudnak a játékosok egymással versenyezni. A rendszer oktatási vagy szórakoztatási célokra egyaránt használható, rugalmasan alkalmazkodva a különböző felhasználói igényekhez.

A kvízek létrehozásához először regisztrálni kell a felhasználónak a platformon. A sikeres bejelentkezést követően azonnal lehetősége nyílik kérdések megadására és saját kvízek összeállítására. A rendszer többféle testreszabási lehetőséget kínál az egyes kérdésekre vonatkozóan, mint például a pontszám vagy a válaszadási idő meghatározása. Az egész kvízt el lehet látni egyedi névvel és rövid leírással, amely segít a játékosoknak megérteni a téma jellegét.

Egyedi funkcióként a felhasználóknak lehetőségük van egy Gemini [1] API kulcsot hozzáadni a fiókjukhoz, amelyet az alkalmazás felhasználva automatikusan tud helytelen válaszlehetőségeket generálni az egyes kérdésekhez. Ez jelentősen felgyorsítja a kvízkészítés folyamatát. A felhasználók különböző előfizetési csomagok közül választhatnak, amelyek eltérő funkciókhoz és korlátokhoz biztosítanak hozzáférést. A csomag kiválasztása után az admin felhasználók jóváhagyhatják a váltást, miután a fizetési összeg átutalásra került.

Az alkalmazás három különböző felhasználói szerepkörre bontható. A felhasználói és az admin jogosultság már a regisztráció során meghatározásra kerül. Az adatbázis első inicializálását követően a legelső felhasználó automatikusan admin

jogosultságot kap, így lehetőséget kap a további regisztrált felhasználók menedzselésére, csomagváltások jóváhagyására és a létrehozott kvízek módosítására. Ezt követően a rendszer már csak standard felhasználói jogosultságokat oszt ki, így kizárólag az első admin felhasználó, vagy az általa kinevezett adminisztrátorok tudnak további admin jogosultságot adni más felhasználóknak az admin panel segítségével.

A harmadik szerepkör a játékos, akinek nincs szüksége regisztrációra a platformon. Minden játék elején egy egyedi, 6 jegyű belépési kód megadásával tudnak csatlakozni a játékmenethez. Amennyiben ez sikeresen megtörtént, egy egyedi játékosnév megadása után a játékos bekerül a váróterembe. Itt várakoznia kell, amíg a játékmester el nem indítja a játékot és ki nem küldi az első kérdést a játékosok számára.

2.2. Rendszerkövetelmények

Annak érdekében, hogy minden felhasználó egyszerűen és gyorsan tudja használni a webalkalmazást lokális fejlesztői környezetben is, létrehoztam egy `startup.sh` nevű *bash* szkriptet. Ez az automatizációs szkript minden szükséges lépést elvégez, amelyet manuálisan kellene megtennünk a backend szerver és a frontend fejlesztői környezet elindításához. Ez jelentősen leegyszerűsíti a projekt telepítését és indítását, különösen új fejlesztők vagy tesztelők számára.

A szkript futtatása előtt szükséges, hogy a következő alapvető eszközök telepítve legyenek a számítógépünkre:

- **Python 3.x:** A backend (FastAPI) futtatásához szükséges. Ellenőrizhetjük a `python --version` vagy `python3 --version` paranccsal.
- **Node.js és npm:** A frontend (React) és a Vite build tool futtatásához elengedhetetlen. Ellenőrizhetjük az `npm --version` paranccsal.

Az alkalmazás fejlesztéséhez használt számítógép tulajdonságai:

- Operációs rendszer: macOS Sequoia 15.7.1
- Chip: Apple M2
- 8 GB memória

2.3. Lokális futtatás a használathoz

Először adjunk futtatási jogosultságot a `startup.sh` nevű *bash* szkriptnek a következő paranccsal: `chmod +x startup.sh`. Ezt követően futtassuk a `./startup.sh` paranccsal. A szkript elvégzi az összes szükséges beállítást és elindítja mind a backend, mind a frontend szervert.

Miután a szkript sikeresen lefutott és mindkét szerver elindult, már csak egy tetszőleges modern webböngészőre van szükségünk (pl. Google Chrome, Mozilla Firefox, Microsoft Edge vagy Safari). A böngésző címsorába írjuk be a `http://localhost:5173` URL-t, és máris használhatjuk a webalkalmazást teljes funkcionalitással a helyi gépünkön.

Ha pedig le szeretnénk állítani a szerver futását, akkor az *ENTER* megnyomásával tehetjük ezt meg. Ekkor automatikusan mind a két szerver leáll. Esetleg a szerveroldal egy gyerekfolyamata marad futva, amit MacOS és Linux rendszeren *Control + C*, Windows rendszeren pedig *Ctrl + C* billentyűkombinációval lehet leállítani.

2.4. Az alkalmazás oldalai

2.4.1. Nyitó oldal

A kezdőlap elsősorban a játékosoknak készült, hiszen a felület központi eleme egy jól látható, nagy méretű gomb, amely lehetővé teszi, hogy egy 6 jegyű kód megadásával azonnal csatlakozhassanak egy folyamatban lévő játékmenethez. Ez a design döntés biztosítja, hogy a játékosok gyorsan és intuitív módon tudjanak bekapcsolódni a kvízbe, anélkül hogy bonyolult navigációs lépéseken kellene keresztülmenniük.

2.1. ábra. A játékhoz csatlakozás gombja

A regisztrált felhasználók számára szintén a kezdőlap szolgál kiindulópontként. A menüsorban, a jobb felső sarokban egyaránt megtalálható a bejelentkezés és a regisztráció gomb, amelyek segítségével a felhasználók hozzáférhetnek a fiókjukhoz, vagy új fiókot hozhatnak létre. A bejelentkezést követően a felhasználók már teljes körű hozzáférést kapnak a kvízkészítő és kezelő funkciókhoz, valamint a saját profiljuk beállításaihoz.

2.2. ábra. A bejelentkezett felhasználók menüszalagja

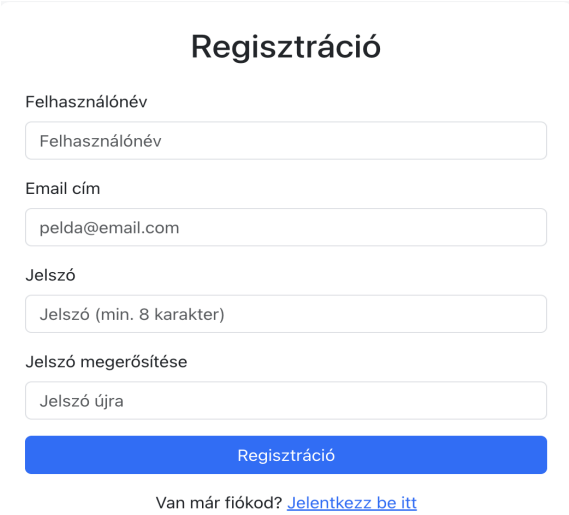
Ez a kétszintű megközelítés – a játékosok számára egyszerű, azonnali hozzáférés, a felhasználók számára pedig jól strukturált navigáció – biztosítja, hogy minden látogató gyorsan megtalálja a számára releváns funkciókat.

2.4.2. Bejelentkezés és Regisztráció

Ahhoz, hogy kvízeket tudjunk létrehozni az alkalmazásban, regisztráció szükséges. Azok a felhasználók, akik még nem rendelkeznek fiókkal, a nyitóoldal jobb felső sarkában található regisztráció gomb megnyomása után kerülhetnek át a regisztrációs oldalra, ahol egy egyszerű űrlap kitöltésével válhatnak regisztrált felhasználókká.

A regisztráció során több kötelező adatot kell megadni. Először is meg kell adni egy felhasználónevet, amely érdemes a keresztnevünknek lennie, mert a továbbiakban ezen a néven fog üdvözölni minket a rendszer minden bejelentkezés után, személyre szabott élményt biztosítva.

Továbbá egy érvényes e-mail címet is meg kell adnunk, amely a továbbiakban a bejelentkezésünk egyik kulcsfontosságú komponense lesz. Ez az e-mail cím szolgál egyedi azonosítóként a rendszerben, így fontos, hogy valós és elérhető legyen.

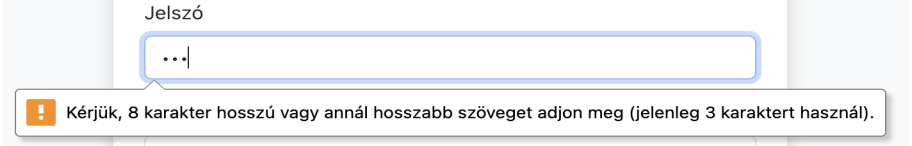


2.3. ábra. A regisztrációs űrlap

Végezetül a jelszót kell megadnunk, és ezt biztonsági okokból kétszer is be kell írnunk a megfelelő mezőkbe. Amennyiben a kétszer megadott jelszó nem egyezik, egy informatív hibaüzenet azonnal tájékoztatja a regisztrálni kívánt felhasználót a problémáról. Ez a biztonsági funkció arról biztosít minket, hogy ne történjen véletlen elgépelés a jelszó megadása során, amely később bejelentkezési problémákhoz vezethetne.

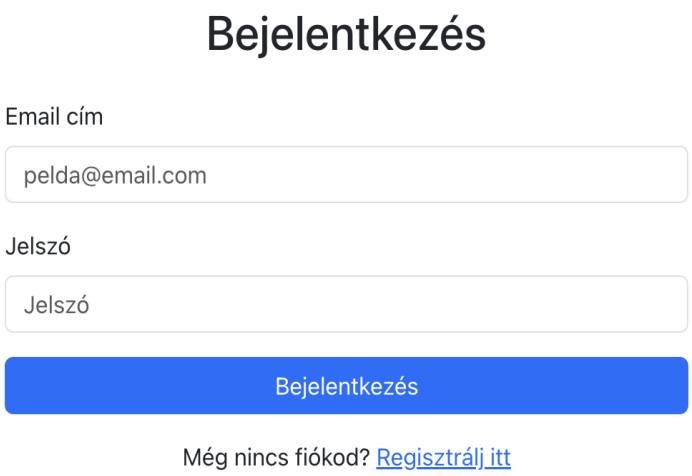
2.4. ábra. Hibaüzenet, amikor különböző a jelszó és a megerősítő

A jelszóra vonatkozóan további biztonsági követelmény is érvényben van: minimum 8 karakter hosszúnak kell lennie. Ez a megkötés azt a célt szolgálja, hogy a felhasználói fiókok minél nehezebben legyenek feltörhetők és a jelszavak ne legyenek túl könnyen kitalálhatók. Bár a 8 karakteres minimum egy alapvető követelmény, természetesen ajánlott ennél hosszabb, összetettebb jelszavakat használni a maximális biztonság érdekében.



2.5. ábra. Hibaüzenet, amikor a jelszó rövidebb, mint 8 karakter.

A sikeres regisztráció után a rendszer automatikusan átnavigál minket a bejelentkezési oldalra, ahol az előzőekben megadott e-mail cím és jelszó párossal azonnal be tudunk jelentkezni. A bejelentkezést követően teljes körű hozzáférést kapunk a kvízkészítő és -kezelő funkciókhoz, valamint a profil beállításokhoz.



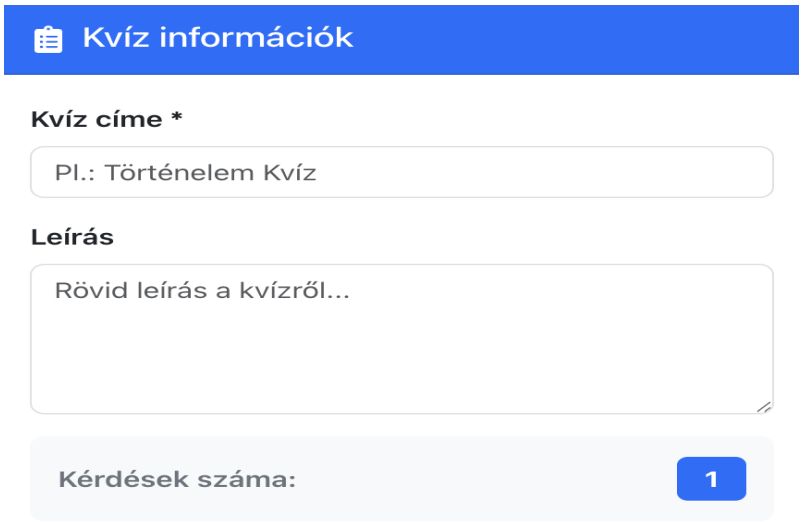
2.6. ábra. A bejelentkezési űrlap.

Ha pedig befejeztük a munkafolyamatunkat és el szeretnénk hagyni a rendszert, a menüsor jobb felső sarkában található kijelentkezés gomb szolgál a biztonságos kijelentkezésre. Ez a funkció lezárja az aktív munkamenetet és megakadályozza, hogy mások hozzáférjenek a fiókunkhoz ugyanazon a számítógépen vagy eszközön.

2.4.3. Kvíz létrehozása

Bejelentkezett felhasználók kvízeket tudnak létrehozni a webalkalmazásban, amelyeket később egy gomb segítségével el tudnak indítani, és a játékosok csatlakozhatnak a játékmenethez. A létrehozás során többféle beállítási lehetőség áll rendelkezésre a kvíz személyre szabásához.

A kvízt el tudjuk látni egy egyedi névvel, amely segít abban, hogy később gyorsan vissza tudjuk keresni a létrehozott kvízünket a kvízlistában. Ezenkívül rövid leírást is megadhatunk, amely hasznos lehet akkor, ha sok idő múlva újra használni szeretnénk a kvízt, mivel emlékeztet minket arra, hogy milyen témát dolgoz fel az adott kvíz. Ez különösen akkor előnyös, ha sok különböző témájú kvízt hozunk létre.



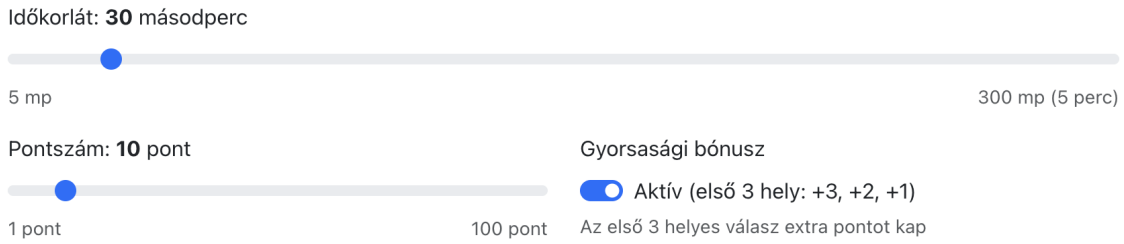
2.7. ábra. A kvízről információkat bekérő űrlap.

A kvízhez természetesen kérdéseket kell hozzáadnunk, amelyekből több különböző fajtát különböztetünk meg a változatosság és a játékélmény érdekében:

- **Egyválasztós kérdés:** A válaszlehetőségek között csupán egyetlen helyes válasz van. A játékosoknak ezt az egy helyes választ kell megtalálniuk és megjelölniük a megadott lehetőségek közül.
- **Többválasztós kérdés:** A kérdésre több helyes válasz is létezik, így akár az összes válaszlehetőséget meg tudják jelölni a játékosok. Ilyenkor a játékosoknak minden helyes választ meg kell találniuk a maximális pontszám eléréséhez.
- **Számbekérő kérdés:** Nincsenek előre megadott válaszlehetőségek. A játékosnak magának kell megadnia egy számot, és minél közelebb van ez az érték a helyes válaszhoz, annál több pontot sikerül besöpörnie. Ez a típus különösen jól használható becslési feladatoknál.
- **Sorrendező kérdés:** Vannak válaszlehetőségek, viszont ezúttal nem egyszerűen megjelölni kell őket, hanem helyes sorrendet kell kialakítani a válaszok között. A játékosoknak logikai vagy időrendi sorrendbe kell rendezniük az elemeket.

A kérdésekhez számos egyedi tulajdonságot adhatunk meg a játékmenet finomhangolása érdekében. Szabályozhatjuk, hogy hány pontot kapjanak a versenyzők helyes megoldás esetén, így differenciálhatjuk a kérdések nehézségi szintjét és fontosságát.

A játékmenet dinamikájának megfelelő kihasználása érdekében be tudjuk állítani, hogy az egyes kérdésekre mennyi időt adunk a versenyzőknek a válaszadásra. Így a játékosoknak nincs lehetőségük az idők végtelenségéig gondolkodni, ami fenntartja a játék tempóját és izgalmát.

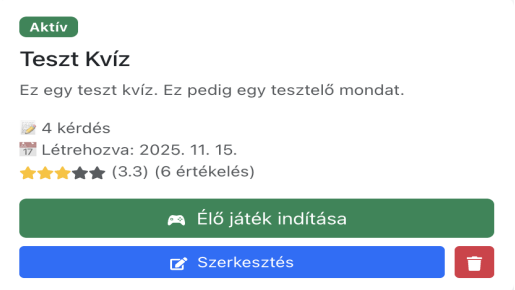


2.8. ábra. A kérdések egyedi tulajdonságainak beállítása.

A kérdések során azonban nem csak az idővel kell megküzdeniük a játékosoknak, hanem az ellenfeleik gyorsaságával is. Minden kérdéshez egyedileg megadható, hogy járjanak-e gyorsasági bónuszpontok. Ha ezt a funkciót aktiváljuk, a három leggyorsabb helyes választ adó versenyző gyorsaságuk szerint 3, 2 és 1 extra pontot kap. Ez a mechanizmus további versenyhelyzetet teremt és jutalmazza a gyors gondolkodást.

2.4.4. Kvíz elindítása

Ha ki szeretnénk próbálni az elkészült kvízeinket, akkor a menüsorban a *Kvízek* menüpontot választva megjelenik az összes általunk létrehozott kvíz áttekinthető listája. A kvízeket kis kártyák formájában jeleníti meg a rendszer, amelyek el vannak látva a kvíz nevével és rövid leírásával, így könnyen áttekinthető, hogy melyik kvízről van szó.



2.9. ábra. Egy kvízkártya.

Minden kvízkártyán három funkciógomb található, amelyek segítségével különböző műveleteket végezhetünk. A *szerkesztés* gomb lehetővé teszi a kvíz utólagos

módosítását, így bármikor hozzáadhatunk új kérdéseket, módosíthatjuk a meglévőket, vagy frissíthetjük a kvíz alapadatait. A *törlés* gomb véglegesen eltávolítja a kvízt a rendszerből, ha már nincs rá szükségünk. Az *indítás* gomb pedig elindítja a kvízt és megnyitja a játékmenetet a játékosok számára.

A játékmenet elindítása után a rendszer automatikusan generál egy egyedi, 6 jegyű kódot, amelyet meg kell osztani a játékosokkal. Ez a kód teszi lehetővé, hogy a résztvevők csatlakozzanak a megnyitott játékhoz a kezdőoldalon található csatlakozás gombbal.

2.10. ábra. Kódgenerálás után várakozás a játékosokra.

Ezután a játékmesternek meg kell várnia, amíg a játékosok egyesével csatlakoznak a váróterembe. A csatlakozott játékosok listája valós időben frissül, így a játékmester folyamatosan látja, hogy hányan vannak jelen. Legalább egy játékos csatlakozása esetén már elindítható a játékmenet, de természetesen érdemes megvárni, amíg minden meghívott résztvevő bejelentkezik a rendszerbe.

Amikor mindenki csatlakozott és a játékmester készen áll, egyetlen gombnyomással elindíthatja az első kérdést. Játékra fel!

2.4.5. Játékos csatlakozása

A játékosok a nyitóoldalon található nagy, központi gomb segítségével tudnak eljutni arra az oldalra, ahol meg kell adniuk a játékhoz való csatlakozáshoz szükséges egyedi, 6 jegyű kódot. Ez a kód osztja be őket a megfelelő játékmenethez, ahol ezután egy egyedi játékosnevet kell megadniuk.

A megadott játékosnév a játék során nem változtatható meg, mivel ez szolgál a játékos egyedi azonosítójaként a rendszerben. Ez a mechanizmus kritikus fontosságú a játék integritásának megőrzése és a újracsatlakozás funkció működése szempontjából.

Ha a játék során a játékos internetkapcsolata megszakad és offline állapotba kerül, de rövid időn belül vissza tud csatlakozni a hálózatra, akkor lehetősége van újra bekapcsolódni ugyanabba a játékmenetbe. Ezt a 6 jegyű kód és a korábban már megadott játékosnév újbóli beírásával teheti meg. A rendszer felismeri a játékost, és automatikusan visszahelyezi őt a játékba.

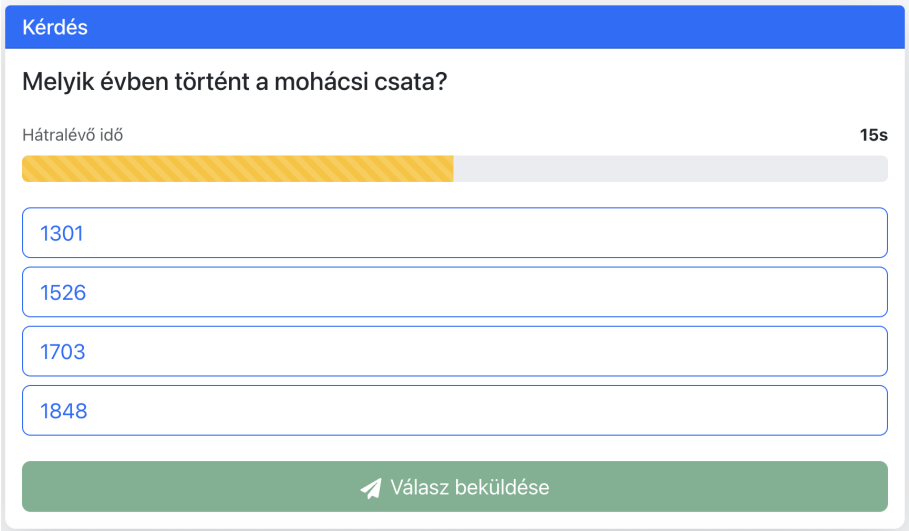
Ilyen esetben azokra a kérdésekre, amelyek az offline időszak alatt kerültek kiküldésre, a játékos automatikusan 0 pontot kap, mivel nem volt lehetősége válaszolni rájuk. Azonban az addig leadott válaszai biztonságosan mentésre kerültek a rendszerben, így a pontszáma megmarad, és az újracsatlakozást követően zavartalanul tudja folytatni a játékot a következő kérdéstől kezdve.

Ha egy játékos egy kérdés teljes időtartama alatt nem tudott jelen lenni a játékmenetben hálózati problémák miatt, ezt a rendszer '#' jellel jelzi a végeredményeket áttekintő fájlban a kvízmester számára. Természetesen ebben az esetben is 0 pont jár az adott kérdésre, viszont a kvízmester dönthet úgy, hogy egyedi módon bírálja el a helyzetet, és az adott játékos esetében más elbírálást alkalmaz.

Ez a funkció biztosítja, hogy átmeneti kapcsolati problémák ne akadályozzák meg véglegesen a játékosokat a részvételben, ugyanakkor átláthatóságot teremt a kvízmester számára, aki így teljes képet kaphat arról, mely játékosok és mely kérdéseknél léptek fel technikai problémák.

2.4.6. Egy játék folyamata

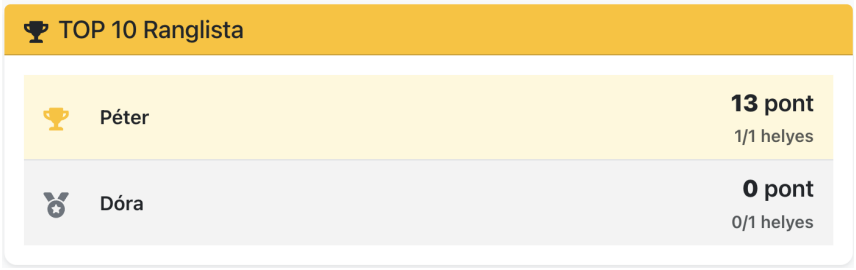
A játékmenet során a játékos kap egy kérdést a játékmestertől, amelyre a megadott időkorláton belül kell választ leadnia. Az időkorlát folyamatosan látható egy visszaszámláló formájában, így a játékos pontosan tudja, mennyi ideje maradt a válaszadásra.



2.11. ábra. Egy kérdés az éppen futó játékmenetben.

A válasz elküldése után a játékosnak várakoznia kell, amíg az összes versenytársa szintén megadja a válaszát, vagy letelik a rendelkezésre álló idő. Ez az átmeneti várakozási fázis biztosítja, hogy minden játékos egyidejűleg lássa az eredményeket, így senki sem szerez előnyt azzal, hogy mások válaszait látja.

A várakozási idő letelte után a rendszer azonnal megmutatja, hogy a játékos helyesen válaszolt-e a kérdésre. Ezenkívül láthatóvá válik, hogy jelenleg hány ponttal és hanyadik helyen szerepel a rangsorban a többi játékoshoz képest. Ez a valós idejű visszajelzés folyamatos versenyhangulatot teremt és motiválja a játékosokat a jobb teljesítményre. A rangsor dinamikusan változik minden kérdés után, így a játékosok láthatják, hogyan alakul a verseny.



2.12. ábra. Összesített ranglista a játékmester oldalán.

Amikor elfogyott az összes kérdés és a játék véget ért, a játékos megtekintheti a saját végeredményét, beleértve az összpontszámát és a végleges helyezését a ranglistán. Ez a képernyő összefoglalja a teljesítményét a teljes játékmenet során.

Ezzel szemben a játékmester sokkal részletesebb betekintést kap: hozzáfér az összes játékos teljesítményéhez, láthatja a pontos eredményeket, a helyezéseket, vala-

mint azt is, hogy ki melyik kérdésre hogyan válaszolt. Ez lehetővé teszi a játékmester számára, hogy átfogó képet kapjon a játék lefolyásáról és a résztvevők tudásáról.

2.4.7. Kvízek értékelése

A játékosoknak a kvíz befejezése után lehetőségük van minden alkalommal értékelni a játékélményt egy átlátható, 5 csillagos értékelő rendszer segítségével. Ez a visszajelzési mechanizmus lehetővé teszi, hogy a játékosok kifejezzék véleményüket a kvíz minőségéről, nehézségi szintjéről és általános élvezeti értékéről.

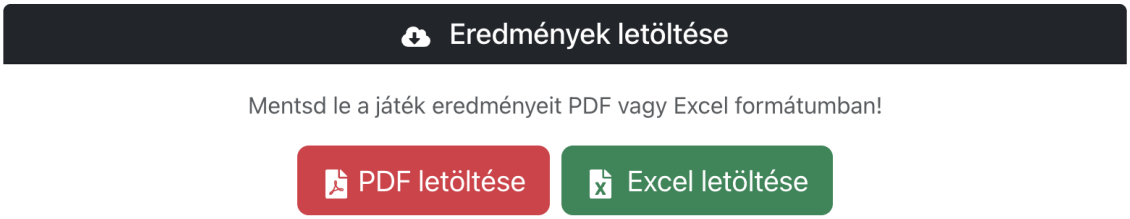
2.13. ábra. A kvízek értékelése a játék vége után.

Az értékelés egyszerű és intuitív: a játékosok 1-től 5 csillagig terjedő skálán jelezhetik elégedettségüket, ahol az 1 csillag a leggyengébb, míg az 5 csillag a legjobb értékelést jelenti. Az értékelés opcionális, így a játékosok szabadon dönthetnek arról, hogy szeretnének-e visszajelzést adni vagy inkább továbblépnek.

Ezek az értékelések rendkívül értékesek a kvízkészítők számára, mivel segítenek nekik megérteni, hogy a játékosok mennyire találták élvezetesnek és minőséginek a létrehozott tartalmat. Az összesített értékelések alapján a játékmesterek láthatják, mely kvízek a legnépszerűbbek, és hol van szükség javításra vagy finomhangolásra. Ez a folyamatos visszacsatolás hozzájárul ahhoz, hogy idővel egyre jobb minőségű kvízek szülessenek a platformon.

2.4.8. Statisztika letöltése

A játékok végeredményét a játékmesterek le tudják tölteni, ha fel szeretnék használni a játékosok teljesítményadatait későbbi elemzéshez, értékeléshez vagy archiváláshoz. A rendszer kétféle formátumban teszi lehetővé a letöltést: PDF és Excel (XLSX) formátumban.



2.14. ábra. Az eredmények letöltésére szolgáló két gomb a játékmester számára.

Mindkét formátumban megtalálható a következő részletes információ minden játékosról: a játékos neve (a játék elején megadott egyedi játékosnév); a végső helyezés a ranglistán; az összpontszám, amelyet a teljes játékmenet során szerzett; valamint kérdésenkénti bontásban, hogy az egyes kérdésekre hány pontot szerzett a játékos.

A PDF formátum különösen hasznos, ha nyomtatható, vizuálisan formázott dokumentumra van szükség, például tanórai értékeléshez vagy hivatalos dokumentációhoz. Ez a formátum könnyen megosztható és minden eszközön azonosan jelenik meg.

Az Excel formátum viszont ideális választás, ha a játékmester tovább szeretné dolgozni az adatokat, statisztikai elemzéseket kíván végezni, vagy egyedi diagramokat szeretne készíteni. Az Excel táblázatban az adatok szerkeszthetők, szűrhetők és számításokhoz használhatók.

Ez a rugalmas exportálási lehetőség biztosítja, hogy minden játékmester a számára legmegfelelőbb formátumban férhessen hozzá a játék eredményeihez.

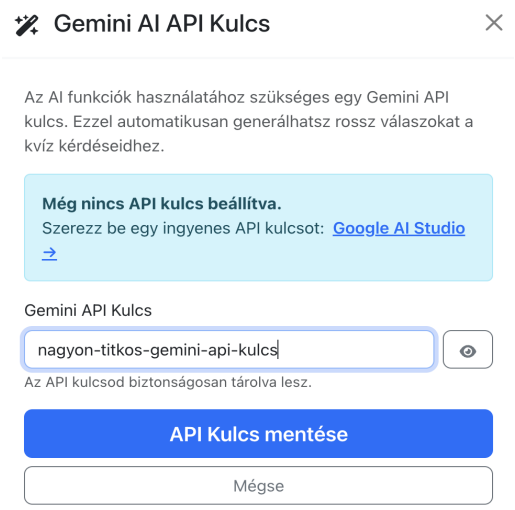
2.4.9. AI funkció használata

A kérdések készítése közben sok kvízkészítőnek okoz fennakadást, hogy a kérdéshez illeszkedő rossz válaszlehetőségeket találjon ki. A rossz válaszoknak egyrészt elég hihető megoldásoknak kell lenniük ahhoz, hogy kihívást jelentsenek, másrészt nem lehetnek túl egyértelműen tévesek, mert akkor a kvíz túl könnyűvé válik.

A jelenlegi AI forradalom hatására rengeteg alkalmazás fejlesztője integrál mesterséges intelligencia alapú megoldásokat a rendszereibe a hatékonyság és a felhasználói élmény javítása érdekében. Ezt a trendet követve én is beépítettem egy ilyen funkciót a webalkalmazásba.

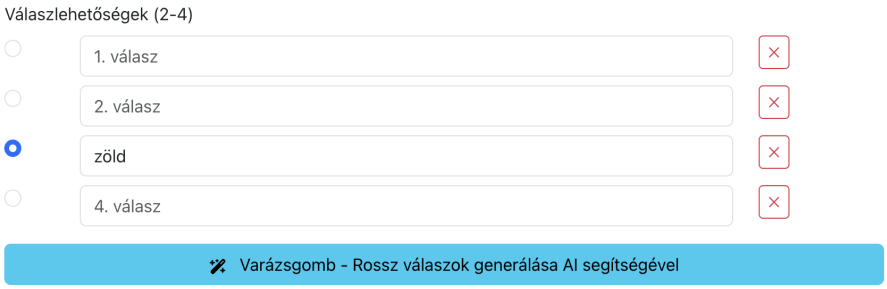
A felhasználó a Google által fejlesztett Gemini mesterséges intelligencia API kulcsát tudja használni a rossz válaszok automatikus generálásához. Az API kulcsot a fiók beállításai között, a profil menüpontban lehet hozzáadni a fiókhoz. Az

API kulcs biztonságosan tárolódik, és kizárólag a felhasználó kvízeihez kapcsolódó válaszgenerálásra használható.



2.15. ábra. A felhasználó megadja a saját Gemini API kulcsát.

A kérdések létrehozása során, miután a felhasználó megadta a kérdés szövegét és kijelölte a helyes választ, automatikusan aktívvá válik egy *Rossz válaszok generálása AI segítségével* feliratú gomb. Ennek megnyomására a rendszer elküldi a kérdést és a helyes választ a Gemini API-nak, amely néhány másodperc alatt a kérdéshez tematikusan illeszkedő rossz válaszlehetőségeket generál.



2.16. ábra. A rossz válaszlehetőségek generálására szolgáló gomb.

Ezek a generált válaszok azonnal megjelennek a kérdésszerkesztő felületen, ahol a kvízkészítő szabadon szerkesztheti, törölheti vagy akár újragenerálhatja őket, ha nem elégedett az eredménnyel. Ez a funkció jelentősen felgyorsítja a kvízkészítés folyamatát, különösen akkor, ha sok kérdést kell rövid idő alatt összeállítani.

2.5. Admin jogosultság

2.5.1. Csomagkezelés

A webalkalmazásban a felhasználók különböző előfizetési csomagok közül választhatnak, amelyek eltérő funkciókhoz és korlátokhoz biztosítanak hozzáférést. A fizetés jelenleg nem az alkalmazáson keresztül történik automatikusan, hanem külső banki átutalással vagy más megállapodott módon.

Amikor egy felhasználó igénylést ad le egy magasabb csomag aktiválására, arról azonnal értesítést kapnak az admin jogosultsággal rendelkező felhasználók. Az adminisztrátorok az admin panelen keresztül megtekinthetik a beérkező kéréseket, és el tudják dönteni, hogy elfogadják vagy elutasítják azokat. Általában a fizetés beérkezésének ellenőrzése után.

2.17. ábra. A beérkezett jóváhagyásra váró csomagigénylések listája.

Az adminisztrátorok rugalmasan kezelhetik a felhasználói csomagokat: igénylés nélkül is képesek módosítani bármely felhasználó csomagját, ami hasznos lehet promóciós akciók, tesztelési célok vagy speciális megállapodások esetén.

Háromféle előfizetési csomag közül választhatnak a felhasználók, amelyek fokozatosan bővülő funkcionalitást kínálnak:

- **Alap csomag (Ingyenes):** Ezt a csomagot minden regisztrált felhasználó automatikusan megkapja a fiók létrehozásakor. A csomag célja, hogy a felhasználó megismerje a rendszert és kipróbálja az alapvető funkciókat költségek nélkül. Az alap csomagban legfeljebb 5 különböző kvízt lehet létrehozni, illetve ebben a szintben még nem érhető el az AI-alapú válaszgenerálás funkció sem. Ez a korlátozott verzió kiválóan alkalmas arra, hogy a felhasználók felmérjék, mennyire illeszkedik az alkalmazás az igényeikhez.

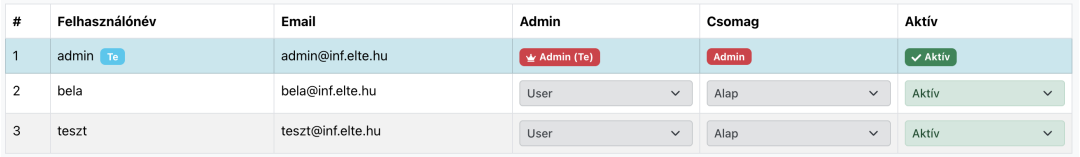
- **Profi csomag:** Ebben a csomagban már jelentősen több lehetőség áll rendelkezésre a kreatív szabadság megélésére. A profi felhasználóknak lehetőségük van használni a Gemini API kulcsukat, amelyet a felhasználói fiókjuk beállításai között tudnak hozzáadni a rendszerhez. Ez lehetővé teszi az automatikus

rossz válaszgenerálást, ami drámaian felgyorsítja a kvízkészítés folyamatát. Ezen kívül már 15 különböző kvízt tudnak létrehozni.

- **Prémium csomag:** Ezzel a csomaggal a felhasználók valóban a kvíz királyaivá válhatnak, mivel minden korlát feloldásra kerül. Korlátlan mennyiségű kvízt lehet készíteni, így akár több száz különböző téma feldolgozására is lehetőség nyílik. A prémium csomag természetesen tartalmazza az AI-alapú válaszgenerálás funkcióját is, hasonlóan a profi csomaghoz. Ez a csomag ideális választás professzionális kvízkészítők, nagyobb oktatási intézmények vagy aktív rendezvényszervezők számára, akik rendszeresen használják a platformot.

2.5.2. Felhasználók jogosultságainak módosítása

Az adminisztrátori jogosultsággal rendelkező felhasználóknak széles körű lehetőségük van a többi felhasználó fiókjának kezelésére. Az admin panel segítségével inaktívvá tehetnek felhasználókat, vagy éppen adminisztrátori jogosultságot adhatnak számukra, attól függően, hogy milyen intézkedésre van szükség.



2.18. ábra. Adminisztrátor számára a felhasználók kezelésére szolgáló beállítások.

A fiók inaktiválása különösen akkor lehet hasznos moderációs eszköz, ha nem szeretnénk véglegesen törölni egy felhasználói fiókot, de átmenetileg fel kell függeszteni a hozzáférését. Ez tipikusan akkor fordulhat elő, amikor a felhasználó megsértette az alkalmazás használati irányelveit vagy közösségi szabályzatát – például nem megfelelő témájú kvízeket hozott létre, sértő kérdéseket fogalmazott meg, vagy visszaélésszerűen használta a rendszert.

Ilyen esetekben érdemes átmenetileg inaktívvá tenni a felhasználó fiókját, majd e-mailben felszólítani őt és tájékoztatni arról, hogy pontosan miért lett felfüggesztve a hozzáférése.

Amennyiben sikerült tisztázni a helyzetet, megbeszélni a problémát és a felhasználó vállalja a szabályok betartását, az adminisztrátorok újra aktiválhatják a fiókot. Ezután a felhasználó akadálytalanul tovább tudja használni az alkalmazást, és ismét

hozzáférhet a kvízkészítő és egyéb funkciókhoz. Ez a kétlépcsős moderációs megközelítés lehetőséget ad a felhasználóknak a javításra, miközben megvédi a közösséget a nem megfelelő tartalomtól.

Az adminisztrátorok azt is eldönthetik, hogy megbízható, aktív felhasználóknak adminisztrátori jogosultságot adnak, így megosztva a moderációs és kezelési feladatokat. Ez különösen akkor lehet hasznos, ha az alkalmazást nagyobb szervezet vagy intézmény használja, és több személy részvételére van szükség a felhasználók kezeléséhez.

2.5.3. Kvízek moderálása

Az adminisztrátornak széles körű jogosultsága van az összes felhasználó által létrehozott kvízek felügyeletére és moderálására. Lehetősége van bármely kvízt megtekinteni, áttekinteni annak tartalmát, és ha szükségesnek látja, szerkesztéseket végrehajtani rajta – például helyesírási hibák javítása, pontatlan információk korrigálása vagy nem megfelelő kérdések módosítása céljából.

#	Kvíz neve	Készítő	Kérdések	Értékelés	Státusz	Létrehozva
3	**Tesztelés** Tesztelés a dokumentáció számára.	bela	1	-	Aktív	2025. 11. 26.
2	**Három kérdéses kvíz** Ez egy 3 kérdésből álló kvíz, ami tesztelésre szolgál.	bela	3	-	Aktív	2025. 11. 18.
1	**Teszt Kvíz** Ez egy teszt kvíz. Ez pedig egy tesztelő mondat.	bela	4	3.3 (6 db)	Aktív	2025. 11. 15.

2.19. ábra. A kvízek moderálására szolgáló műveletek az adminisztrátorok számára.

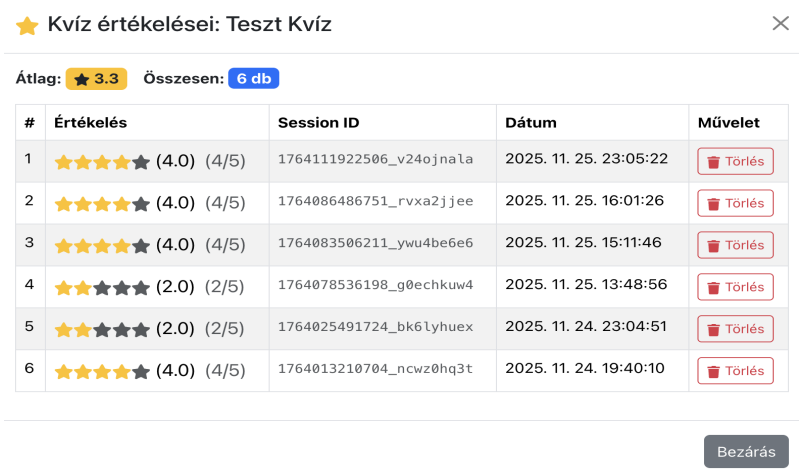
Ha egy kvíz olyan kérdéseket vagy válaszlehetőségeket tartalmaz, amely etikátlan, sértő, vagy akár jogszabályokat sért, akkor az adminisztrátornak jogában áll az egész kvízt inaktívvá tenni. Ez egy átmeneti moderációs intézkedés, amely korlátozza a kvíz használatát annak teljes törlése nélkül.

Az inaktiválás időtartama alatt a kvíz tulajdonosa nem tud játékmenetet indítani azzal a kvízzel, nem tudja szerkeszteni annak tartalmát, és nem tudja törölni sem. Ez megakadályozza, hogy a problémás tartalom továbbra is terjedjen, miközben megőrzi azt a lehetőségre, hogy a helyzet tisztázása után a kvíz visszaállítható legyen.

Az inaktív kvízek továbbra is láthatóak maradnak a felhasználó kvízlistájában, de egyértelműen meg vannak jelölve *Inaktív* státusszal. Mellettük egy informatív üzenet is megjelenik, amely tájékoztatja a felhasználót arról, hogy a kvíz átmeneti-

leg fel van függesztve, és szükséges felvenni a kapcsolatot az adminisztrátorokkal a probléma tisztázása és a kvíz esetleges visszaállítása érdekében.

Ha egy értékelés nem megfelelő (például spam vagy visszaélésszerű használat) formában jelenik meg, az adminisztrátorok jogosultak azt törölni, így biztosítva a platform tisztaságát és az értékelési rendszer hitelességét. Ez a többszintű moderációs rendszer garantálja, hogy az alkalmazás minőségi és etikus tartalmat kínáljon minden felhasználó számára.



2.20. ábra. A kvízekre küldött értékelések áttekintése, törlése.

3. fejezet

Fejlesztői dokumentáció

3.1. Tervezés és specifikáció

3.1.1. A megoldandó feladat

A szakdolgozatom során egy olyan full-stack webalkalmazást fejlesztettem, ahol tisztán elkülönül a három fejlesztési réteg: a kliensoldal, a szerveroldal és az adatbázis réteg. Ez a háromrétegű architektúra biztosítja a kód karbantarthatóságát, skálázhatóságát és a fejlesztési folyamat hatékonyságát.

A frontend részre a React [2] keretrendszert választottam, mert fontos célkitűzés volt, hogy kisebb változtatások során az alkalmazás ne töltse mindig újra a teljes oldalt, hiszen ez nagy mértékben lassítaná a felhasználói élményt és rontaná az interaktivitást. A React virtuális DOM [3] mechanizmusa és komponens-alapú architektúrája lehetővé teszi, hogy csak a szükséges részek frissüljenek, ami gyors és gördülékeny felhasználói élményt biztosít.

A backend keretrendszernek szintén egy gyors, modern és jól dokumentált eszközt kerestem, amely Python [4] nyelvre épül, mivel ebben a nyelvben érzem magam a legkomfortosabban. Ezeket a célokat szem előtt tartva találtam rá a FastAPI [5] keretrendszerre, amely tökéletesen megfelelt az elvárásoknak: rendkívül gyors aszinkron működést biztosít, automatikusan generálja az API dokumentációt, és kiválóan támogatja a modern Python funkciókat, mint például a type hints használatát.

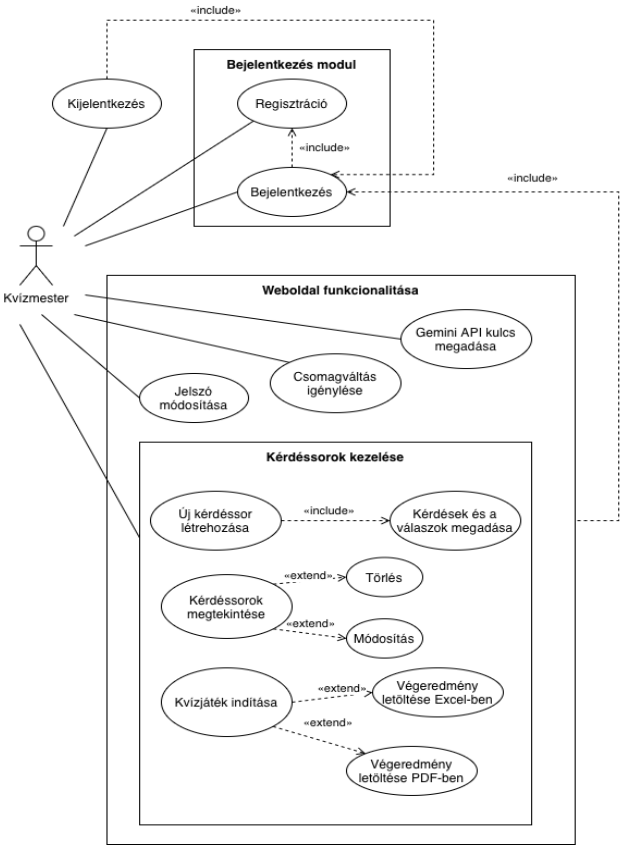
Az adatbázis réteg kiválasztásán hosszasan gondolkodtam, mert kezdetben nem voltam teljesen jártas abban, hogy milyen adatbázis-technológiák léteznek és melyik lenne a legmegfelelőbb a projektem számára. Így jelentős időt kellett szánnom a téma alapos feltérképezésére, különböző adatbáziskezelő rendszerek összehasonlítására.

Végül az SQLite [6] adatbázisra esett a választásom, amely több szempontból is ideális választásnak bizonyult. Egyrészt szakmai vélemények és dokumentációk szerint tökéletesen megfelelő kisebb és közepes méretű alkalmazások adatainak tárolására, másrészt nem igényel külön szervert, ami egyszerűsíti a telepítést és a fejlesztést.

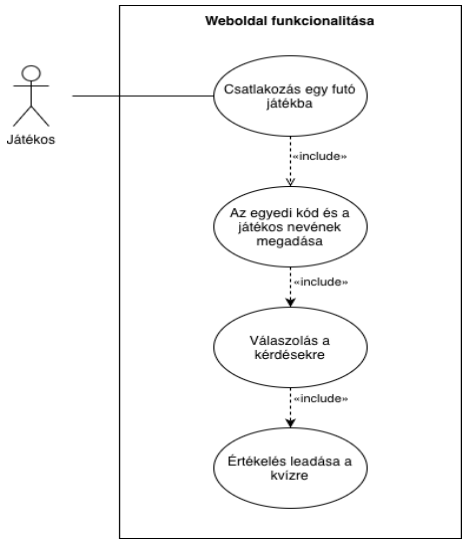
A feladat megoldása során alapvető célkitűzés volt, hogy egy gyors, responzív alkalmazást hozzak létre, amely egy olyan intuitív felhasználói felülettel rendelkezik, amely a felhasználók számára könnyen elsajátítható és természetes módon használható. A jó felhasználói élmény kritikus fontosságú minden modern webalkalmazás esetében.

A webalkalmazás funkcionalitását és működését használati eset diagramok segítségével terveztem meg a *draw.io* [7] UML készítő webalkalmazás használatával. Ez egy böngészőből közvetlenül elérhető, ingyenes webalkalmazás, ahol drag-and-drop módon tudunk különféle UML diagramokat készíteni egyszerűen és vizuálisan.

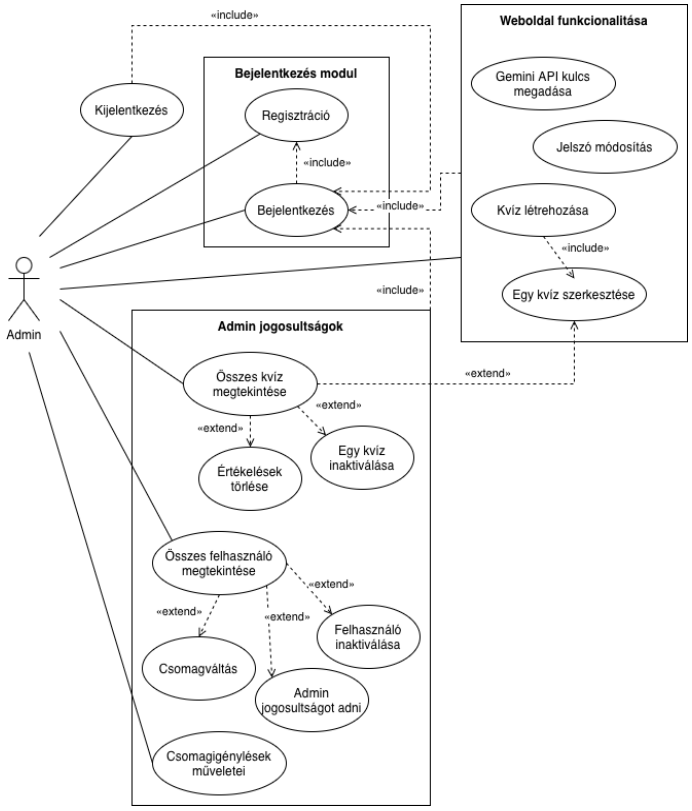
A használati eset diagramokat a három különböző szerepkör szerint hoztam létre külön-külön, hogy világosan láthassam az egyes szereplők funkcionalitását: adminisztrátor, regisztrált játékmester és játékos.



3.1. ábra. A kvízmester használati eset diagramja.



3.2. ábra. A játékos használati eset diagramja



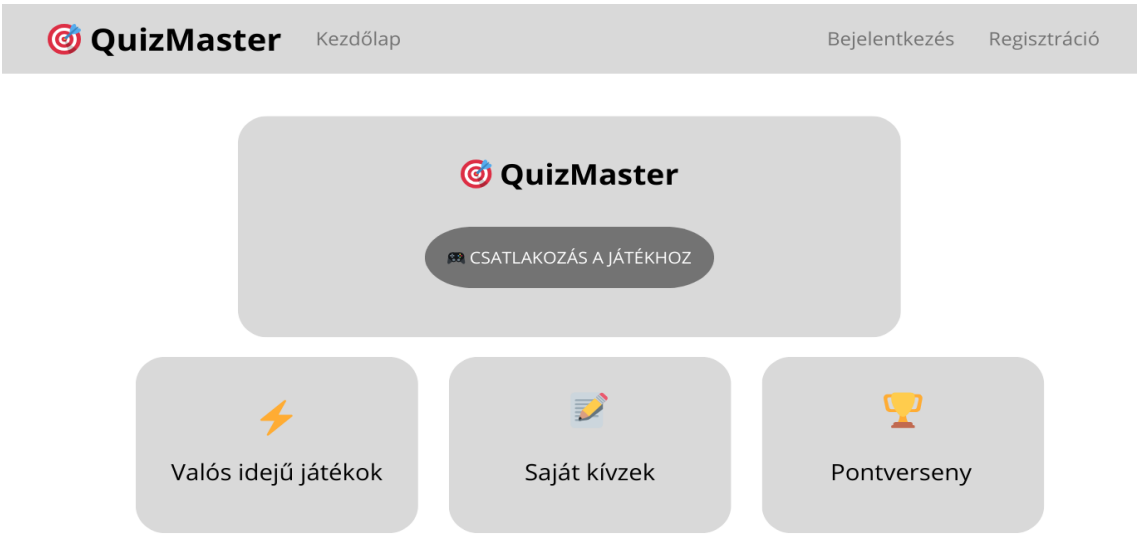
3.3. ábra. Az adminisztrátor használati eset diagramja.

Ezek a diagramok segítettek strukturálni a fejlesztési folyamatot és egyértelműen definiálni az egyes szerepkörök jogosultságait és funkcióit.

3.1.2. A felület terve

A drótvázterveket a *canva.com* [8] oldal segítségével készítettem el, ami a különböző előre létrehozott komponensei alapján egyszerű tervezést biztosított. Mivel nem mondhatom magam erősnek a design világában és nem szerettem volna nagy hangsúlyt fektetni a design-ra, így a React-hez készített *react-boostrap* [9] csomagot használtam. Ez megfelelő alapokat biztosított ahhoz, hogy ne 0-ról kezdjem el az egész design világot. Fejlesztés során pedig már amikor láttam milyen formai világot kezd ölteni a webalkalmazásom, akkor már könnyebben alakultak ki ötleteim. Hiszen láttam, hogy az a gomb jobban nézne ki, ha rendelkezne egy kis árnyékkal, vagy kattintás közben változtatna a színén, így jobban keltene kattintás funkciót.

Ezek a diagramok segítettek strukturálni a fejlesztési folyamatot és egyértelműen definiálni az egyes szerepkörök jogosultságait és funkcióit.



3.4. ábra. A kezdőlap drótvázterve.

A kezdőlap három fő feladatot lát el a felhasználók számára. Elsősorban lehetőséget biztosít a játékosoknak arra, hogy csatlakozzanak egy folyamatban lévő játékhoz. Másodsorban átfogó tájékoztatást nyújt azoknak a látogatóknak, akik először keresik fel a weboldalt, és szeretnének többet megtudni a platform működéséről. Harmadrészt a kvízmesterek is itt érhetik el a belépési és regisztrációs felületet, ahol kezelhetik a saját kvízjátékaikat.

A kezdőlap így egyszerre szolgálja ki a különböző felhasználói csoportokat: a játékosokat, akik gyorsan szeretnének bekapcsolódni a szórakozásba, az új látogatókat,

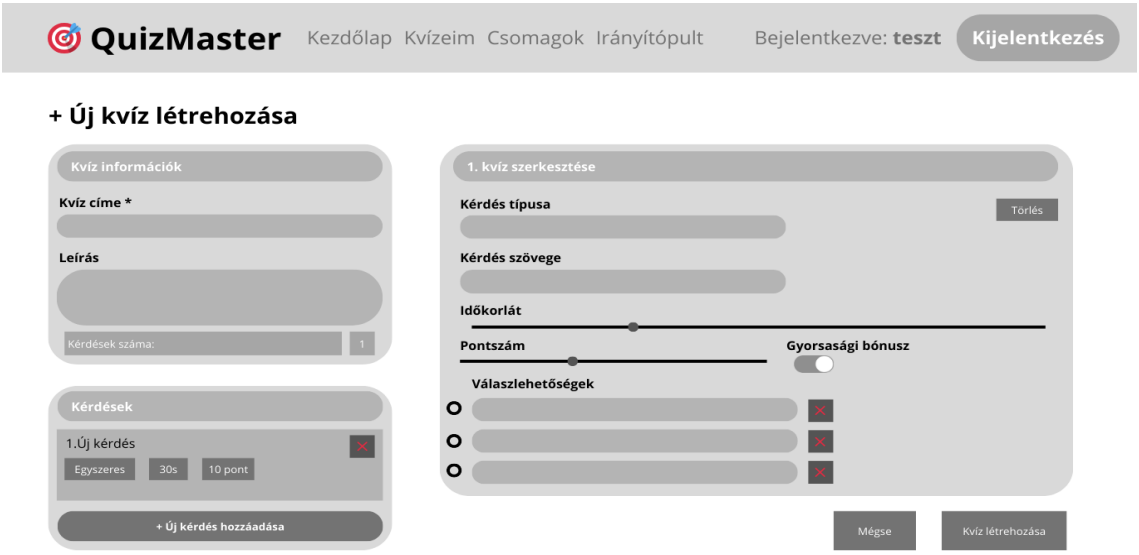
akik ismerkednek a szolgáltatással, valamint a kvízmestereket, akik saját kvízjátékokat hoznak létre és vezetnek.

3.5. ábra. A felhasználó által készített kvízek terve.

A *Kvízeim* felület átlátható és praktikus megoldást kínál a felhasználók számára. Itt egy központi helyen tekinthetik meg az összes általuk létrehozott kvízt, így egyszerűen kezelhetik a saját tartalmaikat.

A kvízek kártyás nézetben jelennek meg, amely vizuálisan is könnyen áttekinthető formát biztosít. Minden kártya tartalmazza a kvíz legfontosabb információit, így a felhasználók gyorsan tájékozódhatnak a saját kvízeik között.

A kártyákon elérhető műveleti gombok három alapvető funkciót tesznek lehetővé. Az *Indítás* gombbal azonnal el lehet kezdeni a kvízt, a *Szerkesztés* opcióval módosíthatók a kvíz tartalmai és beállításai, míg a *Törlés* funkcióval véglegesen eltávolíthatók a már nem szükséges kvízek.



3.6. ábra. A kvízek szerkesztésének felületi terve.

A kvízek szerkesztési felülete teljes körű rugalmasságot biztosít a felhasználóknak. Bármikor lehetőségük van módosítani a már létrehozott kérdéseket, valamint az azokhoz kapcsolódó különböző tulajdonságokat. Így a kvízmesterek folyamatosan frissíthetik a tartalmaikat, hogy azok mindig naprakészek és pontosak legyenek.

Ez a szerkesztői felület az adminisztrátorok számára is hozzáférhető, ami fontos biztonsági szempontból. Az adminisztrátorok átláthatják és szükség esetén módosíthatják az összes felhasználó által készített kvízt. Ez lehetővé teszi, hogy kiszűrjék a nem megfelelő tartalmakat, ellenőrizzék a kvízek minőségét, és biztosítsák, hogy a platform összes tartalma megfeleljen a közösségi irányelveknek.

Az adminisztrátori hozzáférés révén garantálható a platform biztonságos és minőségi működése, miközben a kvízkészítők szabadon alakíthatják saját tartalmaikat.

3.7. ábra. A játékosok válaszolási felületének terve.

A játékosok számára az egyik legfontosabb felület az, ahol a kérdéseket megválaszolhatják. Ennek a felületnek a kialakításánál tudatos designdöntéseket hoztam, hogy a lehető legegyszerűbb és legáttekinthetőbb élményt nyújtsam.

A felület célja, hogy a játékosok teljes mértékben a kérdésekre tudjanak koncentrálni, és semmi ne vonja el a figyelmüket. Ezért kerültem minden felesleges vizuális elemet vagy bonyolult funkciót, ami megzavarhatná őket. A használat rendkívül egyszerű: a játékosoknak csak ki kell választaniuk a szerintük helyes választ, majd egy gomb segítségével elküldeni azt.

A játékosok folyamatosan láthatják, mennyi idő van még hátra a kérdésből. Ezt az információt egy vizuális időszalagon is megjelenítettem, amely áttekinthető formában mutatja a játék előrehaladását. Az időszalag egyszerű, de hatékony visszajelzést ad az idő múlásáról anélkül, hogy elvonná a figyelmet a lényegről: magukról a kérdésekről.

3.2. Fejlesztői környezet és konfiguráció

3.2.1. A szerveroldali fejlesztői környezet előkészítése

A szerveroldal fejlesztői környezetének kialakításához elengedhetetlen egy Python virtuális környezet létrehozása. Ez Linux és macOS rendszereken a `python -m venv .venv` paranccsal, Windows rendszeren pedig a `python3 -m venv .venv`

paranccsal tehető meg a projekt gyökérkönyvtárában. Ezzel kialakítottuk a szerveroldal számára egy izolált virtuális környezetet.

A virtuális környezet használata rendkívül hasznos gyakorlat a Python fejlesztésben, hiszen egy elkülönített térben történnek meg a Python csomagok telepítései, nem pedig az egész számítógép globális Python környezetében. Ez azt jelenti, hogy különböző projektek különböző csomagverziókat használhatnak anélkül, hogy konfliktusba kerülnének egymással, és nem szennyezzük be a rendszer szintű Python telepítést.

A virtuális környezet létrehozását követően telepíteni kell a projekt függőségeit. Ehhez létrehoztam egy `requirements.txt` nevű fájlt, amely a projekt *backend* könytárában található. Ebben a fájlban vannak felsorolva a webalkalmazás működéséhez szükséges összes Python csomag a pontos verziószámaikkal együtt.

A csomagok telepítése a `pip install -r requirements.txt` paranccsal történik Linux és macOS rendszereken, míg Windows rendszerben a `pip3 install -r requirements.txt` parancsot kell használni. A parancs futtatása után az összes felsorolt csomag telepítődik a virtuális környezetbe a `requirements.txt` fájlban megadott verziószámokkal, így biztosítva a konzisztens fejlesztői környezetet minden fejlesztő számára.

Ha készen áll a virtuális környezetünk és telepítettük a függőségeket, akkor aktiválni kell azt a használat megkezdése előtt. Linux és macOS rendszereken ezt a `source .venv/bin/activate` paranccsal, Windows rendszeren pedig a `.venv\Scripts\activate` paranccsal tehetjük meg.

Sikeres aktiválás után a terminálban megjelenik egy `(venv)` vagy `(.venv)` jelölés a számítógépünk felhasználói neve előtt, ami egyértelműen jelzi, hogy sikeresen beléptünk a virtuális környezetbe. Ettől kezdve minden `pip` parancs és Python szkript futtatása ebben az izolált környezetben történik.

Az utolsó lépés már csupán a fejlesztői szerver elindítása. Ehhez navigáljunk a *backend* mappába a terminálban, majd futtassuk a `uvicorn main:app --reload` parancsot. Ez elindítja a FastAPI alkalmazást a Uvicorn ASGI [10] szerveren keresztül.

A `--reload` kapcsoló különösen hasznos fejlesztés közben, mivel gondoskodik arról, hogy ne kelljen minden apró kódváltoztatás után manuálisan újraindítani a szervert. Ehelyett a kapcsoló automatikusan figyeli a fájlrendszert, és ha bármilyen módosítást észlel a forráskódban, azonnal újraindítja a szervert, így mindig az aktu-

ális, legfrissebb változat fut. Ez jelentősen felgyorsítja a fejlesztési ciklust és javítja a produktivitást. Viszont figyeljünk arra oda, hogy éles környezetben nem szabad használni a kapcsolót, mivel teljesítménycsökkenést okoz és biztonsági kockázatot jelenthet.

3.2.2. A kliensoldali fejlesztői környezet előkészítése

A kliensoldal fejlesztői környezetének előkészítéséhez elengedhetetlen a megfelelő JavaScript [11] függőségek telepítése. Ezek a függőségek a projekt `package.json` fájljában találhatók felsorolva, a *dependencies* és *devDependencies* kulcsok alatt, pontos verziószámokkal együtt.

Az `npm install` parancs kiadása után automatikusan letöltésre kerülnek az összes szükséges csomagok és azok tranzitív függőségeik a *node_ modules* mappába. Ez a folyamat igénybe vehet néhány másodpercet vagy akár több percet is, különösen sok függőség esetén vagy lassabb internetkapcsolat mellett. A telepítés során az `npm` létrehoz egy `package-lock.json` fájlt is, amely rögzíti a pontos verziókat és biztosítja a reprodukálható telepítéseket.

A függőségek sikeres telepítését követően az `npm run dev` paranccsal elindíthatjuk a React fejlesztői szervert. Ez a parancs a Vite [12] build tool-t használja, amely rendkívül gyors Hot Module Replacement [13] funkciót biztosít, így a kódváltozások szinte azonnal megjelennek a böngészőben újratöltés nélkül.

Amennyiben már fut a szerveroldalunk egy másik terminálablakban (általában a 8000-es porton), és most már a kliensoldal fejlesztői szervere is aktív, akkor a `localhost:5173` címen érhetjük el a teljes webalkalmazásunkat.

Ehhez nyissunk meg egy általunk preferált modern webböngészőt (pl. Google Chrome), és az URL mezőbe írjuk be a `http://localhost:5173` címet. Miután betöltött az alkalmazás, szabadon használhatjuk és tesztelhetjük az összes funkciót lokálisan. A fejlesztői környezet lehetővé teszi a valós idejű kódmódosításokat és a konzol üzenetek megtekintését.

3.3. Felhasznált technológiák

3.3.1. FastAPI

Amikor körvonalazódni kezdett a szakdolgozatom témája, az első kérdés, amit feltettem magamnak, az volt, hogy milyen backend és frontend technológiákkal érdemes megvalósítani a projektet. A backend részhez mindenképpen Python alapú keretrendszert kerestem, mivel ebben a programozási nyelvben érzem magam a legkomfortosabban, és több éves tapasztalattal is rendelkezem benne.

Az általam fejlesztett alkalmazás korai prototípusa Flask keretrendszerben készült el. A Flask [14] egyszerűsége és rugalmassága miatt népszerű választás, azonban hamar kiderült, hogy ez a megoldás lassúnak bizonyul egy valós idejű, többjátékos kvízjátékhoz. A Flask szinkron működése nem volt alkalmas arra, hogy egyszerre több felhasználó kéréseit hatékonyan kezelje, különösen akkor, amikor a játékmenet gyors interakciókat igényel.

Ezért elkezdtem kutatni, hogy milyen alternatív technológiákat ajánlanak hasonló feladatokhoz. A legtöbb fejlesztői fórumon és szakmai anyagban a FastAPI keretrendszert találtam a legmegfelelőbb választásnak. Ekkor ismerkedtem meg vele részletesebben, és a váltás gyorsan indokolttá vált.

A FastAPI számos előnnyel rendelkezik a Flask-hez képest. Sokkal gyorsabb, köszönhetően az aszinkron működésének és a modern Python funkciók kihasználásának. Ezen túlmenően a fejlesztési időt is jelentősen csökkentette az egyszerű szintaxisa és az átlátható struktúrája. A keretrendszer rendkívül jó dokumentációval rendelkezik, így könnyen elsajátítható még kezdők számára is.

További nagy előnyt jelentett az automatikusan generált, folyamatosan frissülő interaktív API dokumentáció, amelyet a FastAPI beépítetten biztosít Swagger és ReDoc formátumban. Ez az eszköz nagymértékben elősegítette a gyors fejlesztést, hiszen nem kellett minden API végpontot manuálisan dokumentálnom. A dokumentáció mindig naprakész maradt a kóddal, ami jelentősen megkönnyítette a tesztelést és a hibakeresést is.

A FastAPI másik nagy előnye, hogy beépítetten támogatja a WebSocket kapcsolatok kezelését. Ennek köszönhetően nem kellett alacsony szintű Python kóddal manuálisan implementálnom a WebSocket csatornák kezelését. A keretrendszer magas szintű absztrakciót biztosít erre a funkcióra, így egyszerűbben és hatékonyabban tudtam megvalósítani a valós idejű kommunikációt az alkalmazásban.

3.3.2. Websocket

A WebSocket [15] egy kétirányú kommunikációs protokoll, amely a hagyományos HTTP kérésekkel szemben, amelyek egyirányúak, teljes duplex kommunikációt tesz lehetővé. Ez azt jelenti, hogy mind a kliens, mind a szerver bármikor küldhet üzeneteket a másik fél felé, anélkül hogy új kapcsolatot kellene létesíteni.

Erre a technológiára azért láttam szükséget, mert amikor a játékmester elindítja a kvízsort, azonnal küldi is az első kérdést a játékosoknak. Hagyományos HTTP kérések esetén ez a folyamat nehézkesen működne. A játékosoknak valamilyen módon meg kellene tudniuk, hogy már elérhető az első kérdés, majd ennek alapján kellene HTTP kérést indítaniuk a szerver felé. Ez folyamatos lekérdezést (ún. polling) igényelne, ami jelentős terhelést jelentene mind a szerverre, mind a hálózatra, ráadásul késleltetést is okozna.

Ezzel szemben, ha a játékosok a szobakód beírása után felcsatlakoznak egy WebSocket csatornára, a szerver képes valós időben jelzéseket küldeni számukra arról, hogy mikor érhető el a következő kérdés. Ez azonnali értesítést biztosít minden csatlakozott játékos számára, késleltetés nélkül.

A gyakorlatban azonban egy hibrid megközelítést alkalmazok. A WebSocket csatornát csak értesítésekre használom, míg a tényleges kérdések lekéréséhez a játékosok hagyományos HTTP kéréseket küldenek a szerver felé. Ez a vegyes kommunikációs ötlet sokkal hatékonyabb és megbízhatóbb, mintha kizárólag WebSocketet vagy csak HTTP kéréseket használnánk.

A megoldás legnagyobb előnye, hogy garantálja a játék folytonosságát még instabil internetkapcsolat esetén is. Nincs olyan helyzet, hogy valaki véglegesen lemaradna egy kérdésről, csak azért, mert átmenetileg lecsatlakozott a WebSocket csatornáról. Amikor egy játékos kapcsolata helyreáll és újracsatlakozik, bár elszalasztotta a WebSocket értesítést, még mindig tud HTTP kérést indítani a szerver felé, jelezve hogy megérkezett és kéri az aktuális kérdést. A szerver pedig azonnal elküldi számára az éppen aktuális vagy következő kérdést, így a játékos zökkenőmentesen folytathatja a kvízt. Ez a rugalmas architektúra biztosítja, hogy minden játékos optimális élményben részesüljön, függetlenül a hálózati körülményeiktől.

3.3.3. React

Kezdetben az egyszerűség miatt a Jinja2 [16] templating engine-t használtam a frontend megjelenítésére. Ez a megoldás hagyományos szerver oldali renderelést biztosít, amely kisebb projekteknél teljesen elegendő lehet. Azonban egy valós idejű, interaktív kvízjátékhoz ez a technológia túl lassúnak bizonyult. A HTML oldalak visszaadása hosszú időt vett igénybe, és minden apró változtatás után is teljes oldalfrissítésre volt szükség, hiszen ilyenkor mindig újra le kellett kérni az adott oldal teljes tartalmát a szerverről. Ez különösen zavaró volt a játék dinamikus részeiben, ahol másodperceken belül változnak az információk.

Ekkor felismertem, hogy olyan keretrendszert kell választanom, amely csak a tényleges változásokat kéri le és frissíti, nem pedig az egész oldalt. Ez egy sokkal gyorsabb és felhasználóbarátabb frontend technológiát tesz lehetővé, mivel a felhasználói élmény folyamatos marad, megszakítások nélkül.

A választásom a React [2] keretrendszerre esett. A React használható JavaScripttel [11] és TypeScripttel [17] egyaránt, de mivel egyetemi tanulmányaim során JavaScript nyelven készítettem el több tárgy beadandóját is, ezért a gyors fejlesztés érdekében a JavaScript nyelvet választottam. Ez lehetővé tette, hogy a meglévő tudásomra építve hatékonyan haladjak a fejlesztéssel.

Fontos kiemelni a React több kulcsfontosságú jellemzőjét. Az alkalmazások kisebb, önálló komponensekből épülnek fel, ami jelentősen segíti az újrafelhasználhatóságot és a kód átláthatóságát. Egy jól megtervezett komponens többször is felhasználható az alkalmazás különböző részein, ami csökkenti a kód duplikációt és megkönnyíti a karbantartást.

A React másik alapvető technológiai előnye a virtuális DOM használata. Ahelyett, hogy közvetlenül és költségesen módosítaná a böngésző valódi DOM-ját, a React fenntart egy könnyűsúlyú másolatot memóriában, az úgynevezett virtuális DOM-ot. Amikor változás történik, a React összehasonlítja a virtuális DOM előző és jelenlegi állapotát, majd csak a különbségeket alkalmazza a valódi DOM-on. Ez a megközelítés drasztikusan javítja a teljesítményt, különösen olyan alkalmazásoknál, ahol gyakori frissítések történnek.

3.3.4. Adatbázis

Az egyszerűség és a gyors fejlesztés kedvéért adatbázisként SQLite adatbáziskezelőt használok a projektben. Ez a választás különösen előnyös fejlesztői környezetben, hiszen nem kell külön adatbázisszervert telepíteni és konfigurálni, mint például PostgreSQL vagy MySQL esetében. Az SQLite egy fájl-alapú adatbáziskezelő rendszer, amelyben az összes adat egyetlen `.db` fájlban tárolódik a szerveren, ami jelentősen leegyszerűsíti a telepítést és a hordozhatóságot.

Annak érdekében, hogy ne kelljen nyers SQL utasításokat beágyazni a Python kódba, az SQLAlchemy ORM (Object-Relational Mapping) [18] keretrendszert használom, amely tökéletesen illeszkedik a Python nyelvhez és filozófiájához. Az ORM egyik legnagyobb előnye, hogy adatbázis-sémákat tudok létrehozni tisztán Python nyelven, különböző osztályok (models) és metódusok segítségével definiálva, hogy milyen táblák, oszlopok és relációk legyenek az adatbázisban.

Ez az absztrakciós réteg lehetővé teszi, hogy objektum-orientált módon dolgozzak az adatokkal ahelyett, hogy SQL lekérdezéseket írnék. Például egy *User* osztály automatikusan egy *users* táblává válik az adatbázisban, és az osztály attribútumai lesznek a tábla oszlopai.

A következő Python kód segítségével tudom beállítani és inicializálni az SQLite adatbázis kapcsolatot:

```
DATABASE_URL = os.getenv("DATABASE_URL", "sqlite:///./app.db")

engine = create_engine(
    DATABASE_URL,
    connect_args={"check_same_thread": False} if "sqlite" in
        DATABASE_URL else {}
)
```

3.1. forráskód. SQLite beállítások

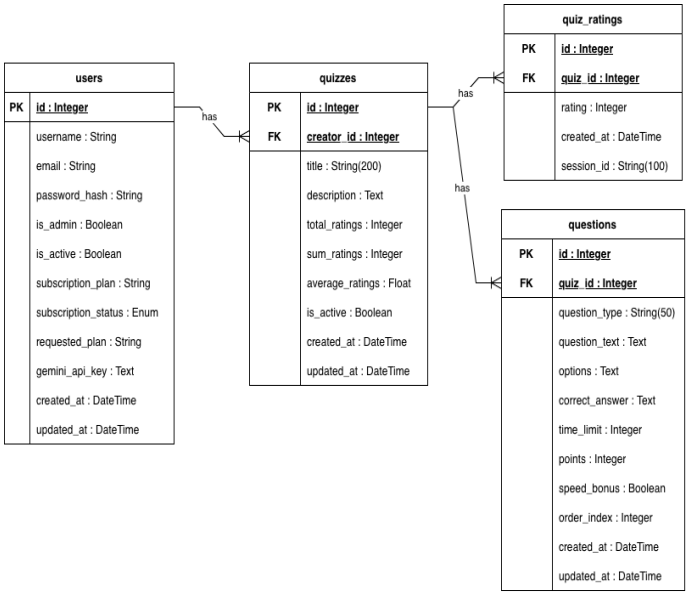
Ez a kód először megpróbálja beolvasni a `DATABASE_URL` környezeti változót, amely lehetővé teszi, hogy production környezetben más adatbázist használjunk (pl. PostgreSQL). Ha a környezeti változó nem létezik, akkor alapértelmezetten az `sqlite:///./app.db` kapcsolati stringet használja, amely egy helyi SQLite fájlt hoz létre.

A `check_same_thread: False` paraméter SQLite-specifikus beállítás, amely lehetővé teszi, hogy több szál is hozzáférjen ugyanahhoz az adatbázis kapcsolathoz. Ez

azért szükséges, mert a FastAPI aszinkron környezetben fut, ahol több párhuzamos kérés is érkezhet különböző szálakon.

Az SQLAlchemy a háttérben automatikusan SQL utasításokra fordítja a Python kódot. Amikor például egy új felhasználót hozunk létre Python objektumként és meghívjuk a `session.add()` és `session.commit()` metódusokat, az SQLAlchemy legenerálja és végrehajtja a megfelelő `INSERT INTO users ...` SQL utasítást, amelyet az SQLite motor tud értelmezni és végrehajtani. Ez az absztrakció jelentősen növeli a kód olvashatóságát, karbantarthatóságát, és csökkenti az SQL injection sebezhetőségek kockázatát is.

Az adatbázis struktúrájának jobb megértése és dokumentálása érdekében az adatbázis tábláit és azok kapcsolatait egy külön egyed-kapcsolat diagramban összeszedtem és vizualizáltam.



3.8. ábra. Az adatbázis egyed-kapcsolat diagramja.

3.3.5. Prompt engineering

Ahogyan az elmúlt években egyre inkább nőtt a nagy nyelvi modellek fejlesztése, úgy szaporodtak az ezzel járó új fogalmak is. Ilyen fogalom a prompt engineering, vagyis a prompttervezés. A nagy nyelvi modellek hatékony használatához rendkívül pontos és precízen megfogalmazott szövegekre, úgynevezett promptokra van szükség. Ezeknek a megalkotása jelentős hatással van az AI válaszainak minőségére.

Kezdetben csak egyszerű, direkt kéréseket tudtunk küldeni a nagy nyelvi modelleknek, azonban a fejlesztők hamar felismerték, hogy ezek az AI modellek al-

kalmazásokba is integrálhatók. Jelenleg, 2025 őszén, az IT szektor jelentős része AI-integrációs megoldások kidolgozására specializálódik. Ez a tendencia átformálja a szoftverfejlesztés teljes területét.

Én is szerettem volna egy olyan webalkalmazást készíteni, amely tükrözi a legfrissebb technológiai trendeket, ezért egyszerű módon integráltam az AI funkciót a projektbe. A megoldás lényege, hogy a regisztrált játékmesterek számára lehetőséget biztosítok helytelen válaszok automatikus generálására a kérdéseikhez. Ez jelentősen meggyorsítja a kvízjátékok összeállítását.

A funkció egy egyedi prompt segítségével működik. Ebben a promptban szerepel a kérdés szövege, a helyes válasz, valamint az is meghatározható, hogy hány darab helytelen választ kérünk az AI modelltől. A munkám során a Google által fejlesztett Gemini AI modellt találtam a legjobban dokumentáltnak és a legmegbízhatóbbnak, ezért a webalkalmazás jelenleg kizárólag ezt a modellt támogatja.

A dolgozatom készítése idejében a *Gemini 2.5 Flash* verziója volt elérhető az ingyenes fiókok számára is, így azt használja a webalkalmazás. Azonban fontos figyelemmel kísérni a technológia fejlődését, mivel néhány hónap múlva ez a modell már elavulttá válhat, vagy a Gemini API változásai miatt esetleg már nem lesz elérhető ebben a formában. Az AI modellek területe rendkívül dinamikusan fejlődik, ezért szükséges lehet a jövőben a modell frissítése vagy cseréje egy újabb verzióra.

A megfelelő modell kiválasztása a `routes` könyvtárban található `ai.py` fájlban lehetséges. Itt egy egyszerű kódmódosítással állítható be, hogy a rendszer melyik Gemini modellt használja. A következő kódrészlettel végezhető el a modell kiválasztása:

```
1 genai.configure(api_key=current_user.gemini_api_key)
2 model = genai.GenerativeModel('gemini-2.5-flash')
```

3.2. forráskód. Gemini modell beállítása

Biztonsági okokból nem szerettem volna saját API kulcsot nyilvánosan megosztani a szakdolgozatomban, ezért egy másik megoldást választottam. Minden felhasználó egyénileg, a saját fiókjához rendelten tudja megadni a saját Gemini API kulcsát. Így garantálható, hogy a felhasználók a saját erőforrásaikat használják, és az alkalmazás működése sem függ egyetlen központi API kulcstól. Ez a megközelítés nemcsak biztonságosabb, hanem skálázhatóbb is, mivel minden felhasználó saját tokenekkel rendelkezik.

3.3.6. JWT Autentikáció

A JSON Web Token (JWT) [19] egy nyílt szabványon alapuló technológia, amely biztonságos információcserét tesz lehetővé a különböző felek között. A JWT-k kompakt és önálló tokenek formájában továbbítanak adatokat, amelyek digitális aláírással vannak ellátva, így biztosítva azok hitelességét és sérthetetlenségét.

A JWT működése viszonylag egyszerű elveken alapul. Amikor egy felhasználó bejelentkezik a rendszerbe, a szerver generál számára egy tokent, amely tartalmazza a felhasználó azonosításához szükséges információkat. Ez a token ezután minden további kérés során elküldhető a szervernek, amely ellenőrzi annak érvényességét anélkül, hogy újra hitelesítenie kellene a felhasználót. A token három részből áll: a fejlécből, amely a token típusát és az aláírási algoritmust tartalmazza, a hasznos adatokból, amelyek a felhasználóra vonatkozó információkat hordozzák, valamint az aláírásból, amely biztosítja a token integritását.

A JWT használata különösen előnyös webalkalmazásokban, mivel nem igényel szerveroldali munkamenet tárolást. A tokenek állapotmentesek, ami azt jelenti, hogy a szerver nem köteles megjegyezni a bejelentkezett felhasználók adatait. Ez jelentősen csökkenti a szerver terhelését és egyszerűsíti a horizontális skálázást. Ráadásul a JWT-k könnyen továbbíthatók HTTP fejlécekben, így tökéletesen illeszkednek a modern webalkalmazások architektúrájába és a RESTful API-k használatához.

3.3.7. REST API

A REST API (Representational State Transfer Application Programming Interface) [20] egy széles körben elterjedt szoftverarchitektúra stílus, amely a webes szolgáltatások tervezésének alapelveit határozza meg. A REST alapvetően HTTP protokollon keresztül működik, és szabványos HTTP metódusokat használ, mint például a *GET, POST, PUT és DELETE*. Ez az architektúra állapotmentes kommunikációt biztosít a kliens és a szerver között, ami azt jelenti, hogy minden egyes kérés önmagában tartalmazza az összes szükséges információt a feldolgozáshoz. A REST API-k egyszerűek, skálázhatók és könnyen integrálhatók különböző rendszerekkel, ezért rendkívül népszerűek a modern webalkalmazások fejlesztésében.

Az én webalkalmazásomban azonban nem tisztán REST alapú kommunikációt alkalmaztam, hanem egy vegyes megközelítést választottam, amely ötvözi a REST API és a WebSocket technológia előnyeit. Amikor a játékosok csatlakoznak egy já-

tékmenethez, tulajdonképpen egy WebSocket csatornára iratkoznak fel, és ezen a csatornán keresztül várják a kvízmester jelzéseit. A WebSocket valós idejű, kétirányú kommunikációt tesz lehetővé a kliens és a szerver között.

Amikor a kvízmester elindít egy új kérdést, jelzést küld a WebSocket csatornán keresztül minden csatlakozott játékosnak. Ezt követően azonban a játékosok nem a WebSocket csatornán keresztül kapják meg magát a kérdést, hanem HTTP kéréssel töltik le azt a szervertől. Ez a hibrid megközelítés számos előnnyel jár a tisztán WebSocket alapú megoldással szemben.

Az egyik legfontosabb előny a megbízhatóság növelése. Technikai hibák vagy átmeneti kapcsolati problémák miatt előfordulhat, hogy valamelyik játékos lemarad a WebSocket csatornán küldött üzenetről. Ha a kérdéseket is ezen a csatornán küldenénk, akkor ezek a játékosok végleg elveszítenék a lehetőséget az adott kérdés megválaszolására. Azzal viszont, hogy a kérdések HTTP kérésen keresztül töltődnek le, még azok a játékosok is hozzáférhetnek a kérdéshez, akik átmenetileg offline módba kerültek, de a kérdés ideje alatt sikerült újra csatlakozniuk. Ez a funkció biztosítja, hogy kisebb technikai nehézségek ne fosszák meg a játékosokat a részvétel lehetőségétől.

A válaszok beküldése viszont ismét WebSocket csatornán keresztül történik, mivel ez a kommunikációs mód jelentősen gyorsabb, mint a hagyományos HTTP kérések. Ez különösen fontos annál a kvíz beállításnál, ahol pontokat ér a gyorsaság. A WebSocket alacsony késleltetése biztosítja, hogy a játékosok válaszai a lehető leggyorsabban eljussanak a szerverhez, így a pontozás pontosan tükrözi a játékosok reagálási idejét.

3.4. Tesztelés

A tesztelés során külön hangsúlyt fektettem a kliens- és a szerver oldal tesztelésére. Szerettem volna két különböző tesztelési típust is megismerni, így a szerver oldal tesztelésére egységteszteket hoztam létre, a kliensoldalt pedig manuálisan teszteltem. Továbbá szerettem volna a grafikai elemekre is tesztelést kitalálni, amire az ismerőseimet és barátaimat kértem meg.

3.4.1. Egységtesztek

Az egységteszteket a szerveroldal tesztelésére hoztam létre. A tesztelés során *pytest* keretrendszert és a FastAPI-hoz beépített *TestClient* tesztelő klienst alkalmaztam. A teszteket külön az alkalmazástól egy *tests* nevű csomagban hoztam létre. Így sokkal strukturáltabb alkalmazást tudtam létrehozni, viszont a *pytest* keretrendszer számára nem elvárás, hogy külön csomagban legyenek, mert az összes olyan fájlt tesztelő fájnak vesz, ami `test_*.py` vagy `*_test.py` névvel rendelkezik.

Az egységtesztelés automatizálása és egyszerűsítése érdekében létrehoztam egy `run_tests.sh` nevű *bash* szkriptet, amely futtatja az összes tesztesetet. Ez a szkript lehetővé teszi, hogy egyetlen paranccsal elindítsuk a teljes tesztcsomagot, ami jelentősen megkönnyíti a fejlesztési és tesztelési folyamatot.

Mielőtt először futtatnánk a szkriptet, szükséges futtatási jogosultságot adni neki. Ezt Linux és macOS rendszereken a `chmod +x run_tests.sh` paranccsal tehetjük meg a terminálban. Ez a parancs hozzáadja a végrehajtási jogosultságot a szkriptfájlhoz, így az operációs rendszer futtatható programként fogja kezelni.

A jogosultság megadását követően a `./run_tests.sh` parancs kiadásával indíthatjuk el a teszteket. A szkript végrehajtása során részletesen kiíródnak az egyes tesztmetódusok eredményei a terminálra.

```
(.venv) taksimon@Takacs-MacBook-Pro backend % pytest tests/ --tb=short -q
========================================= test session starts ==========================================
platform darwin -- Python 3.13.1, pytest-9.0.1, pluggy-1.6.0
rootdir: /Users/taksimon/Desktop/elte-ik-szakdolgozat-v4/backend
configfile: pytest.ini
plugins: anyio-4.11.0, cov-7.0.0
collected 46 items

tests/test_auth.py ................                                                          [ 34%]
tests/test_game.py ..............                                                           [ 65%]
tests/test_quiz.py ................                                                         [100%]

========================================== 46 passed in 3.53s ==========================================
```

3.9. ábra. A szerveroldal teszteseteinek eredménye.

3.4.2. Manuális tesztelés

A manuális tesztelés során szerettem volna a kliensoldal összes funkcióját tesztelni. Ezeket egy táblázatban szeretném megmutatni különböző csoportokra bontva.

Teszteset	**Eredmény**
Kezdőlap elérése a '/' url használatával.	Sikeresen betöltődik a kezdőlap.
Kezdőlap elérése a logóra kattintáskor.	Sikeresen betöltődik a kezdőlap.
Kezdőlap elérése a Kezdőlap menüpontra kattintva.	Sikeresen betöltődik a kezdőlap.
Kattintás a Csatlakozás játékhoz feliratú gombra.	Sikeres átirányítás az oldalra, ahol csatlakozni tudunk egy éppen futó játékhoz.
Kattintás a weboldal alján található Kvízeim gombra.	Sikeres átirányítás az oldalra, ahol a bejelentkezett felhasználó a kvízeit tudja megtekinteni.
Kattintás a weboldal alján található Új kvíz készítése gombra.	Sikeres átirányítás az űrlapra, ahol az új kvíz adatait adhatjuk meg.

3.1. táblázat. Kezdőlap tesztelése

Teszteset	**Eredmény**
Regisztráció az összes kért adat megadásával.	Sikeres regisztráció, majd átirányítás a bejelentkező oldalra.
Regisztráció hiányzó adatokkal.	Sikertelen regisztráció.
Regisztráció már regisztrált felhasználónévvel vagy email címmel.	Sikertelen regisztráció.
Regisztráció olyan email címmel, amiben nincs '@' karakter.	Sikertelen regisztráció.
Regisztráció különböző jelszóval és jelszó megerősítővel.	Sikertelen regisztráció.
Regisztráció kevesebb, mint 8 karakterrel.	Sikertelen regisztráció.

3.2. táblázat. Regisztráció tesztelése

Teszteset	**Eredmény**
Bejelentkezés a regisztrált adatokkal.	Sikeres bejelentkezés, majd átirányítás a felhasználó irányítópultjára.
Bejelentkezés nem regisztrált email címmel.	Sikertelen bejelentkezés.
Bejelentkezés helytelen jelszóval.	Sikertelen bejelentkezés.
Bejelentkezés hiányzó adatokkal.	Sikertelen bejelentkezés.

3.3. táblázat. Bejelentkezés tesztelése

Teszteset	**Eredmény**
Kattintás a Kvízeim menüpontra.	Sikeres átirányítás a Kvízeim oldalra, ahol a bejelentkezett felhasználó által készített kvízek tekinthetők meg.
Kattintás a Csomagok menüpontra.	Sikeres átirányítás a Csomagok oldalra, ahol a bejelentkezett felhasználó a jelenlegi csomagját tekintheti meg, illetve választhat további 3 csomagból.
Kattintás az Irányítópult menüpontra.	Sikeres átirányítás az Irányítópult oldalra, ahol a bejelentkezett felhasználó a fiókjáról talál információkat.
Kattintás a Kijelentkezés menüpontra.	Sikeres kijelentkezés, majd átirányítás a Bejelentkezés oldalra.

3.4. táblázat. Menüsor tesztelése

Teszteset	**Eredmény**
Kattintás az Új kvíz létrehozása gombra.	Sikeres átirányítás az űrlapra, ahol megadhatjuk a kvíz adatait.
Kattintás egy kiválasztott kvíz Élő játék indítása gombra.	Egy tájékoztató ablak jelenik meg, ahonnan tájékozódhatunk a kvízről és a játék működéséről. Továbbá innen indítható el a kvíz, illetve dönthetünk úgy, hogy visszalépünk az indítástól.
Kattintás egy kiválasztott kvíz Szerkesztés gombra.	Sikeres átirányítás az oldalra, ahol szerkeszteni tudjuk a kvízt.
Kattintás egy kiválasztott kvíz szemetessel jelölt gombjára.	A böngésző értesítő ablaka jelenik meg, ahol tájékozódhatunk arról, hogy a törlés nem visszavonható.

3.5. táblázat. Kvízeim oldal tesztelése

Teszteset	**Eredmény**
Kattintás egy kiválasztott csomag Csomag igénylése gombra.	Megjelenik egy tájékoztató üzenet, ami szerint az admin felhasználó tudja elfogadni az igénylést.
Kattintás egy leadott igénylés visszavonására.	Egy tájékoztató üzenet jelenik meg, ami szerint az igénylést visszavontuk.

3.6. táblázat. Csomagok oldal tesztelése

Teszteset	**Eredmény**
Kattintás a Jelszó módosítása gombra.	Egy ablak ugrik fel, ahol a felhasználó előző jelszavát tudja változtatni.
Jelszó módosítása az összes kért adat helyes megadásával.	Sikeres jelszóváltoztatás, majd kijelentkezés után átirányítás a Bejelentkezés oldalra.
Jelszó módosítása kevesebb, mint 8 karakter megadásával.	Sikertelen jelszóváltoztatás.
Jelszó módosítása nem megfelelő jelenlegi jelszóval	Sikertelen jelszóváltoztatás.
Jelszó módosítása különböző jelszóval és jelszó megerősítővel.	Sikertelen jelszóváltoztatás.
Kattintás az AI API Kulcs beállítása gombra.	Egy ablak ugrik fel, ahol egy Gemini API kulcsot tudunk megadni.
Egy helytelen, de 10 karakternél hosszabb API kulcs megadása.	Sikeres API kulcs beállítás.
Egy helytelen és 10 karakternél kevesebb API kulcs megadása	Sikertelen API kulcs beállítás.
Kattintás az API Kulcs törlése gombra.	Sikeres törlés egy felugró ablak elfogadása után.

3.7. táblázat. Irányítópult oldal tesztelése

Teszteset	**Eredmény**
Egy helyes kód, majd a becenevünk megadása a játékhoz csatlakozáshoz.	Sikeres csatlakozás.
Egy helytelen kód, majd a becenevünk megadása a játékhoz csatlakozáshoz.	Sikertelen csatlakozás.
Csatlakozás a játékhoz egy helyes kóddal, majd a játékban már szereplő becenévvel.	Sikertelen csatlakozás arra hivatkozva, hogy a játékos már online van.

3.8. táblázat. A játékhoz csatlakozás tesztelése

Teszteset	**Eredmény**
Kattintás játékmesterként a Játék indítása gombra, miután csatlakozott legalább 1 játékos.	Sikeresen elinduk a játék.
Kattintás játékmesterként a Kérdés befejezése gombra.	Megjelenik az aktuális kérdés eredménye a játékos és a játékmester oldalán egyaránt.
Kattintás játékmesterként a Következő kérdés gombra.	Megjelenik a következő kérdés a játékos és a játékmester oldalán egyaránt.
Játékosként elhagyni az oldalt, majd egy másik kérdés közben újra visszalépni.	Sikeresen megjelenik az éppen aktuálisan futó kérdés a még hátralevő idővel.
Játékosként elhagyni az oldalt, majd a kérdés után (az eredményjelzéskor) visszalépni az oldalra.	Megjelenik a következő kérdésre várás információ.
Kattintás játékmesterként a Játék befejezése gombra.	Megjelenik a végeredmény a játékos és a játékmester oldalán egyaránt.
Kattintás játékmesterként a végeredményt jelző oldalon a PDF letöltése vagy Excel letöltése gombra.	Elindul a PDF vagy Excel fájlt letöltése.
Értékelni játékosként a játék végén a kvízt a csillagok segítségével.	Sikeres értékelés.

3.9. táblázat. A játékmenet tesztelése

Teszteset	**Eredmény**
Kattintás az Admin beállítások menüpontra.	Sikeres átirányítás az oldalra, ahol az admin jogosultsággal rendelkező felhasználók tudnak beállításokat végezni.
Egy kiválasztott felhasználót admin jogosultsággal ellátni.	A kiválasztott felhasználó admin jogusltságot kap és nem változtathat csomagot.
Egy kiválasztott felhasználónak módosítani az aktuális előfizetési csomagját.	Sikeresen módosul a kiválasztott felhasználó előfizetési csomagja.
Egy kiválasztott felhasználót inaktívvá tenni.	A kiválasztott felhasználónak sikertelen bejelentkezés. Továbbá egy hibaüzenet, ami a fiók inaktívitására hivatkozik.
Az Összes kvíz fülön egy kiválasztott kvíz szerkesztése.	Sikeresen megjelenik a kvíz szerkesztési felülete.
Kattintás az értékelések megtekintésénél a törlés műveletre.	Egy böngésző értesítés elfogadása után törlődik a kitörölt értékelés.
Egy kiválasztott kvízt inaktívvá tenni.	A felhasználó kvíze inaktívvá vált, azaz elérhetetlenek lettek a kvízhez tartozó gombok. Illetve egy hibaüzenet figyelmeztet arra, hogy keresse fel az adminisztrátort.
Egy függőben lévő csomagkérés elfogadása.	A felhasználónak módosul az aktuális előfizetési csomagja.
Egy függőben lévő csomagkérést elutasít.	A felhasználónak nem módosul az aktuális előfizetési csomagja és törlődik a csomagkérelme.

3.10. táblázat. Az admin jogosultságok tesztelése

Teszteset	Eredmény
Egy kiválasztott szöveges mező szerkesztése, majd kattintás a Frissítés gombra.	Sikeresen megtörtént a szerkesztés.
Egy kiválasztott válaszlehetőség törlése	Törlődik a kiválasztott válaszlehetőség.
Érvényes Gemini API kulcs megadása után a kérdés és a helyes válasz megadása, majd kattintás a Varázsgombra.	Rossz válasz generálódik a helytelen válaszok szöveges mezőbe.
Érvénytelen Gemini API kulcs megadása után a kérdés és a helyes válasz megadása, majd kattintás a Varázsgombra.	Sikertelen válaszok generálása, hivatkozva az érvénytelen Gemini API kulcsra.

3.11. táblázat. A kvíz szerkesztése funkció tesztelése

3.4.3. Tapasztalatok, vélemények

Amikor a webalkalmazás már kezdett kialakulni, megkértem a barátaimat, ismerőseimet és családtagjaimat, hogy teszteljék az alkalmazást. Ez tökéletes lehetőséget biztosított arra, hogy feltárjam a rejtett hibákat és hiányosságokat, amelyeket egyedül nem vettem volna észre.

Azonban nemcsak a technikai problémák felderítésében segítettek. Rengeteg értékes visszajelzést kaptam tőlük a felhasználói felület kialakításával kapcsolatban is. Rámutattak arra, mely design ötletek működnek jól a gyakorlatban, és melyek igényelnek további változtatásokat. Javaslatokat tettek arra vonatkozóan is, hogy egyes gombok és funkciók hol lennének könnyebben elérhetők a felhasználók számára.

Ez a visszajelzési folyamat kulcsfontosságú volt az alkalmazás fejlesztésében, mivel a valódi felhasználói tapasztalatok alapján tudtam tökéletesíteni a webalkalmazást. A külső szemmel történő tesztelés jelentősen javította a végleges alkalmazás minőségét és használhatóságát.

3.5. Továbbfejlesztési lehetőségek

A szakdolgozatom írása közben is folyamatosan az volt a célom, hogy hogyan tudnám hatékony funkciókkal bővíteni az alkalmazásomat. Az idő limitáltsága miatt azonban nem tudtam minden ötletet megvalósítani. Így szeretnék arról is írni, hogy milyen ötletek maradtak megvalósítatlanul.

Részletesebb statisztikát lehetne a játékmestereknek, ami nem letölthető, hanem az adatbázisból lekérhető, amihez különböző Python adatelemző csomagokat lehetne használni, pl. *pandas*, *NumPy*, *matplotlib*. Ezen ötlet mentén, akár különböző kvíz-analitikai algoritmusokat is lehetne implementálni, amivel a játékmesterek tudomást szerezhetnek arról, hogy melyik kérdések voltak a legnehezebbek és legkönnyebbek.

A gyorsasági bónusz pontok mintájára lehetne implementálni sorozat alapú bónusz pontokat, amit az alapján lehetne kapni, hogy egyfolytában hány kérdésre sikerült jó választ adni.

Tematikus kvízcsomagokat lehetne létrehozni a játékmestereknek, amiket később fel tudnak használni játékra. A játék közben különböző hangeffektusok lehetnek, amik izgalmasabbá tehetik a játékot. Illetve lehetne olyan típusú kérdés is, amikor egy hanganyagot hallgatnak meg a játékosok, majd utána egy kérdés jelenne meg a hanganyagra vonatkozóan, így kideríthető ki mennyire figyelt a hanganyag lejátszásakor.

Technológiai szempontokban is lehetne különböző továbbfejlesztesi lehetőség, például a *Redis* használata, ami egy esetleges áramkimaradás során nem veszti el a gyorsítótárba maradt információkat. Jelenleg a ranglistát a játék közben csak szerver memóriájába mentjük, ami egy éles rendszerkörnyezetben nem túl hatékony.

4. fejezet

Összegzés

A szakdolgozatomban olyan webalkalmazást szerettem volna létrehozni, amelyet mindenki a saját igényei és elképzelései szerint alakíthat. A platform rugalmassága lehetővé teszi, hogy rendkívül sokféle kontextusban használható legyen.

A tanárok például kiválóan alkalmazhatják az órák eleji ismétlésekhez, ahol gyorsan felmérhetik a diákok tudását egy-egy tananyagról. A cégek nyílt napokon használhatják arra, hogy a látogatók számára interaktív, szórakoztató kvízeket készítsenek a vállalat tevékenységéről vagy az iparágukról. De akár baráti társaságokban is remek szórakozást nyújthat, ahol a résztvevők egymás között versenyezhetnek különböző témákban.

Az alkalmazás fejlesztése során rengeteg értékes fejlesztői tudást sikerült megszereznem. Olyan modern technológiákkal ismerkedtem meg, amelyek a mai munkaerőpiacon keresettek és hasznosak. Ez a gyakorlati tapasztalat tökéletesen kiegészíti az egyetemen megszerzett elméleti tudásomat. Az egyetem szilárd alapokat biztosított számomra, amelyekre építve folyamatosan fejlődhetek és mélyíthetem a szakmai ismereteimet.

Remélem, hogy sikerült olyan webalkalmazást létrehoznom, amely nemcsak a saját szakmai fejlődésemet szolgálta, hanem más emberek számára hasznos és értékes is lehet.

Irodalomjegyzék

[1] *Gemini API*. URL: https://ai.google.dev/gemini-api/docs (elérés dátuma 2025. 11. 20.).

[2] *React*. URL: https://react.dev/learn (elérés dátuma 2025. 11. 20.).

[3] *DOM*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model (elérés dátuma 2025. 11. 20.).

[4] *Python*. URL: https://www.python.org (elérés dátuma 2025. 11. 20.).

[5] *FastAPI*. URL: https://fastapi.tiangolo.com (elérés dátuma 2025. 11. 20.).

[6] *SQLite*. URL: https://sqlite.org (elérés dátuma 2025. 11. 20.).

[7] *draw.io*. URL: https://app.diagrams.net (elérés dátuma 2025. 11. 20.).

[8] *Canva*. URL: https://www.canva.com/templates (elérés dátuma 2025. 11. 20.).

[9] *React Bootstrap*. URL: https://react-bootstrap.netlify.app (elérés dátuma 2025. 11. 20.).

[10] *Uvicorn ASGI*. URL: https://uvicorn.dev (elérés dátuma 2025. 11. 20.).

[11] *JavaScript*. URL: https://devdocs.io/javascript/ (elérés dátuma 2025. 11. 20.).

[12] *Vite*. URL: https://vite.dev (elérés dátuma 2025. 11. 20.).

[13] *Hot Module Replacement*. URL: https://webpack.js.org/concepts/hot-module-replacement/ (elérés dátuma 2025. 11. 20.).

[14] *Flask*. URL: https://flask.palletsprojects.com/en/stable/ (elérés dátuma 2025. 11. 20.).

[15] *FastAPI Websockets*. URL: https://fastapi.tiangolo.com/advanced/websockets/ (elérés dátuma 2025. 11. 20.).

[16] *Jinja2*. URL: https://pypi.org/project/Jinja2/ (elérés dátuma 2025. 11. 20.).

[17] *TypeScript*. URL: https://www.typescriptlang.org (elérés dátuma 2025. 11. 20.).

[18] *SQLAlchemy*. URL: https://www.sqlalchemy.org (elérés dátuma 2025. 11. 20.).

[19] *JWT (JSON Web Tokens)*. URL: https://www.jwt.io (elérés dátuma 2025. 11. 20.).

[20] *REST API*. URL: https://www.geeksforgeeks.org/node-js/rest-api-introduction/ (elérés dátuma 2025. 11. 20.).

Ábrák jegyzéke

Forráskódjegyzék