ZÁRÓDOLGOZAT

Készítették:

Kerényi Tamás – Czibik Lóránt Patrik – Nagy Gergő

Konzulens:

Németh Bence

Miskolc

2023.

Miskolci SZC Kandó Kálmán Informatikai Technikum

Miskolci Szakképzési Centrum

SZOFTVERFEJLESZTŐ- ÉS TESZTELŐ SZAK

ZÁRÓDOLGOZAT

Quizter

Kvízes weboldal

Kerényi Tamás – Czibik Lóránt Patrik – Nagy Gergő

2022-2023

Tartalomjegyzék

TÉMAVÁLASZTÁS

A szakdolgozatunk témájára sok ötletünk volt, de végül egy kvízes weboldal mellett döntöttünk. Szerettünk volna valami olyat alkotni, ami interaktív, szórakoztató és később is szívesen tekintünk rá vissza.

A Quizter a játszmán felül megvalósít egy felhasználói platformot is, ahol megtekinthető mások profilja vagy eredménye a ranglistán. Ezen kívül képes az adminisztrátorok számára a kérdések listázására, módosítására vagy törlésére.

HASZNÁLT TECHNOLÓGIÁK, KÖRNYEZETEK, NYELVEK

Vue.js – Frontend

A Vue egy javascript keretrendszer felhasználói felületek fejlesztésére. Az alap HTML-re, CSS-re, Javascriptre épül, és biztosít egy deklaratív és komponens alapú programozási modellt, amely elősegíti a felhasználói felületek hatékony fejlesztését, legyen az egyszerű vagy komplex.

1. ábra: A Vue logója [1]

A Vue egy olyan keretrendszer és ökoszisztéma, amely a frontend fejlesztéshez szükséges általános funkciók többségét lefedi. A web azonban rendkívül változatos - a dolgok, amelyeket a webre építünk, drasztikusan eltérhetnek formájukban és méretükben. Ezt szem előtt tartva a Vue-t úgy tervezték, hogy rugalmas és fokozatosan átvehető legyen. [2]

A felhasználási esettől függően a Vue különböző módon használható:

- Statikus HTML továbbfejlesztése építési lépés (build step) nélkül.
- Webkomponensként történő beágyazás bármely oldalra.
- Egyoldalas alkalmazás (SPA).
- Fullstack / Server-Side Rendering (SSR).
- Jamstack / Statikus oldal generálás (SSG).
- Asztali, mobil, WebGL és akár a terminál célzása.

Single-File Component

A legtöbb Vue projektben a komponenseket egy HTML-szerű fájl formátumban, az úgynevezett Single-File Component (más néven *.vue fájlok, rövidítve SFC) segítségével írják. A Vue SFC, ahogy a neve is jelzi, egyetlen fájlba foglalja a komponens logikáját (JavaScript), sablonját (HTML) és stílusait (CSS). Az SFC a Vue egyik legmeghatározóbb jellemzője, és a Vue komponensek írásának ajánlott módja. [2]

2. ábra: A HTML5 logója [3]

3. ábra: A CSS3 logója [4]

4. ábra: A Javascript logója [5]

API stílusok

A Vue komponensek két különböző API stílusban írhatóak: Options API és Composition API.

Az Options API segítségével egy komponens logikáját beállítás részekkel, mint például *data*, *methods* és *mounted* segítségével határozzuk meg. Az beállítások által definiált tulajdonságok ezen belül függvények, amelyek a komponens példányára mutatnak.

A Composition API segítségével egy komponens logikáját importált API-funkciók segítségével határozzuk meg. Az SFC-kben a Composition API-t jellemzően a *<script setup>* segítségével használjuk. A *<script setup>* részben deklarált import-ok és felső szintű változók / függvények közvetlenül felhasználhatóak a *template*-ben. [2]

A frontend kódja az **Options API** stílusban íródott.

Pinia

A projekthez a Pinia *állapotkezelő keretrendszert* használtuk. A Pinia egy tároló könyvtár a Vue-hoz, ami lehetővé teszi az állapot megosztást komponensek vagy oldalak között.

5. ábra: A Pinia logója [6]

A Pinia az alábbikat nyújtja:

- Devtools támogatás:
  - Idővonal az akciók, mutációk követésére.
  - A store-ok a használt komponensekben jelennek meg.
  - Időutazás (time travel) és könnyebb hibakeresés.
- Hot Module Replacement (HMR):
  - A store-ok módosítása az oldal újratöltése nélkül.
  - Minden meglévő állapot megtartása fejlesztés közben.
- Pluginok: bővítse ki a Pinia funkcióit pluginekkel.
- Megfelelő TypeScript támogatás vagy automatikus kitöltés Javascript használók számára.
- Szerver oldali renderelés támogatása.

A Vuex-hez képest a Pinia egyszerűbb API-t kínál kevesebb komplikációval, és Composition stílusú API-val. [7]

A frontend felépítése

A frontend kinézete egy minimalista dizájnt követ, de azért figyeltünk arra, hogy ne legyen minden túl minimalista. Törekedtünk arra, hogy az oldal reszponzív legyen a mobilos használat érdekében. A bemutatás a komponensekkel kezdődik, majd az oldalakkal folytatódik, ami azért fontos, mivel a komponensek többször előfordulhatnak, és nem önállóan, hanem az alábbi oldalakon használva. Az oldalak leírása az **App** résznél kezdődik.

Navigációs Sáv

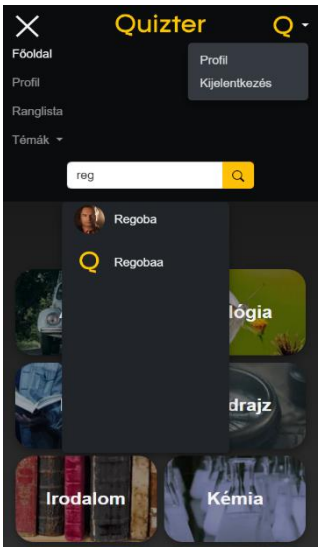
A navigációs sávval (navigation bar) a fő-, profil, ranglista, kvíz beállító, bejelentkezés és regisztráció oldalakra lehet navigálni. A kereső segítségével más felhasználókat lehet keresni.

Felépítése:

- A logó ami a főoldalra navigál.
- Navigációs linkek a fő-, profil és ranglista oldalakra.
- A “Témák” lenyíló menü, ami átnavigál a kvíz beállító oldalra a beállított téma alapján.
- Keresőmező és keresőgomb.
- Bejelentkezett felhasználó esetén a felhasználónév és alatta a szint komponens. Mellette a felhasználó képe, amire kattintva egy lenyíló menü segítségével megtekinthető a profil vagy ki lehet jelentkezni.
- Nem bejelentkezett felhasználó esetén egy felhasználó ikon, amire kattintva egy lenyíló menü segítségével elérhető a bejelentkezés vagy regisztráció oldal.

Az oldalon mobilos nézetben egy hamburger menü jelenik meg a bal oldalon, a logó középen, és a felhasználói kép vagy felhasználó ikon jobb oldalt.

6. ábra: A navigációs sáv asztali nézete, témák lenyíló menü megnyitva



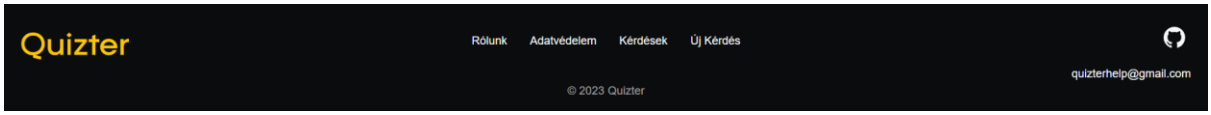
7. ábra: A navigációs sáv mobilos nézete, felhasználó lenyíló menü megnyitva

Lábléc

A lábléccel (footerrel) a fő-, rólunk és adatvédelem oldalakra lehet navigálni. Adminként a kérdések és új kérdés oldalakat is el lehet érni innen.

Felépítése:

- A logó, ami a főoldalra navigál.
- Ha nem admin a felhasználó akkor csak a rólunk és adatvédelem oldalakra navigációs linkek.
- Ha admin a felhasználó akkor az előbb említett oldalakon kívül a kérdések és új kérdés oldalakra navigációs linkek.
- Egy github ikon, ami a frontend forráskódjának a gyűjteményére (repository) linkel.
- Az email cím, kattintásra egy gmail ablakot nyit meg ahol írhat a felhasználó.
- A szerzői jog szövege.



8. ábra: A lábléc asztali nézete

9. ábra: A lábléc mobilos nézete

Szint

A szint komponens a felhasználó tapasztalat pontja alapján adja vissza a szintet és azt, hogy milyen távol van még a következő szint.

Szint	Szükséges xp
1	0
2	100
3	200
4	300
5	500
6	750
7	1,000
8	1,500
9	2,000
10	3,500
11	5,000
12	7,500
13	10,000
14	15,000
15	20,000
16	30,000
17	50,000
18	75,000
19	100,000
20	200,000
21	300,000
22	400,000
23	500,000
24	600,000
25	700,000
26	800,000
27	900,000
28	1,000,000
29	1,100,000
30	1,200,000

11. ábra: A szint haladás számítása

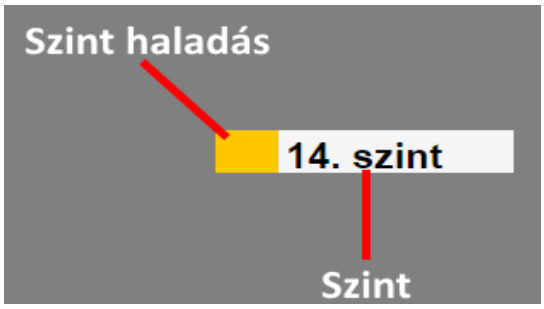
Felépítése:

- A szint középre igazítva.
- A szint haladás, ami a tároló szélességének az adott százaléka.

exp: A jelenlegi tapasztalatpont
szXpMin: A jelenlegi szinthez szükséges tapasztalatpont
szXpMax: A következő szinthez szükséges tapasztalatpont

$$\underbrace{\frac{100}{szXpMax - szXpMin}}_{\text{Slope}} * exp + \underbrace{\left(-1 * \underbrace{\frac{100}{szXpMax - szXpMin}}_{\text{Slope}} * szXpMin \right)}_{\text{yIntercept}}$$

10. ábra: A szint számítása



12. ábra: A szint komponens

Téma gomb

A téma gomb komponens jeleníti meg a főoldalon a témákat. Paraméterként kapja a téma nevét, amit a tároló közepén jelenít meg, és a téma azonosítóját, ami szerint beállítja a téma képét és átnavigál a kvíz beállító oldalra.

13. ábra: A téma gomb komponens

Töltés

A töltés komponens egy forgó körszelet, ami a Quizter logó ’Q’ betűjének egy része. Ez jelenik meg amíg a felhasználó vár a szerver válaszára.

14. ábra: A töltés komponens

Hiba

A hiba komponens csakis akkor jelenik meg, ha probléma akad a töltés során. A “Vissza” gombbal a főoldalra lehet navigálni.

15. ábra: A hiba komponens

App

Itt jelennek meg az oldalak a navigációs sáv (navigation bar) és a lábléc (footer) közé ágyazva. Az oldalak egy tárolóban vannak azért, hogy a lábléc (footer) az oldal alján maradjon minden méretben. Itt történik a bejelentkezett felhasználó automatikus beléptetése, ami a bejelentkezés során generált token alapján működik. Ezt a tokent az oldal eltárolja egy sütiben (cookie), ami 1 hét után lejár. Erre azért van szükség, hogy a böngésző bezárása után ne kelljen újra bejelentkezni.

Főoldal

A főoldalon keresztül lehet témát választani, vagy az előbb említett módon, a navigációs sávban található “Témák” lenyíló menü segítségével.

A rendelkezésre álló témák:

- Autók
- Biológia
- Fizika
- Földrajz
- Irodalom
- Kémia
- Sport
- Szórakoztatás
- Technológia
- Történelem
- Zene
- Vegyes

16. ábra: A főoldal asztali nézete

Felépítése:

- A logó, alatta a “Teszteld a tudásod” szöveg.
- A téma gombok, amik a kvíz beállító oldalra navigálnak.

17. ábra: A főoldal mobilos nézete

Kvíz Beállító

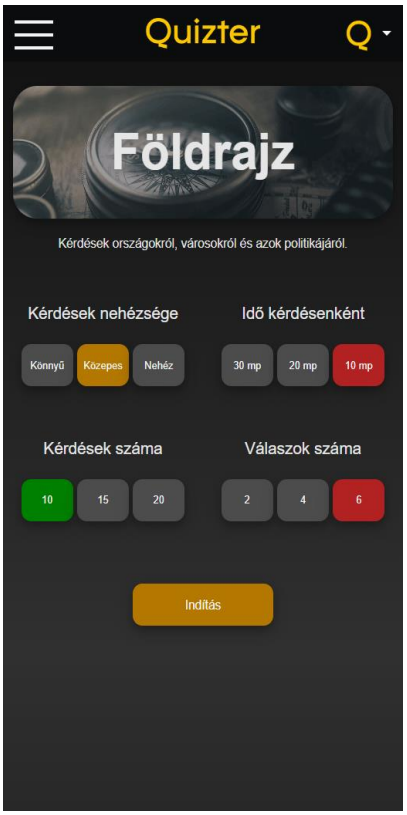
A kvíz beállító oldalon lehet a játszmát személyre szabni. A zöld szín jelöli a könnyű, a sárga a közepes és a piros a nehéz nehezítéseket. A beállítások és paraméterei nehézség szerint növekvő sorrendben:

1. **A kérdések nehézsége:**
    - Könnyű
    - Közepes
    - Nehéz

2. **Az idő kérdésenként:**
    - 30 másodperc
    - 20 másodperc
    - 10 másodperc

3. **A kérdések száma:**
    - 10 kérdés
    - 15 kérdés
    - 20 kérdés

4. **A válaszok száma:**
    - 2 válasz
    - 4 válasz
    - 6 válasz



18. ábra: A kvíz beállító oldal mobilos nézete

Felépítése:

- A téma képe és szövege.
- A téma magyarázata.
- A beállítások, amik tartalmazzák a beállítás nevét, és alatta a 3 hozzá tartozó gombot.
- Az “Indítás” gomb, ami átnavigál a játékmenet oldalra, ha mind a 4 beállítás megvan adva.

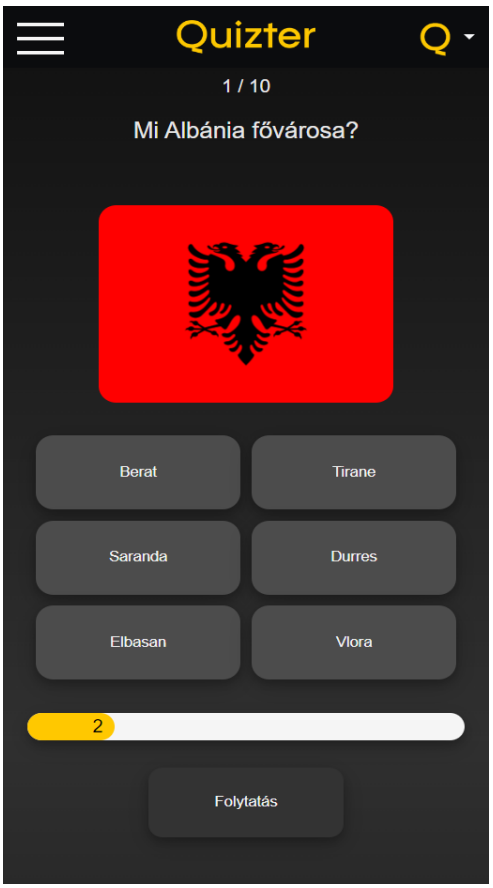
19. ábra: A kvíz beállító oldal asztali nézete

Játékmenet

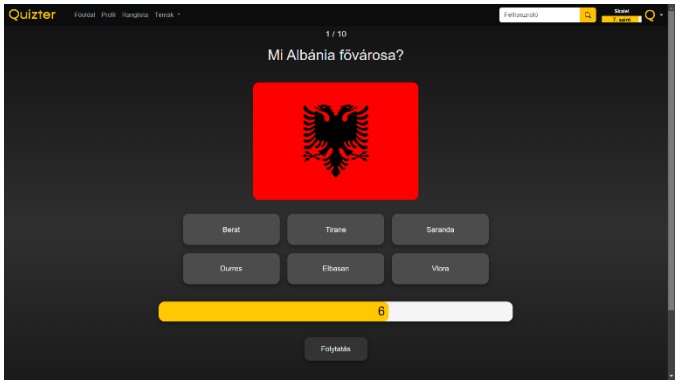
A játékmenet oldalon körönként jelennek meg új kérdések. Ezekre az adott időn belül kell választ adni. Ha ez nem sikerül, akkor az helytelen válasznak számít. Az adott időn belül megadott helytelen válasz nem ér pontot. Viszont ha helyes, akkor a maradt idő szerint növekszik a pontszám. A játszma végén a nehézség és válasz szám nehezítések alapján szorzódik a pontszám (közepes: 1,25, nehéz: 1,5), frissül a bejelentkezett felhasználó tapasztalat pontja, játszma száma, és személyes rekordja, ha nagyobb a pontszám az előzőnél, majd egy listában jelennek meg a játszma adatai.

Felépítése:

- Egy számláló, ami jelzi a jelenlegi és maradandó körök számát.
- A kérdés szövege.
- A kérdés képe.
- A választott mennyiségű (2, 4 vagy 6) válasz gombja, ami kattintás után annak helyessége szerint lesz kiszínezve (zöld, ha helyes, piros, ha helytelen). Válaszadás után kikapcsolnak.
- Egy visszaszámláló, ami jelzi a maradandó időt, és jobbról balra csökken a szélessége másodpercenként. Az animáció fél másodperc alatt történik.
- A “Folytatás” gomb, ami elindítja a következő kört. Az utolsó körben lecseréli a “Befejezés” gomb. Válaszadás után használható.

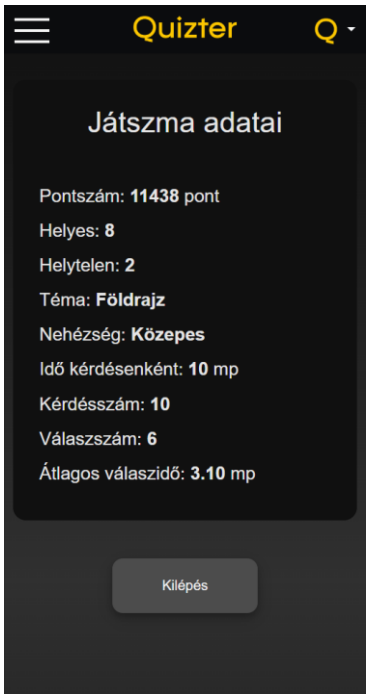


20. ábra: A játékmenet oldal mobilos nézete



21. ábra: A játékmenet oldal asztali nézete

A játszma végeztével egy lista jelenik meg, ami tartalmazza az adott játszma beállításait és a felhasználó teljesítményét. Ha a játékos új személyes rekordot döntött, akkor azt is kijelzi. Alatta egy gomb, ami visszanavigál a főoldalra.



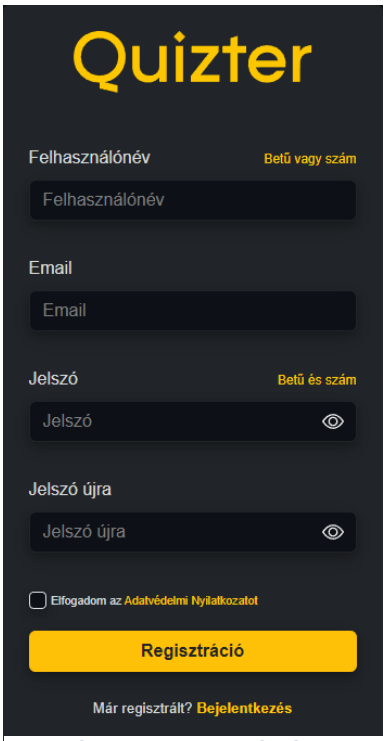
22. ábra: A játszma adatai lista

Regisztráció

A regisztráció oldalon hozható létre egy Quizter profil. A 4 beviteli mező reguláris kifejezés (regex) alapján vizsgálja a beírt tartalmat. Ha a felhasználónév vagy email foglalt, vagy nem egyezik az ismételt jelszó, akkor a beviteli mező alatt egy üzenet jelenik meg.

Felépítése:

- A felhasználónév beviteli mező, a beírt szöveg betű vagy szám lehet.
- Az email beviteli mező, a beírt szövegnek tartalmaznia kell a “@” és “.” karaktereket.
- A jelszó beviteli mező, a beírt szövegnek tartalmaznia kell betűt és számot.
- Az ismételt jelszó beviteli mező, a beírt szövegnek egyeznie kell a jelszóéval.
- Egy jelölőnégyzetet (checkbox), ami azt jelzi, hogy a felhasználó elfogadja az adatvédelmi nyilatkozatot.
- A “Regisztráció” gomb.



23. ábra: A regisztráció oldal asztali és mobilos nézete

Bejelentkezés

A bejelentkezés oldalon is vizsgálva van a felhasználónév és jelszó formátuma. Ezen felül, ha nem helyesek a bejelentkezési adatok, akkor a bemeneti mezők alatt megjelennek tájékoztató üzenetek.

24. ábra: A bejelentkezés oldal asztali és mobilos nézete

Felépítése:

- A felhasználónév beviteli mező, a beírt szöveg betű vagy szám lehet.
- Az “Elfelejtett jelszó?” link, ami előhoz egy felugró ablakot. Itt a felhasználó megadja az email címét, ahova a jelszó visszaállítással kapcsolatosan küldönk egy emailt.
- A jelszó beviteli mező, a beírt szövegnek tartalmaznia kell betűt, számot és speciális karaktert.
- A “Bejelentkezés” gomb.

Jelszó visszaállítása

A jelszó visszaállítása oldalon módosítható a felhasználó jelszava. Az előbb említett email ide irányítja át a felhasználót. Ha nem megfelelő az url-ben található jelszó változtatási kérelem tokenje, akkor az oldal nem tölt be, és a kérelem automatikusan elutasított.

25. ábra: A jelszó visszaállítása oldal asztali és mobilos nézete

Felépítése:

- A jelszó beviteli mező, a beírt szövegnek tartalmaznia kell betűt és számot.
- Az ismételt jelszó beviteli mező, a beírt szövegnek egyeznie kell a jelszóéval.
- A “Jelszó módosítása” gomb.

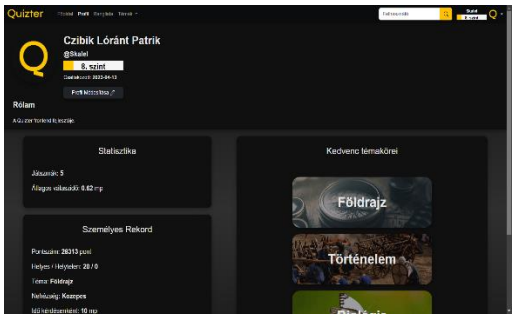
Profil

A profil oldalon tekinthetőek meg a felhasználók adatai. Ha a bejelentkezett felhasználó a saját profilját nyitja meg, akkor szerkeszthető a profilkép, név, leírás és a 3 beállítható téma.

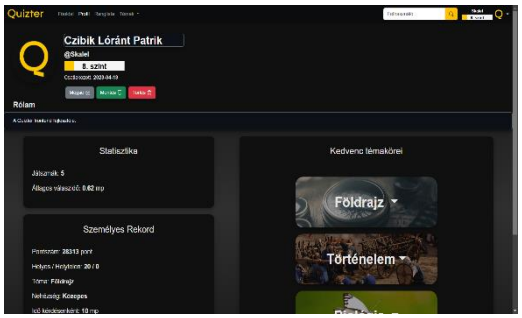
Felépítése:

- A profilkép.
- A név.
- A felhasználónév.
- A szint komponens.
- A csatlakozási dátum.
- A felhasználó leírása.
- Szerkesztés esetén három gomb: “Mégse”, “Mentés” és “Törlés”.
- A felhasználó statisztikája, ami tartalmazza azt, hogy mennyi játszmája van és az átlagos válaszidejét.
- A felhasználó személyes rekordja, ami tartalmazza a pontszámot, a helyes / helytelen válaszok számát, a témát, nehézséget, kérdésszámot éd válasszámot.
- A felhasználó által beállított 3 téma.

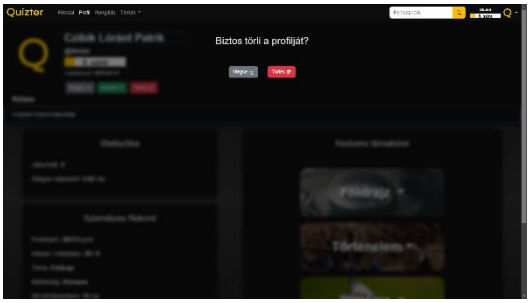
26. ábra: A profil oldal mobilos nézete



27. ábra: A profil oldal asztali nézete



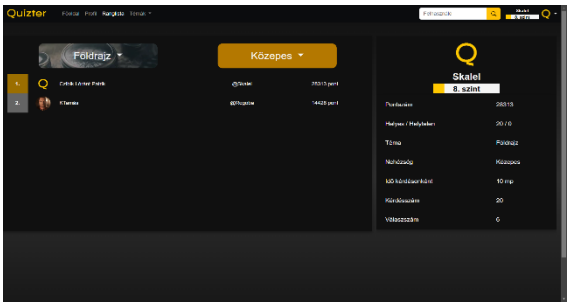
28. ábra: A módosítás opció kiválasztva



29. ábra: A törlés opció kiválasztva

Ranglista

A ranglista oldalon téma és nehézség szerint szűrhetőek a személyes rekordok. 2 részre van bontva, az elsőben a felhasználók eredményei, és a második részben pedig a választott rekord részletesebb leírása. Mobilos nézetben érintésre automatikusan legördül ehhez a részhez, és nem jelenik meg a név a ranglistán.



30. ábra: A ranglista oldal asztali nézete

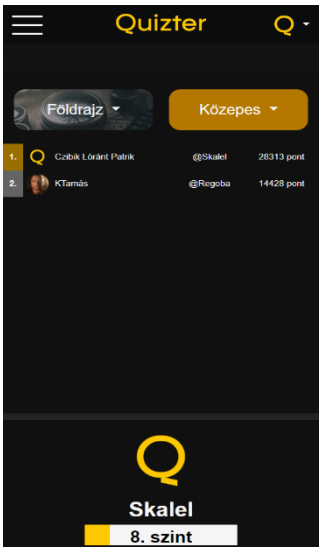
Felépítése:

1. <u>Ranglista:</u>

- A téma és nehézség szűrője, amit egy lenyíló menü segítségével lehet beállítani.
- A helyezés.
- A profilkép.
- A felhasználónév.
- A név.
- A pontszám.

2. <u>Rekord:</u>

- A profilkép, ami kattintásra átirányít az adott felhasználó oldalára.
- A felhasználónév.
- A szint komponens.
- A pontszám.
- A helyes / helytelen válaszok száma.
- A téma.
- A nehézség.



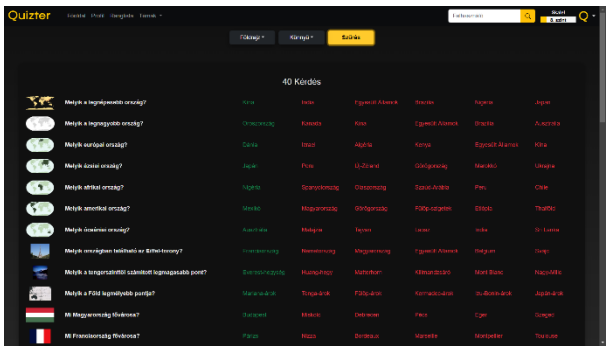
31. ábra: A ranglista oldal mobilos nézete

Kérdések

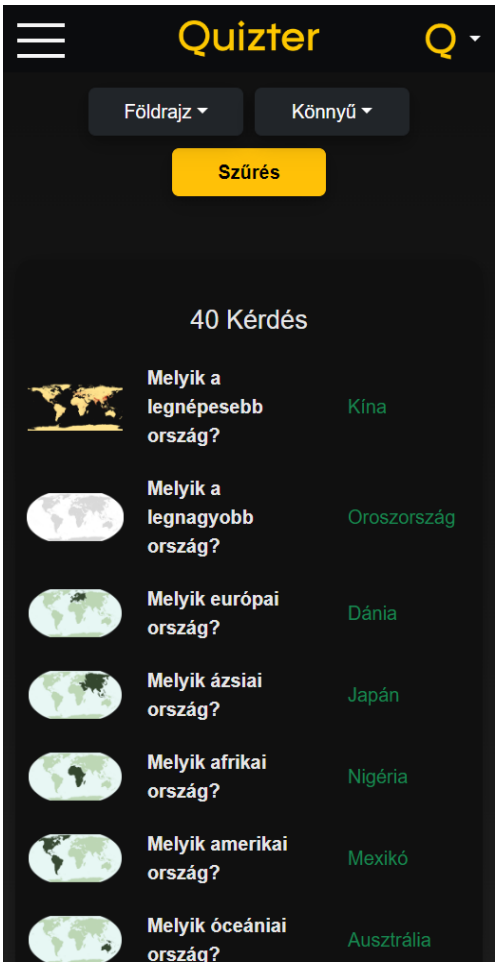
A kérdések oldalon jelennek meg a kérdések téma és nehézség szerint. Egy számláló visszaadja a kérdések számát. Ez mind egy táblázatban van listázva, aminek az egyik sorára kattintva megjelenik a kérdés oldalon a választott elem. Mobilos nézetben nem jelennek meg a kérdések helytelen válaszai.

Felépítése:

- A téma és nehézség lenyíló menü.
- A “Szűrés” gomb, ami ezek szerint listáz.
- A kérdés számláló.
- A kérdések képei, szövegei, 1 helyes és 5 helytelen válaszai.



32. ábra: A kérdések oldal asztali nézete



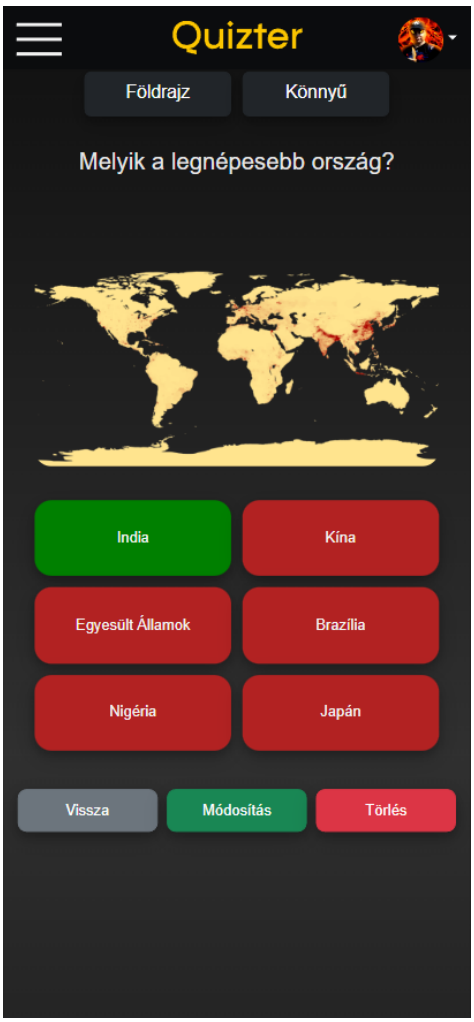
33. ábra: A kérdések oldal mobilos nézete

Kérdés

A kérdés oldalon jelenik meg az adott kérdés témája, nehézsége, szövege, képe és 6 válasza. A kérdés megtekinthető, módosítható, vagy törölhető.

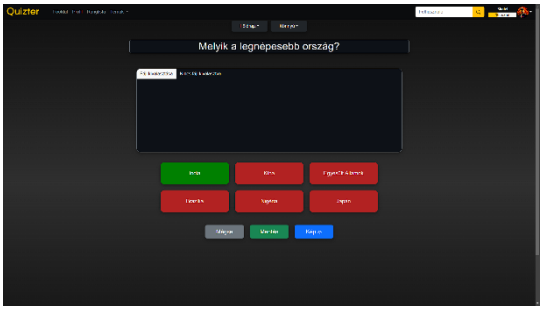
Felépítése:

- A téma és nehézség.
- A kérdés szövege.
- A kérdés képe.
- A kérdés 6 válasza.
- A “Vissza” gomb, ami a kérdések oldara navigál.
- A “Módosítás” gomb, ami a témát és nehézséget lenyíló menüvé, a kérdés szövegét, képét és válaszait pedig beviteli mezővé alakítja. Egy “Mégse”, “Mentés” és kép megjelenítő gombot is létrehoz.
- A “Törlés” gomb, ami a felhasználó megerősítése után eltávolítja a kérdést.

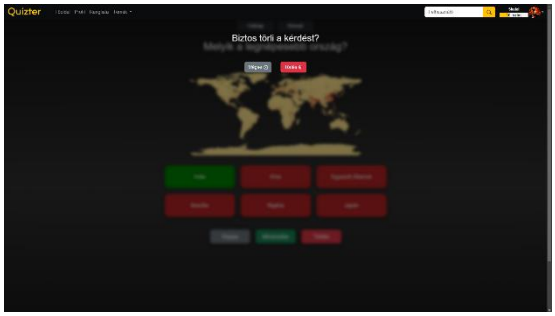


34. ábra: A kérdés oldal mobilos nézete

35. ábra: A kérdés oldal asztali nézete



36. ábra: A módosítás opció kiválasztva



37. ábra: A törlés opció kiválasztva

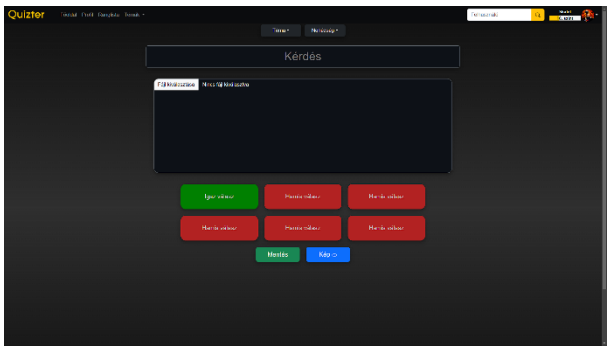
Új kérdés

Az új kérdés oldalon lehet létrehozni kérdéseket. Ehhez be kell állítani a kérdés témáját, nehézségét, szövegét, képét és 6 válaszát. A “Mentés” gomb lenyomásával egy vizsgálat történik, hogy be lett-e állítva a téma, nehézség és kép. Ha igen, akkor az adatok elküldésre kerülnek.

Felépítése:

- A téma és nehézség lenyíló menü.
- A kérdés szöveg beviteli mező.
- A kérdés kép beviteli mező.
- A 6 válasz beviteli mezője.
- A “Mentés” gomb.
- A kép megjelenítő gomb, ha a felhasználó megadott képet.

39. ábra: Az új kérdés oldal mobilos nézete



38. ábra: Az új kérdés oldal asztali nézete

40. ábra: A kép opció kiválasztva

Rólunk

A rólunk oldalon megjelennek a képeink, a nevünk és a szerep, amit a projektben betöltöttünk. Ezen kívül még egy carousel tartalmazza a fejlesztés néhány kezdetleges képeit. Az oldal alján pedig az email, amin keresztül el lehet minket érni.

Felépítése:

- A logó, alatta a “Rólunk” szöveg.
- A bemutatkozás szövege.
- A 3 kép, alatta a név és a betöltött szerep(ek).
- A “Galéria a fejlesztésről” szöveg, alatta a fejlesztés carousel.
- Az “Elérhetőség” szöveg, alatta az email.

41. ábra: A rólunk oldal asztali nézete

Adatvédelem

Az adatvédelem oldal a felhasználót tájékoztatja arról, hogy hogyan kerülnek kezelésre és tárolásra az adatai. Ezen kívül tartalmazza az emailt, amin keresztül el lehet érni minket.

42. ábra: Az adatvédelem oldal

Node.js – Backend

43. ábra: A Node.js logója [8]

A Node.js egy olyan futtató környezet, ami lehetőséget nyújt JavaScriptben írt programok futtatására szervereken. JavaScriptet leginkább a kliens oldalon szokták használni, a böngészőn keresztül. Könnyebbséget nyújt, ha már dolgoztunk frontend oldalon, hogy nem kell egy teljesen új nyelvet megtanulni ahhoz, hogy a backend-et is implementáljuk (Pl: Express.Js, Koa, Meteor.Js). A másik előnye az, hogy a non-blocking természetéből adódóan alkalmas egyszerre több ezer bejövő 'egyszerű' kérés kezelésére. Ezzel szemben egyetlen számításigényes utasítássorozat végrehajtására nem tűnik éppen ideális választásnak. Rengeteg fontos modullal rendelkezik, ezek többek között: HTTP, HTTPS, UDP, Net, DNS, TLS/SSL, URL, Query String, File System, Path, Events, Process, Child Processes, Buffers, Streams, Crypto. A listából látszik, hogy a Node.js igen alacsony szintű hozzáférést tesz lehetővé a szerver különböző rendszereihez, viszont magasabb szintű rendszereket önmagában nem kínál. Ehhez külső modulokat használhatunk, amik JavaScript és C++ komponensekből állhatnak (ahogy a beépített modulok is). [9]

Express.js

Ezt a keretrendszert úgy tervezték, hogy olyan funkciókkal rendelkezzen, amelyeket más keretek, például a **Rails** vagy a **Sinatra** ihlette. Ez azt jelenti, hogy amikor egy tapasztalt fejlesztő kezét rá helyezi, akkor kényelmesebbnek tartja a már ismert fogalmak kezelését, de a **Node.js** és Csak a JavaScript használatával megnövekszik a fejlesztési sebesség.
Az **Express.js** legfontosabb jellemzői:

1. **Minimalista**
   A keretrendszer mögött az a gondolat nem szerepel, hogy más kérdésekbe kerüljön, amelyek nem közvetíthető az alkalmazásunk és a szerver között, mert ez túlbonyolítja, csakis a legszükségesebb dolgokat biztosítja számunkra.

2. **Rugalmas**
   Minimális tulajdonsága miatt nagy rugalmasságot élvez, mivel a fejlesztő választhatja meg a munkavégzési módját, és megengedheti magának, hogy saját megoldásait megvalósítsa, vagy kész megoldásokat, például **ORM**-et építsen be különféle típusú adatbázisokkal.

3. **Alkalmazásokhoz készült**
   Az **Express.js** célja, hogy a webes alkalmazások, oldalak, akár **REST** szolgáltatások, akár hibrid alkalmazások készítésének megkönnyítése. [10]

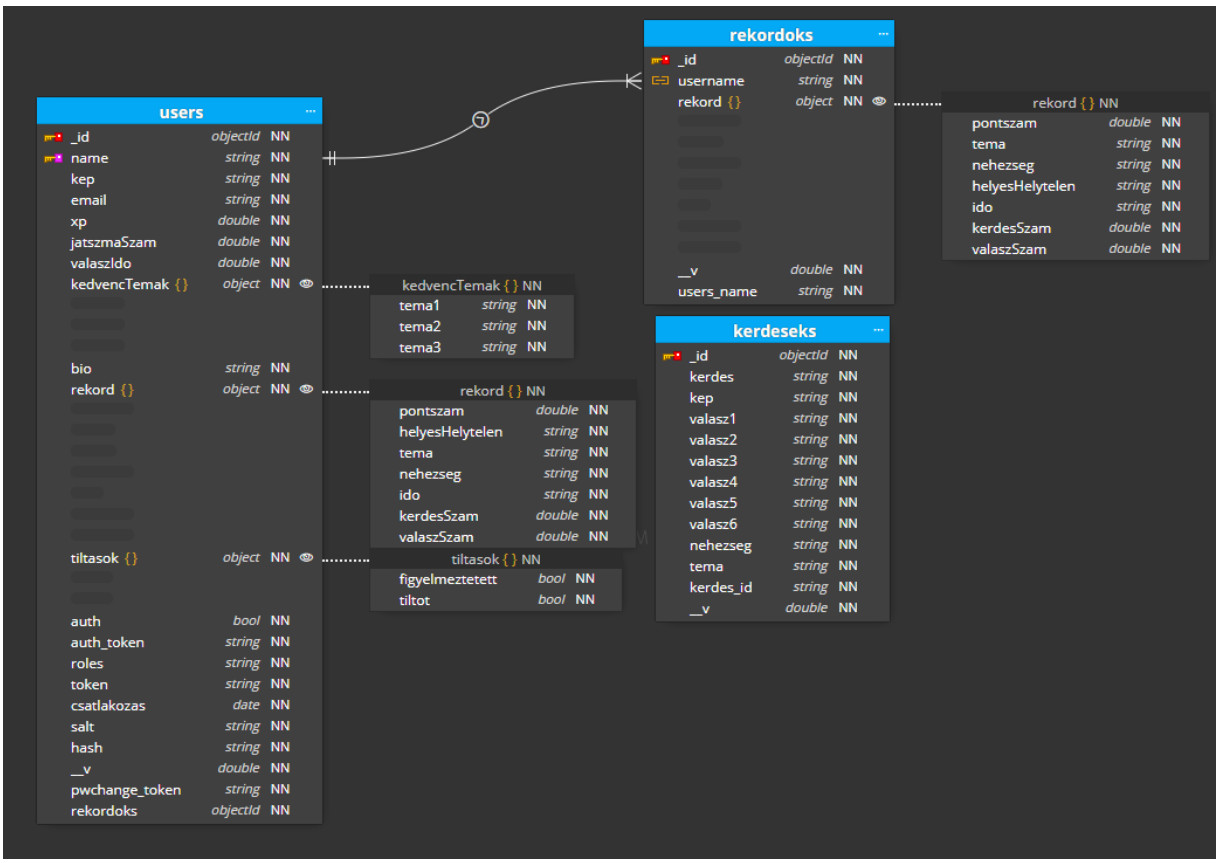
MongoDB – Adatbázis

A legnépszerűbb és legelterjedtebb dokumentum-orientált adatbázis rendszer, ami:

- No-SQL adatmodelre épül
- JSON vagy BSON formában tárolja az adatokat
- nagy performancia jellemzi
- nagy skálázhatóság
- nincs SQL injection

44. ábra: A mongoDB logója [11]

A NoSQL (Not only SQL) adatbázisok népszerűsége az utóbbi években kezdett meredeken emelkedni. A relációs adatbázisoktól eltérő megközelítés fő céljai között említhetjük az óriási mennyiségű adatok feldolgozását (bigdata), a skálázhatóságot és a valós idejű rendszereket. Egyre több fejlesztési projekt alapjául szolgál NoSQL adatbázis és az újszerű felépítés új megközelítést igényel a fejlesztőktől és a felhasználóktól. [12]

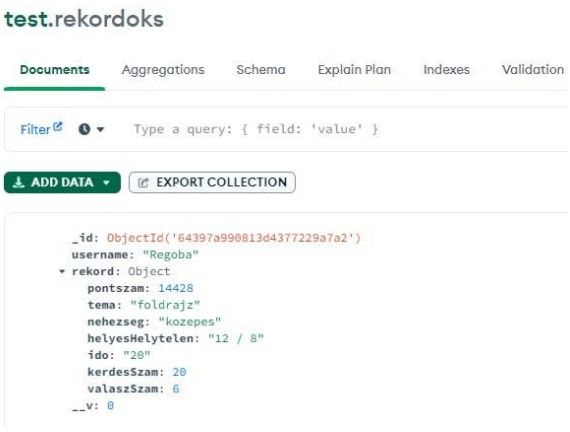


45. ábra: Az adatbázis felépítése

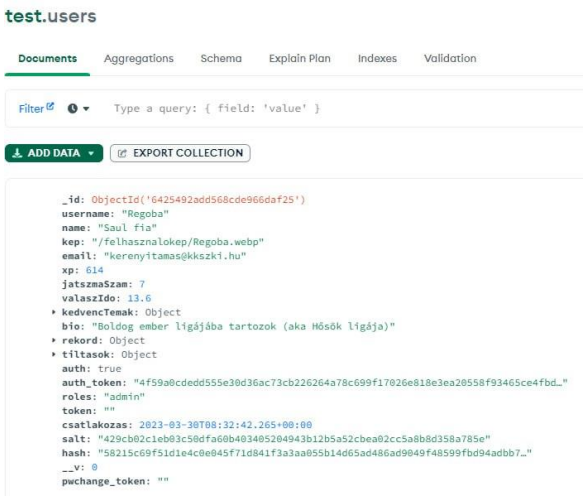
MongoDB schéma

Az adatbázisban 3 különböző collection létezik, mind a három független egymástól viszont egy célt szolgálnak, hogy az adatok épp és naprakészek legyenek.

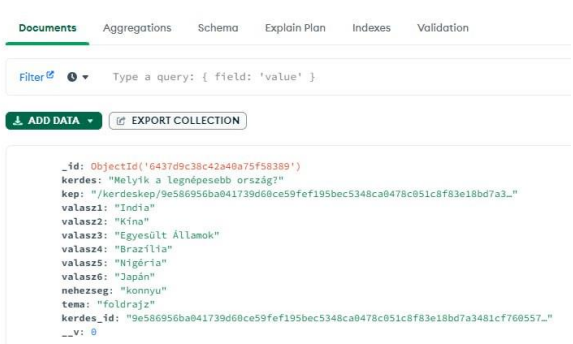
46. ábra: A schémák



47. ábra: A rekordoks schéma



48. ábra: A users schéma



49. ábra: A kerdeseks schéma

Az adatbázisban közvetlen nincsenek kulcsok meghatározva, sőt NoSQL adatbázisban összeférhetetlen, hogy relációs kapcsolatok legyenek a kollekciók közt,
ez első sorban azért van mert a MongoDB skálázás elvét követi, mi az összes kollekciót (próbálja) függetlenné tenni, amikor nagyobb változásokat szeretnénk beiktatni a meglévő projektünkbe, vagy csak simán szeretnék törölni egy meglévő kollekciót, és újratervezni a semmiből azt. Viszont felkínálja annak a lehetőségét az 'aggregation pipeline'-ban, hogy a lekérdezésben összekötéseket alkossunk, amivel könnyen lehet dolgozni.

A backend és a frontend kapcsolata

A frontend kötött a backend szerverhez, így külön kategorizálva vannak a backend-en, hogy egyszerűsítse a navigálást a forráskódban. A backend fontosabb végpontjai:

Ranglista:

- Téma és nehézség szerint visszaad rekordokat, pontszám szerint csökkenő sorrendben.
- Új vagy meglévő rekordok kezelése adatbázisban.

```
exports.getLeaderboard = ((req, res) => {
  Rekordok.aggregate([{ $match: { "rekord.tema": `${req.params.tema}`, "rekord.nehezseg": `${req.params.nehezseg}` } },
  { $sort: { "rekord.pontszam": -1 } },
  { $lookup: { from: "users", localField: "username", foreignField: "username", as: "felhasznalo" } },
  { $project: { "_id": 0, "felhasznalo._id": 0, "felhasznalo.auth_token": 0, "felhasznalo.salt": 0, "felhasznalo.hash": 0, } }])
    .exec(function (err, ranglista) {
      let valasz = {}
      ranglista.forEach(function (userlist, index) {
        valasz[`felhasznalo${index + 1}`] = {
          felhasznalonev: userlist.felhasznalo[0].username,
          kep: `${process.env.HOST_URL}:${process.env.PORT || 2000}${userlist.felhasznalo[0].kep}`,
          nev: userlist.felhasznalo[0].name,
          exp: userlist.felhasznalo[0].xp,
          rekord: {
            pontszam: userlist.rekord.pontszam,
            helyesHelytelen: userlist.rekord.helyesHelytelen,
            tema: userlist.rekord.tema,
            nehezseg: userlist.rekord.nehezseg,
            ido: userlist.rekord.ido,
            kerdesSzam: userlist.rekord.kerdesSzam,
            valaszSzam: userlist.rekord.valaszSzam,
          },
        }
      })
      res.send(valasz);
    })
});
```

50. ábra: A getLeaderboard végpont

Felhasználó:

- Profil személyre szabása.
- Statisztika, személyes rekord nyilvántartás.

```
exports.updateUserPage = (async (req, res) => {
  const filter = { _id: `${req.user._id}` };
  let update = {
    kep: "placeholder",
    name: req.body.nev,
    bio: req.body.bio,
    kedvencTemak: {
      tema1: req.body.tema1,
      tema2: req.body.tema2,
      tema3: req.body.tema3,
    }
  }
  await User.findOne({ _id: `${req.user._id}` }).then(felhasznal => {
    if (felhasznal.kep == "/felhasznalokep/default-user.webp" && req.file) update.kep = req.file.path.substring(5);
    else update.kep = felhasznal.kep;
  }).catch();
  await User.updateOne(filter, update).then(res.send("Sikeres update!")).catch();
});

exports.deleteUser = ((req, res, next) => {
  User.findOneAndRemove({ _id: `${req.user._id}` }).then(res.send("Sikeres törlés!")).catch(res.send("Sikertelen törlés!"))
});
```

51. ábra: A felhasználói végpontok kódrészlete

Game:

- Adott mennyiségű kérdés lekérdezése.
- Statisztika mentése, a felhasználói tapasztalatpont (xp), játszamászám, vagy személyes rekord frissítése.

```
exports.KerdesValasz = (async (req, res) => {
  if (!['2', '4', '6'].includes(req.params.valaszszam)) { req.params.valaszszam = 6 }
  if (!['10', '15', '20'].includes(req.params.kerdesszam)) { req.params.kerdesszam = 10 }
  if (!['autok', 'biologia', 'fizika', 'foldrajz', 'irodalom', 'kemia', 'sport', 'szorakoztatas', 'technologia', 'tortenelem', 'zene', 'vegyes']
    .includes(req.params.tema)) { req.params.tema = "foldrajz" }
  if (!['konnyu', 'kozepes', 'nehez'].includes(req.params.nehezseg)) { req.params.nehezseg = "konnyu" }

  if (req.params.tema != 'vegyes') Kerdesek.aggregate([{ '$match': { tema: `${req.params.tema}`, nehezseg: `${req.params.nehezseg}` } },
  { '$sample': { size: parseInt(req.params.kerdesszam) } },
  { '$project': { _id: 0 } }],).exec(async function (err, kerdes) {
    NoLoginKerdesFormazas(req, res, kerdes)
  })

  else
    Kerdesek.aggregate([{ '$match': { nehezseg: `${req.params.nehezseg}` } },
    { '$sample': { size: parseInt(req.params.kerdesszam) } },
    { '$project': { _id: 0 } }],).exec(async function (err, kerdes) {
      NoLoginKerdesFormazas(req, res, kerdes)
    })
});
```

52. ábra: Az egyik játszmával kapcsolatos végpont

Auth:

- Felhasználó authentikálása, Salt + Hash egyeztetése a beírt jelszóval.
- Felhasználói jogkörök vizsgálata.
- Regisztráció & Elfelejtett jelszó visszaállítás email segítségével.

```
exports.checkAuthentication = ((req, res, next) => {
  if (req.headers.authorization) {
    const token = req.headers.authorization.split(' ')[1]
    User.findOne({ auth_token: `${token}` }).exec(function (err, user) { if (!user || !user.auth) { res.sendStatus(401) } else { req.user = user; next() } });
  }
  else res.sendStatus(401);
});

exports.checkAuthForGameSession = ((req, res, next) => {
  if (req.headers.authorization) {
    const token = req.headers.authorization.split(' ')[1]
    User.findOne({ auth_token: `${token}` }).exec(function (err, user) { if (!user || !user.auth) { res.sendStatus(401) } else { req.user = user; next() } });
  }
  else next();
});

exports.isAdmin = ((req, res, next) => {
    if (req.user.roles == 'admin') { next(); }
    else res.sendStatus(401);
});
```

53. ábra: Az autentikációs végpontok kódrészlete

Middleware

A backenden meghatározott eljárások vannak arra, hogy hitelesítse, illetve megvizsgálja a fő eljárás előtt, hogy ott a hibakezelés már rugalmasan megtörténjen.
Vegyük példának a gyakran megtalálható 'checkAuthentication'-t a végpontok eljárásaiban:

```
//User Rekord feltöltése & babrálása
app.patch('/updateUserRecord', checkAuthentication, UpdateRecord);
//Kérdés felvétele, req.body.kerdes alapján vizsgál, ha talál hasonló szövegű kérdést nem viszi fel
app.post('/createQuestion', checkAuthentication, isAdmin, kerdeskep.single(`file`), getQuestionImage, createQuestion);
//Kérdés update, képet is lehet benne változtatni gyönyörűen (temp fájllal cseréljük a meglévőt)
app.patch('/updateQuestion', checkAuthentication, isAdmin, kerdeskep.single(`file`), updateQuestion);
//Kérdés törlés, note: Ne vedd ki a kerdeskep eljárást vagy pofánszarja a req.body-ban lévő formod a túl sok adat mani
app.delete('/deleteQuestion/:k_id', checkAuthentication, isAdmin, deleteQuestion);
```

54. ábra: Autentikáció a védett végpontokon

Elsődleges célja, hogy hitelesítse minden API lekérdezésnél a felhasználónál tartott tokent, viszont a másodlagos célja, hogy a Backend közvetlen tudjon dolgozni a legfrissebb felhasználói adatokkal és azt forgatva dolgozzon a MongoDB-ben vagy más eljárásban.

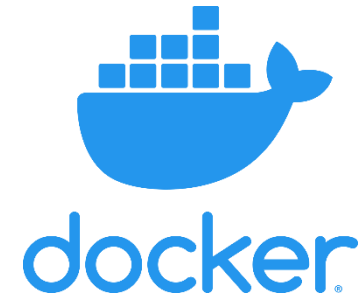
A következő kódrészlet a regisztrációnál, illetve az elfelejtett jelszónál érvényesül, a form-ból kinyeri az adatokat és reguláris kifejezésekkel hitelesíti, majd az előbbieket felsorolva máris hibakezeléssel indul, és hibát küld vissza, ha probléma van:

```
const { check } = require('express-validator');
//Regisztrációnál vizsgálja a bekért adatokat, ha ezeknek a feltételeknek nem felelnek meg akkor ezen hibaüzeneteivel visszadobja.
exports.RegisterValidation = (
[
    check('username').isLength({ min: 3, max:12 }).withMessage("Név legalább 3 karakterből és maximum 12 karakterből áljon!").matches(/^(?=.{3,12}$)(?![_.])(?!.*[_.]{2})[a-zA-Z0-9._]+(
    check('email').isEmail().withMessage("Nem email").normalizeEmail(),
    check('password').isLength({min: 8}).withMessage("Fos a jelszó, 8 karaktert").matches(/\d/).withMessage("Jelszóba legyen 1 szám"),
    check('password2').custom((value, { req, res }) => {if(value !== req.body.password){return false;} return true;}).withMessage("Nem azonos jelszó")
]);
exports.ChangePWValidation = ([
    check('password').isLength({min: 8}).withMessage("Fos a jelszó, 8 karaktert").matches(/\d/).withMessage("Jelszóba legyen 1 szám"),
    check('password2').custom((value, { req, res }) => {if(value !== req.body.password){return false;} return true;}).withMessage("Nem azonos jelszó")
])
```

55. ábra: Küldött adatok vizsgálata reguláris kifejezésekkel

Docker

Mi az a Docker?



56. ábra: A Docker logója [13]

A Docker egy platform és eszközkészlete, amellyel a fejlesztők, rendszer adminisztrátorok könnyedén tudják telepíteni, kihelyezni (deploy) az alkalmazásaikat egy homokozóba (sandbox, mint konténer), hogy ott futtassák őket a gazda gép operációs rendszerén, például egy Linux-on. A kulcsfontosságú nyereség a Docker használatával az az, hogy ez engedi a felhasználóknak, hogy becsomagoljanak egy alkalmazást annak minden egyes függőségével együtt egy szabványosított egységbe. Nem úgy, mint a virtuális gépek, a konténerek nem igényelnek túlzottan sok erőforrást, így még hatékonyabban tudja felhasználni az „alatta lévő” rendszer erőforrásait.

Mik azok a konténerek?

Összehasonlítva őket a virtuális gépekkel, amelyeknél a gazda gép operációs rendszere és a gazda gép erőforrásait használja a „vendég” virtualizált operációs rendszer, amely nagyon sok erőforrást igényel az ő saját működéséhez. Ezzel szemben a konténerek más megközelítést alkalmaznak: a konténerek a gazda operációs rendszer alacsony szintű mechanikáját kihasználva a virtuális gépek elszigeteltségének nagy részét biztosítják, a számítási igény és teljesítmény töredéke mellett.

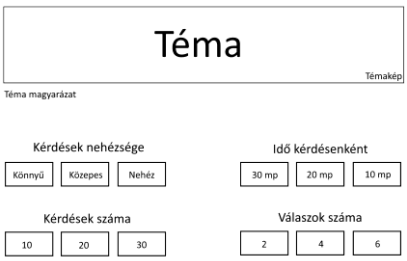
Miért használunk konténereket?

A konténerek olyan logikai csomagolási mechanizmust kínálnak, amelyben az alkalmazások elvonatkoztathatók attól a környezettől, amelyben ténylegesen futnak. Ez a szétválasztás lehetővé teszi, hogy a konténer alapú alkalmazások könnyen és következetesen telepíthetők legyenek, függetlenül attól, hogy a célkörnyezet egy privát adatközpont, a nyilvános felhő vagy akár egy fejlesztő személyes laptopja. A fejlesztők így kiszámítható (értsd: mindig mindenhol nagyjából ugyanannyi erőforrást használó) környezeteket hozhatnak létre, amelyek elszigeteltek a többi alkalmazástól, és bárhol futtathatók.
Üzemeltetési szempontból a hordozhatóság mellett a konténerek az erőforrások pontosabb ellenőrzését is lehetővé teszik, így az infrastruktúra hatékonyabbá válik, ami a számítási erőforrások jobb kihasználását eredményezheti.
Ezen előnyök miatt a konténerek (és a Docker) széles körben elterjedtek. Az olyan vállalatok, mint a Google, a Facebook, a Netflix és a Salesforce a konténereket a nagy mérnöki csapatok produktívabbá tételére és a számítási erőforrások jobb kihasználására használják. [14]

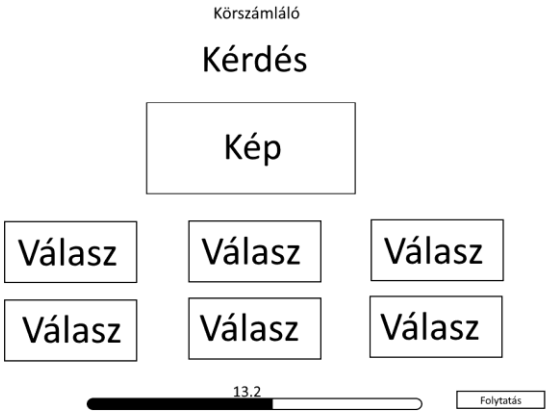
VÁZ-SZERKEZET RAJZOK

Az alábbi váz-szerkezet rajzok az oldal kezdetleges kinézetét demonstrálják. A fejlesztés során néhol módosultak az oldalak kinézetei.

57. ábra: A főoldal váz-szerkezete

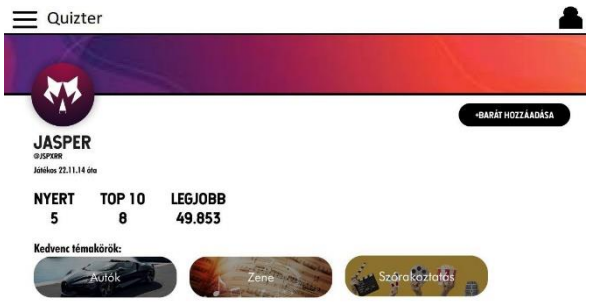


58. ábra: A kvíz beállító oldal váz-szerkezete

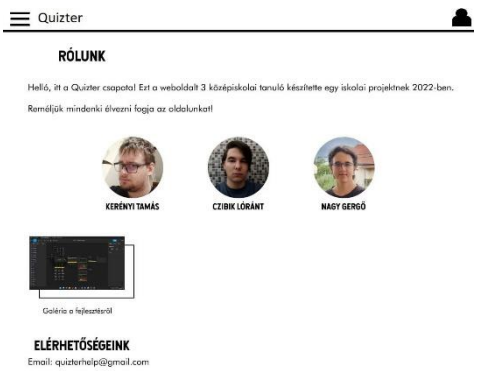


59. ábra: A játékmenet oldal váz-szerkezete

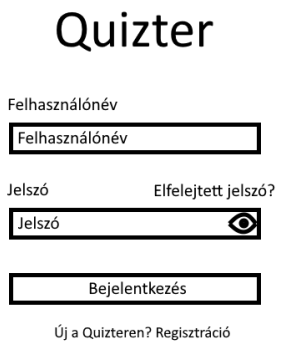
60. ábra: A ranglista oldal váz-szerkezete



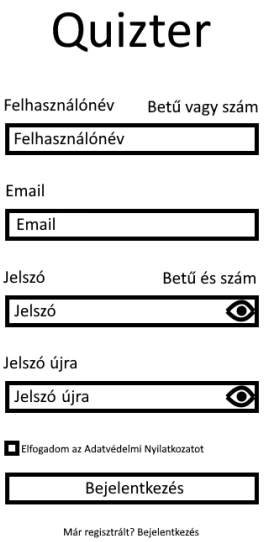
61. ábra: A profil oldal váz-szerkezete



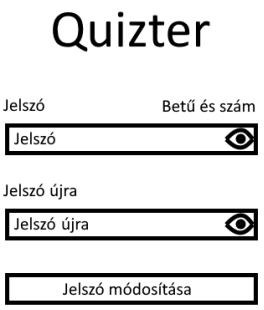
62. ábra: A rólunk oldal váz-szerkezete



63. ábra: A bejelentkezés oldal váz-szerkezete



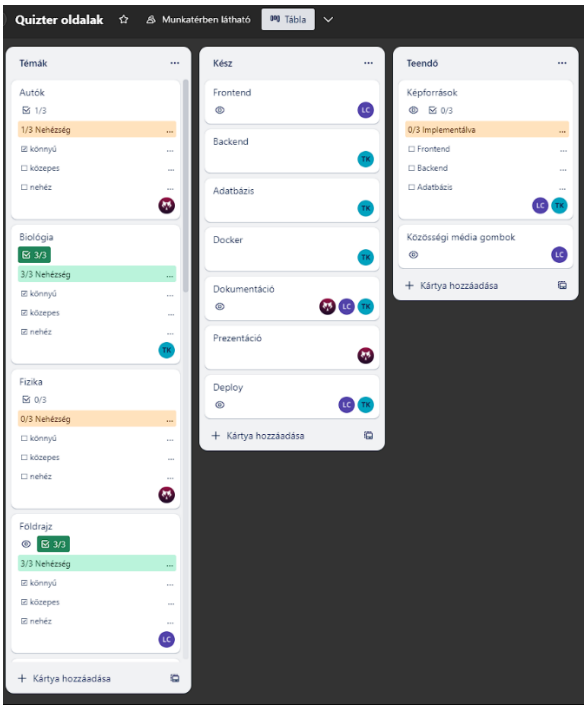
64. ábra: A regisztráció oldal váz-szerkezete



65. ábra: A jelszó visszaállítása oldal váz-szerkezete

KOLLABORÁCIÓS SZOFTVER

A projekt tervezéséhez a **Trello** listakészítő alkalmazást használtuk. Az első lista felsorolja az adott témákat, és azokat nehézség jelölőnégyzetekre bontja. A második lista a kész, és a harmadik lista a teendő dolgokat jelzi. Minden feladat ki van osztva a tagok számára.



66. ábra: A Trello listák

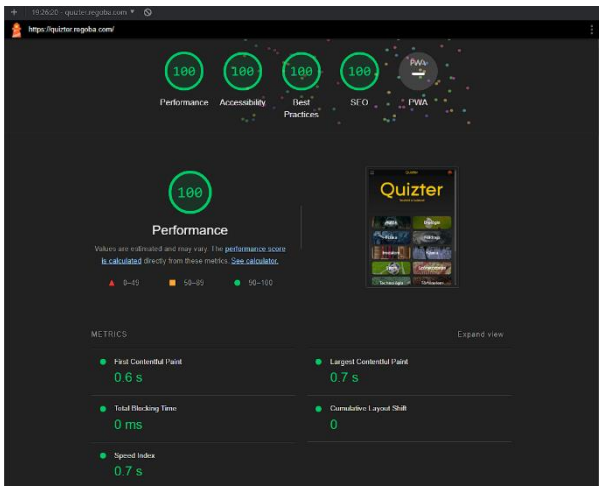
TESZTELÉS

Lighthouse

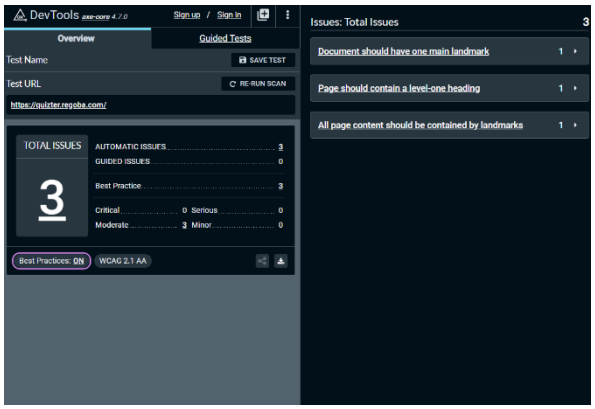
A Lighthouse egy nyílt forráskódú, automatizált eszköz a weboldalak minőségének javítására. Bármilyen weboldallal szemben futtatható, legyen az nyilvános vagy hitelesítést igénylő. Rendelkezik teljesítményre, hozzáférhetőségre, progresszív webes alkalmazásokra, SEO-ra és még sok másra vonatkozó ellenőrzésekkel. [15]

Axe Devtools

Az **axe DevTools** egy automatizált és irányított akadálymentesítési tesztelési megoldás komponensfejlesztők, frontend fejlesztők, natív mobilalkalmazás-fejlesztők és tesztmérnökök számára, amely lehetővé teszi a kisegítő lehetőségek 76-84%-ának egyszerű megtalálását és kijavítását, mielőtt alkalmazásai kikerülnének a fejlesztésből. [16]



67. ábra: Lighthouse teszt



68. ábra: Axe Devtools teszt

FORRÁSKÓD

A szakdolgozat alapját képező alkalmazás forráskódja, valamint jelen szakdolgozat a csapat közös GitHub repositoryjában található, a következő elérhetőség alatt:

https://github.com/CLorant/quizter

FORRÁSJEGYZÉK

[1] Vue logó
Letöltés dátuma: 2023. 04. 19.
https://upload.wikimedia.org/wikipedia/commons/thumb/9/95/Vue.js_Logo_2.svg/1024px-Vue.js_Logo_2.svg.png

[2] Introduction | Vue.js
Letöltés dátuma: 2023. 04. 20.
https://vuejs.org/guide/introduction.html

[3] HTML5 logó
Letöltés dátuma: 2023. 04. 19.
https://en.wikipedia.org/wiki/File:HTML5_logo_and_wordmark.svg

[4] CSS logó
Letöltés dátuma: 2023. 04. 19.
https://en.wikipedia.org/wiki/File:CSS3_logo_and_wordmark.svg

[5] Javascript logó
Letöltés dátuma: 2023. 04. 19.
https://upload.wikimedia.org/wikipedia/commons/thumb/3/3b/Javascript_Logo.png/640px-Javascript_Logo.png

[6] Pinia logó
Letöltés dátuma: 2023. 04. 19.
https://en.wikipedia.org/wiki/Pinia#/media/File:Pinialogo.svg

[7] Introduction | Pinia
Letöltés dátuma: 2023. 04. 20.
https://pinia.vuejs.org/introduction.html

[8] Node.js logó
Letöltés dátuma: 2023. 04. 21.
https://upload.wikimedia.org/wikipedia/commons/thumb/d/d9/Node.js_logo.svg/640px-Node.js_logo.svg.png

[9] Node.js
Letöltés dátuma: 2023. 04. 20.
http://www.inf.u-szeged.hu/~tarib/javascript/nodejs.html#irodalomjegyzek

[10] Megkezdjük az Express.Js használatát
Letöltés dátuma: 2023. 04. 20.
https://hu.admininfo.info/empezando-trabajar-con-express

[11] MongoDB logó
Letöltés dátuma: 2023. 04. 21.
https://upload.wikimedia.org/wikipedia/commons/thumb/9/93/MongoDB_Logo.svg/640px-MongoDB_Logo.svg.png

[12] No-SQL – MongoDB adatbáziskezelő
Letöltés dátuma: 2023. 04. 20.
https://masterfield.hu/hu/tanfolyamok/nosql-mongodb-adatbaziskezelo-tanfolyam

[13] Docker logó
Letöltés dátuma: 2023. 04. 29.
https://www.docker.com/wp-content/uploads/2022/03/vertical-logo-monochromatic.png

[14] Docker - 1. rész: Alapok, telepítés, első használatba vétel
Letöltés dátuma: 2023. 04. 20.
https://attila.gludovatz.hu/posts/docker-1-resz-alapok-telepites-elso-hasznalatba-vetel

[15] Lighthouse overview - Chrome Developers
Letöltés dátuma: 2023. 05. 08.
https://developer.chrome.com/docs/lighthouse/overview/

[16] Deque Product User Guides
Letöltés dátuma: 2023. 05. 08.
https://dequeuniversity.com/guide/