

VMSZC Nádasdy Tamás Technikum és Kollégium

Szoftverfejlesztő

54 213 05

SZAKDOLGOZAT

ReactGame (webalkalmazás)

Konzulens:

Balics Gábor

Készítette:

Károlyi Krisztián, 13/A

Csepreg, 2021

NYILATKOZAT

Alulírott Károlyi Krisztián kijelentem, hogy ez a szakdolgozat a saját kutatásom, önálló munkám terméke, a más forrásból átvett szó szerinti vagy tartalmi idézést a szabályoknak megfelelően jelöltem.

Nyilatkozatommal hozzájárulok ahhoz, hogy az iskola szakdolgozatomat – a jogvédelemszerzés kivételével – felhasználhatja.

Kijelentem, hogy a szakdolgozatom bekötöttség és elektronikus formában leadott példányai mind formátumban, mind tartalomban egyezők, eltérést nem tartalmaznak.

.....

Dátum

.....

Vizsgáló aláírása

Tartalom

| | |
|---|----------|
| Bevezetés..... | 3 |
| 1. Játékprogramok | 4 |
| 1.1 Hasonló rendszerek jellemzése..... | 4 |
| 1.1.1 Futtató motorok | 4 |
| 1.2 Design és grafikai elemek..... | 5 |
| 2. ReactGame | 6 |
| 2.1 Felhasznált programozási nyelvek, keretrendszerek | 6 |
| 2.1.1 HTML..... | 6 |
| 2.1.2 CSS..... | 6 |
| 2.1.3 Bootstrap | 7 |
| 2.1.4 JavaScript | 7 |
| 2.1.4.2 jQuery..... | 8 |
| 2.1.5 PHP..... | 8 |
| 2.1.6 Laravel..... | 9 |
| 2.2 A felhasználói felület működése..... | 10 |
| 2.2.1 A program elindítása, elérése | 10 |
| 2.2.2 Regisztráció | 10 |
| 2.2.3 Bejelentkezés | 11 |
| 2.2.4 Kezdőlap és menü..... | 12 |
| 2.2.5 Profil, és annak szerkesztése..... | 13 |
| 2.2.6 Eredményeim..... | 14 |
| 2.2.7 Új játék | 14 |
| 2.2.8 Barátok | 15 |
| 2.2.9 Ranglista..... | 15 |
| 2.2 Felhasználók jelentése, blokkolt felhasználók..... | 16 |
| 2.3 Az adatbázis..... | 17 |
| 2.3.1 Tervezési szint..... | 17 |
| 2.3.2 Fizikai szint (implementálás)..... | 19 |
| 2.3.3 Migrációk és seederek a Laravelben..... | 20 |
| 2.4 A szerver működése..... | 21 |

| | | |
|-----------|---|-----------|
| 2.4.1 | Az MVC | 21 |
| 2.4.2 | Blade..... | 22 |
| 2.4.3 | A szerver működése, a program beüzemelése | 23 |
| 3. | Tesztelés..... | 26 |
| | Összegzés | 27 |
| | Bibliográfia..... | 28 |
| | Ábrajegyzék | 29 |
| | Melléklet | 30 |

Bevezetés

Szakedolgozatom témájául a reakcióidőt választottam, mivel úgy gondolom, ez egy nagyon fontos dolog a hétköznapi életben, például a közlekedésben, játékokban, sportban, vagy akár életveszélyes szituációkba keveredhetünk az utcán is.

De mi is az a reakcióidő? A legtöbb meghatározás szerint a reakcióidő a minket (agynkat) érő inger és a rá adott reakció közt eltelt időt jelenti. Természetesen ezt rengeteg tényező befolyásolja, hogy mennyi idő alatt tudunk lereagálni egyes helyzeteket. Például nem mindegy, hogy idős vagy fiatal emberről beszélünk, kipihent állapotban van-e az illető, van-e állapotára kiható anyag a szervezetében (alkohol, gyógyszer, drog, vagy bármilyen tudatmódosító tulajdonsággal rendelkező anyag). Ezeken kívül még sok dolog befolyásolja, de most nem kifejezetten ezzel foglalkozom a programommal, hanem inkább a reakcióidő mérésével, s ezzel létrehozva egy saját többfelhasználós, webalapú játék platformot.

Mióta az internet szinte bárki számára elérhető a világon, nagyon népszerűek lettek az online (többfelhasználós) játékok. Jómagam is szeretek időt tölteni velük, noha mióta programozást tanulok, már más szemmel látom ezeket a dolgokat. Vannak böngészőben futtathatóak (mobilról és asztali számítógépről egyaránt), például az én szakedolgozatomhoz készült játék, valamint vannak olyanok, amikhez telepíteni kell klienst. Ennek megvannak az előnyei és a hátrányai egyaránt. Például a platformfüggetlenség csak úgy oldható meg így, ha külön készítenek operációs rendszerekre (pl. Android, Windows IOS) klienst. Viszont „nagyobb” játékoknál (pl. League of Legends) nem ajánlott böngészőben futtatni a játékot, mert az erőforrásigényesebb, rengeteg hibát okozhat, és a böngészős játékoknak megvannak a határai. (Nagyobb játékok külön keretrendszereket, drivereket igényelnek a hardver kihasználása miatt).

Tehát azért választottam ezt a témát, mivel bárki lemérheti a saját reakcióidejét, és természetesen gyakorolhatja azon a játékon, amelyikben fejlődnie kell a statisztikákhoz mérten. Persze sok hasonló játék van az interneten, de olyat nem láttam, ahol nyilvántartásban vannak a játékosok eredményei, ami alapján ranglistákat mutat a barátok között. Mivel ez egyszerűbb, kevés erőforrást igénylő játéknak számít, ezért tökéletes megoldásnak láttam a böngészős megoldást. Ennek működéséről a továbbiakban részletesen fogok írni.

A programomat mindenkinek ajánlanám értelemszerűen, nem árt, ha tornáztatjuk vele agynkat. A játék nem kerül sok erőforrásba, ezért természetesen ingyenes felhasználásra szántam.

1. Játékprogramok

Az interneten rengetegféle játék létezik. Én azért találok a saját elképzelésem érdekesnek, mivel sok olyan játékot ismerek, ahol barátokat lehet jelölni és a játékosok eredményei alapján folyamatosan frissülő ranglista van (pl. Subway Surfers), és rengeteg hasonló tartalmú program létezik, ahol a reakcióidőt mérik fel, de a kettőt „egybegyúrva” még nem nagyon láttam. Ezért jó ötletnek gondoltam ezt megvalósítani, hasznos unaloműző játék is lehet. Igyekeztem egyedivé varázsolni a játékot, bár néhol nehézséget okozott, hogy maga a játék lényege mindenhol ugyanaz végeredményben.

1.1 Hasonló rendszerek jellemzése

Felhasználói szemmel nagyon nehéz lenne több részre tagolni a játékot. Az én véleményem szerint azonban a barátok és maga a programban játszható játékok egymástól függetlenek, habár a baráti listának csak az eredmények láthatósága miatt van létjogosultsága nálam. Másrészt, a felhasználói felület, amit a program a játékosok elé tár (frontend) ugyan megjelenne, de nem csinálna semmit a programot futtató motor (nálam a szerver) nélkül. Mi is a motorja a programomnak?

1.1.1 Futtató motorok

A rendszer fontos központi része a szerver (kiszolgáló), aminek folyamatosan működésben kell lennie, rendelkezésre kell állnia ahhoz, hogy a játékosok tudjanak játszani. De ez minden (webalapú) programhoz elengedhetetlen, nemcsak a játékoknál. Például egy webshop is hasonlóan működik. Az sem árt, ha a szervert futtató számítógép megfelelő hardverrel rendelkezik, ugyanis a gyenge gépet hamar túl lehet terhelni, ha egyszerre túl sok felhasználó szeretné elérni az oldalunkat, nem beszélve a DDOS-támadás elleni védelem fontosságáról.

Általában PHP szervert használ a legtöbb ilyen program, ami feldolgozza az oldal felé küldött kéréseket. A PHP-ről és a többi webszerver-komponensről a „felhasznált programozási nyelvek” című fejezetben írok részletesen. Viszont nem árt figyelni arra, hogy ennek is vannak sebezhetőségei, noha folyamatosan fejlesztik és adják ki az újabb PHP verziókat. Ugyanakkor ajánlanám minden kezdőnek vagy haladónak egyaránt, ugyanis egy nyitott és könnyen megtanulható, objektumorientált programozási nyelvről van szó. A legtöbb weboldal ezt használja.

Azonban maga a program motorja több komponensből áll. Nálam egy MySQL-adatbázis tárolja az összes adatot, amivel a PHP kommunikál. A webszerver, ami elérhetővé teszi gépünket a beállított hálózaton a gép IP-címével és a beállított port segítségével, nem más, mint az Apache (ezenkívül még ismert az EnginX is), ami egy ingyenes webszerver, több operációs

rendszer is támogatja. Lényegében ezek a legfontosabbak egy hasonló webalkalmazás futtatásához. Én a Laragon nevű program segítségével állítottam be ezeket, mert nagyon könnyen beszerezzük egy lépésben ezeket a komponenseket, és be sem kell konfigurálni. Tehát a PHP értelmez, feldolgoz, az Apache felel maga a webservert működéséért, míg a MySQL az adatbázist tartalmazza.

1.2 Design és grafikai elemek

A kinézetre/felhasználói élményre általában nem nagyon adnak az ilyen egyszerű játékoknál, azonban én igyekeztem a modern „material design”, illetve a „flat style” tematikája szerint felépíteni az oldalt. Céлом volt, hogy egyszerű, letisztult legyen az oldal, ne keljen egyes dolgokat keresgélgni. Napjainkban a reszponzivitás elvárás a weboldallal szemben, így a fejlesztés során végig teszteltem mobiltelefonon, illetve a böngésző fejlesztői eszközei segítségével lehetőségünk van egyes modellek kijelzőjének megfelelően megjeleníteni a weboldalt, ami nagyon hasznos. A reszponzív design azt jelenti ebben az értelemben, hogy a weboldal rugalmasan alkalmazkodik az eszközünk képernyőjének méreteihez, aszerint méretezi az oldal elemeit. A felhasználói élményt ezzel lehet leginkább fokozni. (mobil, asztali pc, laptop egyaránt jól fogja megjeleníteni az oldalt). Ezt legkönnyebben a Bootstrap megfelelő felhasználásával érhetjük el. Erről a későbbiekben írok részletesen.

2. ReactGame

2.1 Felhasznált programozási nyelvek, keretrendszerek

2.1.1 HTML

A HyperText Markup Language egy leíró nyelv, amely maga a böngészőben megjelenő oldal szerkezetét, megjelenését írja le.

A webfejlesztés alapja, hogy cakk-pakk tudjunk egy komplett HTML oldalt írni, ismerjük annak legfontosabb elemeit. Azonban önmagában csak egy statikus oldalt tudunk vele készíteni, ez korántsem elegendő egy komplex webapplikációhoz, azonban mégsem lenne lehetséges a HTML nélkül mindezt megvalósítani.

A HTML kialakulása az internet korai időszakába nyúlik vissza, amikor egy olyan szabványos jelölő nyelvet akartak kialakítani, amivel egy dokumentumot bármilyen gépről meg lehet jeleníteni böngésző segítségével. Az internet terjedésével együtt elterjedt a használata. Természetesen a korral együtt fejlődött, mai változata a HTML5, ami elég jól támogatja a böngészős játékokat is.



1. ábra: a HTML logója

2.1.2 CSS

A CSS jelentése **Cascading Style Sheets**¹, azaz egymásba ágyazott stíluslapok. A HTML oldalaink megjelenését befolyásoló egyszerű nyelvről van szó, mely segítségével meghatározhatjuk, hogy hogyan (és hol) jelenjenek meg az egyes HTML elemek (paragrafusok, címsorok, stb.), többek között befolyásolhatjuk a színüket, méretüket, elhelyezkedésüket, margóikat, stb. Az egymásba ágyazhatóság (kaszkádozás) arra utal, hogy több stíluslapot, meghatározást is megadhatunk egyszerre, illetve egy stílus lehet több



2. ábra: a CSS logója

elemre is érvényes, amit egy másik stílussal felüldefiniálhatunk. A stílusok öröklődnek az oldal hierarchiája szerint, ha például a gyökér elemre definiálunk egy stílust, akkor az többnyire az oldal összes elemére érvényes (a tulajdonságok örökölhetségétől függően).

A HTML kezdetben nem a szép megjelenést szolgálta, csupán a tartalom egyszerű megjelenítését. Azonban a terjedésével egyre többféle oldal született és igény lett a dokumentum megformálására, szép megjelenítésére. Emiatt ki kellett alakítani egy rendszert, ami leírja a HTML elemek kinézetét. Így alakult ki lényegében a CSS, amit azóta is előszeretettel használunk.

¹ CSS: [CSS alapjai I. Weblabor](#) [2020.12.06]

2.1.3 Bootstrap

A **Bootstrap**² egy nyílt forráskódú keretrendszer (framework), mely HTML, CSS, JavaScript technológiákat használ. Alapvetően arra jó, hogy nagyon könnyedén, és minimális energia befektetéssel tudjon valaki jól kinéző, bármilyen képernyőméreten szépen megjelenő weboldalakat készíteni. Habár nem csupán CSS keretrendszernek minősül, valójában csak a CSS miatt használtam jómagam is. Könnyű használni: Csupán be kell importálni dokumentumunkban, mint egy stíluslapot, azonban ez internet-hozzáférést igényel. Másik megoldás, hogy egyszerűen letöltjük a stíluslapot, ami tartalmazza az stílusokat. Ha be van importálva, akkor csak a HTML osztályaira kell alkalmazni a Bootstrap osztályokat (pl. container, row, form-control, stb...), és oldalunkon alkalmazva lesz a stílus. A reszponzív oldalakban nagy segítség. A Bootstrap hivatalos oldalán egyébként nagyszerű dokumentáció van, amivel könnyen tudjuk használni. Jelenlegi legfrissebb verzió a Bootstrap 4.



3. ábra: a Bootstrap logója

2.1.4 JavaScript

A **JavaScript**³ programozási nyelv egy objektumorientált, prototípus alapú szkriptnyelv, amelyet weboldalakon elterjedten használnak, eredetileg Brendan Eich, a Netscape Communications mérnöke fejlesztette ki



4. ábra: a JavaScript logója

A JavaScript kód vagy a HTML fájlban vagy külön (jellemzően .js kiterjesztésű) szövegfájlban van. Ezek a fájlok tetszőleges szövegszerkesztő (nem dokumentumszerkesztő) programmal szerkeszthetők. A JavaScript esetében a futási környezet jellemzően egy webböngésző (JavaScript-motorja). Windows-os környezetben futtatható a wscript.exe segítségével vagy a Node-dal a Node.js telepítésével. Linux-os környezetben pedig Node-dal és szintén a Node.js installálása után futtatható.

Könnyedén tehetjük vele dinamikussá az oldalunkat, habár fontos megjegyezni, hogy ez a kliens oldalon fut le, ezért nem szabad autentikációs, vagy bármilyen szerveroldali feladatot ebben megoldani. Ugyanakkor játékokhoz szinte elengedhetetlen, és sokkal gyorsabb, mintha minden tartalomfrissítés az oldal újbóli betöltésével történne. Csak akkor működnek a javascriptes funkciók, ha az engedélyezve van a böngészőnkben. Az én programomban ezzel oldottam meg a reakcióidő mérését, ugyanis azt kliensoldalon célszerű lefuttatni, hisz a szerver lelassítaná a folyamatot, valótlan eredményt mutatna.

² **Bootstrap:** [Mi az a Bootstrap 4? Hogyan érdemes használnunk? Élő példákkal \(gremmedia.hu\)](#) [2020.12.06]

³ **JavaScript:** [JavaScript – Wikipédia \(wikipedia.org\)](#) [2020.12.06]

2.1.4.2 jQuery

A **jQuery**⁴ népszerű JavaScript könyvtár, mely a HTML kód és a kliensoldali JavaScript közötti kapcsolatot hangsúlyozza. 2006 januárjában jelentette meg a Mozilla Alapítvány népszerű JavaScript evangélistája, John Resig. A függvénykönyvtár MIT és GNU kettős licenc alatt jelent meg. A jQuery ingyenes, nyílt forrású szoftver. Számos ismert IT cég is alkalmazza a jQuery-t saját projektjeiben, például a Microsoft erre építette a Visual Studióban is elérhető ASP.NET AJAX platformját. A jQuery célja, hogy segítsen minél inkább leválasztani a JavaScript kódot a HTML-ről, és kényelmes kommunikációt biztosítson a weblap elemeivel – eseményvezérlők és azonosítók (ún. CSS szelektorok) használatával.



5. ábra: a jQuery logója

2.1.5 PHP

A PHP jelentése: Hypertext Preprocessor. Egy objekumorientált programozási nyelv. Működését tekintve ez egy szerver oldali nyelv. A PHP program egy szerveren kell, hogy fusson (ami lehet a saját gépünk is). A PHP oldalak elkészítésénél a HTML-t gyakorlatilag csak mint formázást használják. Ezen lapok teljes funkcionalitása a PHP-re épül. Amikor egy PHP-ben megírt oldalt akarunk elérni, a kiszolgáló először feldolgozza a PHP utasításokat, és csak a kész (HTML) kimenetet küldi el a böngészőnek. Így kliens oldalról nem látható a program kód. A feldolgozáshoz egy úgynevezett interpretert (értelmezőt) használ, amely általában a webservert egy külső modulja. Ebben hasonlít más szerver oldali szkript nyelvekre, mint a Microsoft ASP.NET-je, a Sun Microsystems JavaServer Pages és a mod_perl. Az 5.4-es verzió óta a nyelv tartalmaz egy egyszerű, parancssorban beállítható webservert. Legfrissebb verziója PHP 8.0, ami szakdolgozatom írása közben jött ki. A PHP 1995 óta van jelen Rasmus Lerdolf jóvoltából.



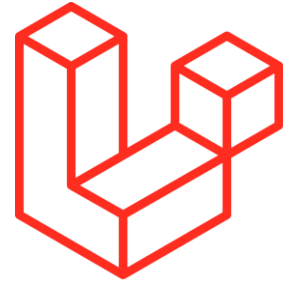
6. ábra: a PHP logója

Az én szakdolgozatom „motorja” a PHP, ugyanis a felhasználó csak a PHP nyelven írt kontrollerek (osztályok) segítségével tud kommunikálni az adatbázissal (regisztrálni, be lépni, játszani, stb.).

⁴ **jQuery**: <https://hu.wikipedia.org/wiki/JQuery> [2020.12.07]

2.1.6 Laravel

Az én projectem alapja a **Laravel**⁵. A Laravel a PHP nevű serveroldali szkriptnyelvhez tartozó egyik keretrendszer, egy olyan nyílt forráskódú webes alkalmazás, amellyel gyorsan és egyszerűen lehetséges testre szabott webes alkalmazásokat tervezni. A fejlesztők azért részesítik előnyben a Laravelt a többi keretrendszerhez képest, mert kiemelkedő teljesítményt nyújt, sok funkciót kínál, illetve skálázható is.



7. ábra: a Laravel logója

Legfrissebb verziója a Laravel 8. Az MVC-t, vagyis a Model View Controllert követi, ezért hatékonyabb és biztonságosabb, mint a natív PHP. Megkönnyíti az olyan általános feladatok elvégzését, mint a hitelesítés (authentication), a munkafázis (sessions) a routing, vagy a gyorsítótárba másolás (caching). Ennél a típusnál az osztályok ényegében megegyeznek az adatbázis modelljeivel, a laravel a beállított táblák segítségével automatikusan megcsinálja a modellt, elég csak magát az osztályt létrehoznunk, a property-ket már automatikusan létrehozza magának, persze ez nem azt jelenti, hogy manuálisan nem adhatunk meg saját tulajdonságokat vagy függvényeket az osztályban. szakdolgozatom tehát egy Laragon parancsorból kiadott paranccsal jött létre, ami egy Laravel projectet jelent. Természetesen ezután még rengeteg munkám volt, azonban jelentős mértékben segített a Node modulok egyszerű beszerzésében, felépíteni az alap struktúrát.

⁵ **Laravel:** https://www.prooktatas.hu/hir.php?hirek_cim=Laravel,%20a%20legjobb%20PHP%20keretrendszer
[2020.12.07]

2.2 A felhasználói felület működése

Igyekeztem minél egyértelműbb, egyszerűbb és szebb felületet biztosítani a felhasználóknak, hogy a legkevesebb informatikai tudással rendelkező ember is könnyedén élvezhesse. Bemutatom logikai sorrendben a webalkalmazásom felhasználói felületét:

2.1.5 A program elindítása, elérése

Mint kiderült, a programomat böngészővel, vagyis a projectet futtató webszerver címét (IP-cím vagy Domain) kell megnyitni a böngészőben. A tesztelés alatt nyilván csak localhoston (saját gépemen), illetve LAN-on volt érhető el a webalkalmazásom, de könnyen elindíthatjuk egy rendes webszerveren is a forrásfájlok rendelkezésre állása esetén.

2.2.2 Regisztráció

The screenshot shows a registration form with the following fields and labels:

- Játssz velünk, ingyen!** (Title)
- Felhasználónév** (Username): Input field with value 'xkiki2001@gmail.com'
- Email-cím** (Email): Input field with value 'pl.admin@admin.com'
- Jelszó** (Password): Input field with masked characters '.....'
- Jelszó megerősítése** (Password Confirmation): Input field with placeholder text 'Figyelj rá, hogy megegyezzen a két beírt jelszó!'
- Nem** (Gender): Dropdown menu with 'Férfi' selected
- Születésnap** (Birthdate): Input field with placeholder 'éééé. hh. nn.' and a calendar icon
- Regisztráció** (Registration): Red button

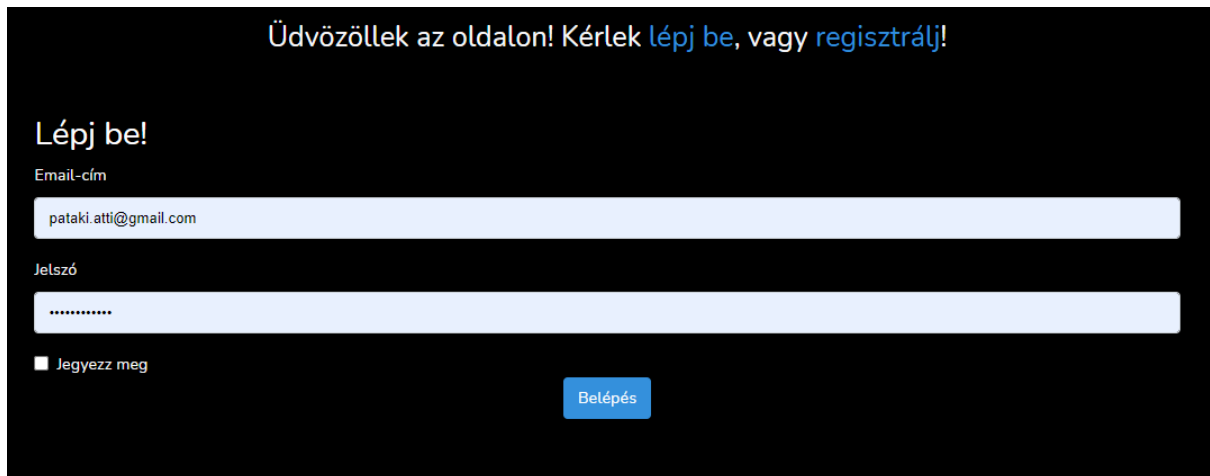
8. ábra: Regisztráció

A ReactGame több felhasználó részére készült, emiatt is fontos a regisztráció. A regisztráció során meg kell adni egy nevet (mindegy, hogy teljes név, vagy csak becenév), ami abban segít majd, hogy megtaláljuk barátainkat. Ezután kell egy érvényes email-cím, ami a felhasználók pontos azonosítását szolgálja a program működése során, ezért nyilván egy email címet csak egy felhasználó használhat. (szólni fog a program, ha már foglalt az e-mail cím). Következik a jelszó, amit kétszer kell beírni, nehogy rosszul adja meg véletlenül a felhasználó (ha nem egyezik a két jelszó, hibát fog írni rá és félbeszakad a regisztráció). Ezenkívül statisztikai okokból meg kell adni a nemünket, illetve a születési dátumunkat. A jelszó természetesen erős titkosítással kerül az adatbázisba, aminek biztonsági okokból legalább 8 karakternek kell lennie. Sikeres regisztráció esetén rögtön átirányít minket a program a bejelentkező felületre, egy

üzenet kíséretében. Természetesen nem csak így érjük el a bejelentkező oldalt. Nem alkalmaztam email-cím megerősítést vagy Capatcha-ellenőrzést, ugyanis ennél a programnál nem tartottam fontosnak, de persze ezt meg tudom oldani később, ha komolyabb felhasználásra kerülne a program.

2.2.3 Bejelentkezés

A bejelentkezés értelemszerűen a regisztrált email-cím és jelszó segítségével történik. Az adatok helytelen megadása esetén mindig csak annyit árul el a program, hogy „az email-cím vagy a jelszó nem megfelelő”, mivel ez megnehezíti a fiók feltörésének lehetőségét, hiszen nem tudja a próbálkozó, hogy melyik adatot rontotta el. Ha sikeres a belépés, akkor átirányít minket a program a kezdőlapra, ahonnan a menüből könnyen navigálhatunk.



Üdvözöllek az oldalon! Kérlek [lépj be](#), vagy [regisztrálj](#)!

Lépj be!

Email-cím

pataki.attii@gmail.com

Jelszó

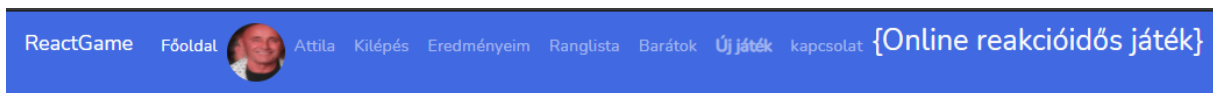
.....

☐ Jegyezz meg

Belépés

9. ábra: bejelentkező felület

2.2.4 Kezdőlap és menü

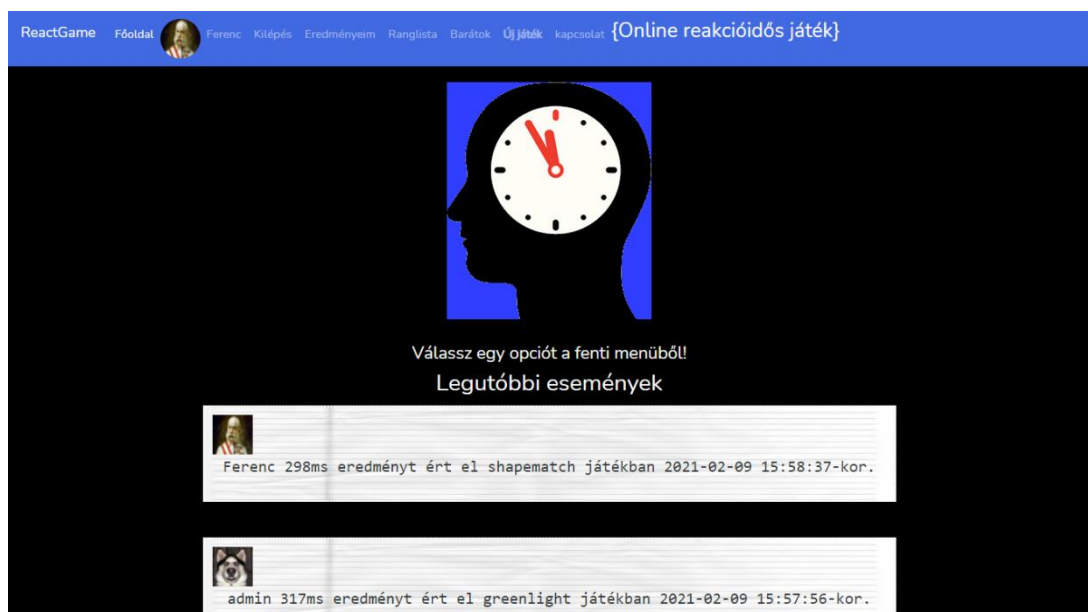


10. ábra: Menüsáv

Asztali és mobil nézetben is felül egy navigációs sávban vannak a linkek, de mobilon ezt jobboldalt ki kell nyitni egy gombbal. A menüben a következő elemek vannak:

- Főoldal: visszavisz a főoldalra
- A bejelentkezett játékos aktuális profilképe és neve: Ezekre kattintva a játékos profiljára visz minket, ahol megnézhetjük regisztrált adatainkat, illetve szerkeszthetjük is azokat (profil)
- Kilépés: Kijelentkezés a webalkalmazásból
- Eredményeim: A játékos elmentett eredményeihez visz
- Ranglista: Az felhasználó és barátai legjobb eredményét sorrendben mutató táblázathoz visz
- Barátok: Arra az oldalra visz, ahol lehetőségünk van barátaink megtekintésére, baráti kérelmek elfogadására/elutasítására, illetve új barát bejelölésére email-cím vagy felhasználónév segítségével.
- Új játék: Arra az oldalra visz, ahonnan elindíthatjuk a játékokat.
- Kapcsolat: Hibajelentés, kapcsolat a fejlesztővel, elérhetőségek

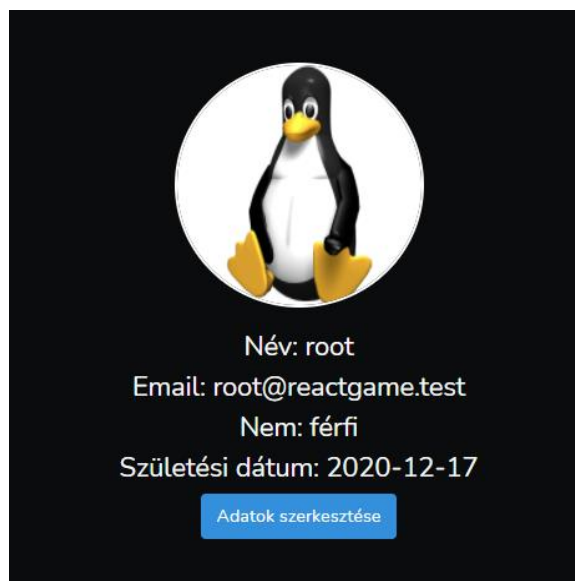
Bejelentkezés után az üdvözlőképernyőt látjuk. Ha már vannak eredményeink, ugyanitt lejjebb láthatjuk az utolsó 10 eseményt, ha barátaink is vannak, természetesen az övükét is mutatja:



11. Ábra: A kezdőlap

2.2.5 Profil, és annak szerkesztése

Kattintsunk a menüben a képünkre vagy a nevünkre. (Alapértelmezetten egy fantomkép lesz az avatarunk regisztráció után.) Ez az oldal megjeleníti a profilunkat, valamint az adatok szerkesztése gombra kattintva megjelenik egy felület, ahol lehetőségünk van nevünk, email-címünk, avatarunk (képünk), valamint jelszavunk megváltoztatására. Más felhasználó profilját is lehetőség megtekinteni, ha rákattintunk a képre vagy a nevére. Hasonlóan fog megjelenni, minr a saját profilunk.



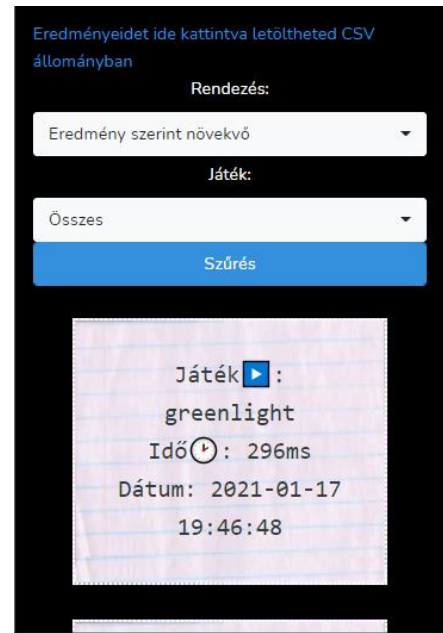
12. ábra: Profil

Az avatart úgy oldottam meg, hogy a kép URL-jét kell bemásolni a mezőbe (képen jobb klikk, képhivatkozás/kép címének másolása). Természetesen vissza is lehet vonni a 'mégsem' gombbal a műveletet. A jelszót csak akkor kell beírni, ha meg szeretnénk változtatni, ellenkező esetben csak üresen kell hagyni. Azért oldottam meg URL-lel, mert a képek feltöltése kicsit bonyolultabb megoldás és tárhelyet is foglal, ráadásul ez egy elhanyagolható funkció a játékban, mindössze jobban néz ki vele a profil. Bármennyi alkalommal változtathatjuk adatainkat.

13. ábra: profil szerkesztése

2.2.6 Eredményeim

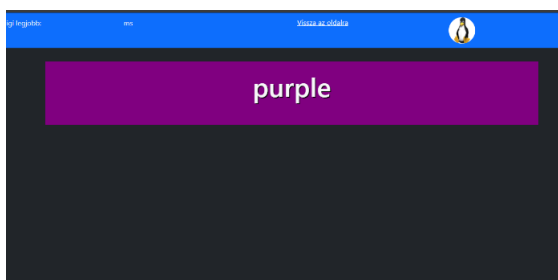
Ez a nézet mutatja azokat az eredményeket, amiket valamely játékkal értünk el. Láthatjuk a játék nevét, amiben elértük az eredményt, az időt, valamint a teljesítés dátumát. Amint az ábrán látható, lehetőségünk van dátum, illetve eredmények szerint rendezni és játékok szerint szűrni. Alapból az összes játékot listázza természetesen. Nem bonyolítottam túl a nézetet, csak a szükséges adatok állnak rendelkezésre. Igazából csak az a lényege, hogy a játékos tudja ellenőrizni bármikor, hogy mikor, hogyan, miben teljesített. A beállított szűrő alapján generál egy CSV fájlt a szerver, amit le tudunk tölteni az oldal tetején lévő linkkel.



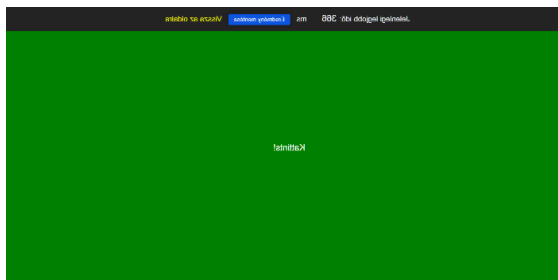
14. ábra: Eredményeim

2.2.7 Új játék

Itt tudjuk megtekinteni, hogy milyen játékokat kínál az oldalam, és innen indíthatunk egy játékot. Minden játékhoz van leírás, hogy könnyen megértsük, mi az adott játék lényege. A mellettük szereplő kép csak illusztráció, figyelemfelkeltő hatással van ott. Természetesen minden játék kezdőképernyőjén megtalálhatóak az utasítások. Kiemelném, hogy **minden játékmenetből csak akkor lesz elmentve az eredmény, ha megnyomtuk a mentés gombot**. Így a nem kívánt eredmények nem lesznek eltárolva és tárhelyet takarítok meg.



15. ábra: A colormatch játék



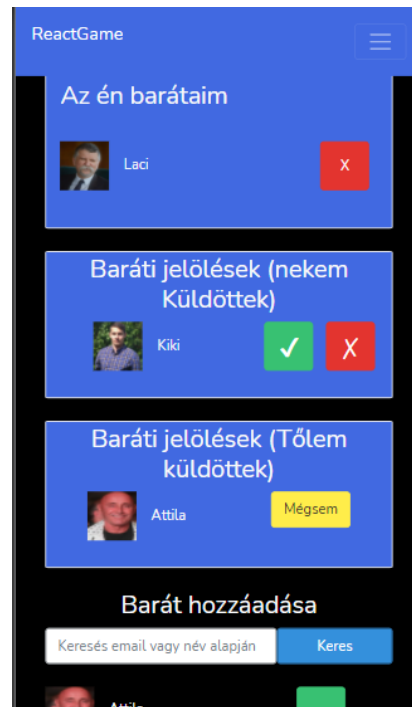
16. ábra: A greenlight játék



17. ábra: Az „új játék” menü

2.2.8 Barátok

Ezen az oldalon tudjuk megtekinteni meglévő barátainkat, elfogadni vagy elutasítani a baráti felkéréseket, illetve keresni és bejelölni játékosokat. A keresés email-cím és felhasználónév alapján egyaránt működik. Ha a kereső talált játékost, a mellette lévő „+” gombra kattintva elküldi a baráti kérelmet, amit majd el kell fogadnia a célszemélynek. Természetesen el is utasítható a kérelem. Ugyanakkor lehetőség van meglévő barátok eltávolítására, vagy a saját felkérésünket visszavonni. A ranglistán a barátok legjobb eredményével összehasonlítva fog a mi legjobb eredményünk megjelenni, ezért fontos a baráti szerep. Ha valakit már bejelöltünk, vagy a barátunk, ugyanúgy megtalálja a kereső és ott van a + gomb, azonban semmi sem fog történni.



18. ábra: a barátok menü

2.2.9 Ranglista

A ranglista funkciója, hogy (az egyes játékokat különválasztva) kilistázza a játékos és annak barátainak a legjobb eredményét az adott játékban és ez alapján összeállít egy helyezési sorrendet. Természetesen ez folyamatosan változik, amint valaki jobb eredményt ér el a másiknál. A fenti menüvel ki kell választani a kívánt játékot, majd a „mutasd” gombra. A saját (belepett felhasználó) helyezésünket kiemeli kékes színnel. A képen például „Pista” a beletett felhasználó (4. hely), root érte el a baráti körben a legjobb eredményt a greenlight nevű játékban, a maga 274 ezredmásodperces eredményével. A colormatch nevű játékban is ő a legjobb jelenleg.

| Ranglista (greenlight) | | | |
|------------------------|---------|----------|--|
| Helyezés | Játékos | Idő (ms) | |
| 1 | root | 274 | |
| 2 | Laci | 280 | |
| 3 | admin | 282 | |

| Ranglista (colormatch) | | | |
|------------------------|---------|----------|--|
| Helyezés | Játékos | Idő (ms) | |
| 1 | root | 258 | |
| 2 | Béla | 351 | |
| 3 | Laci | 459 | |
| 4 | Pista | 488 | |

19. ábra: ranglista

2.2 Felhasználók jelentése, blokkolt felhasználók

Lehetőségünk van bejelenteni gyanús eredményeket elérő játékosokat, mindezt a játékos nevére kattintva találjuk a megjelenített profilján. A „report” hivatkozásra kattintva megjelenik egy ablak, amiben meg kell adnunk az okot, ezzel segítve az admin munkáját a döntéshozatal során.



20. ábra: Felhasználók profilja és bejelentés

A gombra kattintás után kapunk egy üzenetet: „Bejelentésed elküldtük”. Ha valami oknál fogva mégsem sikerülne, akkor az üzenet ez lesz: „Nem sikerült elküldeni a bejelentést. Azonban ilyenkor nem nagyon találkozhatunk, a tesztelés alatt egyszer sem fordult ez elő.

Ha egy profilt blokkol az admin, akkor azzal a fiókkal nem fog tudni belépni, senki sem fogja látni a játékon belül, azonban profilját bármikor visszaállíthatja az admin. Ha egy blokkolt fiókkal próbálunk belépni, akkor a következő üzenetet kapjuk a bejelentkező felületen: „A fiókot blokkolták!” – természetesen a kapcsolat menü ilyenkor is elérhető, és jelezheti az adminnak, ha valamiben nem ért egyet.

2.3 Az adatbázis

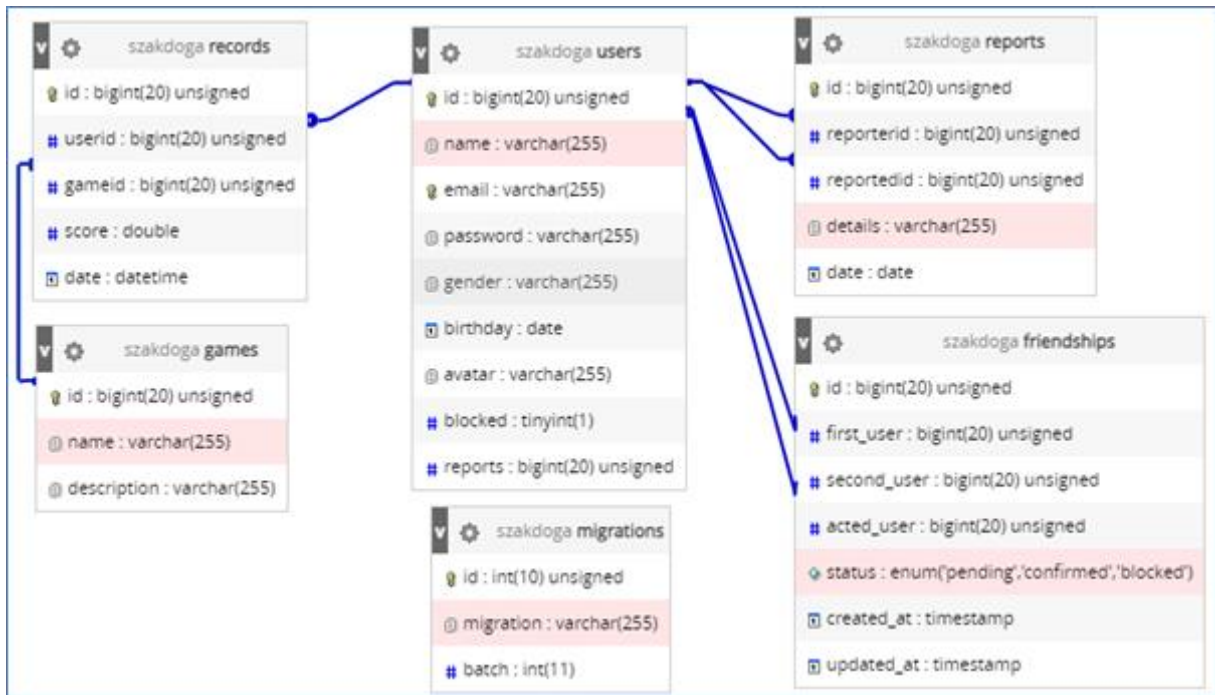
Az adatokat valamilyen formában tárolni kell. Ehhez elengedhetetlen egy adatbázis, ami megvalósítja az adatok rendszerezett tárolását. Az én adatbázisom technikai hátterét a Laragon csomag részét alkotó MySQL adatbázis szerver biztosítja. Ezt az adatbázist közvetlenül, a tesztelés erejéig a HeidiSQL nevű programmal felügyeltem, ami közvetlen lehetőséget biztosít az adatbázisok, táblák, rekordok, kapcsolatok teljeskörű menedzselésére egy felhasználóbarát környezetben. A programom a Laravel egyik beépített funkciója segítségével kommunikál az adatbázissal: csupán be kell állítani a programrendszer konfigurációs fájljában az adatbázis-szerver címét, portját, jelszavát, valamint a projectünkhöz készített adatbázis nevét. Ezek segítségével elérem az adatbázisomat. Az adatbázis-műveleteket ún. Eloquent segítségével hajtom végre, ami egy object-relational mapper (ORM), hasonló a C# Entity Framework-ben használt LINQ-hez. Sokkal egyszerűbb a szintaktika, mint a hagyományos PHP - SQL kérések.

2.3.1 Tervezési szint

Először is meg kell tervezni az adatbázisunk felépítését. A tervezés során figyelembe kell vennünk az elvárásainkat az adatok tárolásával és azok kapcsolataival kapcsolatban:

- Először is szükségünk van egy felhasználók (**users**) táblára, ahol a játékosok adatait fogjuk tárolni, hiszen ők szolgálnak a játék alapjaként (a legtöbb adat az egész adatbázisban hozzájuk fog kapcsolódni valamilyen szinten). El kell tárolni a felhasználók neveit (**name**) és jelszavait (**password**). Természetesen a jelszó már titkosítva kerül az adatbázisba regisztráció, belépés vagy jelszómódosítás során. Én egy PHP beépített függvénnyel, a **bcrypt**-tel oldottam meg, ami saltolva egy 192 bites hasht generál a jelszóból, ami elég erős védelmet nyújt a feltöréssel szemben. Minden ilyen kódolási algoritmus egyirányú, tehát nem tudjuk visszafejteni, csupán ellenőrizni, hogy egyezik-e a begépett szöveggel. Ezenkívül szükséges az **email**-cím, ami egyértelműen azonosítja a játékost a játékosok számára, mivel egy email-címet csak egy felhasználó használhat, valamint statisztikai okokból a felhasználó nemét (**gender**) és születési dátumát (**birthday**) is tárolom, természetesen nem kötelező valódi adatokat megadni. Ahhoz, hogy lehetséges legyen blokkolni egy játékost, egy **blocked** oszlopban tárolom azt, hogy blokkolva van-e (igen/nem) és a játékosra érkezett bejelentések számát a **reports** oszlopban tárolom.

- A játékosok elért eredményeit egy **records** nevű táblában tárolom, amiben egy sor a következőképp néz ki: a rekord azonosítója (**id**), az eredményhez tartozó játékos azonosítója (**userid**), a játék azonosítója, amiben az eredmény létrejött (**gameid**), maga az eredmény, mint idő (**score**), végül az eredmény dátuma (**date**).
- Mivel több, egyszerű játékról van szó, ezért egy **games** táblában tárolom a játékok adatait, ahol van egy azonosító (**id**), egy, a játék nevét tartalmazó **name** oszlop, egy játékhoz tartozó leírás (**description**), valamint egy hozzá tartozó kép URL-jét tároló **image** oszlop.
- A gyanús eredményeket elérő játékosok bejelentéséhez készítettem egy **reports** nevű táblát, ami azt tárolja, hogy ki (**reporterid**), kit (**reportedid**), mikor, miért (**details**) jelentett be.
- Végül a baráti kapcsolatok megvalósítása érdekében egy **friendships** nevű tábla tárolja a barátságokat. Ebben a két felhasználó azonosítója van (**first_user**, **second_user**), valamint annak a játékosnak az azonosítója, aki kérelmezte a kapcsolatot (**acted_user**). Utóbbi azért szükséges, mert ezzel könnyen meg tudjuk különböztetni, hogy a belépett játékos kapta, vagy küldte a kérelmet. Végül a **status** oszlopban tárolom azt, hogy a baráti viszony vissza van-e igazolva, vagy még függőben van. Például, ha A bejelöli B-t, akkor létrejön egy rekord, „pending” státusszal. Ha a B megerősíti, akkor már „confirmed” lesz. Ha viszont elutasítja B a felkérést, akkor egyszerűen törlődik a kérelem.
- A **migrations** táblát a Laravel automatikusan generálja az első adatbázis migrációnál. A migráció lényegében az, hogy adott osztályok és annak függvényeinek segítségével a program magától létrehozza az adatbázisát (persze ehhez az adatbázis-kapcsolatot jól kell definiálni az ENV fájlban). Azért jó, mert rendkívül felgyorsítja az adatbázis felépítését, amikor frissen állítjuk fel a programot akár egy új gépen. Tehát a migrations tábla csupán a végrehajtott migrációk adatait tárolja, igazából elhagyható, de jó, ha van, hisz nyomon tudjuk követni ezeket a műveleteket.



21. ábra: az adatbázis modell (osztálydiagram)

2.3.2 Fizikai szint (implementálás)

Az előző részben meghatároztam az elvárásokat, ezek alapján kellett megvalósítani az adatbázist. Mint említettem, egy MySQL adatbázis felel az adatok tárolásáért. Mint látható, valójában csak két közvetlen, egy az egyhez kapcsolat van az adatbázisomban, ugyanis így egyszerűbb és hatékonyabb volt számomra megoldani az adatokkal való műveleteket végrehajtani a játék futása során. Természetesen minden táblában nélkülözhetetlen a primary key (elsődleges kulcs), valamint foreign keyek (idegenkulcsok), mivel ezek alapján tudjuk meghatározni a kapcsolatokat. Minden kulcsnál előjel nélküli intet használok, mivel így kétszer annyi felhasználót tudunk tárolni, hisz negatív számot nem kaphat egy AI oszlop.

- A felhasználókat egy külön táblában tárolom, ahol azonban csak a személyes adataikat tárolom. Elsődleges kulcs az id, azonban az email egy unique érték, vagyis a táblában csak egyszer szerepelhet ugyanaz az email. Minden oszlop maximum 255 karakter hosszú varchar, amit könnyen stringként tudunk kezelni a programban, kivéve a születnapot, ami egy Date típus. A **reports** egy nagy int, a **blocked**et tinyint-nek kezeli, valójában csak 0 és 1 érték kerül bele, bőven jó ez a típus.
- Magukat a játékokat a tulajdonságaik alapján tárolom, mint rekordokat a games táblában. Elsődleges kulcs az id, a leírás (**description**) és a játék neve (**name**), valamint a játékhoz tartozó kép URL-je varchar típusként tárolódik.

- A **records** tábla tárol minden egyes lejátszott játékot. Ehhez tárolom a következőket: ki (**userid**), milyen játékot (**gameid**), milyen eredménnyel (**score**), Mikor (**date**) játszott? Természetesen egy AutoIncrementes ID is tartozik minden rekordhoz, hogy könnyen beazonosíthassuk. Itt tehát két külső kulcs is van: **gameid** és **userid**. Ezek segítségével könnyen megkapjuk a hozzá tartozó játékos és játékot, akár mint objektumot (a modell függvényeinek segítségével). Mindkettő egy a többhöz kapcsolat, mivel egy játékhoz pontosan egy felhasználó és egy játék tartozik, azonban egy játék vagy egy játékos bármennyi rekordhoz tartozhat. Az id-t tartalmazó oszlopok előjel nélküli bigintek, a score egy double típus, végül a date egy dátum természetesen.
- A barátságokat úgy oldottam meg, hogy a **first_user** és a **second_user** idegenkulcsként kapcsolódik a users tábla id oszlopához, ezzel a kapcsolat szereplőit megkapjuk. Az **acted_user** oszlop abban segít, hogy melyik fél kezdeményezte a kapcsolatot. A status egy enumerable típus, ami segítségével könnyen meghatározható, hogy jelölről, vagy megerősített barátságról van-e szó. Látható, hogy van még a „**blocked**” lehetőség is, azonban ezt egyelőre nem tartottam fontosnak, mivel egyszerűen ki lehet törölni, mint barátot a nem kívánt személyt. Ámbár a jövőben könnyen bele tudom építeni a tiltás funkciót, ha lesz rá igény. Tehát a programon belül, a users táblához tartozó modellben írtam függvényeket, amik lekérdezésekkel visszaadják a baráti kéréseket, illetve barátokat. Tehát lényegében a user objektum egyik függvényeinek megkapjuk a barátait, mint adatgyűjteményt.

2.3.3 Migrációk és seederek a Laravelben

Érdemesnek tartottam még leírni, hogy a Laravel egyik előnye az, hogy úgynevezett migrációs osztályokat és függvényeket tudunk készíteni, amelyek a megadott adatbázisban létrehozzák a táblákat. A programunkban az egyes táblák modelljeit egyszerűen megkapjuk, csupán a tábla nevét célszerű megadni az osztályban. Ezek alapján könnyen tudjuk használni rekordjainkat, mint objektumokat. Ezt sokat használtam, ugyanis volt, hogy újra kellett csinálnom az adatbázist, de ennek a funkciónak köszönhetően pár másodperc alatt töröltem és újra létrehoztam a táblákat kettő, terminálba kiadott paranccsal.

A seederek is osztályok, melyek segítségével rekordokat tudunk írni a táblákba. Én személy szerint arra használtam, hogy előre definiált felhasználókat hozzak létre rögtön a migráció lefutása után, hogy teszteljem az autentikációt hamar.

2.4 A szerver működése

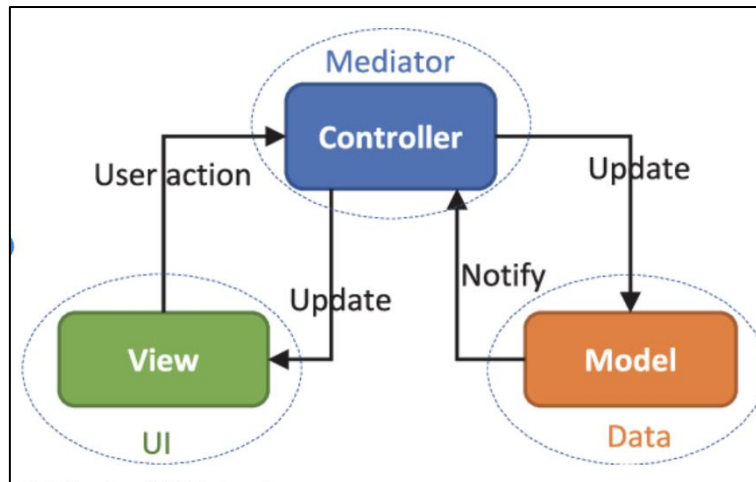
A szerver a rendszer legösszetettebb része, amely feladata, hogy futtassa azokat a szolgáltatásokat, amelyek szükségesek a program használatához (MySQL szerver, Apache szerver, PHP, stb...), valamint, hogy a klienseket összekösse az adatbázissal. Nyilván csak akkor érhető el a játék, hogyha fut a szerver. A HTTP kérésekkel kommunikál a kliensekkel. A szerver futhat bármilyen (kellően modern) eszközön, ami alkalmas arra. Nekem például a saját laptopomon fut, és helyi hálózaton egyszerre több eszközt is problémamentesen kiszolgál. Azonban, ha nagyobb igény lenne, akkor lehetséges, hogy egy erősebb, direkt erre a célra gyártott szervergépre ajánlott mozgósítani a rendszert.

2.4.1 Az MVC

A **modell-nézet-vezérlő**⁶(MNV) (angolul model-view-controller) a szoftvertervezésben használatos programtervezési minta. Összetett, sok adatot a felhasználó elé táró számítógépes alkalmazásokban gyakori fejlesztői kíváncsi az adathoz (modell) és a felhasználói felülethez (nézet) tartozó dolgok szétválasztása, hogy a felhasználói felület ne befolyásolja az adatkezelést, és az adatok átszervezhetők legyenek a felhasználói felület változtatása nélkül. A modell-nézet-vezérlő ezt úgy éri el, hogy elkülöníti az adatok elérését és az üzleti logikát az adatok megjelenítésétől és a felhasználói interakciótól egy közbülső összetevő, a vezérlő bevezetésével. Előnyei:

- ✓ Egyidejű fejlesztés – Több fejlesztő tud egyszerre külön a modellen, vezérlőn és a nézeteken dolgozni.
- ✓ Magas szintű összetartás – MNV segítségével az összetartozó funkciók egy vezérlőben csoportosíthatóak. Egy bizonyos modell nézetei is csoportosíthatóak.
- ✓ Függetlenség – MNV mintában az elemek alapvetően nagy részben függetlenek egymástól
- ✓ Könnyen változtatható – Mivel a felelősségek szét vannak választva a jövőbeli fejlesztések könnyebbek lesznek
- ✓ Több nézet egy modellhez – Modelleknek több nézetük is lehet
- ✓ Tesztelhetőség - mivel a felelősségek tisztán szét vannak választva, a külön elemek könnyebben tesztelhetők egymástól függetlenül

⁶**Modell-nézet-vezérlő:** <https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91>
[2020.12.23]

22. ábra: az MVC modell⁷

2.4.2 Blade

Az én projectem felhasználói oldala is ezen az elven működik. A felhasználó, mikor egy kérést küld a szerverem felé (pl. megnyitja a ranglistát, vagy belép, stb.), akkor az ahhoz az URL-hez (route) tartozó kontroller metódusát hívja meg a routes fájl. Igazából minden nézet dinamikus, a Blade-nek köszönhetően, ugyanis mikor a metódus visszaadja a felhasználónak a nézetet, azzal együtt olyan változókat/gyűjteményeket is küld a view-nak, amik szükségesek az adatok megjelenítéséhez (pl. ranglista lekérésekor visszaadja a barátainak a legjobb eredményét az adott játékban egy tömbben, amit a Blade segítségével megjelenít a HTML elemekben egy ciklussal). A Blade egy Laravel által is biztosított, egyszerű, de hatékony sablonmotor. A többi sablonmotorral ellentétben lehetővé teszi a PHP kódok futtatását is. A dinamikus tervezésnek köszönhetően bármikor tudok új játékot hozzáadni, vagy módosítani a meglévőket.

⁷ MVC ábra: <https://medium.com/@rhodunda/mvc-design-pattern-fe76175a01de> [2020.12.23]

2.4.3 A szerver működése, a program beüzemelése

2.4.3.1 Rendszerkövetelmények

Ahhoz, hogy egy gépen fusson rendesen egy webszerver és adatbázis szerver, a következő specifikációk ajánlottak egy számítógépben:

- 2 magos, 1,6Ghz CPU
- 4GB RAM
- 40GB merevlemez terület

A következő szoftverek szükségesek:

(mindezek megoldhatóak az XAMPP vagy Laragon program telepítésével)

- PHP 7.4.2
- Apache 2.4.36
- MySQL 5.7.24
- Szükséges a composer csomag, amit a következő pontban telepítünk

2.4.3.2 A rendszer beüzemelése

- A webszerver www (vagy htdocs) mappájába másolja bele a 'reactgame' nevű mappát.
- A működéshez szükséges telepíteni kiegészítő csomagokat, amit a következőképpen tehetünk meg:
- Nyissunk egy terminált/parancssort a reactgame (forráskód) mappában és adjuk ki a következő parancsokat:
 1. composer install
 2. npm install && npm run dev
 3. npm audit fix
 4. php artisan storage:link (Ez a generált csv fájlok elérése érdekében kell)
- Az adatbázis létrehozásához futtassuk le a reactgme_db.sql fájlt egy adatbázis kezelő programmal.
- Fontos! A reactgame mappában lévő env konfigurációs fájlban van lehetőség az adatbázis nevét, az adatbázis kapcsolathoz szükséges adatokat megadni. A felhasználónév jelenleg root, jelszó nincs beállítva. Ezeket értelemszerűen be kell állítani, ha mások a MySQL szerver autentikációs adatai.
- Mindezek után a webalkalmazásnak működni kell, a webböngészőben (alapértelmezett webszerver beállítások mellett) <http://localhost/reactgame/public> címen érjük el.

2.4.4 A vezérlőpult

A vezérlőpultot azért csináltam, hogy:

- Az adatbázist bármikor (újra) fel lehessen építeni, vagy visszaállítani eredeti állapotára. Ezzel az alapvető adatok is bekerülnek a táblákba.
- Nyomon lehessen követni a játékosokra beérkezett panaszokat
- Játékosok blokkolásának, illetve feloldásának lehetővé tételéhez.
- Mindezt egy felhasználóbarát környezetben

A letiltott játékosok nem tudnak bejelentkezni, barátaik nem látják őket és eredményeiket sehol, azonban megmaradnak az adatbázisban. A blokkolásnál minden szükséges adatot láthatunk a játékosal kapcsolatban, egy listából kell kiválasztani.

Mivel feltételezhetően nem létezik az adatbázis, mikor ezt az oldalt először látogatja a kezelő, ezért nyilván nem az adatbázisban tárolom az ehhez szükséges admin jelszót, hanem egy konfigurációs fájlban tárolom hashelve, ami természetesen teljesen biztonságos.

Érdekesnek tartom asztali gépen, vagy laptopon használni a felületet, ugyanis a reportok táblázat sokkal átláthatóbban jelenik meg, mint mobilon (a reszponzivitás ellenére is). Természetesen lehet szűrni a bejelentések között, hogy könnyebben megtaláljuk a keresett esetet.

Ezt a webalkalmazás címe/controlpanel néven érjük el (pl. <http://localhost/reactgame/public/controlpanel>). Csak annak van jogosultsága bármilyen műveletet végezni, akinek birtokában van az admin jelszó és be tud lépni. A **jelszó**: v8w8He3NsKEsC8q6

ADMIN BELÉPÉS

FELHASZNÁLÓNÉV

JELSZÓ

SIGN IN

Jelentkezz be!

Adatbázis-kezelő

Figyelem! Ezen műveletek elvégzéséhez csak admin joggal rendelkező személy jogosult!

Adatbázis felépítése & feltöltése alapvető adatokkal, illetve a meglévő visszaállítása alap állapotra

Művelet elvégzése

Reportok

Itt láthatóak a bejelentések

| Dátum | Bejelentő | Bejelentett | Indok |
|------------|----------------------|----------------------|---|
| 2021-02-08 | admin@reactgame.com | ferenc@reactgame.com | Szerintem csal... |
| 2021-02-08 | admin@reactgame.com | ferenc@reactgame.com | Érdekes eredmények hmm |
| 2021-02-08 | ferenc@reactgame.com | admin@reactgame.com | Colormatch_hard-ban lehetetlen eredményt ért el |
| 2021-02-08 | ferenc@reactgame.com | admin@reactgame.com | Nem tetszik a profilképe |
| 2021-02-08 | admin@reactgame.com | ferenc@reactgame.com | Nem tudok játszani |
| 2021-02-08 | ferenc@reactgame.com | admin@reactgame.com | ja |

Játékos blokkolása/feloldása

Figyelem! a játékos nem fog végleg törölődni fog a rendszerből! Csak indokolt esetben (pl. csalás) ajánlott bannolni! Bármikor visszaállítható a játékos profilja.

Blokk magyarázat: 1=igen, 0=nem

Blokkolás

Feloldás

23. ábra: A kontroll panel

3. Tesztelés

A játék fejlesztése során a saját laptopomon futtattam végig a szervert, ami egy Dell Latitude e6430 üzleti laptop, amely fontosabb paraméterei:

- ♠ Intel Core i5 3rd Gen 3210M processzor (max 3,2 Ghz)
- ♠ 8GB DDR3 1333Mhz memória
- ♠ 500GB HDD
- ♠ Intel HD Graphics 4000 integrált grafikus kártya

Mivel ez egy kevés erőforrást igénylő rendszer, könnyedén elbírt ezzel a géppel egyszerre 4 felhasználót LAN-on (ennyi eszközöm volt rendelkezésre otthon). Tesztelés során a következő eszközökkel vizsgáltam a program viselkedését:

- A szervert futtató laptop (localhoston)
- Xiaomi Black Shark 2 (Android 10) mobiltelefon
- Apple Iphone 7 (IOS 14.2)
- HP laptop

A fent említett eszközökön a reszponzív felépítésnek köszönhetően szépen jelent meg a weboldal, nem csúszott el semmi és minden elem megfelelő méretben mutatkozott a weblap. Véleményem szerint problémamentesen kiszolgálna ezzel a szerény lappal is több tíz felhasználót. Nyilván több ideig tart betölteni egy új oldalt, mint amikor egy konkrét játékon játszik a játékos, mivel ott JavaScript dolgozik a betöltött weboldalon.

Egy webszerverként működő asztali gépen is futtattam, tökéletesen működött a program. Természetesen léptek fel kisebb hibák a játék fejlesztése során a kliens oldalán, például a reakcióidő telefonon 300 ezredmásodperccel több volt a valódi eredményeknél. Ezt egyszerűen meg tudtam oldani (a hibát a telefonos böngészők zoom opciója okozta). Mivel a játékok JavaScriptet használnak, ezért csak akkor működnek, ha a böngészőben nincs letiltva a JavaScript. A hálózati sebesség nem befolyásolja maguknak a játékoknak az eredményét, mivel kliensoldalon fut betöltés után.

Összegzés

A szakdolgozatom, illetve a programom elkészítése során rengeteget tanultam, hisz én 5 éve kezdtem el programozást tanulni, mindvégig Javat és C#-ot főleg. Úgy vélem, kicsit bátor dolog volt PHP-ban és Laravellel elkészíteni a programom, ugyanis csak 4 hónapja kezdtem el ezeket tanulni. Mindazonáltal, ez egyben egy kihívást is jelentett számomra, amit végig szerettem csinálni. Sokat kellett tanuljak, de szerintem megérte. A sok elméleti tudást most gyakorlatban is alkalmazhattam egy nagyobb szabású programnál. Nem elég csak elméletben tudni a dolgokat, a gyakorlat is ugyanolyan fontos szerintem. A programozás is egy olyan szakma, amiben az ember folyamatosan tanul, régebbi kódjaira nézve lehet, hogy elcsodálkozik az illető, hogy miért úgy csinálta anno a programot, ahogy.

Természetesen nem tökéletes a program, vannak kisebb hibák, amiket valójában nem biztos, hogy hamar észrevenne egy játékos. Azonban betölti funkcióját, véleményem szerint teljesítette a specifikáció elvárásait.

A szakdolgozatban több funkciót is említettem, amit nem valósítottam meg ebben a verzióban. Ezek mind a **további fejlesztés** részét képezik:

- ♠ több játék, az admin felületről lehessen új játékot hozzáadni
- ♠ üzenetküldés más játékosoknak
- ♠ hatékonyabb, gyorsabb működés

Szerencsére ez a project típusa könnyen fejleszthető, így nem okoz majd akadályt egy-egy funkció beépítése (feature update, új játékok). Remélem mások is hasznosnak tartják majd a programomat, szívesen haladnék tovább a webfejlesztés irányában is. Úgy érzem, egy viszonylag jól működő, hasznos programot sikerült kiadni a kezemből.

Bibliográfia

1. CSS alapjai I. (weblabor)

letöltés ideje: 2020.12.06

<http://weblabor.hu/cikkek/cssalapjai1>

2. Mi az a Bootstrap4? Hogyan érdemes használnunk? Élő példákkal (gremmedia.hu)

letöltés ideje: 2020.12.06

<https://gremmedia.hu/educacio/bejegyzes/mi-az-a-bootstrap-4-hogyan-hasznaljuk>

3. JavaScript – Wikipédia

letöltés ideje: 2020.12.06

<https://hu.wikipedia.org/wiki/JavaScript>

4. jQuery – Wikipédia

letöltés ideje: 2020.12.07

<https://hu.wikipedia.org/wiki/JQuery>

5. Programozó oktatás - Laravel, a legjobb PHP keretrendszer

letöltés ideje: 2020.12.07

https://www.prooktatas.hu/hir.php?hirek_cim=Laravel,%20a%20legjobb%20PHP%20keretrendszer

6. Modell-nézet vezérlő – Wikipédia

letöltés ideje: 2020.12.23

<https://hu.wikipedia.org/wiki/Modell-n%C3%A9zet-vez%C3%A9rl%C5%91>

7. MVC ábra - Bryan Rhodunda

letöltés ideje: 2020.12.23

<https://medium.com/@rhodunda/mvc-design-pattern-fe76175a01de>

Ábrajegyzék

| | |
|---|----|
| 1. ábra: a HTML logója..... | 5 |
| 2. ábra: a CSS logója | 5 |
| 3. ábra: a Bootstrap logója | 6 |
| 4. ábra: a JavaScript logója | 6 |
| 5. ábra: a JQuery logója..... | 7 |
| 6. ábra: a PHP logója..... | 7 |
| 7. ábra: a Laravel logója..... | 8 |
| 8. ábra: Regisztráció..... | 9 |
| 9. ábra: Bejelentkezés..... | 10 |
| 10. ábra: A menüsáv..... | 11 |
| 11. ábra: Kezdőlap..... | 12 |
| 12. ábra: Profil..... | 13 |
| 13. ábra: Profil szerkesztése | 13 |
| 14. ábra: Eredményeim | 14 |
| 15. ábra: A colormatch játék | 14 |
| 16. ábra: A greenlight játék | 14 |
| 17. ábra: Új játék menü | 14 |
| 18. ábra: Barátok menü | 15 |
| 19. ábra: Ranglista..... | 15 |
| 20. ábra: Felhasználók profilja és bejelentés..... | 16 |
| 21. ábra: Az adatbázis-modell (osztálydiagram)..... | 19 |
| 22. ábra: Az MVC modell | 22 |
| 23. ábra: Kontroll panel..... | 25 |

Melléklet

1. Egy db CD-ROM a következő adatokkal:
 - Szakdolgozat digitális változata (reacgame_dokumentacio.pdf)
 - Eredetiségnyilatkozat
 - Szakdolgozathoz tartozó programrendszer forráskódja (reactgame mappa)
 - Az adatbázist leíró SQL utasítás (reactgame_db.sql)