Czibulya Márk, Kovács Máté, Szklenár Andos Milán

Game2racker

Gyűjtsd össze a játékaidat!

Tartalom

[Bevezetés 3](#_Toc132959569)

[1. Felhasználói dokumentáció 5](#_Toc132959570)

[1.1. Főoldal 5](#_Toc132959571)

[1.2. Játék bemutató oldala 5](#_Toc132959572)

[1.3. Keresés 6](#_Toc132959573)

[1.4. Profil 7](#_Toc132959574)

[1.5. Beállítások 7](#_Toc132959575)

[1.6. Bejelentkezés 7](#_Toc132959576)

[1.7. Regisztráció 7](#_Toc132959577)

[1.8. Lábléc (footer) 8](#_Toc132959578)

[2. Felhasznált technológiák 9](#_Toc132959579)

[2.1 HTML 9](#_Toc132959580)

[2.2 CSS 9](#_Toc132959581)

[2.3 Javascript 10](#_Toc132959582)

[2.4 PHP 10](#_Toc132959583)

[2.5 Visual Studio Code 10](#_Toc132959584)

[2.6 Github 11](#_Toc132959585)

[2.7 XAMMP 11](#_Toc132959586)

[3. Adatbázis dokumentáció 12](#_Toc132959587)

[3.1. Adatbázis célja 12](#_Toc132959588)

[3.2. API használata 12](#_Toc132959589)

[3.3. Szükséges információk 14](#_Toc132959590)

[3.4. Kapcsolatok meghatározása 14](#_Toc132959591)

[3.5. Users tábla 16](#_Toc132959592)

[3.6. Sessions tábla 16](#_Toc132959593)

[3.7. RatingTable tábla 17](#_Toc132959594)

[3.8. Comment tábla 18](#_Toc132959595)

[3.9. Ratios tábla 18](#_Toc132959596)

[3.10. lists Tábla 19](#_Toc132959597)

[19](#_Toc132959598)

[3.11. listGames tábla 20](#_Toc132959599)

[3.12. games tábla 21](#_Toc132959600)

[3.14. Táblák 22](#_Toc132959601)

[4. Fejlesztői dokumentáció 23](#_Toc132959602)

[4.1. Kezdőlap 23](#_Toc132959603)

[4.2. Regisztráció 23](#_Toc132959604)

[4.3. Bejelentkezés 26](#_Toc132959605)

[4.4. Footer 27](#_Toc132959606)

[4.5. Hozzászólások 29](#_Toc132959607)

[4.1. Felmerült problémák a fejlesztés során 34](#_Toc132959608)

[Összefoglalás 35](#_Toc132959609)

[Források: 36](#_Toc132959610)

[Ábrajegyzék 37](#_Toc132959611)

# Bevezetés

Csapatunk 3 főből áll, Czibulya Márk, Kovács Máté és Szklenár Andos Milán. A témaválasztás kapcsán több ötletünk volt, ezek közül volt olyan is, amit el is kezdtünk megvalósítani, de különböző fejlesztési és feladat megosztási akadályokba ütköztünk.

Az első elképzelésünk a COVID-19 vírus okozta iskolaszünet kapcsán született meg, ugyanis a lezárás időszakában nem volt lehetőségünk a mindennapos iskolába járásra fél éven keresztül, ezért otthonról online oktatás formájában tudtunk részt venni a tanórákon. A bezártság kapcsán jutott az eszünkbe egy olyan szolgáltatás vagy koncepció, hogy együtt tudjunk videókat nézni az interneten. Így megálmodtuk a Gather2Watch-t.

Ilyen alkalmazás már létezik a világhálón, de kutakodva arra a döntésre jutottunk, hogy mi szeretnénk ezt egy tovább fejlesztett formában létrehozni, illetve ingyenessé tenni, mert jelenleg csak fizetős alkalmazás létezik az interneten.

Ezt úgy terveztük meg, hogy a weboldalunkra történő authentikációt követően a belépett felhasználók közösen, egy adott videót egyszerre tudják nézni a társaikkal, anélkül, hogy külön-külön meg kellene nyitniuk egy videómegosztó weboldalt, hanem csak egy linkre van szükségük a videó megtekintéséhez egyszerre egy időben. Ugyanakkor a regisztrált felhasználók a beépített chat felületen meg tudták volna beszélni a látottakat.

Ennek megvalósításához olyan websocket megoldást találtunk, amihez Nodejs backend technológiát kellett volna használnunk. El is kezdtük vele az ismerkedést, és a regisztrációt meg is tudtuk valósítani, valamint a chat is kiválóan működött, azonban itt elakadtunk. Egyrészt technológiai nehézségek miatt, mivel backend tantárgyból a PHP-t tanultuk és nem a Nodejs-t. Másrészt a feladatokat sem tudtunk igazán egyenlően elosztani egymás között, hogy az mindannyiunk számára értékelhető projekt legyen.

Ezt követően gondolkoztunk tovább, hogy valamilyen focival kapcsolatos alkalmazást készítsünk, de ez a téma nem érdekelte a csapatunk mindent tagját, így ez már az ötletelés fázisában elvetésre került.

Végül kézenfekvő volt a következő ötlet, mi az, ami mindenkit érdekel, és ez a számítógépes játékok. Ez alatt, nem egy konkrét játék fejlesztését terveztük, hanem szintén egy közösségi oldalban gondolkoztunk, ahová azok regisztrálhatnak, akik érdeklődnek a számítógépes játékok iránt, az általunk létrehozott felületen megoszthatják a tapasztalataikat, és információkat szerezhetnek a játékokról, szavazhatnak a különböző játékokra 1-től 10-ig terjedő pontszámozással.

A különböző feladatokat is fel tudtuk osztani egymás között. A csapatunkból Máté készítette el a regisztrációs, belépési felületet és a minden oldalon megjelenő „footert” ami olyan hasznos lehetőségeket kínál, mint például a hírlevélre való feliratkozás. Márk az Internet Game Database oldaláról a különböző lekérdezéseket írta meg, Milán pedig a főoldalon megjelenő tartalmat fűzte össze, valamint ha a felhasználó kattint a főoldalon megjelenő egyes játékok kártyáira, akkor onnan a kiválasztott játék egyedi oldalára jut, ahol további információk jelennek meg róla, valamint ide készítettük el a hozzászólási lehetőségeket is.

Fontos megjegyeznünk, hogy az alkalmazásunkat eleve angol nyelven készítettük el, mivel nem szeretnénk, ha a felhasználók köre egy szűk földrajzi környezetre korlátozódna. Jövőben célunk, hogy egy nemzetközi célcsoportot érjünk el, és a világ minden tájáról regisztráljanak. Ezt azért is tartjuk fontosnak, egyrészt az angol nyelv gyakorlása mellett, hogy nemzetközi kapcsolatokat lehet kiépíteni.

A megvalósításhoz a PHP, Javascript, MYSQL technológiákat használtuk. Az adatok tárolása adatbázisban történt, melyből PHP segítségével nyerjük ki az adatokat, ezáltal egy nagyon könnyen karbantartható weboldalt készítettünk.

Dolgozatunk első részében bemutatjuk a fentebb említett, felhasznált technológiákat, és a fejlesztői környezetet, ezt követően a felhasználói dokumentációban az oldal részletes leírását, a fejlesztői dokumentációban az oldal mind frontend mind pedig backend oldali programozását.

Külön fejezetbe gyűjtöttük össze a programozás során felmerült akadályokat, valamint azt, hogyan oldottuk meg ezeket. Végezetül az összefoglalásban értékeljük a közös munkánkat és ismertetjük azokat az ötleteinket, amelyek már nem kerültek megvalósításra, de úgy gondoljuk, hogy majd a jövőben mindenképpen érdemes lenne megvalósítanunk ezeket a tovább fejlesztési lehetőségeket.

# Felhasználói dokumentáció

A weboldalon számítógépes játékok gyűjteménye jelenik meg. A felhasználók regisztrálhatnak a weboldalra, értékelhetik a játékokat és a felhasználókat is a profiljuk és a hozzászólásaik alapján.

## Főoldal

A Game2Racker weboldalt megnyitva a főoldalon találjuk magunkat. A navigációs sáv közepén található egy két részre osztott keresőmező. Az első részben egy lenyitható választómenü látható, ahol a választhatunk a játékok, a DLC-k, a felhasználók által írt modok, illetve az oldalon regisztrált felhasználók között. Ha kiválasztjuk például a felhasználókat és végrehajtunk egy keresést, akkor a felhasználók profiljai jelennek meg az oldalon.

A jobb felső sarokban található meg a belépés és a regisztráció gomb. Bejelentkezés után a jobb felső sarokban megjelenik felhasználó neve, amelyre rákattintva megjelenik egy lenyíló menü melynek segítségével megtekinthetjük a saját profilunkat, vagy a beállításainkat esetleg ki is tudunk jelentkezni a fiókunkból.

A főoldalon találhatóak meg a legújabb megjelenésű játékok listája, illetve a legtöbbet játszott játékok a felhasználók értékelései alapján. Egy csúszka segítségével könnyen tudunk böngészni a játékok között. Megvalósításra került még egy lábléc is a főoldalon, amelyre több hasznos funkció került. Itt vissza tudunk navigálni a főoldalra a „Home” gombra kattintva, valamint található még egy ,,Profile” gomb is, mely a saját profiljára navigálja a felhasználót ha be van jelentkezve. Szintén megtalálható még egy úgynevezett ÁSZF, amely a chat funkció szabályait tartalmazza, valamint sok más hasznos dolog is, melyekre később részletesebben kitérünk.

## Játék bemutató oldala

Ha a főoldalon vagy a keresésből rákattintunk egy játékra, megjelenik egy új oldal, ahol a játékról minden lényeges információt megtalálunk. Megjelenik a játék borítóképe, a felhasználók értékelése, a listához adáshoz szükséges gomb, a játéknak a neve, megjelenési dátuma, játékmódja, kategóriája, fejlesztők és kiadók nevei, platformok melyekre megjelent az adott játék, mint például Steam vagy Epic Games ahol könnyedén megvásárolható a játék. Szintén található még egy rövid leírás a játékokról, egy videó illetve képek a játékról és a felhasználóink által írt hozzászólások.

Ha a felhasználó szeretne értékelni egy játékot akkor ahhoz be kell jelentkeznie, ellenkező esetben a csillagok nem kattinthatók. Egy 1-10 skálán értékelheti a játékot. A Score funkció a következő képpen működik: a felhasználók értékelésiek alapján számol egy átlagot és így jelenik meg egy kapott szám a weboldalon a Score mellett.

A pluszjelű gombbal egy saját maga által létrehozott listába tudja rakni a játékot, melyet egy felugró ablakban teheti meg. A felugró ablakban lehetősége van a felhasználónak új listát is létrehozni, nevet adni a listának és játékokat elmenteni a kiválasztott listába és törölni a létrehozott listáját.

## Keresés

A navbárban mikor kiválasztotta a felhasználó milyen keresést szeretne végrehajtani (All,Games,DLC,mods) és beírja a keresőbe a játék nevét vagy olyan szót amilyen játékot szeretne megtalálni egy új oldalon egymás alatti kártyákban, megjelennek a játékok képe és neve. Ezek a kártyák kattinthatók és átirányít az adott játék bemutató oldalára. A játék neve mellett még a listához adás is megtalálható, ha a felhasználó ismeri a játékot és egyből listához szeretné adni.

## Profil

A navbárban a felhasználónak lehetősége van megtekinteni saját profilját a profilképére kattintva vagy a nevére kattintva lefelé nyíló menüben a „View Profile”-ra lépve. A profil oldalon látható a felhasználó profilképe, a Score-ja amit más felhasználók adtak neki. A saját profilját nem tudja értékelni a felhasználó de más profilját igen amit ugyanúgy egy 1-10 skálán tud megtenni és átlagolja a Score-t mint ahogy a játékoknál.

A profilkép mellett megjelennek a felhasználó által létrehozott listák is ahol egy csúszkán látja az összes játékot, amit a listájához adott. A játékok alatt pedig látható felhasználó játékhoz tartozó értékelése. A profil oldal alján megjelenik a comment szekció.

## Beállítások

A navbárban a nevünkre kattintva lenyílik egy almenü ahol a fogaskerékre kattintva átvisz a beállítások oldalra. Itt a felhasználó be tud állítani egy profilképet egy linken keresztül (Képeket nem tárolunk az adatbázisban csak a hozzáférési útját) illetve a Steam profiljának a linkjét ami megjelenik a profilján egy kattintható Steam ikonként.

## Regisztráció

A regisztráció gombra kattintva, megjelenik egy regisztrációs felület, ahol a felhasználónak meg kell adnia egy általa választott egyedi felhasználónevet, az email címét, illetve a jelszavát. Létrehoztunk egy jelszómegerősítő mezőt is, hogy a regisztráló meggyőződön arról, hogy a jelszavát jól írta e be, ezáltal elkerülve a későbbi esetleges fennakadásokat. A regisztrációt nagyon fontosnak tartjuk, mivel nagyon sok funkció csak regisztrálás után érhető el, ennek érdekében több helyen is feltüntettük a regisztráció gombot az oldalunkon.

## Bejelentkezés

A bejelentkezés gombra kattintva megjelenik a belépési felület, ahol be tudunk lépni a profilunkba előzetes regisztrációt követően. A felhasználónak a belépéshez meg kell adnia az email címét illetve a regisztráció során megadott jelszavát. Itt egy olyan lehetősége is van a felhasználónak, ha például még nem rendelkezik regisztrációval az oldalunkon, akkor ,,Register Now” gombra kattintva megteheti ezt. Sikeres bejelentkezés a főoldalon találjuk magunkat. Ha a belépés sikertelen, akkor hibaüzenetet kapunk: ,,Incorrect password or email”. Ilyenkor lehetőségünk van újból megpróbálni bejelentkezni oldalunkra.

## Lábléc (footer)

Létrehozásra került a weboldal alján egy lábléc, mely nagyon hasznos információkat rejt a felhasználók számára. Itt lehetőségünk van olyan funkciók használatára, amit a felhasználó esetleg feljebb nem vett észre, mint például a bejelentkezés, regisztráció vagy éppen a kijelentkezés. Ezeket pontosan azért raktuk ide, hogy ha a felhasználó elfelejt regisztrálni vagy éppen belépni, akkor ezt még egy helyen könnyedén megtalálja. Ezért tartottuk ideálisnak a láblécbe elhelyezni ezeket a hasznos funkciókat.

Létrehozásra került még egy Home gomb, mely visszanavigál minket a főoldalra, egy Profile gomb, amelyre kattintva eljuthatunk a profilunkra könnyedén, illetve egy General Terms and Conditions, amely az általános szerződési feltételekre navigál minket, ahol megtekinthetjük a Game2Racker oldal legfontosabb szabályait.

A harmadik menüpontban megismerkedhetünk a közösségi oldalainkkal, ahol könnyedén felvehetjük a kapcsolatot az adminokkal, illetve hasznos posztokat, híreket is találhatunk a közösségi médiánkban. Facebook, Twitter, Instagram közösségi oldalakon leszünk megtalálhatók.

## Comment szekció

A Comment szekcióban a jelen oldalra, írhat hozzászólást a jelenleg belépett felhasználó. (legyen a jelen oldal egy adott felhasználó profilja, vagy egy adott játék oldala).

Egy comment felépítése:

Egy like és dislike gombbal reagálhat minden felhasználó egy hozzászólásra ez az összes like és dislike-ot számolja ezért lehet mínusz értékelése is egy hozzászólásnak. Megjelenik még a felhasználó avatárja és felhasználó neve is. A felhasználó névre kattintva átirányít a profiljára. Mellette látható az is, hogy mikor írta a felhasználó a hozzászólást. Egy felhasználó a saját hozzászólását is tudja törölni ha elgépelt valamit vagy csak szimplán meggondolta magát, Ha a felhasználó a saját profilján van akkor mindenkiét tudja törölni.

# Felhasznált technológiák

## HTML

HTML (HyperText Markup Language) egy szöveges leírónyelv, amelyet az interneten megjelenő oldalak létrehozásához használnak. Magyarul a "hipertext jelölőnyelv" kifejezést használhatjuk rá. Az HTML segítségével lehetővé válik az oldal szerkezetének és formázásának megadása, valamint különböző interaktív elemek (pl. hivatkozások, képek, videók, űrlapok) beillesztése az oldalra.

A HTML-t egy szerver oldali jelölő nyelvként használják, ami azt jelenti, hogy az oldalt kérő kliens a szerverről kapja meg az HTML forráskódját, amit a böngésző megjelenít a felhasználó számára. Az HTML forráskódja szöveges állományokban tárolódik, és a böngészők a kód alapján jelenítik meg az oldalt a felhasználó számára. A forráskódban az HTML elemeket címkék (tag)-ek között írják le, és az elemek tulajdonságait attribútumok segítségével állítják be.

Az HTML-t általában CSS (Cascading Style Sheets) segítségével formázzuk, ami lehetővé teszi az oldal megjelenésének (pl. színek, betűtípusok, elrendezés) testreszabását. Ezen kívül az HTML-t JavaScript segítségével is lehet interaktívvá tenni, ami lehetővé teszi az oldalakon előforduló események kezelését és az oldal dinamikus működését.

## CSS

A CSS (Cascading Style Sheets) egy nyelv, amely lehetővé teszi a weboldalak formázásának és kinézetének szabályozását. A CSS segítségével beállítható a szöveg betűtípusa, színe, háttér, elrendezése, stb. A CSS lehetővé teszi az oldalak egységes kinézetének kialakítását, és elkülöníti a tartalmat a formázástól, ami javítja a a weboldal karbantarthatóságát és a fejlesztői hatékonyságot. Egy stílus lapot több oldalhoz is hozzá rendelhetünk, így rugalmasságot és időt spórolhatunk velük. Ha a böngészőben megnyitunk egy oldalt akkor az egy úgynevezett cache (gyorsítótár) mappába lementi a stíluslapot, és, ha legközelebb az oldalra navigálunk, akkor nem kell megvárni, hogy letöltődjön a CSS fájl, mert a böngésző már a gyorsítótárban eltárolta azt.

A CSS használatával egy weboldal HTML kódjában található elemekhez rendelhetők stílusok, ami nagymértékben növeli a weboldal esztétikai megjelenését.

## Javascript

A JavaScript egy szkript nyelv, ami azt jelenti, hogy a fordítás nélkül, közvetlenül a böngészőben fut. A JavaScript-et általában HTML és CSS dokumentumokkal együtt használják, hogy weboldalakat hozzanak életre. A HTML adja meg az oldal alapstruktúráját, míg a CSS formázást és stílust ad az oldalnak. A JavaScript pedig lehetővé teszi az oldal interaktív funkcióinak létrehozását.

A JavaScriptet a W3C (World Wide Web Consortium) fejlesztette ki és az ECMAScript (European Computer Manufacturers Association) standard alapján működik. Széleskörben használt nyelv, minden modern webes böngészőben támogatja, és lehetővé teszi számos különböző eszközön való futtatását: asztali számítógépeken, mobil eszközökön, IoT eszközökön, stb.

## PHP

A PHP egy szerveroldali script nyelv, amelyet weboldalak és webalkalmazások készítésére használnak. Az "PHP: Hypertext Preprocessor" rövidítése, és egy nyílt forráskódú programozási nyelv, amelyet széles körben használnak dinamikus weboldalak létrehozására. A PHP kódot közvetlenül HTML kódba lehet beágyazni, és a kiszolgáló feldolgozza, mielőtt az eredményként kapott HTML küldené el a felhasználó webböngészőjébe.

A PHP alkalmas webfejlesztésre, mert széles körű webalkalmazásokat tudunk vele készíteni, egyszerű weboldalaktól a bonyolult webalapú rendszerekig. Nagy és aktív közösséggel rendelkezik, amely sok útmutatót és segítséget kínál egymásnak, és számos készen álló keretrendszert és könyvtárat tartalmaz, mint például a Laravel, CodeIgniter, Symfony és még sok más, amelyek segítenek a fejlesztőknek gyorsan és egyszerűen webalkalmazásokat készíteni.

A PHP többféle adatbázissal is kommunikálhat, beleértve a MySQL, PostgreSQL és SQLite, és kompatibilis számos különböző operációs rendszerrel, beleértve a Windows, Linux és macOS. A PHP jól működik más technológiákkal, mint például a HTML, CSS és JavaScript, amelyeket általában a weboldal elsődleges részének létrehozására használnak.

## Visual Studio Code

A Visual Studio Code egy ingyenes és nyílt forráskódú szoftverfejlesztői környezet, amelyet a Microsoft fejleszt. Segítségével többféle programozási nyelven írhatunk kódot, például C#, C++, JavaScript, Python. Emellett számos hasznos eszközzel rendelkezik, mint például a kód automatikus kiegészítése, hibakeresés, a kód formázása és a verziókezelés.

Az egyik legnépszerűbb szoftverfejlesztői környezetnek számító VSCode használatával egyszerűen és hatékonyan tudjuk fejleszteni a különböző projekteket, ráadásul több platformon is elérhető Windows, Mac, és Linux rendszeren egyaránt .

## Github

GitHub egy web alapú platform, amelyet főként verziókezelésre és együttműködő szoftverfejlesztésre használnak. Többféle funkciót biztosít a kód tárolók kezeléséhez, mint például forráskódkezelés és verziókezelő eszközök. A fejlesztők többnyire GitHub-ot használják, hogy meg tudják osztani kódjukat egymással, együttműködjenek projekteken és hozzájáruljanak az open-source szoftverekhez.

GitHub 2008-ban alakult és azóta az egyik legnagyobb kódtároló platformmá vált a világon. Milliók használják fejlesztők és szervezetek, köztük néhány legnagyobb tech cég is, mint a Microsoft, a Google és a Facebook.

## XAMMP

Az XAMPP egy ingyenes és nyílt forráskódú szoftvercsomag, amely tartalmazza az Apache web szerver, a MySQL adatbázis kezelő rendszer, valamint az PHP és a Perl programozási nyelveket. A neve a csomagban található komponensekre utal: "X" az operációs rendszer (Cross-platform), "A" az Apache, "M" a MySQL, "P" pedig a PHP rövidítése.

Az XAMPP célja, hogy egyszerűen telepíthető és használható környezetet biztosítson a webfejlesztéshez. Általában lokális számítógépeken használják, hogy teszteljék és fejlesszék a weboldalakat, mielőtt azokat az internetre feltöltik.

Az XAMPP telepítése és konfigurálása viszonylag egyszerű, sok dokumentáció és támogatás áll rendelkezésre az interneten. Az XAMPP használatával a webfejlesztők könnyen kialakíthatnak egy komplett fejlesztési környezetet a saját gépükön, amelyben tesztelhetik és finomíthatják a webalkalmazásokat.

Jelen esetünkben, mi a XAMMP-ot nem lokális gépen használtuk, hanem a PhpMyAdmin-t összekötöttük a szerverünkkel.

# Adatbázis dokumentáció

## Adatbázis célja

Az adatbázisunk célja, hogy a weboldalon regisztráló felhasználók adatait, és a weboldalon történő funkciókat kezeljük és eltároljuk. Az oldalunkon lehetőség van a játékok, illetve a felhasználók értékelésére, hozzászólásra, illetve a hozzászólások értékelésére is, ezért tartottuk nagyon fontosnak egy relációs adatbázis létrehozását.

## API használata

Az API az "Application Programming Interface" rövidítése. Az API egy olyan szoftver interfész, amely lehetővé teszi a különböző szoftveralkalmazások közötti kommunikációt és adatcsere lehetőségét.

Az API-k általában előre meghatározott szabványokat és protokollokat használnak, amelyek lehetővé teszik az alkalmazások közötti adatátvitelt. Az API-k lehetővé teszik, hogy az alkalmazások különféle funkciói és adatok elérhetők legyenek más alkalmazások számára, hogy azok használni tudják őket. Például egy webhely, mint például a Facebook, rendelkezik egy API-val, amely lehetővé teszi a harmadik feleknek, hogy az alkalmazásukba integrálják a Facebook bejelentkezési rendszert vagy a Facebook felhasználók adatait.

Az API-k használata széles körben elterjedt a szoftverfejlesztésben, és lehetővé teszi az alkalmazások közötti integrációt, ami növeli az alkalmazások funkcionalitását és használhatóságát.

Ahhoz ,hogy az oldalunkon több mint kétszázezer játékot megjelenítsünk, keresnünk kellett egy olyan megoldást, ahol nem mi manuális visszük fel a saját adatbázisunkba a játékokról az adatokat.

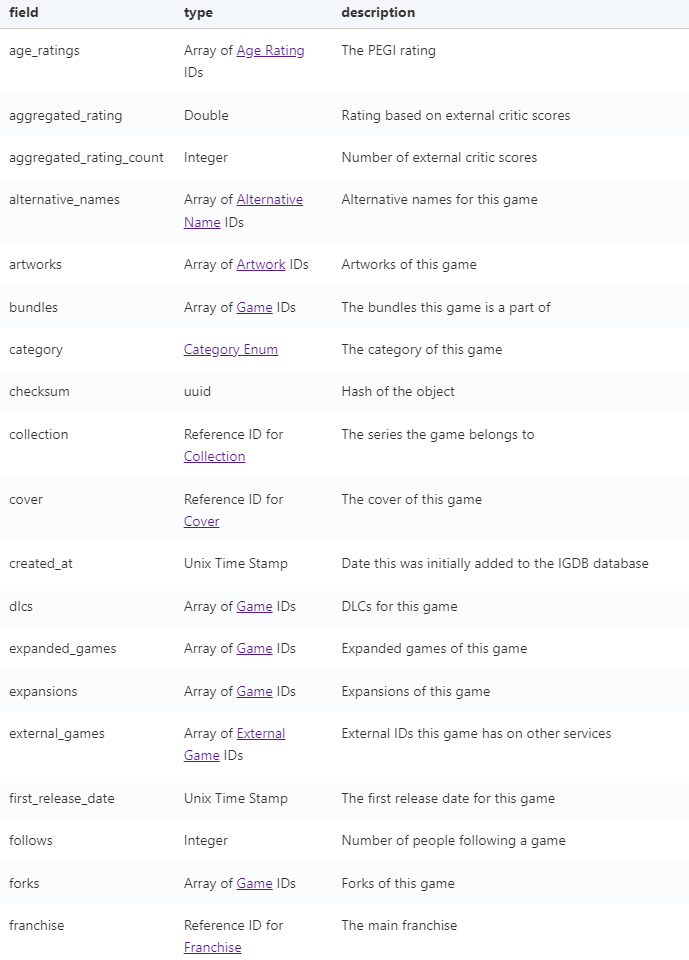
Egy olyan adatbázis forrást kellett találnunk, amit ingyenesen is fel tudunk használni, így találtunk meg az Internet Game Database (IGDB) nevű oldalt.

Az IGDB-hoz tartozik egy Developer API dokumentáció (<https://api-docs.igdb.com/#getting-started>) ami részletesen leírja, mit kell tenni ahhoz, hogy elérjük az adatbázist. Egy Twitch fiókon keresztül kellett kérnünk egy Tokent amit az IGDB oldalán kellett regisztráltatni, hogy megkapjuk a hozzáférést.

Az adatbázisukhoz való hozzáférést Postman-ben végeztük ahol SQL-hez hasonló lekérdezéseket végeztünk pl.:

*fgame.name, url, game.genres.name,game.first\_release\_date,game.summary,game.platforms.slug,game.platforms.name, game.total\_rating; sort total\_rating\_count desc; where game.platforms.slug = "win" & game.total\_rating >= 90; l 500;*

Ezzel a lekérdezéssel minden 90-nél többre értékelt játékról lekérünk adatokat mint a nevét, kategóriáját ,megjelenési dátumát ,leírását. Pontosan mi sem tudjuk, hogy hány játék található meg az oldalunkon, mivel a játékok száma napról napra változik, de nagyjából 230.000 játékkal dolgozunk.

A dokumentáció részletesen végigmegy a különböző lekérdezési módokon.

Az API-nak korlátai is vannak mivel az API-t szolgáltató cég nem szeretné, hogy leterhelve legyen a saját szervere. A korlátok a következők: egyszerre 500 játék adatait kérhetjük le és másodpercenként 4 különböző lekérdezést folytathatunk egy oldalon.

Api-ból lekért mezők amiket használunk az oldalon:

|  |  |
| --- | --- |
| id | Játékhoz tartozó egyedi azonosító |
| cover | Játék borítója |
| name | Játék neve |
| release\_dates | Játék megjelenési ideje |
| game\_modes | Játék módja |
| genres | Játék kategóriája |
| involved\_companies | Játékfejlesztő cég és kiadója |
| platforms | Minden platform, amire a játék megjelent |
| websites | Játékhoz tartozó weboldalak |
| summary | Játékhoz tartozó leírás |
| videos | Játékhoz tartozó videók |
| screenshots | Játékhoz tartozó képek |
| rating | Játékhoz tartozó értékelés az IGDB oldala szerint |
| follows | Játékhoz tartozó játékosok álltai követés az IGDB oldala szerint |
| dlcs | Játékhoz tartozó letölthető tartalmak |

## Szükséges információk

kieg.

kieg.

táblanevek!

* Felhasználó adatai: felhasználónév, email cím, jelszó, profilkép (avatar)
* Sessions: A regisztrált felhasználók alkalmankénti bejelentkezését kezeli.
* Felhasználó értékelése: A felhasználók értékelhetik egymás profilját az oldalon történő aktivitásuk alapján, illetve magát a játékokat is értékelhetik.
* Hozzászólások értékelése: A felhasználók hozzászólhatnak a játékok alá, illetve egymás profilja alá is. A játék, illetve a profil kommentelés külön táblában szerepel.

## Kapcsolatok meghatározása

**user => sessions:** 1:N-es kapcsolat, mivel egy felhasználóhoz több session is tartozhat egy időben de fordítva ez nem így van, egy sessionhöz csak egy felhasználó tartozhat

**user => comment:** 1:N-es kapcsolat, mivel egy felhasználóhoz több komment tartozhat, egy hozzászólásnak csak egy írója lehet.

**user => ratingTable:** 1:N-es kapcsolat, mivel egy felhasználóhoz több értékelés tartozhat, de egy értékelés egyedelőfordulása csak egy felhasználóé lehet (aki értékelte).

**user => ratios:** 1:N-es kapcsolat, mivel egy felhasználóhoz több értékelés (like, dislike) tartozhat, de egy értékeléshez csak egy felhasználó tartozhat.

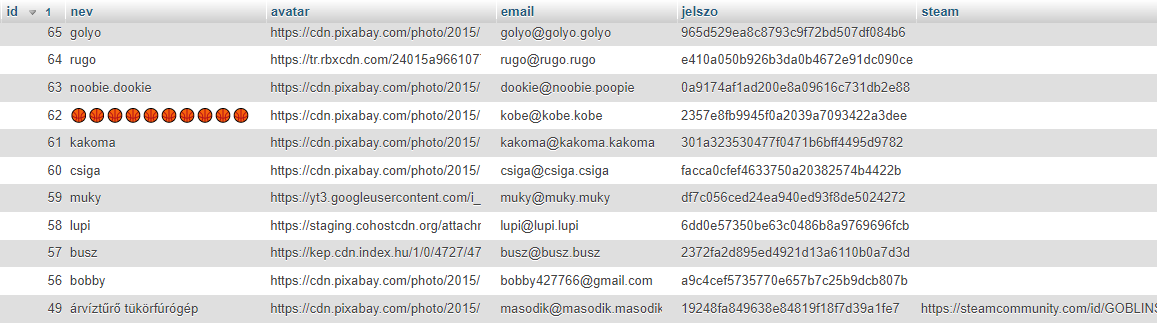
**komment => ratios :** 1:N-es kapcsolat, mivel egy kommenthez szintén több értékelés tartozhat, de az értékelés egy egyedelőfordulása nem tartozhat több hozzászóláshoz.

**lists <=> games:** N:M kapcsolat, mivel egy listához több játék tartozik, egy játék is tartozhat és ez fordítva is igaz, egy játékhoz több lista is tartozhat. Felbontottuk két darab 1:N-es kapcsolatra.

**lists => listGames:** 1:N kapcsolat, a list egy egyedelőfordulásához a listGames több egyedelőfordulása tartozhat, de fordítva ez nem igaz.

**games => listGames**: **:** 1:N kapcsolat, a games egy egyedelőfordulásához a listGames több egyedelőfordulása tartozhat, de fordítva ez nem igaz.

## Users tábla

A felhasználói adatok tárolásáért felel.

1. ábra Users tábla

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| nev | varchar(255) | Felhasználónév. kötelező kitölteni a regisztráció során |
| avatar | varchar(255) | A felhasználó által beállítható kép, amely a felhasználó profiljánál jelenik meg és ide a kép linkjét lehet feltölteni. |
| email | varchar(50) | Email cím, kötelező kitölteni a regisztráció során. |
| jelszo | varchar(255) | Jelszó, amit a regisztráló megad, ez titkosítva kerül az adatbázisba biztonsági okok miatt. Kötelező kitölteni. |
| steam | varchar(200) | A felhasználó megadhatja a steam profil linkét, amit itt tárolunk el. |

## Sessions tábla

Eltárolja a felhasználók munkameneteit.

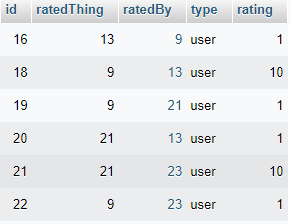
|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| UserID | int | A felhasználó azonosítója, akihez az adott session tartozik. |
| active | tinyint(1) | Az active mező arra szolgál, hogy éppen a felhasználó munkamenete aktív-e, tehát nem járt le. |
| token | varchar(255) | Egy egyedi kulcs, mely munkamenet azonosítót jelent. |
| acquired | datetime | A session kezdési időpontját jelenti, amikor például a felhasználó bejelentkezik a profiljába. |
| expires | datetime | A session lejárati időpontját jelenti. |

#### ábra Session tábla

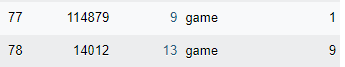
## RatingTable tábla

Ebben a táblában a felhasználók és a játékok értékelésére vonatkozó sorokat tároljuk.

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| ratedThing | int | Az adott értékelt játék vagy felhasználó azonosítója, a type mezőtől függ hogy ez melyikre vonatkozik |
| ratedBy | int | Ki értékelte az adott felhasználót vagy játékot. |
| type | varchar(10) | Típus, ami vagy user vagy game. |
| rating | int | Maga az értékelés jelenik meg ebben a mezőben. 1-10 ig terjed. |



#### ábra Rating table



2. ábra Rating tábla

## Comment tábla

Ebben a táblában a hozzászólásokhoz kapcsolódó sorokat tároljuk.

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| madeBy | int | Ki írta az adott a hozzászólást. |
| madeOn | int | Az adott értékelt játék vagy felhasználó azonosítója, a type mezőtől függ hogy ez melyikre vonatkozik |
| type | varchar(10) | Típus, ami vagy user vagy game. |
| date | datetime | Mikor írták az adott hozzászólást. |
| text | varchar (6000) | A hozzászólás szövege, maximum 600 karakter |

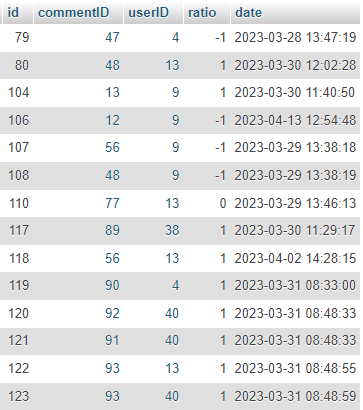


3. ábra Comment tábla

## Ratios tábla

Itt található a hozzászólások értékelése.

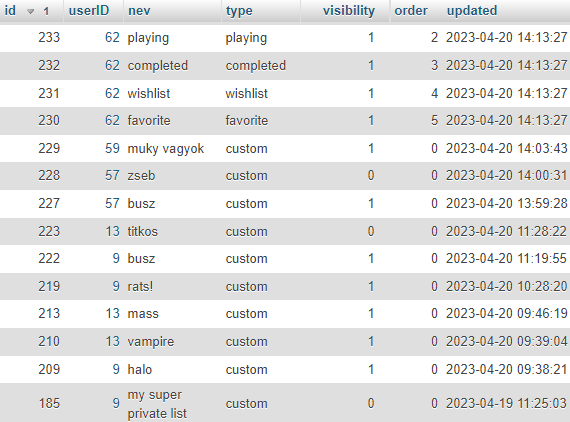
|  |  |  |
| --- | --- | --- |
| mező | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| commentID | int | Annak a kommentnek az azonosítója amihez ez az értékelés tartozik |
| userID | int | A felhasználó azonosítója. |
| ratio | tinyint(1) | Lehet (+1) és (-1). Ha pozitív akkor az értékelés tetszik a felhasználónak, ha pedig negatív akkor nem tetszik. Akkor lehet az értéke 0, ha az adott felhasználó visszavonja az értékelését mivel ilyenkor nem törli a sort az adatbázisból |
| date | datetime | Mikor történt az adott értékelés |



#### ábra Ratios tábla

## lists Tábla

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| userID | int | A felhasználó azonosítója, akihez a lista tartozik. |
| nev | varchar(200) | Lista neve. |
| type | varchar(12) | Lista típusa. |
| visibility | tinyint(1) | Lista láthatósága, lehet privát vagy publikus. (0 vagy 1) |
| order | int | Milyen sorrendben jelenik meg a lista. |
| updated | datetime | Mikor lett frissítve a lista. |



## listGames tábla

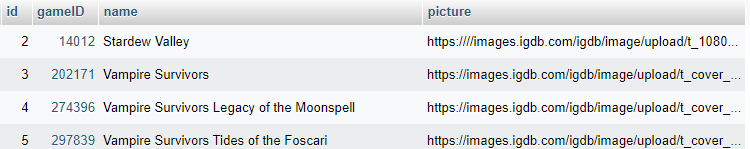
Kapcsolótábla a games és a lists között

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| listID | int | A lista egyedi azonosítója. |
| gameID | int | A játék azonosítója. |
| date | datetime | Mikor került be a listába az adott játék. |



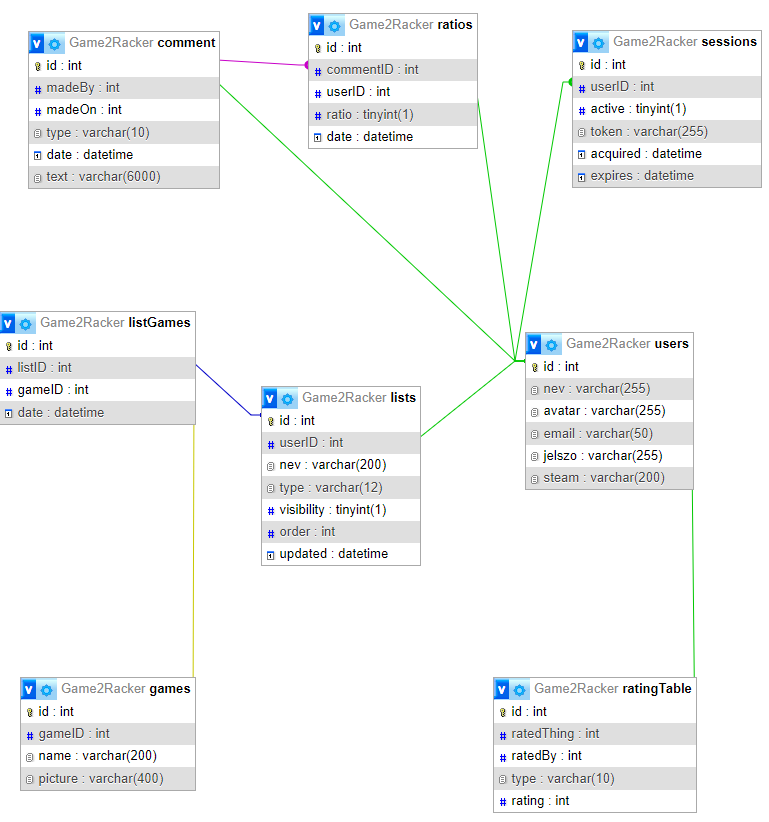
## games tábla

|  |  |  |
| --- | --- | --- |
| **mező** | **típus** | **leírás** |
| ID | int | Egyedi azonosító, ami folyamatosan növekvő szám. |
| gameID | int | A játék egyedi azonosítója.  Ez az azonosítója a játéknak az IGDB adatbázisban |
| name | varchar(200) | A játék neve. |
| picture | varchar(400) | A játék képe. |



## Táblák

Egy 8 táblából álló adatbázist hoztunk létre.



# Fejlesztői dokumentáció

## Kezdőlap

## Regisztráció

A regisztráció megvalósításához szintén a PHP-t választottuk, mert így könnyedén létre tudtunk hozni egy egyszerűen működő regisztrációs űrlapot. Az űrlapban a regisztrálni kívánó személyeknek meg kell adniuk a felhasználónevüket, email címüket, a jelszót, illetve a beírt jelszót meg is kell nekik erősíteni, hogy biztosan jól írták-e be, ezzel elkerüljük azt, hogy esetleg egy rossz jelszó megadása esetén a felhasználónak úgy kell találgatnia a megadott jelszó után.

A kódunk először ellenőrzi, hogy az űrlapot elküldték-e a "submit" gomb megnyomásával. Ha igen, akkor a kód feldolgozza a felhasználó által megadott adatokat és ellenőrzi, hogy a megadott e-mail cím vagy felhasználónév már létezik-e az adatbázisban. Ha létezik, akkor hibaüzenetet jelenít meg és nem engedi a felhasználónak a regisztrációt. Ha nincs ilyen felhasználó, akkor ellenőrzi a megadott jelszavak egyezőségét. Ha a jelszavak egyeznek, akkor hozzáadja az új felhasználót az adatbázishoz.

A kód használja a POST módszert a bejelentkezési adatok elküldéséhez a szervernek, biztonságosabbá teszi az adatokat az SQL Injection támadások elleni védelem érdekében. A jelszó md5 hash-elt, ami biztonságosabb, mint a sima szövegjelszó tárolása, bár a hash függvények ma már nem javasoltak jelszavak hash-elésére, de jelenlegi ismereteink alapján választottuk ezt a titkosítási módszert.

Az űrlap tartalmazza a regisztrációhoz szükséges mezőket, és egy "Register now" gombot a regisztráció elküldéséhez. Ha a felhasználó már rendelkezik fiókkal, akkor az "Already have an account?" felirat alatt található linkre kattintva átirányítják őt a bejelentkezési oldalra.

<?php

require("../depend/connection.php");

if(isset($\_POST['submit'])){

$name = htmlspecialchars($\_POST['name']);

$email = htmlspecialchars($\_POST['email']);

$pass = md5($\_POST['password']);

$cpass = md5($\_POST['cpassword']);

$select = "SELECT \* FROM `users` WHERE email = ? OR nev = ?";

$stmt = $conn->prepare($select);

$stmt->bind\_param('ss', $email,$name);

$stmt->execute();

$result = $stmt->get\_result();

$stmt->reset();

if($result->num\_rows > 0){

$error[] = 'User already exist!';

}else{

if($pass != $cpass){

$error[] = 'Password not matched!';

}else{

$insert = "INSERT INTO `users`(nev, email, jelszo) VALUES(?,?,?)";

$stmt = $conn->prepare($insert);

$stmt->bind\_param('sss', $name,$email,$pass);

$stmt->execute();

$stmt->reset();

echo $name;

header("Location: login.php");

exit();

}

}

};

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Game2Racker</title>

<link rel="stylesheet" href="../css/login.css">

</head>

<body>

<div class="form-container">

<form action="" method="post">

<h3>Register now</h3>

<?php

if(isset($error)){

foreach($error as $error){

echo '<span class="error-msg">'.$error.'</span>';

};

};

?>

<input type="text" name="name" required placeholder="Enter your name">

<input type="email" name="email" required placeholder="Enter your email">

<input type="password" name="password" required placeholder="Enter your password">

<input type="password" name="cpassword" required placeholder="Confirm your password">

<input type="submit" name="submit" value="Register now" class="form-btn">

<p id="account">Already have an account? <a href="login.php">Login now</a></p>

</form>

</div>

</body>

</html>

## Bejelentkezés

Létrehoztunk a már említett bejelentkezési felületet is, hogy az oldalon történő hasznos funkciókat a felhasználók könnyedén tudják használni. A bejelentkezésnél a kódunk használ egy MySQL adatbázist, amit egy kapcsolatfájl (connection.php) segítségével ér el.

A megírt script a bejelentkeztetési adatokat (e-mail és jelszó) ellenőrzi, majd ha azok helyesek, akkor létrehoz egy munkamenetet (session), amelyet a sessions táblában tárolunk. Az aktív munkamenet azonosítására egy token-t generál, amelyet a felhasználó böngészőjébe helyezett süti (cookie) tárol. Az aktív munkameneteknek egy lejárati idejük van, amelyet 3 napra állítottunk. A munkamenet azonosítója és a felhasználó ID-je szintén a sessions táblában van tárolva.

if(isset($\_POST['submit'])){

$name = mysqli\_real\_escape\_string($conn, $\_POST['name']);

$email = mysqli\_real\_escape\_string($conn, $\_POST['email']);

$pass = md5($\_POST['password']);

$select = "SELECT \* FROM `users` WHERE email = ? && jelszo = ?";

$stmt = $conn->prepare($select);

$stmt->bind\_param('ss', $email,$pass);

$stmt->execute();

$result = $stmt->get\_result();

$row = $result->fetch\_assoc();

$stmt->reset();

if($result->num\_rows == 1){

$bytes = random\_bytes(20);

$token = bin2hex($bytes);

$newId = $row['id'];

$stmt = $conn->prepare("INSERT INTO `sessions`(`userID`, `active`, `token`, `expires`) VALUES (?,true,?,DATE\_ADD(CURRENT\_TIMESTAMP(), INTERVAL 3 DAY));");

$stmt->bind\_param('ss', $row['id'],$token);

$stmt->execute();

$stmt->reset();

setcookie("session", $token, time() + (60\*60\*24\*7), "/");

Ha a sikeres bejelentkezés megtörténik, akkor a script átirányítja a felhasználót az aktuális oldalra ($\_SESSION['current\_page']), amelyet a munkamenet létrehozása előtt azonosítottak.

## Footer

Ahhoz, hogy még több lehetőséget kínáljon a weboldalunk létrehoztunk egy úgynevezett láblécet, mely különböző funkciókat, illetve fontos információkat kínál.

Az oszlopok címei "quick acces", "navigation" és "follow us". Az első oszlop tartalmaz lista elemeket, amelyek tartalmazzák a "Login", "Register" és "Logout" hivatkozásokat. Az "id" attribútummal rendelkező "kijelentkezes" elem egy bejelentkezett felhasználó kijelentkezési lehetőségét biztosítja. A láblécen történő kijelentkezés után a főoldalon találjuk magunkat.

A második oszlop címe a "navigation" tartalmaz egy "home" hivatkozást, amely visszavezet minket a kezdőoldalra. Az echo függvény segítségével egy dinamikus "profile" hivatkozást is létrehoztunk a felhasználó nevével és azonosítójával. Ezenkívül tartalmaz egy "General Terms and Conditions" hivatkozást is, amelyre kattintva eljuthatunk az Általános Szerződési Feltételekhez és részletesen áttekinthetjük ezeket.

A harmadik blokk a "follow us" címszó alatt található, amely lehetővé teszi a felhasználóknak, hogy a weboldal szociális média oldalaira navigáljanak, mint például a Facebook, Twitter és Instagram oldalakra.

<footer class="footer">

<div class="footer-container">

<div class="row">

<div class="footer-col">

<h4>Quick access</h4>

<ul>

<li><a href="login/login.php">Login</a></li>

<li><a href="login/register.php">Register</a></li>

<li id="kijelentkezes"><a href="#">Logout</a></li>

</ul>

</div>

<div class="footer-col">

<h4>Navigation</h4>

<ul>

<li><a href="index.php">home</a></li>

<?php

echo "<li><a href=\"profile.php?user=" . $userData['nev'] . "&userid=" . $userData['userID'] . "\">profile</a></li>";

?>

<li><a href="termsangol.php">General Terms and Conditions</a></li>

</ul>

</div>

<div class="footer-col">

<h4>follow us</h4>

<div class="social-links">

<a href="https://www.facebook.com/"><i class="fab fa-facebook-f"></i></a>

<a href="https://twitter.com/Game2Racker"><i class="fab fa-twitter"></i></a>

<a href="https://www.instagram.com/"><i class="fab fa-instagram"></i></a>

</div>

</div>

</div>

</footer>

## Hozzászólások

$commentField = "<section class=\"commentSection\">

<div class=\"innerComments flexcol littleGap\">

<h1>Comments</h1>

<form class=\"writeComment flexcol\">

<h2>Write a comment...</h2>

<textarea name=\"\" id=\"{$commentInput}\" cols=\"30\" rows=\"10\" style = \"resize: none\" onkeypress=\"Enter(event)\"></textarea>

<input type=\"submit\" value=\"Comment\" class=\"comment-send\" id=\"{$sendComment}\">

</form>

<div class=\"commentContainer littleGap\">";

Ebben a kódsorban megkezdjük felépíteni a hozzászólás szekciót, az egészet egy php változóban tároljuk el stringként (ami jelen esetben html kód lesz). A végén az egészet ki írjuk.

$commentsSQL = "SELECT

comment.id AS 'comment\_id' ,comment.parentID,comment.madeBy,comment.madeOn,comment.type,comment.date,comment.text, `users`.id AS 'user\_id' , `users`.nev, `users`.avatar, IFNULL(SUM(ratios.ratio),0) as ratio

FROM `comment` LEFT JOIN `users` ON comment.madeBy = `users`.id

LEFT JOIN `ratios` ON ratios.commentID = comment.id

WHERE comment.type = '{$pageData['type']}' AND comment.madeOn = '{$pageData['id']}' GROUP BY comment.id ORDER BY comment.date DESC;";

$queriedComments = mysqli\_query($conn, $commentsSQL);

Ez a PHP kód egy SQL lekérdezést hajt végre az adatbázisban tárolt kommentek lekérdezésére. Lekérdezzük az összes hozzászólást a pagedata alapján, és a hozzá tartozó szavazatokat a commentid idegen kulcs alapján. LEFT JOINT használunk, hogy azok a hozzászólások is megjelenjenek amikhez nem tartozik szavazat.

GROUP BY comment.id miatt tudjuk SUM-olni a ratiokat- amik egy egyedelőforduláshoz tartoznak, hogy egyszerre egy komment csak egyszer legyen kilistázva, mellette kollektívan a ratio tábla lényege

látszik hogy egy kommentnek milyen egy kommentnek az értéke(felhasználók által)

//IFNULL, ha egy komment még nem lenne értékelve akkor NULL érték helyett 0-t tudunk adni neki így ha ki akarjuk írni ez sem fog hibát okozni.

if (mysqli\_num\_rows($queriedRatio) > 0) {

$ratiorow = mysqli\_fetch\_array($queriedRatio);

if ($ratiorow['ratio'] > 0) {

$commentField .= "<div class=\"upbutton\"><i class=\"fa fa-thumbs-up liked\"></i></div>

<div class=\"downbutton\"><i class=\"fa fa-thumbs-down\"></i></div>";

}

else if($ratiorow['ratio'] < 0) {

$commentField .= "<div class=\"upbutton\"><i class=\"fa fa-thumbs-up\"></i></div>

<div class=\"downbutton\"><i class=\"fa fa-thumbs-down disliked\"></i></div>";

}

else {

$commentField.="

<div class=\"upbutton\"><i class=\"fa fa-thumbs-up\"></i></div>

<div class=\"downbutton\"><i class=\"fa fa-thumbs-down\"></i></div>";

}

}

else {

$commentField.="

<div class=\"upbutton\"><i class=\"fa fa-thumbs-up\"></i></div>

<div class=\"downbutton\"><i class=\"fa fa-thumbs-down\"></i></div>";

}

Ez a kódsor a tetszik és nem tetszik gombok megjelenéséért felel. Ha a belépett felhasználó interaktált egy kommenten ezekkel (like -ot, vagy dislike -ot adott) akkor azt megfelelően beszínezi.

Ha a like -ot adott akkor azt a stringet adja hozzá a komment php változóhoz amiben a html kód tartalmazza azt a classt ami a like gomb szinezéséért felel

ha dislike-ot akkor azt, ha meg nem interaktált még a felhasználó a jelen vizsgált kommentel akkor alap kinézettel kerül oda.

{$sendComment}.addEventListener('click',function $functName(event){

event.preventDefault();

if ({$commentInput}.value != \"\") {

{$sendComment}.removeEventListener('click',$functName);

let dataForPHP = new FormData();

dataForPHP.append(\"madeOn\", {$pageData['id']});

dataForPHP.append(\"text\", {$commentInput}.value);

dataForPHP.append(\"type\", \"{$pageData['type']}\");

fetch(`depend/pushComment.php`, {

method: \"POST\",

body: dataForPHP

})

.then(response => response.text())

.then(data => {

console.log(data)

if (data == \"NOPLIZ\") {

{$commentInput}.value = \"\";

{$sendComment}.addEventListener('click',$functName);

window.location.reload();

}

else if(data == \"GANTZ\"){

setTimeout({$sendComment}.addEventListener('click',$functName),5000)

}

})

.catch(error => console.log(error));

console.log(\"gag\");

}

});

először megnézzük hogy üres e a komment input mező és csak akkor haladunk tovább ha nem az.

létrehozunk egy formdata változót és hozzá adjuk a komment adatait: mi alá került a komment: a játék vagy felhasználó idje; a típusa: játék vagy felhasználó alá került; és magát az üzenetet.

fetch apival meghívjuk a pushcomment.php filet post metódussal átadjuk neki a formDatát.

a pushComment file ezeket a post-olt form adatokat várja.

let downbuttons = document.querySelectorAll(\".downbutton\");

function PushVotes(element,like,dislike,val,stat=\"\"){

let parentID = element.parentElement.parentElement.parentElement.parentElement.id;

let ratio = \"\";

if (stat == \"plus\") {

if (like.classList.contains(\"liked\")) {

val.innerHTML = parseInt(val.innerHTML) -1;

like.classList.remove(\"liked\");

ratio = 0;

}

else if(dislike.classList.contains(\"disliked\"))

{

val.innerHTML = parseInt(val.innerHTML) +2;

dislike.classList.remove(\"disliked\");

like.classList.add(\"liked\");

ratio = 1;

}

else{

val.innerHTML = parseInt(val.innerHTML) +1;

dislike.classList.remove(\"disliked\");

like.classList.add(\"liked\");

ratio = 1;

}

}

else if (stat == \"minus\") {

if (dislike.classList.contains(\"disliked\")) {

val.innerHTML = parseInt(val.innerHTML) +1;

dislike.classList.remove(\"disliked\");

ratio = 0;

}

else if(like.classList.contains(\"liked\"))

{

val.innerHTML = parseInt(val.innerHTML) -2;

like.classList.remove(\"liked\");

dislike.classList.add(\"disliked\");

ratio = -1;

}

else{

val.innerHTML = parseInt(val.innerHTML) -1;

like.classList.remove(\"liked\");

dislike.classList.add(\"disliked\");

ratio = -1;

}

}

A like és dislikeokat kezeli, ha már be van like olva akkor le veszi róla és levon egyet a számból, ugyan ez a dislike val csak ott hozzá ad, ha meg likeről rakja át dislike ra akkor

kettőt von ki (1- le veszi a like ott, és még 1, rárakja a dislikeot). Ugyan ez a dislikeről like ra csak ott 2-t ad

## Felmerült problémák a fejlesztés során

Munkáink során nagyon sok akadályba ütköztünk. Először is egy teljesen más projektet álmodtunk meg a csapattal és elkezdtük ezt megvalósítani. Ez volt a Gather2Watch, ami egy úgymond együtt néző alkalmazás létrehozása volt. Elég jól haladtunk vele az elején, aztán amikor következett a Nodejs használata sajnos rájöttünk, hogy ismeretek hiányában más projektet kell választanunk.

Amikor a felhasználó interaktál a like és a dislike gombokkal, akkor eltelik egy kis idő az interakció és annak az adatbázisba kerülése között. Ezt ha megismétli a felhasználó, akkor mivel az adatbázisba még nincs adat a szelekció elengedi maga mellett és a többi interakció is felkerül időközben.

# Összefoglalás

Elért eredmények, tovább fejlesztési lehetőségek

Amikor már az oldalunk olyan mennyiségű látogatót tud felmutatni, amely alapján a Google engedélyezi a reklámjainak az elhelyezését, ezekből bevételre is szert tudunk tenni, amiből majd finanszírozhatjuk szerverbérletet, a további fejlesztői munkáinkat is. Ebből fakadóan született meg egy másik tovább fejlesztési ötletünk is, a VIP regisztráció, ami annyit jelentene, ha valaki hajlandó egy jelképes összeget fizetni, akkor nem jelennének meg a profiljába belépve a felületen a reklámok. Továbbá a VIP regisztrációval rendelkezőknek lehetőségük lenne különböző avatarokat választani, melyekre a normál regisztrációval rendelkezőknek nincs lehetőségük. Egy VIP jelvény is beépítésre kerülne, így a felhasználók láthatnák, hogy például az adott felhasználó VIP regisztrációval rendelkezik, mikor megnézik egymás profilját. Így viszont majd egy fizetési rendszert is ki kell alakítani, hogy pl. PayPalon keresztül tudjanak fizetni a felhasználók a reklámmentes felületért, illetve a VIP regisztrációért.

Mivel mindannyian folytatjuk tanulmányainkat ezen a területen, minden bizonnyal a későbbiekben fogunk hozzá készíteni mobil applikációt is.

# Források:

abc betűrendben: Szerző, cím, megjelenés dátuma, ha van, link (excelben érdemes gyűjteni és sorba rendezni)

# Ábrajegyzék

[1) ábra User tábla magyarázata 16](#_Toc132708257)

[2) ábra User tábla 16](#_Toc132708258)

[3) ábra Session tábla magyarázata 17](#_Toc132708259)

[4) ábra Session tábla 17](#_Toc132708260)

[5) ábra RatingTable magyarázata 17](#_Toc132708261)

[6) ábra Rating table 17](#_Toc132708262)

[7) ábra Comment tábla magyarázata 18](#_Toc132708263)

[8) ábra Comment tábla 18](#_Toc132708264)

[9) ábra Ratios tábla magyarázata 19](#_Toc132708265)

[10) ábra Ratios tábla 19](#_Toc132708266)