BGSzC Pestszentlőrinci Közgazdasági és Informatikai Szakgimnáziuma

1184 Budapest Hengersor 34.

1. Záró dolgozat

Játék Fórum

Konzulens tanár: Készítette:

Bencze István Herczegfalvi-H. Áron

Tartalom

[1 Bevezetés 3](#_Toc1525023802)

[1.1 Feladat leírás 3](#_Toc957017017)

[1.2 A felhasznált ismeretek 3](#_Toc112859204)

[1.3 A felhasznált szoftverek 4](#_Toc1497629432)

[2 Felhasználói dokumentáció 5](#_Toc1306443123)

[2.1 A program általános specifikációja 5](#_Toc2143618530)

[2.2 Rendszerkövetelmények 5](#_Toc1591163845)

[2.2.1 Hardver követelmények 5](#_Toc1700339184)

[2.2.2 Szoftver követelmények 6](#_Toc631023026)

[2.3 3. A program telepítése 6](#_Toc988686508)

[2.4 A program használatának a részletes leírása 11](#_Toc1011243290)

[3 Fejlesztői dokumentáció 16](#_Toc1160597969)

[3.1 Az alkalmazott fejlesztői eszközök 16](#_Toc315030325)

[3.2 Adatmodell leírása 18](#_Toc28116281)

[3.3 Részletes feladatspecifikáció, algoritmusok 21](#_Toc1025000380)

[3.4 Tesztelési dokumentáció 24](#_Toc987217259)

[4 Összefoglalás 26](#_Toc2013411872)

[4.1 Önértékelés 26](#_Toc100918224)

[4.2 Továbbfejlesztési lehetőségek 27](#_Toc765012318)

[5 Felhasznált irodalom 28](#_Toc1436628119)

[6 Ábrajegyzék 29](#_Toc690447064)

# Bevezetés

## Feladat leírás

Az internet közösségi korát éljük, lényegében mindegyik népszerű platform rendelkezik valamilyen szintű közösségi elemmel, ami miatt szociális hálónak hívhatjuk, ezeknek az oldalaknak szoktak fórum szerű részei lenni, de ez volt a leglényegesebb számomra, ezért szerettem volna létrehozni ezt a fórumot.

A Játék Fórum jelenleg egy baráti társaságnak készül, de a potenciál a bővülésre megvan, akár széleskörű adoptációra is kész lehet pár módosítással, éles szerverek bevetésével, akár teljes közösségihálóvá nőheti ki magát.

Különféle közösségek létrehozására, posztok és kommentek létrehozására, értékelésére van lehetőség ezen a weboldalon.

## A felhasznált ismeretek

Front-End téren a szakgimnáziumban megszerzett alap HTML és Bootstrap tudás mellett LikeAll kurzuson szerzett alap PHP tapasztalataimat, idén külső gyakorlati helyen gyakorolt rutinnal bővítetten használtam fel.

Saját kíváncsiságom által CSS-t kezdtem tanulni, mely tudásomat szintén prezentáltam a projekt készítése során.

MySQL adatbáziskezelő szerver phpMyAdmin XAMPP-be beépített adminisztrációs felületén keresztül gyakoroltam rekord olvasás, szűrés, összekapcsolás, hozzáadás, frissítés, törlés operációkat, mely ismereteimet idén PHP-val összekötve gyarapítottam.

Alapvető JavaScript tudásomat idén TypeScripttel bővítettem, a NodeJS szerverrendszert Full-Stack környezetben megtanultam alkalmazni. Kezdetben az iskolában Angular keretrendszerben, majd a projekt fejlesztése közben Reactban.

A React Horgok alapvető ismeretét a projekt készítése közben sajátítottam el, saját próbálkozással és videó tutorialok segítségével. A komponensek használata nagyon hasonló sima HTML tag-ekhez így ahhoz nem kellett extra.

REST API HTTP-n keresztül történő lekéréseket, feltöltéseket autodidakta módon sajátítottam el, első sorban a legnépszerűbb videómegosztóról szerzett hozzáértés, amit a dokumentáció forrásmegjelölésénél meg is találhatnak.

## A felhasznált szoftverek

Dokumentáció készítéséhez:

* Microsoft Word: dokumentum szerkesztő program,
* Paint.NET: képszerkesztő szoftver,
* Lucidcharts: diagramm szerkesztő weboldal,

Program írásához:

* Visual Studio Code: kódszerkesztő program,
* Git SCM: Github elérés,
* Github: verziókövető rendszer

Kiegészítők, pluginek:

* PHP Intelephense (Intellisence, snippets for PHP)
* PHP Debug
* ES7+ React snippets
* TS React code snippets
* Simple React snippets
* Prettier kód formázó

Könyvtár, keretrendszer:

React Typescript

Modulok kiegészítők:

* Axios,
* React Router,
* React Transition Group,
* Framer Motion,
* React Markdown,
* React Responsive,
* Remark GFM,
* React Timeago,
* Use Local Storage

# Felhasználói dokumentáció

## A program általános specifikációja

A weboldal igazából egy multi-fórum, több téma megvitatása lehetséges rajta.

Egy “látogató” felhasználó, aki nincs felregisztrálva, nincs belépve csak nézni tudja a fórumot, egyedül regisztrálni tud.

A regisztrált felhasználó a bejelentkezés után a következő műveleteket tudja elvégezni:

* Közösségeket lehet létrehozni a fórumon, lényegében több kisebb fórumot, ezekbe lehet csatlakozni és a közösség részévé válni.
* A posztokat a közösségeken belül lehet közzé tenni, a posztok alatt kommentszekcióban lehet hozzáfűzni a hozzászólásokat.
* Meglévő közösségeket törölni, módosítani nem tud.
* Egyedül saját posztjait, hozzászólásait tudja utólag törölni, módosítani.

Az adminisztrátor bármelyik profil törlésére jogosult, erre van egy külön irányítópultja, ahol látja a profilokat listába szedve.

A tulajdonos tudja törölni a felhasználókat, közösségeket, posztokat, kommenteket.

## Rendszerkövetelmények

### Hardver követelmények

Kliens számára:

Minimum Rendszerkövetelmény:

* CPU: Dual-core, 1 gigahertz vagy jobb;
* RAM: 2 gigabájt;

Ajánlott Rendszerkövetelmény:

* CPU: Quad-core, 1 gigahertz vagy jobb;
* RAM: 8 gigabájt vagy több;

Szerver számára:

Minimum Rendszerkövetelmény:

* CPU: Quad-core, 4 gigahertz vagy jobb;
* RAM: 8 gigabájt;

Ajánlott Rendszerkövetelmény:

* CPU: Hexa-core, 3 gigahertz vagy jobb;
* RAM: 16 gigabájt vagy több;

### Szoftver követelmények

Kliens:

* Operációs Rendszer: Windows 10, Fedora Linux 32+, Manjaro Linux 21+ vagy macOS High Sierra,
* Böngésző: Bármilyen Chromium 99+ alapú böngésző legfrissebb verziója (Új Microsoft Edge, Chrome, Opera, Brave)
* JavaScript engedélyezése

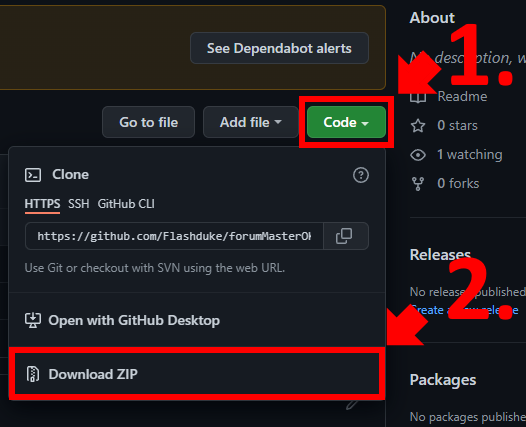
Szerver:

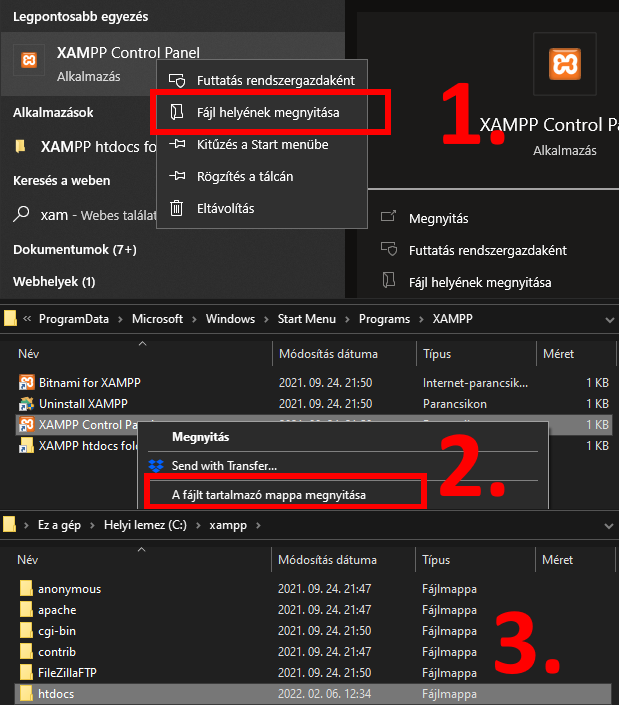
* Operációs rendszer: Windows Server 2022+, Ubuntu Linux Server 20+, Red Hat Enterprise Linux 7.2+
* XAMPP Control Panel 7.2 (Tartalmazza az Apache Szervert, mySQL szervert)
* NodeJS 16.14+

## A program telepítése

A legfrissebb verzió telepítéshez:

Látogasson el a Github fiókomon található „forumMasterOKJ” című kódgyűjeteményemhez (<https://github.com/Flashduke/forumMasterOKJ>), kattintson a zöld „Code” feliratú gombra, majd válassza ki a lenyíló menüből a „Download ZIP” elemet, ez Windowsos rendszeren a „Letöltések” mappába tölti le a tömörített állományt.



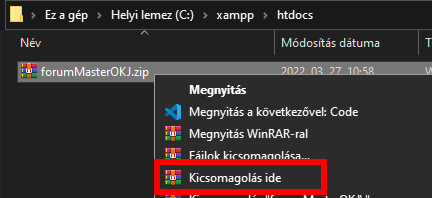
1. ábra Githubról letöltés

Ha viszont a CD-ről szeretné telepíteni akkor a fenti lépést kihagyhatja.

Következő lépés: látogasson a XAMPP Control Panel „htdocs” mappába amit úgy ér el, hogy az Ön által használt operációs rendszer keresősávjában rákeres a XAMPP-re majd az ikonra tett jobb kattintással eléri a menüt ahol a „Fájlt tartalmazó mappa megnyitása” elemet kiválasztva megnyitja a fájlkezelőjében a program parancsikonjainak a mappáját ahol az előző lépés megismétlésével a „XAMPP Control Panel” parancsikonján megnyitja a program gyökérkönyvtárát ahol a „H” gomb lenyomásával kijelölhetjük a keresett „htdocs” mappát.

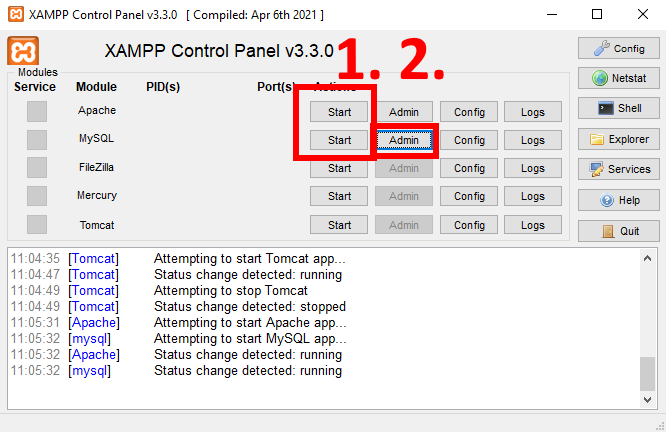
2. ábra htocs mappa elérése

Lépjen be a kijelölt mappába, majd a letöltött állományt csomagolja ki erre a helyre. Nevezze át a kicsomagolt könyvtárat, ha alapból nem ez a neve „*forumMasterOKJ*”-re.



3. ábra állomány kicsomagolása a htdocs mappába

Nyissa meg a XAMMP alkalmazást, indítsa el az Apache és mySQL szervereket a feliratuk mellé helyezett start gombbal, majd a mySQL „start” mellett az „admin” gomb megnyomásával felhozza a phpMyAdmin felületét, ahol a felső menüsávban az importálás mezőre kattintva az alábbi oldalt láthatja:



4. ábra szerverek kezelése

Itt nincs más dolgunk csak a „Fájl feltöltése” gombra kattintva a kicsomagolt állományban a „db\_forum” című mappán belül válassza ki a „db.sql” állományt, majd kattintson a „Megnyitás” gombra, ezután a jobb alsó sarokban lévő „Indítás”-t kiválasztva fel is töltötte az adatbázist a mySQL szerverre.

A képen szöveg látható

Automatikusan generált leírás

5. ábra phpmyadmin adatbázis feltöltés

Ha beírja az útvonalat a böngésző URL sávlába a „localhost/forumMasterOKJ” címet akkor a következő oldalt kell látnia:

A képen szöveg látható

Automatikusan generált leírás

6. ábra localhost mappák ellenőrzése

Győződjön meg arról, hogy a böngészőjében nincs letiltva a JavaScript:

A képen szöveg, képernyőkép, monitor, képernyő látható

Automatikusan generált leírás

7. ábra javascript engedély

Most nyissa meg a parancssort (CMD, PowerShell, NodeJS command prompt, Bash) másolja ki a „react\_forum” mappához vezető teljes elérési útvonalat:

A képen szöveg, képernyőkép, képernyő látható

Automatikusan generált leírás

8. ábra React App útvonal másolása

Írja be a parancssorba a „cd” parancsot, majd másolja be az előzőleg megkeresett elérési útvonalat a következőképpen:



9. ábra parancssor navigálás

Ha nem a CD-ről telepít akkor írja be a parancsot: „npm install”

Majd itt gépelje be a következő parancsot:



10. ábra Node szerver indítása

Ha ezt a lépést követően megnyílik az alapértelmezett böngészőjében a fórum akkor sikeresen végrehajtotta a telepítést.

Ellenkező esetben nézze meg, hogy telepítés során nem írt el valamit, ha nem találja a problémát akkor próbálja a legelejéről újrakezdeni.

## A program használatának a részletes leírása

1. **Az oldal navigációjának kezelése:**

**Asztali számítógépen:**

Az oldal legtetején megtalálható fixen elhelyezett navigációs sávon keresztül tudjuk az oldalt navigálni. A bal felső sarokban lévő „Forum” gombra kattintva, ha nem alapból ott vagyunk akkor a kezdőoldalon találhatjuk magunkat.

Bejelentkezés előtt a jobb felső sarokban balról jobbra olvasva megtalálhatunk egy bejelentkezés („Login”) és egy regisztrációs („Signup”) gombot, melyekre kattintva egy felülről beúszó modál ablakot nyithatunk meg.

Ha bejelentkezni szeretnénk akkor az első gombra kattintva behozott ablakon keresztül begépelhetjük a regisztrált fiókunk email címét az első „Email” feliratú beviteli mezőbe, amire az oldal automatikusan ráviszi a kurzort.

Ezután a második „Password” feliratú mezőbe rákattintva megadhatja jelszavát, miután mindkettő mezőt kitöltöttük a bejelentkezés („Login”) gomb kattinthatóvá válik, melyet megnyomva, ha sikeresen megtörtént a bejelentkezés akkor bezáródik az ablak és új funkciókhoz kapunk hozzáférést. Ha viszont sikertelen a művelet abban az esetben kapunk egy hibaüzenetet a modál ablak tetején.

Ha regisztrálni szeretnénk egy új fiókot a fórumon, akkor a „Signup” gombra kattintva beúszó modál ablakon keresztül tudjuk adatainkat megadni, minden mezőt szükséges a regisztrációgomb feloldásához megfelelően kitölteni.

Ha a felhasználó valamilyen módon felül akarja írni a gombnak a kattinthatóságát akkor sem fogja elfogadni a helytelenül bevitt vagy kihagyott adatokat a szerver biztonságára figyelemre tekintve. Ez a jelszó bevitelére is igaz, a megfelelő előkészületek ott meg lettek ejtve.

A bevitelimezők pirosra váltva jelzik a felhasználó számára, hogy nem megfelelő a bevitt karakterlánc formátuma, és utasításokkal segítik a felhasználót a javításra. Ezek az utasítások a következőek:

* Első mező:

Email, itt a megfelelő típus úgy néz ki, hogy minimum egy karakter, vagy karakterlánc ami állhat bármilyen alfanumerikus vagy szám típusú karakterből, ami után egy kukac ( @ ) szimbólum, ezt követően egy előzőleg részletezett karakter, vagy karakter lánc folytatásával és egy ponttal elválasztott domain címmel (pl. „ .hu ”, „ .com ” stb.) végződik

Minta egy szabályos Email címre: „minta@domain.hu”

* Második mező:

Felhasználónév, ami egy különleges karaktereket nem tartalmazó, ide nem értve a kötőjelet ( - ), alsó vonalat ( \_ ) karakterlánc lehet, ami minimum 4 karaktert tartalmaz, ezek lehetnek kicsi vagy nagy betűk, számok.

Kettő minta szabályos felhasználónevekre: „aron”, „SzeretemA\_Tejet96”

* Harmadik mező:

Jelszó, ennek tartalmaznia kell minimum egy számot, egy kapitális betűt és különleges karaktert ( !, @, #, $, % ), minimum 8 karakter, maximum 24 karakter hosszú lehet

* Negyedik mező:

Jelszó újra, itt annak érdekében, hogy későbbi problémákat megelőzzük, újra meg kell adnia az egyel feljebb begépelt jelszavát, aminek pontosan meg kell egyeznie a fent megadottal különben a mező nem fog zöldre váltani.

A mezők zöldre váltanak, ha az összes adatot megfelelően vittük fel.

A sárga „Signup” gombra kattintva befejeztük a regisztrációt. Itt is, ha bármilyen probléma történik a regisztráció során akkor az a felugró ablak tetején jelenik meg, különben sikeres a regisztráció és be is jelentkezhetünk az újonnan létrehozott profilunkba.

A gomb alatt megtalálható egy link arra az esetre, ha már rendelkezik fiókkal és inkább bejelentkezni szeretne. Erre a linkre kattintva egy másik modál ablak nyílik meg ahol a bejelentkezés mezői szerepelnek.

**Mobiltelefonon:**

A navigációs sáv alul található, fixen elhelyezve, fehér körvonalú ikonok jelzik a nem megnyitott lehetőségeket és a kitöltött lila színű ikonok azt, ami meg van nyitva, ezek a szimbólumok gomboknak felelnek meg.

A házikóra koppintva a kezdőoldalra ugorhatunk, ha alapesetben nem.

Az emberalakra koppintva bejelentkezés előtt a fentiekben írt bejelentkezés és regisztrációs gombot találhatjuk működésük ugyanaz, mint a fentiekben leírtaknál.

1. **Modál ablakok kezelése:**

A megnyitott modál ablakból a lila háttérre kattintva, vagy koppintva tudunk kilépni.

1. **Az oldal tartalmának kezelése:**

Tovább haladva az oldalon bal-középen a legfrissebb posztok találhatóak az összes közösségből, egy helyen.

Jobb oldalt fent pedig egy információs ablak van a segítségünkre, ezen az ablakon van kettő gomb,

* „Create Post”: új posztot tudunk létrehozni

Megnyit egy felugró ablakot, amibe be tudjuk gépelni a poszt részleteit,

* „Create Community”: új közösséget tudunk létrehozni

Minden posztot láthatunk az oldalon. A lila hátterű fejlécen attól függően, hogy hol vagyunk:

* Kezdőoldal,
* Egy közösség oldala,
* Egy felhasználó profilja,

Az oldalhoz tartalmat hozzáadni, hozzászólni, értékelni csak olyan felhasználó tud, aki már regiszterált és éppen be van jelentkezve a felületre.

Csak a nem egyértelmű információt tünteti fel a fejléc. Például a közösség oldalán nem fogja írni, hogy melyik közösségre töltöttük fel, hiszen ott vagyunk, vagy egy profil oldalán nem fogja feltüntetni, hogy ki volt a szerzője.

Viszont minden esetben ott lesz, hogy mennyi idő telt el a poszt közzététele óta.

Ha kurzorunkat egy poszt fölé visszük akkor egy lila árnyék jelenik meg a poszt alatt, ami azt indikálja, hogy ez egy kattintható elem az oldalon, ha egy poszt túl hosszú egy halványulás mutatja a tartalom alja felé, a jobb alsó sarkában láthatunk egy interakció mezőt, ahol balról jobbra haladva lájk, diszlájk és komment gombokat láthatunk szimbólumokkal feltüntetve.

Értékelni egy posztot egyszerűen a Tetszik vagy Nem tetszik gombra tett kattintással tudjuk megtenni. Ha véletlen kattintottunk rá az előbb leírt gomb valamelyikére akkor ezt a gombra tett újrakattintással egyszerűen ki is tudjuk venni.

A vezérlők mellett számlálók találhatóak, amik változnak, ha értékeljük a posztot (dinamikusak). Ha a komment gombra kattintunk akkor egyből a hozzászólás űrlaphoz visz minket az adott poszt oldalán és ott egyből tudjuk írni, ami a poszt olvasása során válaszként eszünkbe jutott.

Egy poszt oldalán akár több hozzászólással is találkozhatunk, amit más felhasználók írtak. Ezeket a kommenteket a fenti értékelést részletező leírás szerint működő tetszik-nem tetszik rendszer van a komment bal alsó sarkán elhelyezve.

Ami eltér a poszt értékeléséhez képest az az, hogy a kommentekre választ írni nem lehetséges, erre az az ok, hogy ahol ezt az opciót más oldalakon beépítették az egy nagyon mérgező környezetet hoz létre ezért ebbe a fórumba nem került beágyazásra.

Ha a posztot Ön hozta létre és be van jelentkezve akkor a fejlécen a felhasználóneve helyett a „You” magyarul az „Ön” felirat jelenik meg, és egy kuka ikon a jobb felső sarokban, amivel azonnal törölni tudjuk a posztunkat.

1. **Tartalom létrehozása:**

Poszt létrehozásához a kezdőoldalon, konkrét közösség oldalán a posztok felett található „Create Post” űrlapon megtalálható, ha nem egy külön közösség oldalán vagyunk egy lenyíló menü, ahol ki kell választani, hogy a posztot hova szeretnénk közzétenni.

Címet feltétlenül adni kell a tartalomnak különben nem lesz kattintható a „Post” gomb. A szövegtörzset, ha ismerjük a „markdown” nyelvet akkor a szöveget adott karakterekkel akár formázni is tudjuk például címekkel, idézettel, táblázattal és stb. majd a „Post” gombra kattintva létre tudunk hozni egy új posztot címmel és törzzsel, melyet más felhasználók tudnak értékelni, és válaszolni rá.

Kommenteket csak posztok alatt tudunk létrehozni.

Egy posztnak a beágyazott linkjein és gombjain kattintva az adott poszt oldalára tudunk jutni, ahol a posztnak a szövegét követően megtalálható a komment szekció, itt lehet látni mások milyen véleménnyel, válasszal rendelkeznek a problémára, blogra. Itt megtalálható a komment mező, ahova a saját hozzáfűzni valónkat gépelve majd a „Comment” gombra nyomva ezt közzé is tudjuk tenni. Itt nem kell kiválasztani mihez szólunk hozzá mert csak a posztra tudunk válaszolni.

Saját közösség létrehozásához látogassunk a kezdőlapra, ott kattintsunk a „Create Community” gombra, ez fel fog hozni egy modál ablakot itt meg tudjuk adni a közösség nevét, leírását és a „Create” gombra kattintva létre tudjuk hozni az új közösséget.

Az oldal alkalmazkodik a preferenciáinkhoz is.

Ha a rendszerünkön, legyen az mobiltelefon, asztali számítógép vagy laptop, ha beállítottuk a sötét mód használatát akkor az oldal alapból sötét külsővel fog rendelkezni.

Ha rendszerünkben bekapcsoltuk a mozgás csökkentését akkor az animációk nem fognak megjelenni.

Ezeket a funkciókat az oldalon belül egyelőre sajnálatos módon nem lehet átállítani az alapértelmezetthez (rendszer által preferálthoz) képest.

1. **A hozzáférhetőségi lehetőségek használata:**

Aki valamilyen hátrányossággal él, nem tud egeret használni, vagy nehezen lát annak a weboldal teljesen végig tabulátorral is navigálható, képernyőolvasóval feltérképezhető.

1. **Adminisztrátor jogosultságok:**

Minden poszt, minden komment és minden közösség törlésére lehetőség van.

Ez olyan mintha a saját tartalmainkat törölnénk csak a jobb felső sarokban lévő piros háttéren fehér kuka gombra kattintva ez véglegesen megtehető.

Az adminisztrátornak van egy külön oldala is, ahol ezt a felhasználókkal is megteheti, listába szedve megtalálja a felhasználókat nevek alapján. Jobb oldalt szintén a törlés gombokkal.

# Fejlesztői dokumentáció

## Az alkalmazott fejlesztői eszközök

Az operációs rendszer, amit a fejlesztés során használtam az a Microsoft Windows 10 Pro 21H2-es verziója, kiadó: Microsoft Corporation (Microsoft Software licensz)

A dokumentáció írásához a Microsoft Office Suite-et használtam, kiadó: Microsoft Corporation (Microsoft Software licensz)

Ezen belül az ábrák formázásához a paint.NET-et alkalmaztam, kiadó: dotPDN LLC. (PDN Classic licensz)

További ábrák készítéséhez a Lucidcharts UML szerkesztő webes applikációját használtam, gyártja: Lucid Software Inc. (Trial, Próbaverzió licensz)

Fejlesztői környezetnek a Visual Studio Code kódszerkesztő szoftvert használom, kiadó: Microsoft Corporation

A következő pluginokkal kiegészítve:

* PHP Intelephense (Intellisence, snippets for PHP), kiadója: Ben Mewburn,
* PHP Debug, kiadója: Xdebug,
* ES7+ React snippets, kiadója: dsznajder,
* TS React code snippets, kiadója: infeng,
* Simple React snippets, kiadója: Burke Holland,
* Prettier kód formázó, kiadó: Prettier

Tesztelésre a Google Chrome böngészőt használtam, kiadó: Google LLC.

JavaScript könyvtárnak a React TypeScriptet választottam, kiadója: Meta Platforms, Inc., ez NodeJS runtime-ot használ, typusok használatára a Microsoft Corporation által fejlesztett JavaScript szintaktikán a TypeScript-en, kiegészítve a következő bővítményekkel, modulokkal:

* Axios:

Promise alapú HTTP kliens a böngészőnek, kiadó Matt Zabriskie,

* React Router:

React komponens, útválasztást ad a React klienshez, kiadó: Remix,

* React Transition Group:

React komponens csoport, több áttünés és CSS animációk elhelyezése React komponeneseken belül, kiadó: Toru Kobayashi,

* Framer Motion:

Animáció könyvtár a Reacthoz, kiadó: Framer BV.,

* React Markdown:

Egy React komponens, amelyhez egy markdown karakterláncot lehet adni, amelyet biztonságosan renderel a React elemeknek, kiadó: Espen Hovlandsdal,

* React Responsive:

Egy React horog, amely biztosítja a képernyő méretéhez kötött kondícionális funkciók, komponensek stb. feltételes kirajzolását, renderelését, kiadók: Yocontra, Fractal

* Remark GFM:

Egy plugin a React Markdownhoz, kiegészíti GithubMD-szerű szintaktikával, kiadó: Titus Wormer

* React Timeago:

Egy React komponens, ami a megadott dátumhoz képest eltelt időt formáva adja vissza, kiadó: atool, alanwei0

* Use Local Storage:

Egy React horog ami a useState() horoghoz hasonlóan működve a számítógép saját tárhelyét használja mint hosszútávú memória, kiadó: Nick Scialli

RESTful API:

Natúr PHP-ban írtam kiegészítve a Firebase JWT kezelő könyvtárával (3-Clause BSD licensz) melynek futtatásához a XAMPP Apache szerverét használom

Az API teszteléséhez az Insomnia-t használtam.

Adatbázishoz mySQL-t használok melyet a XAMPP mySQL szerverén futtatok,

A XAMPP kiadója: Apache Friends, licensz: GPL

Verziókezelőrendszernek a Github.com-ot használtam (MS Corp., MS licensz) oda Git Bash-el Push-oltam a fájlokat, kiadó: Software Freedom Conservancy (GPL licensz)

Minden program, ahol nincs máshogy állítva MIT licenszelt.

## Adatmodell leírása

Mindegyik tábla rendelkezik saját azonosítóval: „id” (PRIMARY KEY)

„users” felhasználók tábla: tartalmazza a felhasználók email címét, álnevét, jelszavát, engedélyeit, JWT frissítőjét, ha vannak akkor a képeinek a neveit.

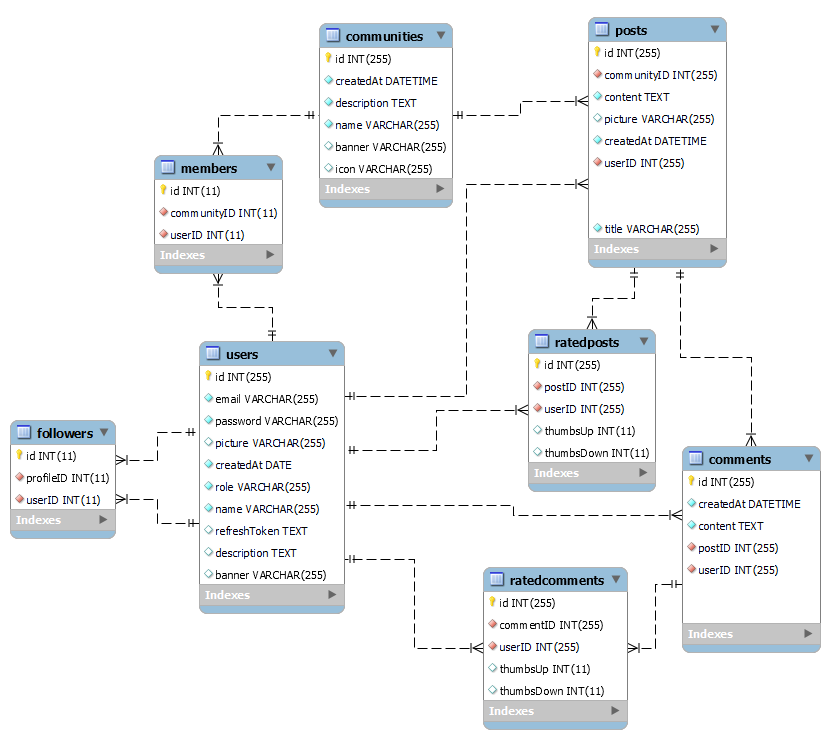
Tartalmi táblák:

* „communities” közösségek tábla: tartalmazza a címét, leírását, mikor készült és ha vannak akkor a képeknek a nevét,
* „posts” posztok tábla: tartalmazza a címét, tartalmát, mikor készült és egy kapcsolatot (FOREIGN KEY) a felhasználó felé, aki közzé tette,
* „comments” kommentek tábla: tartalmazza a tartalmát, mikor készült és egy kapcsolatot (FOREIGN KEY) a felhasználó felé, aki közzé tette

Interakciós táblák:

* „ratedposts” értékelt posztok: ez egy kapcsolótábla a több a többhöz kapcsolatot oldja meg, a posztok és a felhasználók felé (FOREIGN KEY),
* „ratedcomments” értékelt kommentek: ugyanaz, mint az előbbi csak a kommentekre megfeleltetve,
* „followers” követők kapcsolótábla: több a többhöz felhasználó kapcsolatot létesít,
* „members” tagok kapcsolótábla: több a többhöz kapcsolatot létesít, a közösségek és a felhasználók között

Az alábbi diagrammban láthatjuk a kapcsolatokat a tábláink között:



11. ábra adatbázis relációs ábra

Az osztálydiagramm csak a backend API-ról készült, mivel itt az OOP megoldást választottam.

Lényegében a tábláknak az oszlopait jelenítik meg és a metódusaik egy CRUD mintát követnek, létrehoz, olvas, felülír, töröl (Create, Read, Update, Delete) folyamatokat látják el, melyeket az API végpontok példányosítanak és hívnak meg.

Ettől eltér a „User” modell ahol regisztrálás, bejelentkezés és a JWT frissítése, beállítása történik.

A képen asztal látható

Automatikusan generált leírás

12. ábra PHP osztályok UML diagramm

## Részletes feladatspecifikáció, algoritmusok

REST API:

A legtöbb metódus az API végpontjainak modelljében fedi egymást, mondhatni sablon alapján meg lehet őket csinálni, így általánosságban a modelleken belül megegyeznek a metódusok.

* „read()” – olvas metódus:

Egy SQL lekérdezés az adott modell összes mezőjét egy „SELECT” parancs létrehozásával egy előkészített állítással (prepared statement) kéri le az adatbázisból, majd az eredményt visszaadja (return).

A végpontokon egy while ciklussal végig járja a megkapott eredményt és egy tömböt soronként egy asszociatív tömbbel tölt fel, majd ezt JSON kóddá átalakítva visszaadja a lekérdezőnek (kliensnek).

* „readSingle()” – egyet olvas metódus:

Egy szűrt SQL lekérdezés, amely a példányosított objektum elsődleges azonosító „id” alapján szűr és 0 vagy 1 sort szed szét egy asszociatív tömbbe.

Ezt a végponton meghívva JSON formában juttatja a lekérdezőnek.

* „check()” – ellenőriz metódus:

Megegyezik a fenti „egyet olvas” metódussal, de nem „id” hanem más mezők alapján ellenőriz, és logikai változót ad vissza az alapján, hogy létezik-e a keresett sor, vagy sem.

Végpontra meghívva a visszaadott értéket küldi a kliensnek.

* „create()” – létrehoz metódus:

Egy „INSERT” SQL parancsot ad ki, ezt a megfelelő mezőkkel tölti fel amit előtte „htmlspecialchars()” és „strip\_tags()” beépített PHP metódusokkal megtisztít, majd ha sikeresen végrehajtotta a feltöltést akkor egy logikai változóban igazat ad vissza, ellenkező esetben kiírja a hibát és hamisat ad vissza.

Ezt meghívva egy API végponton csak megfelelő hitelesítéssel érhetjük el, ha ez megvan akkor a kapott JSON objektumból kiolvasott mezőket a példányosított objektum mezőihez rendeli, majd egy JSON választ küld vissza.

* „update()” – frissít metódus:

Egy „UPDATE” SQL parancsot ad ki a csatlakoztatott adatbázisnak, egy előzőleg megtisztított „id” mező alapján keresi meg a frissíteni kívánt sort, sikeres operáció során logikai igazat, különben hamisat ad vissza és kiírja a hiba okát.

A végponton való meghívás megegyezik a „létrehoz” metóduséval.

* „delete()” – töröl metódus:

Egy „DELETE” SQL parancsot ad ki a csatlakoztatott adatbázisnak, egy előzőleg megtisztított „id” mező alapján keresi meg a törölni kívánt sort, sikeres operáció során logikai igazat, különben hamisat ad vissza és kiírja a hiba okát.

A végponton való meghívás megegyezik a „létrehoz” és „frissít” metódusokéval.

A felhasználóra specifikus metódusok:

* „login()” – bejelentkezés metódus:

Megfelel az „ellenőriz” mezódusnak, a végponton történő operációkat kivéve. A végponton attól függően, hogy az adatbázisból lekért email és titkosított jelszó megegyezik a kliens általi kérésben megadottakkal visszaad kettő JWT-t (JSON WEB TOKEN-t) amivel a bejelentkezés után kezeli a meghatalmazásait az API kérések felé.

* Az egyik egy frissítő token, ami az adatbázisba is felkerül. A másik egy hozzáférést biztosít, ez minden percben biztonsági okok miatt lejár, ezt majd egy http-only (egyedül http-n keresztül hozzáférhető) sütiben tárol, hogy a felhasználó ne lássa és ne próbálkozzon meg a visszafejtésével, ez 24 óra után jár le.
* „setRefreshToken()” – frissítő tokent beállít metódus:

„UPDATE” SQL paranccsal feltölti a „login” végpont által beállított frissítő tokent az adatbázisba, majd logikai változót ad vissza a művelet végrehajtása alapján, hamis érték mellett hibát is.

* „findRefreshToken()” – frissítő tokent talál metódus:

Az „ellenőriz” metódushoz hasonló, a „refreshToken” mező alapján keres.

A „refresh” végponton meghívva ellenőrzi, hogy van-e az adatbázisban megegyező frissítő token, ha van akkor egy új elérési tokent ad vissza a kliensnek a válaszban.

REACT KLIENS:

„AuthContext” – hitelesítés kontextus:

A React beépített „createContext()” metódusát használva paraméterként egy új objektumot „auth” mezővel és „setAuth(auth)” metódust megadva létrehoz egy új kontextust.

„AuthProvider” – hitelesítés szolgáltató:

Egy beépített állapotkezelő horog a „useState()” segítségével szabályozható kontextus példányt ad vissza:

export function AuthProvider({ children }) {

  const [auth, setAuth] = useState({});

  return (

    <AuthContext.Provider value={{ auth, setAuth }}>

      {children}

    </AuthContext.Provider>

  );

}

„RatingContext” – értékelési kontextus:

A fent leírt kontextussal megegyezik „auth” helyett a „rating” megnevezéseket használva.

„RatingProvider” – értékelési kontextus:

Az hitelesítés szolgátatóval megegyezik az „auth” megnevezés helyett „rating”-et használva.

## Tesztelési dokumentáció

Integrációs tesztek:

A http kérés-válasz tesztelésre az Insomnia nevű program segítségével tudtam próba lekéréseket írni.

Az összes végpontnak a megfelelő metódussal beállított (GET, POST, DELETE, PUT) lekéréseket indítottam, JSON formátumban például:

A képen szöveg látható

Automatikusan generált leírás

13. ábra HTTP lekérés

A válaszként érkező esetleges hibakódok javítását a sor alapú megközelítés miatt sikeresen és gyorsan meg tudtam oldani.

Sikeres kérés esetén a következő válasz érkezik:

A képen szöveg látható

Automatikusan generált leírás

14. ábra HTTP válsz

Fehér dobozos tesztelés:

A tesztkörnyezet: Google Chrome böngésző

A teszt menete:

React komponensek írása közben megnyitott böngésző ablak mellett a Google DevTools fejlesztői eszközök fülön a konzol ablakban lévő információk alapján való kezelése a weboldalnak.

Hibák feltárása:

* Gomb klikkelés hatására megtörténik-e a várt funkció lefutása
* Beviteli mezők kitöltése közben frissülnek-e az állapotok
* React Node-ok kezelése során nem ütközünk DOM hibába, az oldal elemei nem tűnnek el a képernyőről
* http-kérések esetében CORS (Cross Origin Resource Sharing) hibák kijavítása, API tesztelő szoftvernél nem ütközünk ilyen problémába.

Ezeket a hibákat a PHP alapú REST API-on belül a fejlécek módosításával és csak a „localhost:3000” címtől elfogadva történjen.

Extrém esetre való tesztelés:

Ha valami gyártás során nem észrevett hiba miatt vagy gyanús beavatkozás a nem hozzáférhető oldalak, gombok, beviteli mezők kitöltése során is biztosítva legyen a szerver védelme.

Esetleges SQL injekció elleni védelem tesztelése:

SQL parancs megadása a beviteli mezőkben, vagy http-kérések tartalmi részében:

* phpMyAdmin felületen való ellenőrzése az adatbázisnak,
* válszokban való tisztított karakterláncok ellenőrzése

# Összefoglalás

## Önértékelés

A kitűzött céljaim az időhöz mérve nagy részük teljesítve lettek, amit meg akartam csinálni nagyrészt sikerült, ami kivétel:

* részletes profilbeállítások,
* képek kezelése,
* moderáció adott közösségekben

Problémákba ütköztem, melyeket meg is oldottam:

* A http protokoll használatakor, nem vettem figyelembe a CORS irányelveket, ezeket fejlesztés közben tudtam korrigálni,
* A visszajelzési tetszik-nem tetszik rendszer fejlesztése közben, sok logikát helyeztem a frontendbe ezekbe belekavarodtam, a szerverről lefrissített állapotok lekérése miatt, szintén szerverrel való kommunikációs probléma, ezen nem segített a PHP kód debug-olása sem mivel nincsenek szigorúan írva a típusai a nyelvnek és kis problémák nem derülnek ki fejlesztéskor. Végül sikerült megoldani a problémát, egyszerűbb megoldás lett volna a websocketek használata, viszont az PHP backenddel nem lett volna megoldható, és utólag egy Node-Express-Typescript backendre nem akartam átírni a nagyrészt kész REST API-omat.
* A felhasználónak a JWT frissítése megoldására magamtól kellett rájöjjek, mivel PHP-hoz ilyen téren nem sok megoldás, itt egy új oszlop létrehozásával az adatbázisban és egy új API végpont létrehozásával ki tudtam küszöbölni a problémát.

Saját fejlődésem az OOP mélyebb megértésével kezdődik, ezt a PHP kódban alkalmaztam is, megértettem a metódusok és a más nem OOP függvények közötti különbséget.

Teljesen új lehetőségeket ismertem fel a TypeScript-tel fűszerezett React kliens írásánál. Mondhatni egyből beleugrottam a mély vízbe, állapotkezelés, horgok használata, eddig ilyesmiről nem tudtam. Első félévben az Angular használata közben találtam rá erre a keretrendszerre, ami jobban megtetszett a JS-be integrált HTML kódja miatt.

## Továbbfejlesztési lehetőségek

Ami talán a legfontosabb hiányzó funkció, ami hiányzik a weboldalról az a kereső mező. Ez megtalálható a navigációs sávban, viszont nem működik.

Az értesítések még egy nagy tulajdonság, amivel ki lehetne egészíteni az az értesítések támogatása, ami a követett profilok, csatlakozott közösségek új posztjairól értesítene.

A szövegterület kijelöléssel való szerkeszthetősége a META Inc. által készített DraftJS JavaScript könyvtár segítségével, grafikusan szerkeszthető felülettel lecserélni az alap HTML szövegdobozt.

Automatikusan generált profilkép, utólag módosítható avatárok használata a Dicebear API segítségével. Az így generált avatár a Backend szerveren való tárolásának megoldása, ha esetleg nem támogatnák tovább a Node Package Manager által elérhető modult.

Videólejátszó hozzáadása, támogatása, YouTube, Twitter, Reddit API-ok használatával, natív lejátszón belül, szöveg közé helyezve.

Valós vállalkozás esetén a szükséges szerződési és adatkezelési szabályok egy lábjegyzetben való megjelenítése az oldalon, szabadalmi jogok hozzáadása.

Külön stílusos „Rólunk” oldal készítése, ahol minden szükséges információval lehet kalauzolni a felhasználót, hozzáadni látványos elemekkel a felhasználói dokumentáció legkeresettebb részeit.

A jelenleg működő a felhasználó rendszerében beállított preferenciák alapján működő beállítások oldalon belül való kezelése kapcsolókkal megoldva. Ez tartalmazza a jelenlegi hozzáférhetőségi lehetőségeket: Sötét mód használata, Csökkentett mozgás (animációk csökkentése)

# Felhasznált irodalom

React Typescript dokumentáció (<https://hu.reactjs.org/>): Meta Platforms, Inc., 2022. 04. 01.

React Router dokumentációja (<https://reactrouter.com/>): Remix, 2022. 04. 01.

PHP JWT Authentication Tutorial (<https://www.techiediaries.com/php-jwt-authentication-tutorial/>) szerző: Techiediaries Team, 2021. 12. 26.

Csökkentett mozgás (<https://www.youtube.com/watch?v=5DEq5cWNYt8>) szerző: Kevin Powell, 2022. 01. 06.

Router helyzet probléma Typescriptben (<https://stackoverflow.com/questions/59752515/react-typescript-add-location-state-to-react-router-component>) megoldás szerzője: aLLeXUs, 2022. 02. 10.

Traversy Media PHP REST API From Scratch tutorial sorozata (3 részes) (<https://www.youtube.com/c/TraversyMedia>) szerző: Brad Traversy, 2022. 01. 12.

ReactJS Authentication, Authorization sorozat (első 5 rész) (<https://www.youtube.com/c/DaveGrayTeachesCode>) szerző: Dave Gray, 2022. 03. 21.

# Ábrajegyzék

[1. ábra Githubról letöltés 7](#_Toc100849694)

[2. ábra htocs mappa elérése 7](https://plkik-my.sharepoint.com/personal/heraro549_hengersor_hu/Documents/ZÁRÓDOLGOZAT/ZD_Herczegh_Aron.docx#_Toc100849695)

[3. ábra állomány kicsomagolása a htdocs mappába 8](#_Toc100849696)

[4. ábra szerverek kezelése 8](#_Toc100849697)

[5. ábra phpmyadmin adatbázis feltöltés 9](#_Toc100849698)

[6. ábra localhost mappák ellenőrzése 9](#_Toc100849699)

[7. ábra javascript engedély 10](#_Toc100849700)

[8. ábra React App útvonal másolása 10](#_Toc100849701)

[9. ábra parancssor navigálás 10](#_Toc100849702)

[10. ábra Node szerver indítása 10](#_Toc100849703)

[11. ábra adatbázis relációs ábra 19](#_Toc100849704)

[12. ábra PHP osztályok UML diagramm 20](#_Toc100849705)

[13. ábra HTTP lekérés 24](#_Toc100849706)

[14. ábra HTTP válsz 24](#_Toc100849707)