



# Adatbázis alapú web alkalmazás fejlesztése

## Készítette

Lázár Benedek

Programtervező informatikus BSc

## Témavezető

Dr. Tajti Tibor Gábor

egyetemi docens

EGER, 2023

# Tartalomjegyzék

<b>Bevezetés</b>	<b>5</b>
<b>1. Adatbázis alapú web alkalmazások</b>	<b>6</b>
1.1. Bevezetés . . . . .	6
1.2. Az internet fejlődése . . . . .	6
1.3. Web alkalmazások . . . . .	7
1.4. Adatbázisok . . . . .	7
1.5. Közösségi média platformok . . . . .	8
1.6. Hasonló alkalmazások . . . . .	8
1.6.1. Facebook . . . . .	8
1.6.2. YouTube . . . . .	9
1.6.3. Twitter (X) . . . . .	9
1.7. Összegzés . . . . .	9
<b>2. Fejlesztői dokumentáció</b>	<b>10</b>
2.1. Követelmény specifikáció . . . . .	10
2.1.1. Bevezetés . . . . .	10
2.1.2. Az alkalmazás célja . . . . .	10
2.1.3. Jelenlegi helyzet . . . . .	10
2.1.4. Vágyalomrendszer . . . . .	11
2.1.5. Üzleti folyamatok modellje . . . . .	11
2.1.6. Követelménylista . . . . .	13
2.2. Funkcionális specifikáció . . . . .	14
2.2.1. Célok . . . . .	14
2.2.2. Követelmények áttekintése . . . . .	14
2.2.3. Képernyőtervek . . . . .	15
2.2.4. Forgatókönyvek . . . . .	16
2.3. Rendszerterv . . . . .	17
2.3.1. Projekt terv . . . . .	17
2.3.2. Fizikai környezet . . . . .	18
2.3.3. Architektúrális terv . . . . .	18
2.3.4. Adatbázis terv . . . . .	20

2.3.5.	Implementációs terv . . . . .	26
2.3.6.	Telepítési terv . . . . .	27
2.3.7.	Tesztelési terv . . . . .	27
<b>3.</b>	<b>Telepítési útmutató</b>	<b>29</b>
3.1.	Bevezetés . . . . .	29
3.2.	Projekt futtatása XAMPP-al . . . . .	29
3.3.	Projekt futtatása Docker-rel . . . . .	31
<b>4.</b>	<b>Használt technológiák</b>	<b>33</b>
4.1.	Bevezetés . . . . .	33
4.2.	Technológiák . . . . .	33
4.2.1.	XAMPP . . . . .	33
4.2.2.	MySQL . . . . .	33
4.2.3.	PhpMyAdmin . . . . .	34
4.2.4.	Laravel . . . . .	34
4.2.5.	CSS . . . . .	35
4.2.6.	Bootstrap . . . . .	36
4.3.	Egyéb szoftverek és alkalmazások . . . . .	37
4.3.1.	SourceTree . . . . .	37
4.3.2.	Trello . . . . .	38
4.3.3.	Docker . . . . .	39
<b>5.</b>	<b>Megvalósított funkciók</b>	<b>40</b>
5.1.	Bevezetés . . . . .	40
5.2.	Regisztráció és Bejelentkezés . . . . .	40
5.3.	Profil oldalak . . . . .	41
5.4.	Létrehozás . . . . .	41
5.5.	Módosítás . . . . .	42
5.6.	Törlés és Tiltás . . . . .	42
5.6.1.	Automatikus törlés . . . . .	42
5.6.2.	Törlési opciók . . . . .	43
5.7.	Feliratkozás, leiratkozás . . . . .	43
5.8.	Jogosultságok kezelése . . . . .	44
5.9.	Fellebbezés és helyreállítás . . . . .	45
<b>6.</b>	<b>Továbbfejlesztési lehetőségek</b>	<b>46</b>
6.1.	Bevezetés . . . . .	46
6.2.	Néhány bővítési lehetőség . . . . .	46
	<b>Összegzés</b>	<b>48</b>



# Bevezetés

A szakdolgozatom témája egy adatbázisra épülő webes alkalmazás fejlesztése. Gyakran használunk ilyen alkalmazásokat, gondoljunk például a YouTube-ra, vagy a Facebook-ra, ami rengeteg felhasználónak ad lehetőséget, hogy videóit és gondolatait megossza másokkal. Ezeket az adatokat egy adatbázisban tárolják, így a felhasználók feltölthetik ide fájljaikat, vagy letölthetik a mások által feltöltött tartalmakat.

A projektem célja, létrehozni egy adatbázisra épülő webalkalmazást, amelyben a felhasználók csoportokat hozhatnak létre, vagy csatlakozhatnak már meglévő csoportokhoz és ezen belül tudják megosztani egymás között a képeiket és gondolataikat.

A projekt implementálása előtt meg kellett terveznem az alkalmazást és kiválasztani azokat a technológiákat, amiket használni fogok a fejlesztés során.

Először is szükségem volt egy fejlesztői környezetre, én a Visual Studio Code-ot választottam. Szükségem volt még webszerverre és egy adatbázisra is, és mivel a XAMPP és a PhpMyAdmin könnyen használható, és legtöbbször ezeket használtuk a gyakorlatok során, ezért rájuk esett a választás.

A fejlesztés segítése és gyorsítása érdekében érdemes keretrendszert is használni, én pedig a Laravel keretrendszert használtam, mert ezt a technológiát már ismertem és gyorsan illetve átláthatóan lehet vele dolgozni, ezen kívül rengeteg beépített funkció segítette a munkámat.

Ezekon kívül a verziókövetéshez SourceTree-t, a feladatok kiosztásához és nyomon követéséhez pedig Trello-t használtam. Annak érdekében, hogy akár XAMPP használata nélkül is lehessen futtatni az alkalmazást és az adatbázist, létrehoztam konténereket is Docker desktop segítségével.

Az alkalmazás a következő GitHub repository linkről tölthető le:

[https://github.com/benedekklazar/Szakdolgozat\\_HDMIJ3\\_2023](https://github.com/benedekklazar/Szakdolgozat_HDMIJ3_2023)

# 1. fejezet

## Adatbázis alapú web alkalmazások

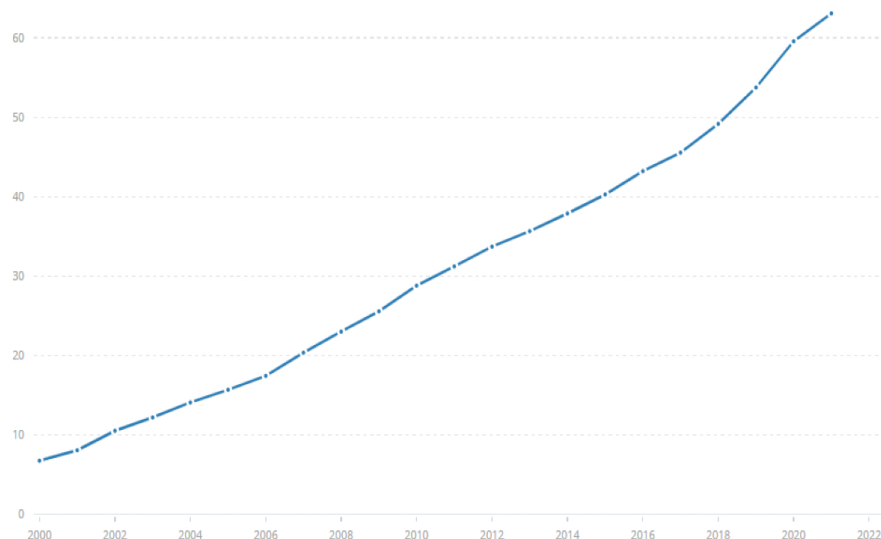
### 1.1. Bevezetés

Ebben a fejezetben fogom bemutatni az adatbázisok és webalkalmazások általános tulajdonságait, illetve bemutatok néhány olyan web alkalmazást is, amely adatbázisban tárolja az adatait.

### 1.2. Az internet fejlődése

Az internet megjelenésével és fejlődésével az emberek gyorsabban és hatékonyabban képesek kapcsolatba lépni más emberekkel, tartalmakat oszthatnak meg egymással és sokan már a munkájukat is internet segítségével végzik. Az Email megjelenésével az embereknek már nincs szükségük valódi levelekre, mert ezek szállítása sokkal több időt és erőforrást vesz igénybe, mint egy elektronikus levél elküldése az interneten keresztül. A közösségi weboldalnak köszönhetően az emberek felvehetik a kapcsolatot távol levő ismerőseikkel, vagy akár idegenekkel is egy pillanat alatt. Az internetről vásárolhatunk könyveket, ruhákat, vagy akár rendelhetünk ételeket is, anélkül, hogy el kéne hagynunk a otthonunkat. Megtekinthetjük a legújabb híreket, videókat és eseményeket, de használhatjuk tanulásra, vagy játékra is.

Az internet nagyon hasznos, de sajnos megvannak a maga veszélyei is. Ilyenek lehetnek például az internetes átverések, vagy csalások. Az internet növekedésével párhuzamosan a hacker-ek elkezdtek olyan vírusokat írni, amelyek interneten keresztül is terjednek. Fontos, hogy ezeket a csalásokat észrevegyük és ne töltsünk le gyanús weboldalakról fájlokat, vagy szoftvereket, illetve ne vásároljunk termékeket megbízhatatlan oldalakról. De az internetnek más veszélyei is lehetnek a társadalomra, mivel az embereknek az internet kora óta egyre kevesebb mozgásra van szükségük, hiszen bármit elérhetnek otthonról egy kattintással, ha tudnak csatlakozni az internethez. Sokan már iskolába sem járnak, hiszen online is tudnak csatlakozni az oktatáshoz.



1.1. ábra. Az internetet használó személyek százalékos aránya a világon

Összességében elmondható, hogy az internetnek köszönhetően rengeteg új munkalehetőség jött létre, gondoljuk a Webfejlesztőkre, vagy akár a YouTube tartalomgyártókra. A weboldalak utat nyitottak a hirdetőknél is, hogy termékeiket mások weboldalain hirdessék, ezáltal a weboldal tulajdonosa bevételhez juthat.

### 1.3. Web alkalmazások

A webalkalmazások olyan szoftverek, amelyek a böngészőnkben futnak. A web alkalmazásokat bárki használni tudja, akinek van számítógépe és képes csatlakozni az internetre. Ezeket az alkalmazásokat nem kell a számítógépünkre külön telepíteni, hanem gyakran egy távoli szerveren futnak. A legtöbb nem webes szoftverrel az a probléma, hogy telepíteni kell, amely sok tárhelyet és erőforrást vesz el a számítógépünktől. Ezen kívül minden frissítésnél le kell tölteni és telepíteni kell a szoftver egy újabb verzióját. Pozitívum, hogy az ilyen alkalmazásoknak általában nincs szükség internetkapcsolatra a használatukhoz és a felhasználói élmény is gazdagabb. Ezeket a nem webes alkalmazásokat **vastag kliensnek** is nevezik, szemben a webes alkalmazások kliensével, melyet **vékony kliensnek** neveznek. A vékony klienshez szükség van egy weboldalra és egy böngészőre, amiben az alkalmazást futtatjuk. Főbb előnyei a nem webes szoftverekkel szemben a kis hardverigény és az automatikus frissítés.

### 1.4. Adatbázisok

Az adatbázisok elektronikusan elérhető, logikailag összetartozó információk és adatok szervezett gyűjteménye. Egy adatbázis táblákból, a táblák pedig mezőkből épülnek fel.

Minden táblában szükség van egy elsődleges kulcsra, ez az azonosító. Minden egyes rekordnak egy egyedi azonosítót kell biztosítani, azaz nem szerepelhet egy azonosítóból több egy táblában. Ez azért fontos, hogy az adatokra egyértelműen lehessen hivatkozni. Léteznek relációs és nem relációs adatbázis kezelő rendszerek. A legnépszerűbb relációs adatbázis kezelő rendszerek közé tartozik a MySQL, amely SQL nyelvet használ az adatok lekérdezésére.

Az adatbázisoknak biztosítani kell az adatok elérhetőségét és védelmét. Oda kell figyelniük, hogy az érzékeny adatokat, például jelszavakat megfelelő biztonsággal tároljuk, ezért a jelszavakat csak hash-elés után mentjük le az adatbázisban.

Az adatok tárolása, titkosítása és védelme mellett biztosítanunk kell, hogy az adminisztrátorok könnyen hozzáférhessenek az adatokhoz. Lehetőséget kell adni, hogy az adatok bármikor létrehozhatóak, megtekinthetőek, törölhetőek és szerkeszthetőek legyenek az adminisztrátorok számára.

## **1.5. Közösségi média platformok**

Mivel a projektem is egy adatbázisra épülő közösségi média alkalmazás, ezért itt bemutatom ezen alkalmazások főbb tulajdonságait. A közösségi média alkalmazások az emberek interakcióin alapszanak. Létezhetnek híroldalak, melyek napról napra információval szolgálnak. A felhasználók hozzászólás formájában fejezhetik ki véleményüket. A blog oldalakon a felhasználók saját cikkeket és bejegyzéseket hozhatnak létre. Ezek lehetnek oktató jellegűek, humorosak, de akár kritikusak is. A közösségi oldalakon a felhasználók megoszthatják másokkal videóikat, vagy fotóikat.

Ezeknek az oldalaknak általában megvannak a saját közösségi irányelveik, melyeket minden felhasználónak be kell tartania. Nem oszthatunk meg akármit, csak olyan tartalmakat, melyek megfelelnek a platform szabályainak. Fontos, hogy ne osszunk meg személyes adatokat, például a jelszavunkat, mert ezeket az információkat bárki láthatja.

## **1.6. Hasonló alkalmazások**

### **1.6.1. Facebook**

A Facebook az egyik legnépszerűbb közösségi hálózat, fő alapítója Mark Zuckerberg. Az alkalmazás felhasználói regisztráció, vagy bejelentkezés után kapcsolatba léphetnek a többi felhasználóval, létrehozhatnak eseményeket, csoportokat és üzleteket. Ismerősnek



jelölhetnek más felhasználókat, vagy elfogadhatják a beérkező ismerősnek jelöléseket. A felhasználók megoszthatnak képeket és videókat ismerőseikkel, vagy reagálhatnak mások posztjaira megjegyzés és "tetszik" formájában.

### **1.6.2. YouTube**

A YouTube a legismertebb videómegosztó platform, három alapítója Steve Chen, Chad Hurley és Jawed Karim. Havonta több mint 1 milliárd aktív felhasználója van. Az első videó 2005 április 24.-én került fel a platformra "Me at the zoo" címmel. A YouTube sokak számára munka és pénzkeresési lehetőséget biztosít, mivel a felhasználók akár bevételt is kereshetnek videók feltöltésével, a videókon megjelenő reklámokon keresztül. A platform főbb funkciói közé tartozik a videó feltöltése, lájkolás (tetszik), diszlájkolás (nemetszik), hozzászólás írása, de létrehozhatunk lejátszási listákat is. A videóknak megadhatunk címet, indexképet és leírást is, illetve bármikor szerkeszthetjük, vagy törölhetjük a videóinkat.

### **1.6.3. Twitter (X)**

A Twitter, vagy "X" egy ismert közösségi hálózat és mikroblog szolgáltatás, négy alapítója Jack Dorsey, Noah Glass, Biz Stone és Evan Williams, jelenlegi tulajdonosa pedig Elon Musk. Az "X", régebben twitter felhasználói rövid szöveges posztokat hozhatnak létre, vagy mások bejegyzéseit oszthatják meg. Népszerű platform a hírességek és cégek körében, hiszen könnyen felvehetik a kapcsolatot követőikkel, felhasználóikkal. Havonta 328 millió felhasználó használja aktívan a platform különböző szolgáltatásait. Főbb szolgáltatásai közé tartozik a "Tweetelés", vagyis egy új poszt létrehozása, a "Re-tweetelés", vagyis egy meglévő poszt újra posztolása, de ezeken kívül van lehetőség hozzászólást írni és lájkolni is. A felhasználók a szöveges posztokhoz mellékelhetnek képeket, vagy videókat is.

## **1.7. Összegzés**

A fent említett adatbázis alapú web alkalmazások inspirálták a szakdolgozatomhoz készült projektet is. A legtöbb ilyen alkalmazás adatbázist használ az adatok tárolására. A projektben (a Facebook-hoz hasonlóan) létre lehet hozni csoportokat és ezeken belül posztokat megosztani. A posztokra hozzászólás formájában tudnak a felhasználók reagálni. A YouTube fellebbezés rendszeréhez hasonlóan, a felhasználók képesek vitatni valamely tartalmuk eltávolítását, ha azt nem érzik jogosnak. A designt főleg a Twitter és a Facebook inspirálta, ahol a posztok egymás után jelennek meg egyesével.

## 2. fejezet

# Fejlesztői dokumentáció

### 2.1. Követelmény specifikáció

#### 2.1.1. Bevezetés

Ebben a fejezetben fogom összefoglalni az alkalmazásom főbb céljait és követelményeit. Bemutatom a jelenlegi helyzetet, a vágyálom rendszert és az üzleti folyamatok modelljét.

#### 2.1.2. Az alkalmazás célja

Az alkalmazás célja, hogy a felhasználók különböző csoportokat hozzanak létre, vagy meglévő csoportokhoz csatlakozzanak, posztokat és hozzászólásokat hozzanak létre és ezáltal megosszák egymás között gondolataikat. Fontos, hogy a megosztott tartalmakhoz csak azok a felhasználók férjenek hozzá, akiknek megvan hozzá a megfelelő jogosultsága. A felhasználóknak legyen lehetőségük létrehozni, megtekinteni, szerkeszteni és törölni posztjaikat és csoportjaikat.

#### 2.1.3. Jelenlegi helyzet

Napjainkban a legtöbb közösségi média felület ugyanazokat a tartalmakat fogja ajánlani minden felhasználónak, ezért a tematikus tartalmak gyakran csak kevés emberhez jutnak el. Az algoritmus azokat a tartalmakat fogja népszerűsíteni, amik a többség számára érdekesek lehetnek, emiatt a kevésbé népszerű témák háttérbe szorulnak, és nehezebben találják meg azok, akiket tényleg érdekel az adott téma.

Célunk, hogy az érdeklődők csatlakozhassanak egy kis közösségbe, ahol láthatják, hogy mások milyen tartalmakat osztottak meg a csoportban, miközben ne zavarjuk azokat, akiket nem érdekel az adott téma. Ez azt jelenti, hogy egy csoportban megosztott poszt csak ott lesz elérhető, és csak a tagok tekinthetik meg.

## 2.1.4. Vágyáalomrendszer

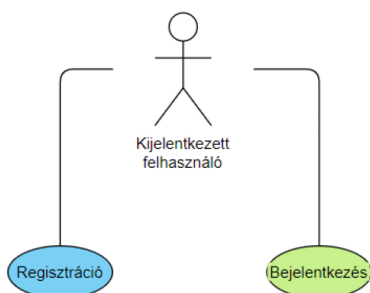
Projektünkben a funkciók eltérőek attól függően, hogy mely csoportoknak vagyunk tagjai, például egy tag kevesebb funkciót érhet el, mint egy moderátor, de többet, mint egy olyan felhasználó, aki nem tagja a csoportnak, vagy nincs bejelentkezve.

Szükség van egy magasabb szintű felhasználóra, azaz egy adminisztrátorra, aki korlátlanul hozzáfér minden csoporthoz, poszthoz és felhasználóhoz, azokat szabadon szerkesztheti vagy törölheti.

Ha bejelentkezek és létrehozok egy csoportot, akkor azon a csoporton belül én leszek az Admin, azaz kezelhetem, hogy kik csatlakozhatnak a csoportomba, illetve moderálhatom a csoportban megjelenő posztokat. Kijelölhetem a csoport moderátorait, akik segítenek a szabályaim betartatásában.

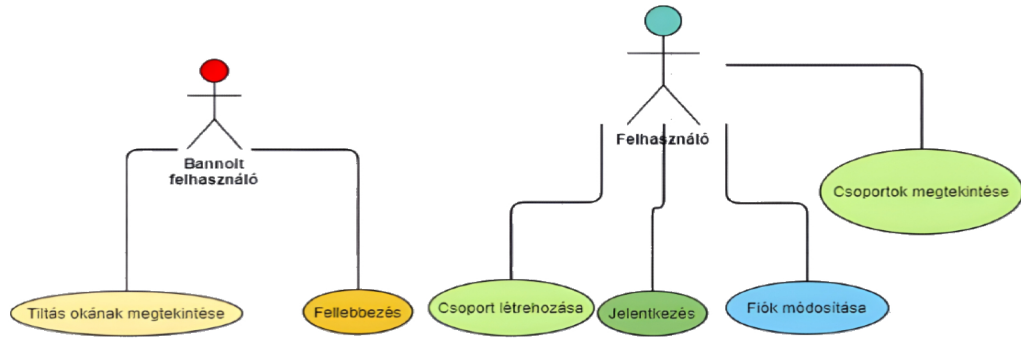
## 2.1.5. Üzleti folyamatok modellje

Ha nem vagyunk bejelentkezve, akkor csak a regisztrációra és bejelentkezésre van lehetőségünk. Ha a felhasználó megpróbál valamilyen egyéb műveletet végrehajtani, vissza lesz irányítva a kezdőoldalra.



2.1. ábra. A nem bejelentkezett felhasználó use case diagrammja

Regisztráció és bejelentkezés után megtekinthetjük a csoportokat, módosíthatjuk vagy törölhetjük a fiókunkat. A módosítások közé tartozik a név, profilkép vagy jelszó megváltoztatása.



2.2. ábra. A bejelentkezett és bannolt felhasználó use case diagrammja

A fenti ábrán látható, hogy bejelentkezés után elérhetővé válik a fiókunk módosítása és törlése funkció, létre tudunk hozni csoportokat, vagy csatlakozhatunk meglévőkhöz, illetve kijelentkezhetünk. Moderátorként lehetőségünk van más felhasználók tartalmának módosítására, vagy eltávolítására is.

Abban az esetben, ha töröljük a felhasználónkat, még van lehetőségünk azt visszaállítani egy meghatározott ideig (egy hét). Ha egy admin tiltja a fiókunkat, akkor fellebbezhetünk és kérhetjük a fiók visszaállítását. Ha egy héten belül nem küldjük el a fellebbezést, akkor a tiltás végleges és a fiókunk automatikusan véglegesen törlődni fog az adatbázisból.

### 2.1.6. Követelménylista

A következő táblázatban foglaltam össze a legfontosabb funkciókat, amiknek meg kell valósulniuk az alkalmazásban. Ezek közé tartozik az oldalak megtekintése, csoportok és posztok létrehozása, szerkesztése és törlése, a hozzászólás írása és a fellebbezés funkciók is.

Modul	Leírás
Felhasználó létrehozása	Itt lehet új felhasználót létrehozni. Amennyiben már létező felhasználót próbálunk létrehozni, vagy hibás adatot adunk meg, hibaüzenetet kapunk.
Autentikáció	Itt lehet bejelentkezni a fiókba megfelelő felhasználónévvel és jelszóval. Amennyiben hibás adatot adunk meg, hibaüzenetet kapunk.
Név módosítása	Egy új felhasználónév megadásával módosíthatja a régit, amennyiben az új név még nem foglalt.
Jelszó módosítása	Először a régi jelszó megadásával hitelesíti magát a felhasználó, majd az új jelszó megadásával módosíthatja azt. Az új jelszót 2x kell megadni annak érdekében, hogy a felhasználó ne zárja ki magát a fiókjából.
Jogosultság	A következő szinteket kell létrehozni: vendég, tag, moderátor, admin.
Csoport létrehozása	A csoport létrehozása funkció segítségével bárki saját csoportokat tud létrehozni, melynek ő lesz az adminja.
Tagság létrehozása	Itt tudnak a felhasználók csoportokhoz csatlakozni, A nyilvános csoportokba azonnal bekerülnek, míg a zárt csoportokba csak egy admin vagy moderátor jóváhagyásával kerülhetnek be.
Csoport módosítása	A csoportadminok szerkeszthetik vagy törölhetik saját csoportjaikat.
Felhasználó oldal Csoport oldal Poszt oldal	A felhasználók megtekinthetik a többi felhasználót, csoportokat, posztokat és a posztokra érkezett megjegyzéseket.
Tiltás és fellebbezés	Az adminok törölhetik a nem helyén való tartalmakat. A tiltott felhasználóknak lehetőségük van megtekinteni a tiltás okát és fellebbezést indítani a tiltás ellen.
Automatikus törlés	Ha véletlenül törölünk valamit, azt még helyre lehet állítani egy adott ideig. Ha az idő letelik, a tartalom automatikusan törlődni fog az adatbázisból.

## 2.2. Funkcionális specifikáció

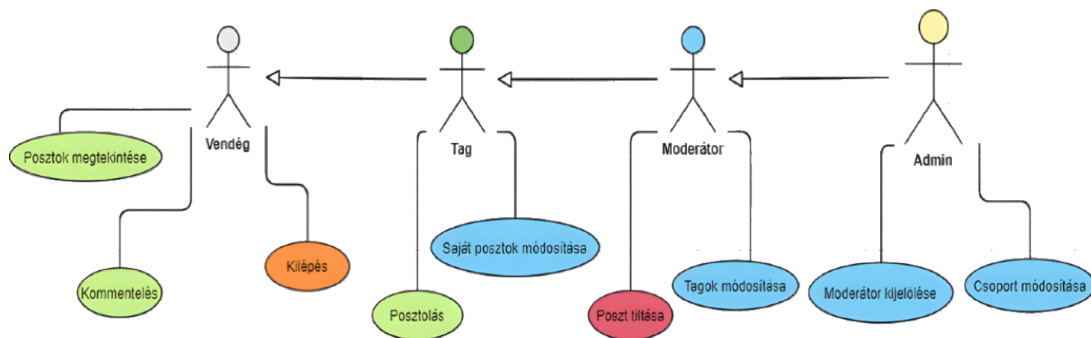
### 2.2.1. Célok

Az alkalmazás célja, hogy a felhasználók ne egy platformtól függjenek, hanem kisebb csoportokat hozhassanak létre.

A felhasználók tematikus csoportokat hozhatnak létre, vagy már meglévőkhöz csatlakozhatnak és megoszthatják saját képeiket a többi hasonló érdeklődésű felhasználóval. A felhasználóknak lehetősége lesz törölni és szerkeszteni saját tartalmaikat, vagy mások tartalmaira reagálni egy hozzászólás írásával.

### 2.2.2. Követelmények áttekintése

Négy fő felhasználói szint van: az adminisztrátor, moderátor, tag és vendég.



2.3. ábra. A csatlakozott felhasználók use case diagrammja

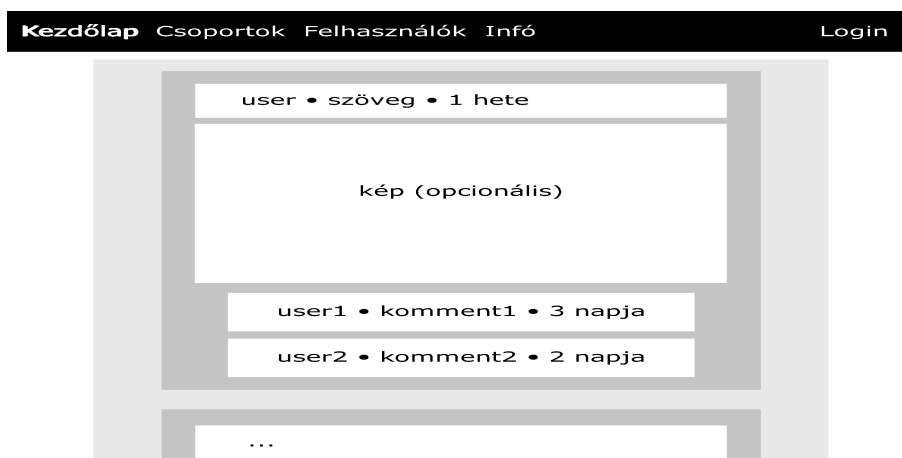
A csoport tagok minden alattuk álló szerepkör képességeit megkapják. Például a moderátor tud posztolni és kommentelni is, mert ezeket egy tag is megteheti.

Ez azt is jelenti, hogy egy moderátor csak a gyengébb szerepkörű felhasználókat módosíthatja, például egy vendégnek adhat tag szerepkört, de moderátor jogot már nem.

A vendégek meg tudják tekinteni a posztokat és írhatnak megjegyzéseket, de nem hozhatnak létre saját posztot. A tagok már létre hozhatnak saját posztokat és ezeket szabadon törölhetik vagy módosíthatják. A moderátorok már bárki posztját törölhetik a csoportból, és módosíthatják vagy törölhetik a csoport többi tagját. Az admin rendelkezhet az egész csoport felett, beállíthatja, a csatlakozás feltételeit, vagy a csoport láthatóságát. A csoport tulajdonosa törölheti, vagy szerkesztheti saját csoportját, illetve kioszthat moderátor és adminisztrátor jogosultságokat a tagok között.

### 2.2.3. Képernyőtervek

A funkciók megtervezése után meg kellett terveznem az alkalmazás megjelenését is. A következő képernyőtervek segítették a munkámat a frontend rész fejlesztésében.



2.4. ábra. A poszt oldal képernyőterve

A csoport oldalon fognak megjelenni azok a posztok, amelyeket a többi tag osztott meg. Minden poszthoz tartozik egy egyedi URL, amit a poszt azonosítója határoz meg. Ezeket csak akkor láthatjuk, ha legalább vendég jogosultságunk van a csoportban.

Azok a posztok nem jelennek meg, melyek törölve lettek, vagy privát láthatóságúak, kivéve abban az esetben, ha mi osztottuk meg őket.

Ha egy nem létező posztra hivatkozunk, vagy nincs jogosultságunk, akkor át leszünk irányítva egy hiba oldalra.



2.5. ábra. A csoport lista oldal képernyőterve

A "csoportok" fülre kattintva fognak megjelenni saját és a többi felhasználó által létrehozott nyilvános csoportok. Ha ezekre kattintunk, akkor, megtekinthetjük a posztokat. Abban az esetben, ha még nem vagyunk tagok, csak egy csatlakozás gomb jelenik meg.

## 2.2.4. Forгатókönyvek

**Regisztráció:** Meg kell adnunk egy legalább 5 karakter hosszú felhasználónevet és kétszer egy minimum 5 karakteres jelszót. Ha túl rövid jelszó vagy felhasználónév lett megadva, vagy már létező fiókot próbálunk regisztrálni, akkor hibaüzenetet kapunk. Ha a megadott jelszavak nem egyeznek, akkor is hibaüzenetet kapunk. Ha minden sikeres, az adatbázisban létrejön egy felhasználó és bejelentkezünk a fiókba.

**Bejelentkezés:** Egy felhasználónév - jelszó párral tudunk bejelentkezni. Ha a felhasználónévhez megfelelő jelszót adtunk meg, bejelentkezünk a fiókba, ellenkező esetben hibaüzenetet kapunk.

**Hozzáférés igénylése:** Ha egy csoportnak még nem vagyunk tagja, akkor kérhetünk hozzáférést. Egyes csoportokhoz automatikusan csatlakozhatunk, míg másokhoz csak admin jóváhagyásával férhetünk hozzá.

**Meghívás kezelése:** Ha meghívást kaptunk egy csoportba, akkor szabadon eldönthetjük, hogy elfogadjuk, vagy elutasítjuk. Abban az esetben, ha az "Elfogad" gombra kattintunk, hozzá leszünk adva a csoporthoz, míg az "Elutasít"-ra kattintva törölhetjük a meghívást az adatbázisból.

**Posztolás és hozzászólás:** Ha megkapjuk a tag rangot egy csoportban, akkor szabadon posztolhatunk. Ehhez meg kell adnunk egy szöveget és megadhatunk egy képet is, ha szeretnénk. Van lehetőségünk eldönteni, hogy posztunk megjelenjen-e a lista oldalon, vagy csak linkel legyen elérhető. A posztok közzétételét ütemezhetjük is egy jövőbeli dátum megadásával.

**Moderálás:** Moderátorként törölhetjük más felhasználók tartalmait a csoportból. Ehhez ki kell jelölnünk a tartalmat és meg kell adnunk egy indoklást. Azt is lehetőségünk van eldönteni, hogy csak a posztot töröljük-e, vagy a hozzá tartozó megjegyzéseket is. Azonos, vagy magasabb rangú felhasználók tartalmainál nem jelenik meg a tiltás lehetősége.



## 2.3. Rendszerterv

### 2.3.1. Projekt terv

A projekt egy közösségi média alkalmazás, ami PHP alapokon nyugszik. Ezek mellett HTML-t, CSS-t és JavaScript-et használok. Az adatbázis MySQL amit PhpMyAdmin segítségével hozok létre és XAMPP-on keresztül csatlakozok. Alkalmazásom részeit külön fejlesztem a fontosabb részekkel kezdve. Az ütemtervet és a feladatok kiosztását Trello-ban végzem, a verziókövetéshez SourceTree-t használok.

#### A projekt ütemterve:

1. Alkalmazásötlet kialakítása, megtervezése
2. A fejlesztőkörnyezetek kiválasztása, telepítése
3. Ütemterv megírása
4. Követelmények és funkciók specifikálása
5. Rendszerterv létrehozása
6. Adatbázis megtervezése és elkészítése
7. Backend funkciók implementálása
8. Felhasználói felület implementálása
9. Tesztelés és véglegesítés

**Menük, gombok, menü hierarchiák:** A menüben bejelentkezés előtt csak a Regisztráció és Bejelentkezés gombok lesznek láthatóak. Log-in után láthatjuk a Kezdőoldal, Csoportok és Felhasználók menügombokat, illetve a Profil gombot. Az utóbbira kattintva egy legördülő menüből érhető el a profilom oldal, ami a bejelentkezett felhasználó oldalára visz. Itt kap helyet egy Téma gomb, amivel a világos és sötét nézet között váltogathatunk. Ezeken kívül találunk itt egy Profil szerkesztése és egy kijelentkezés lehetőséget is.

Abban az esetben, ha van jogosultságunk szerkeszteni valamit, megjelenik egy menü az adott objektum mellett. Ha ez a saját tartalmunk, akkor a menüből a Szerkesztés illetve a Törlés gombokat érhetjük el. Ha valaki más tartalmát van jogosultságom törölni, akkor csak egy Tiltás gomb jelenik meg.

A csoportok oldalán 3 különböző gomb jelenhet meg: Belépés, Jelentkezés és Kilépés. Ha jóváhagyás nélkül jelentkezhetünk akkor a Belépés, ellenkező esetben a Jelentkezés gomb lesz elérhető. Ha már jelentkezünk, akkor a kilépés gombot fogjuk csak látni.

### 2.3.2. Fizikai környezet

Ez a projekt egy webes alkalmazás lesz, amihez szükségünk van webszerverre és adatbázisra. Szerverünk a 80-as porton lesz elérhető. Az adatok egy MySQL adatbázisban lesznek tárolva, amit a PhpMyAdmin szolgáltatás segítségével könnyen el tudunk érni a `http://localhost/phpmyadmin` elérési úton. A kliens lehet bármely számítógép ami képes webböngészőt, például Google Chrome-ot futtatni. A fejlesztéshez Visual Studio Code-ot fogok használni Laravel keretrendszerrel. A megjelenéshez Bootstrap-et használok.

### 2.3.3. Architektúrális terv

Az alkalmazás MVC-modellt (Model-View-Controller) használ. Az MVC egy architektúrális tervezési minta. Három fő rétegből áll: Model réteg, View réteg és Controller réteg.

A **modell** rétegben található az adatbázis. Itt kapnak helyet az adatok, ez felel a tárolásukért és lekérdezésükért is. Itt találhatóak az adatokon módosításokat végző függvények is.

2.1. kód. A poszt modelljének egy részlete

```
1 <?php
2 class Post extends Model
3 {
4     use HasFactory;
5     protected $guarded = [];
6     public $timestamps = false;
7
8     public function status(){
9         return $this
10         ->belongsTo('App\Models\Status', 'status_id');
11     }
12
13     public function group(){
14         return $this
15         ->belongsTo('App\Models\Group', 'group_id');
16     }
17
18     ...
```

A **nézet** réteg felel a felhasználói felület megjelenítéséért. Az adatok itt fognak megjelenni és ez fogja továbbítani a felhasználó által bevitt információkat a kontrollernek. A mi esetünkben itt kapnak helyet a blade.php fájlok.

Minden objektumtípusnál egy külön mappába tettem a megjelenítésért felelős fájlokat, mert egy oldal többféleképpen jelenhet meg. Például a ha egy törölt poszt URL-jét másoljuk a címsorba, akkor az annak megfelelő blade fájlt fogja betölteni.

#### 2.2. kód. Nem található komment oldal

```
1 @include('layouts.app')
2
3 <div class="alert alert-warning">
4     <strong>
5         Ez a komment nem létezik!
6     </strong>
7 </div>
```

A **Kontroller** réteg felelős a vezérlésért. Itt kapnak helyet a kontroller osztályok. Ez fogja meghívni a nézeteket (blade fájlokat) a 'view' rétegben és a függvényeket a modell rétegben.

#### 2.3. kód. A "felhasználó profil oldal" nézetet meghívó kontroller függvény

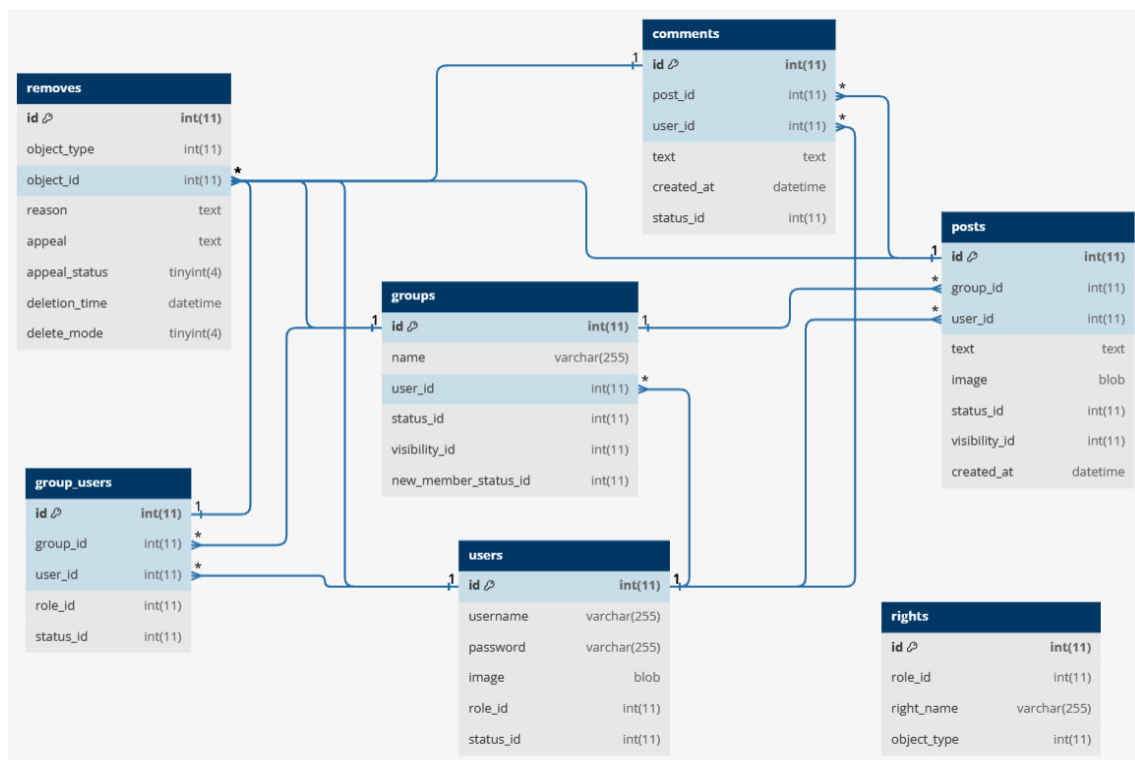
```
1 <?php
2 public function profile($id)
3 {
4     if (!(
5         $this->isLoggedIn() &&
6         $this->isActive()
7     )) return redirect()->back();
8
9     $data = User::where('id', $id) -> first();
10
11     if ($data === null) return view('user.profile.404');
12
13     $status = $data -> status -> name;
14
15     return view('user.profile.'. $status, [
16         'user' => $data,
17
18         'isAdmin' => $this->isAdmin(),
19         'isLoggedIn' => $this->auth('id') == $id,
20     ]);
21 }
```

A 4-7 sorban található egy jogosultság ellenőrzés. Ez a legtöbb metódus előtt lefut, hogy megtudjuk, van-e jogunk az adott műveletet végrehajtani. A példában csak Login után tekinthetjük meg a profil oldalt és csak akkor, ha a fiókunk aktív. (vagyis nincs törölve, vagy tiltva)

Ezután lekérdezzük a megadott ID alapján az adatot. Ha nincs ilyen, akkor a kontroller meghívja a 404.blade.php nézetet. Utána lekérdezzük a megtekinteni kívánt felhasználó státuszát és meghívjuk a megfelelő nézetet. A státusz lehet aktív, törölt vagy tiltott. Csak akkor láthatjuk a tartalmat, ha az aktív.

### 2.3.4. Adatbázis terv

Az adatokat MySQL adatbázisban tároljuk, itt lesznek tárolva a felhasználók, a csoportok, a posztok, a hozzászólások és egyéb adatok, például a felhasználók jelentkezései. Az adatbázis létrehozásához és szerkesztéséhez PhpMyAdmin szolgáltatást használók.



2.6. ábra. Az adatbázis fő táblái

A **Csoportok** (Groups) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
name	varchar	csoport neve.
user-id	int	csoport tulajdonosa.
status-id	int	csoport státusza.
visibility-id	int	csoport láthatósága.
new-member-status-id	int	új tag státusza

Egy **csoport** létrehozásánál meg kell adnunk a *nevét*, *láthatóságát* és azt, hogy az új tagok azonnal, vagy csak jóváhagyás után csatlakozhatnak-e. A *tulajdonos* automatikusan az lesz, aki létrehozta a csoportot. A *státusz* alapértelmezés szerint "aktív" lesz.

A **Felhasználók** (Users) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
username	varchar	egyedi felhasználónév.
password	varchar	hash kódolt jelszó.
image	blob	profilkép, opcionális.
role-id	int	felhasználó szerepköre.
status-id	int	felhasználó státusza.

Regisztráció után egy egyedi *felhasználónév* és egy hash-elt *jelszó* kerül az adatbázisba. A *profilkép* alapértelmezés szerint NULL, de ezt később módosíthatjuk. Ha már van legalább 1 adminisztrátor szerepkörű felhasználó az adatbázisban, akkor a 'user' szerepkört kapjuk meg. Ellenkező esetben 'admin'-ként kerülünk az adatbázisba. A *státusz* alapértelmezés szerint "aktív" lesz.

A **Posztok** (Posts) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
group-id	int	poszt csoportja.
user-id	int	poszt tulajdonosa.
text	varchar	poszt szövege.
image	blob	kép, opcionális.
status-id	int	poszt státusza.
visibility-id	int	poszt láthatósága.
created-at	datetime	létrehozás ideje.

Posztolás után a kiválasztott csoport és a bejelentkezett felhasználó azonosítója automatikusan bekerül az adatbázisba. Ezen kívül meg kell adnunk egy szöveget és fel tölthetünk egy képet is. A kép lehet akár gif formátumú is. Kiválaszthatjuk a poszt láthatóságát. Ha ütemezni szeretnénk, hogy mikor váljon elérhetővé a tartalmunk, akkor meg kell adnunk egy jövőbeli dátumot.

A **Megjegyzések** (Comments) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
post-id	int	poszt, amire a komment érkezett.
user-id	int	komment írója.
text	varchar	komment szövege.
status-id	int	komment státusza.
created-at	datetime	létrehozás ideje.

A felhasználók hozzászólhatnak mások posztjaihoz. Ehhez csak egy szöveget kell megadni. A többi adat automatikusan lesz beállítva.

A **Tagok** (Group-users) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
group-id	int	csoport, amibe a tag jelentkezett.
user-id	int	jelentkezett felhasználó.
role-id	int	tag szerepköre a csoporton belül.
status-id	int	tagság állapota.

A **Group-user** tábla köti össze a felhasználókat a csoportokkal. Ha jelentkezünk egy csoportba, akkor a felhasználónk ID-ja és a csoport ID-ja kerül az adatbázisba. A szerepkörünk kezdetben 'Tag', a státuszunk pedig attól függ, hogy a csoport tulajdonos automatikus, vagy jóváhagyásos csatlakozást állított-e be.

A **Törlések** (Removes) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
object-type	int	törölt objektum típusa.
object-id	int	törölt objektum azonosítója.
reason	varchar	törlés indoklása.
appeal	varchar	fellebbezés szövege.
appeal-status	int	fellebbezés állapota.
deletion-time	datetime	végleges automatikus törlés időpontja.
delete-mode	int	törlés típusa.

Annak érdekében, hogy a törlés még visszaállítható legyen, minden objektumnak van egy 'status-id' nevű mezője. Ennek értéke lehet aktív, törölt vagy tiltott. Alapértelmezetten aktív, de ha töröljük valamelyik tartalmunkat, akkor az állapot 'Törölt' lesz. Megadhatjuk, hogy csak a kijelölt tartalmat, vagy a hozzá fűződő tartalmakat is törölni szeretnénk. Ha nem mi, hanem egy moderátor törli valamely tartalmunkat, akkor annak állapota 'Tiltott' lesz.

A törlést még visszaállíthatjuk, erre 1 hetünk van. Abban az esetben, ha tiltást kapunk, elolvashatjuk a tiltás okát, ami a 'reason' mezőben van eltárolva. Fellebbezésünket az 'appeal' mezőben tárolja a rendszer. Az 'appeal-status' mező tárolja, hogy elküldtük-e már a fellebbezést, vagy sem. Ha elküldünk egy fellebbezést, az egy hetes visszaszámlálás újraindul, és az admin-nak el kell fogadnia, vagy elutasítania a fellebbezést. Egy hét leteltével a rekord automatikusan törlődni fog a **removes** táblából.

### A Jogosultságok (Rights) Tábla:

Mező	Típus	Leírás
ID	int	azonosító, elsődleges kulcs.
role-id	int	szerepkör, amivel a műveletet végrehajthatjuk.
right-name	varchar	jogosultság megnevezése.
object-type	int	objektum típusa, amin a műveletet el akarjuk végezni.

Ebbe a táblába kerülnek a jogosultságok és szerepkörök, amivel rendelkezünk kell a művelet elvégzéséhez. Moderátorként jogunk van mások tartalmát tiltani, de csak akkor, ha az "gyengébb" szerepkörű nálunk.

### Egyéb táblák, kapcsolótáblák:

Ezeknek a tábláknak a szerkezetük azonos, csak a tartalmuk különbözik. Egy **ID**-hoz egy **Név** tartozik.

A **szerepkör** (roles) tábla tartalma:

ID	Név	Prioritás
1	Admin	4
2	Moderátor	3
3	Tag	2
4	Vendég	1

A prioritás mező fogja meghatározni, hogy mely szerepkörnek legyen hatalma a másik felett. Ha két felhasználónak azonos a prioritása, akkor egyiknek sem lesz jogosultsága módosítani a másikat.



Az **állapotok** (status) tábla tartalma:

ID	Név
1	aktív
2	törölt
3	tiltott
4	várólistás
5	meghívott
6	bannolt

A 4-es, 5-ös és 6-os státuszok csak a group-user táblára vonatkoznak. Ha egy tagot kitiltunk valamely csoportból, akkor az 'tiltott' helyett 'bannolt' státuszt kap. Erre azért van szükség, mert a tiltásokat egy héten múlva törölné a rendszer, így bárki újra jelentkezhetne egy csoportba, ahonnan tiltva lett.

A **fellebbezés állapotok** (appeal-status) tábla tartalma:

ID	Név
1	nem elküldött
2	elküldött
3	elutasított
4	elfogadott

A fellebbezéseknek 4 állapotuk lehet. Abban az esetben, ha a tartalmunkat mi töröltük akkor a fellebbezés mező üresen marad, az 'appeal-status' mezőbe pedig a 4-es érték kerül. (ezért azt fellebbezés nélkül vissza tudjuk állítani)

Az 'elutasított' és 'elfogadott' értékek azért fontosak, hogy a felhasználó kapjon egy visszajelzést arról, hogy tartalma vissza lett-e állítva, vagy sem.

A **objektum típus** (object-type) tábla tartalma:

ID	Név
1	felhasználó
2	poszt
3	csoport
4	tagság
5	komment

Mivel lehet 1-es ID-val rendelkező poszt és csoport is, ezért fontos, hogy egyértelműen hivatkozzunk a kijelölt objektumra. Nem elég egy ID-t megadnunk, az objektum típusát is meg kell határoznunk. Ha például lekérdezzük a 2-es típusú 5-ös ID-val rendelkező objektumot, a rendszer egyértelműen tudni fogja, hogy mi az **5-ös** azonosítójú **poszt**-ra hivatkozunk.

A **Láthatóságok** (visibilitys) tábla tartalma:

ID	Név
1	nyilvános
2	nem listázott
3	privát

Beállíthatjuk, hogy kik lássák a tartalmainkat. Ha privátra állítjuk, akkor csak mi láthatjuk. Nem listázott beállítással posztunkat bárki megtekintheti, aki tagja a csoportnak és ismeri a poszt URL-jét. Nyilvános beállítással már meg fog jelenni posztunk a csoport főoldalán, azaz minden tag látni fogja.

Ezt nem csak posztoknál, hanem csoportoknál is beállíthatjuk. A privát csoportokba senki sem jelentkezhet, de meghívhatunk embereket, akik eldönthetik, hogy szeretnének-e csatlakozni.

### 2.3.5. Implementációs terv

Az alkalmazás PHP, HTML, CSS és Javascript nyelven fog elkészülni. A különböző komponenseket külön rétegekre kell bontani. A modell rétegben kapnak helyet az adatok és a rajtuk műveleteket végző metódusok, a nézet (view) rétegben kapnak helyet a blade fájlok amelyek a felhasználói felület megjelenítéséért felelnek. A vezérlő (controller) rétegben kapnak helyet az alkalmazás irányításáért felelős funkciók.

Elsőként a Regisztráció, bejelentkezés és kijelentkezés funkciókat implementálok. Ezután adom hozzá a felhasználó módosításával kapcsolatos opciókat, például a jelszó, vagy a profilkép módosítása lehetőséget. Minden objektumtípushoz meg kell írnom a létrehozás, megtekintés, szerkesztés és törlés funkciókat.

Következő lépés a csoportokat adom hozzá. Ez után következnek a tagságok, posztok és megjegyzések. A tagságokhoz meg kell írnom a csatlakozás, kilépés és új tag meghívása metódusokat.

Végül módosítom a törlés lehetőséget, és soft delete-et vezetek be. Implementálok a fellebbezés és a visszaállítás funkciókat. Létrehozom az automatikus törlés funkciót.

### 2.3.6. Telepítési terv

Első lépésként egy webszervert fog kelleni, ehhez használhatunk XAMPP-ot, vagy Docker konténereket.

A klienshez egy böngészőt futtatni képes operációs rendszerre, például Windowsra és egy webböngészőre, például Google Chrome-ra lesz szükségünk.

Kelleni fog egy adatbázis szolgáltatás, ami támogatja a PHP nyelvet és a MySQL adatbázist. Ehhez én PhpMyAdmin-t használtam.

Ezután jöhet a Laravel keretrendszer telepítése és a forráskód, illetve az adatok importálása.

### 2.3.7. Tesztelési terv

Minél előbb érdemes elkezdeni tesztelni az alkalmazást, már a fejlesztés korai szakaszában. Érdemes a határesetekre koncentrálni, szándékosan hibás adatokat megadni, hogy kiderüljön, megfelelően működik-e a funkció.

A manuális teszteléshez először készítettem egy listát, ahol felsoroltam azokat a komponenteket, modulokat, amelyeket érdemes letesztelni. Ezután végigmentem a projektem főbb funkcióin a tesztelési terv alapján. Ha hibákat találtam, azokat felírtam, majd javítás után újrakezdtem a tesztelést az első tesztlépéstől. Ez azért fontos, mert lehet, hogy a javítással elrontottam egy korábban működő funkciót.

A teszteknel érdemes kipróbálni olyan lehetőségeket is, amire a felhasználó nem is gondolna. Ilyen lehet például egy hibás URL megadása, vagy a profilkép feltöltésnél valamilyen más típusú fájl feltöltése.

Természetesen nincs kimerítő teszt, de megtalálhatjuk és javíthatjuk vele a hibákat. Fontos, hogy azokra a részekre koncentráljunk, ahol magas a hibák lehetősége. Egy részt érdemes többször többféleképpen is tesztelni, mert lehet, hogy bizonyos bemene-tekre kapunk hibát, míg másokra nem.

Minél több tesztet végzünk, annál jobban megbízhatunk a programunkban, de sosem lehetünk biztosak abban, hogy szoftverünk tökéletesen és hibátlanul működik.

ID	Tesztlépés	Siker / Hibaüzenet
1.	Az alkalmazás megnyitása	A program lefut, minden megfelelően megjelenik
2.	Regisztrációs oldal megnyitása	Az oldal megnyitható/elérhető
3.	Regisztráció helyes adatokkal	A regisztráció sikeres
4.	Regisztráció helytelen adatokkal	A hibaüzenet megjelenik, a regisztráció sikertelen
5.	Létező felhasználó regisztrálása	A hibaüzenet megjelenik, a regisztráció sikertelen
6.	Bejelentkezés oldal megnyitása	Az oldal megnyitható/elérhető
7.	Bejelentkezés létező felhasználóval	A bejelentkezés sikeres
8.	Bejelentkezés nem létező felhasználóval	A hibaüzenet megjelenik, a bejelentkezés sikertelen
9.	Bejelentkezés hibás jelszóval	A hibaüzenet megjelenik
10.	Kezdőlap oldal megnyitása	Az oldal minden tartalma megjelenik
11.	Felhasználók oldal megnyitása	Az oldal minden tartalma megjelenik
12.	Csoportok oldal megnyitása	Az oldal minden tartalma megjelenik
13.	Kommentek, posztok, csoportok kipróbálása	Az objektumok létrehozhatóak, törölhetőek, szerkeszthetőek és megtekinthetőek
14.	Kijelentkezés tesztelése	A felhasználó valóban kijelentkezett, csak a regisztráció és log-in funkciók elérhetőek.
15.	Admin, moderátor kipróbálása	Módosíthat-e bárkit/bármit?
16.	Láthatóság tesztelése	Nem láthatóak azok a tartalmak, amelyek megtekintéséhez nincs jogosultságunk
17.	Objektum törlése	A törölt objektum nem jelenik meg a listában
18.	Menük tesztelése	Minden megfelelően megjelenik, működik
19.	Csatlakozás és leiratkozás tesztelése	A jelentkezés, meghívás és kilépés működik A jelentkezők befogadhatóak és kidobhatóak

## 3. fejezet

# Telepítési útmutató

### 3.1. Bevezetés

Ez a telepítési útmutató a Windows operációs rendszerű számítógépekre vonatkozik. Lehet, hogy a telepítés eltérhet más operációs rendszereken.

Ebben a fejezetben fogom bemutatni, hogyan lehet a kész projektet telepíteni. Kétféle lehetőség van: telepíthetjük XAMPP, vagy Docker segítségével is. Első lépésként töltsük le a projektet a megadott github repository-ról:

`https://github.com/benedekklazar/Szakdolgozat\_HDMIJ3\_2023`

Ezután telepítsük a Composer-t a következő linkre kattintva:

`https://getcomposer.org/download/`

*Előfordulhat, hogy a rendszer nem adja hozzá automatikusan a ComposerSetup/bin mappát a rendszerváltozókhoz. Ebben az esetben keressünk rá a start menüben "A rendszer környezeti változóinak módosítása"-ra és a megjelenő ablakban kattintsunk a **Környezeti változók...** gombra. A rendszerváltozóknál jelöljük ki a PATH nevűt és a szerkesztés gombra kattintva adjuk hozzá a ComposerSetup bin mappájának elérési útját. Ez alapértelmezetten a /Users/NÉV/AppData/Local/ComposerSetup mappában található.*

### 3.2. Projekt futtatása XAMPP-al

Telepítsük a XAMPP-ot az Apache Friends weboldaláról. Használjuk a következő linket: `https://www.apachefriends.org/download.html`

Nyissuk meg a XAMPP-ot, majd kattintsunk az Apache és MySQL modulok sorában a Start gombra.



3.1. ábra. A XAMPP control panel

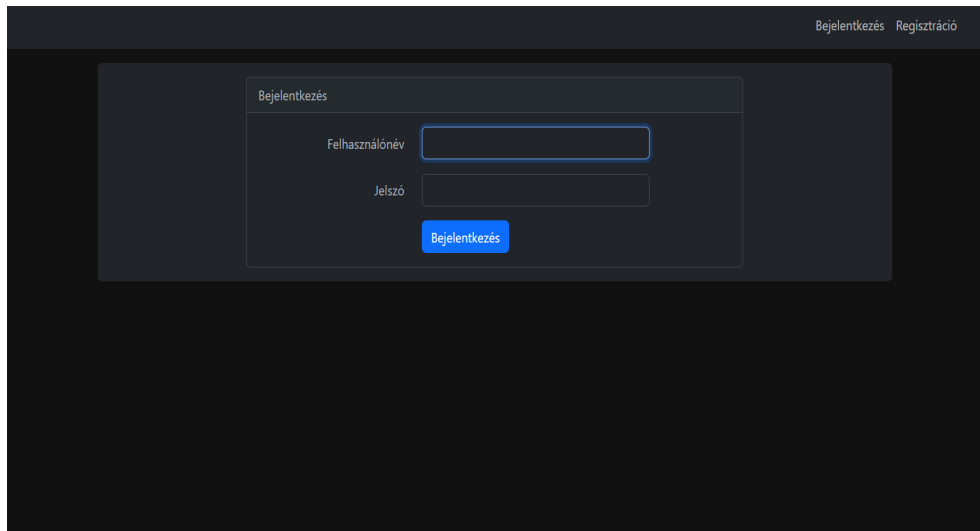
Ha ezekkel megvagyunk, nyissuk meg a projektet és győződjünk meg róla, hogy a projekt gyökérmappájában vagyunk.

Itt található egy start.ps1 nevű fájl. Az előkészítéshez ezt a fájlt kell futtatnunk. Ehhez írjuk be a parancssorba a következő powershell utasítást:

**powershell -ExecutionPolicy Bypass -File start.ps1**

Ezzel elindul a Composer telepítése. Ha mindent jól csináltunk, megjelenik a vendor mappa a navigációs ablakban. Ezután létrejön az .env fájl az .env\_2.example fájl alapján. Majd legenerálja az alkalmazás kulcsot és létrehozza az adatbázist. Következő lépésként futtatja a migrációkat, legenerálja a táblákat és feltölti alapértelmezett adatokkal. Végül elindítja az alkalmazást a 8000-es porton.

Nyissuk meg a böngészőt és írjuk a címsorba következő linket, vagy kattintsunk ide: <http://localhost:8000/>



3.2. ábra. Az alkalmazás nyitóoldala

Az alkalmazás nyitóoldala jelenik meg. Nézzük meg az adatbázist is, ehhez kövessük ezt a linket: <http://localhost/phpmyadmin/>

Ha a generálás és migrációk futtatása során nem kaptunk hibát, itt találjuk a 'szakdolgozat\_hdmij3' nevű adatbázist, benne a táblákkal és kezdő adatokkal.

### 3.3. Projekt futtatása Docker-rel

Telepítsük a Docker desktop-ot a következő linkre kattintva:

<https://docs.docker.com/desktop/install/windows-install/>

A telepítéshez szükség lesz linux kernel-re is. Ezt úgy érhetjük el, hogy a parancssorba begépeljük a **wsl --install** parancsot.

Ezek után, nyissuk meg a projektet és ellenőrizzük, hogy a projekt gyökérmappájában vagyunk-e.

Itt találunk egy `docker.ps1` nevű fájlt. Az előkészítéshez ezt a fájlt kell futtatnunk. Ehhez gépeljük be a terminálba a következő powershell parancsot:

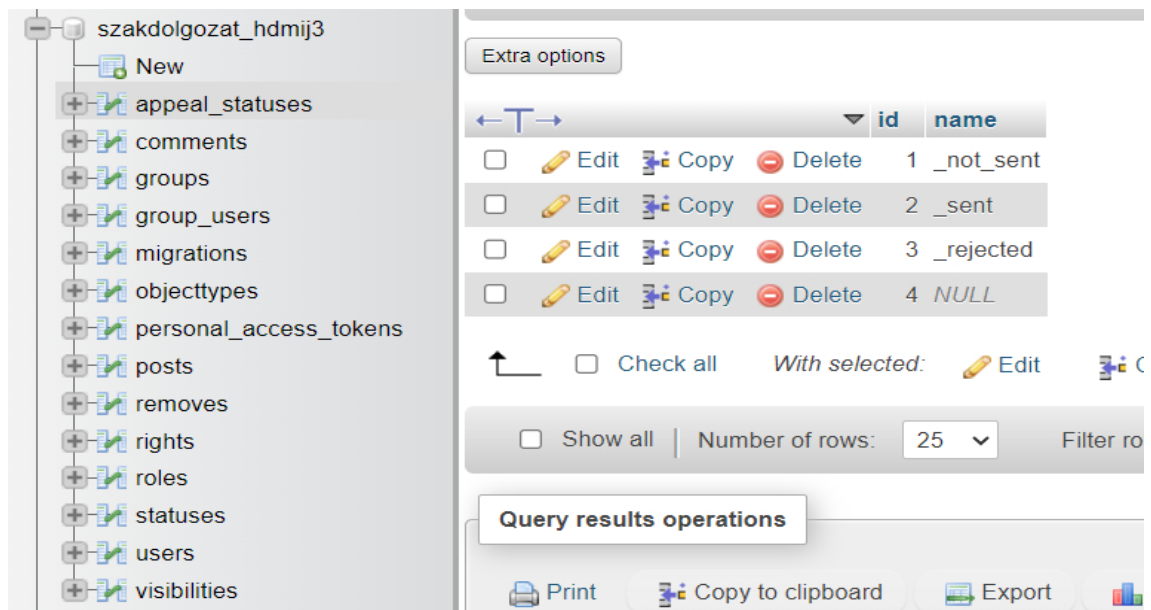
**powershell -ExecutionPolicy Bypass -File docker.ps1**

Először létrejönnek a Docker konténerek, amelyek a `docker-compose.yml` fájlban lettek definiálva. Ezután lefut a Composer install, ha még nem létezne az `autoload.php` fájl a vendor mappában. Következő lépésként az `.env` fájl jön létre az `.env.example` fájl alapján. Majd legenerálja az app kulcsot és futtatja a migrációkat a 'php' nevű docker

szolgáltatásban. Legvégül elindítja az alkalmazásunkat a 8000-es porton.

Kattintsunk a következő linkre: <http://localhost:8000/>

Az alkalmazás nyitóoldalán találjuk magunkat.



3.3. ábra. Az adatbázis PhpMyAdmin-ban

A PhpMyAdmin szolgáltatás a 8090-es porton fog futni. <http://localhost:8090/>  
Először be kell jelentkezünk. Az egyszerűség kedvéért a felhasználónév és a jelszó is 'root'. Ezt megváltoztathatjuk az .env fájl módosításával.

Sikeres bejelentkezés után láthatjuk az adatbázisunkat, táblákkal és adatokkal.



## 4. fejezet

# Használt technológiák

### 4.1. Bevezetés

Ebben a fejezetben fogom ismertetni azokat a technológiákat és szoftvereket, amelyeket a fejlesztés során használtam.

### 4.2. Technológiák

#### 4.2.1. XAMPP

A XAMPP egy nyílt forráskódú szabad, platformfüggetlen webservert. Főbb szolgáltatásai az Apache, a MySQL, a FileZilla, a Mercury és a Tomcat. A projekt adatbázisához és futtatásához csak az Apache és a MySQL szolgáltatásokat kell elindítani. A XAMPP egy betűszó, amely a következőt jelenti:

1. **X** a platformfüggetlenséget jelenti
2. **A**pache webservert
3. **M**ariaDB, vagy **M**ySQL adatbázis
4. **P**HP szervertoldali szkriptnyelv
5. **P**erl általános célú szkriptnyelv

A XAMPP elérhetővé teszi a PhpMyAdmin szolgáltatást is, mely segítségével hoztam létre, és szerkesztettem az adatbázisom tábláit és mezőit.

#### 4.2.2. MySQL

A MySQL egy nyílt forráskódú adatbázis kezelő rendszer, amely SQL nyelvet használ. Az SQL (Structured Query Language) egy strukturált lekérdezőnyelv, ami segítségével adatokat tudunk tárolni, lekérdezni, módosítani és törölni. A rendszer az adatokat

táblákba rendezi. Ezeknek a tábláknak vannak oszlopaik, melyeket mezőknek, illetve soraik, melyeket rekordoknak nevezünk. Az eltárolt adatok gyakran összefüggnek egymással. Az adatok között lehet 1 az egyhez, egy a többhöz, és több a többhöz kapcsolat is.

4.1. kód. A posztok táblát létrehozó kód SQL nyelven

```
1 CREATE TABLE `posts` (  
2   `id` int(11) NOT NULL,  
3   `group_id` int(11) NOT NULL,  
4   `user_id` int(11) NOT NULL,  
5   `text` text NOT NULL,  
6   `image` blob DEFAULT NULL,  
7   `status_id` int(11) NOT NULL,  
8   `visibility_id` int(11) NOT NULL,  
9   `created_at` datetime NOT NULL  
10 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

A .env fájlban adhatjuk meg, hogy milyen fajta adatbázist szeretnénk használni a projektünkhöz. A projektben a DB\_CONNECTION változó értéke 'mysql'-re lett állítva.

### 4.2.3. PhpMyAdmin

A PhpMyAdmin egy adatbázis menedzsment eszköz, amit php nyelven írtak. Segítségével könnyen el lehet érni az adatbázisunkat és módosításokat tudunk rajta végrehajtani az interneten keresztül. Adatbázisokat és táblákat készíthetünk vele, a táblákat adatokkal tölthetjük fel.

A projekt adatbázisát itt hoztam létre, illetve a tesztelés során itt ellenőriztem, hogy egy adat sikeresen bekerült-e az adatbázisba, módosítva, vagy esetleg törölve lett.

### 4.2.4. Laravel

A laravel egy kifejezetten PHP webalkalmazások fejlesztését elősegítő keretrendszer, amely MVC (Model View Controller) tervezési mintát használ. Sokat segít az adatbázis lekérdezések megírásában, hogy a fejlesztőknek azzal már ne kelljen foglalkozniuk. Továbbá készíthetünk alapértelmezett adatokat és migrációkat az adatbázisunkhoz. A telepítés után létrejönnek a model, view és controller mappák, használhatjuk a blade sablonokat a nézet (view) rétegben, illetve az artisan parancsok is a rendelkezésünkre fognak állni. Kapni fogunk egy 'routes' mappát, ahol felsorolhatjuk az alkalmazásunk linkjeit, és megadhatjuk, hogy mely metódusok legyenek meghívva. Legenerálja az autentikációs és regisztrációs fájlokat, az 'auth' mappában, egy alapértelmezett bejelentkezési és regisztrációs oldallal.

#### 4.2. kód. A felhasználóhoz tartozó elérési utak

```
1 Route::get('/user',[UserController::class,'list']);
2 Route::get('/user/{id}',[UserController::class,'profile']);
3 Route::get('/user/edit/{id}',[UserController::class,'edit']);
4 Route::post('/user/edit/{id}',[UserController::class,'update'])
5     ->name('user.update');
6
7 Route::get('/user/delete/{id}',
8     [UserController::class,'delete']);
9 Route::get('/user/remove/{id}',
10    [UserController::class,'remove']);
```

A laravel egy jelenleg is bővülő keretrendszer, melyhez folyamatosan adnak hozzá újabb és újabb funkciókat. A Laravel weboldalán sok hasznos információval találkozhatunk, rengeteg példa kóddal és dokumentációval, amely akár a kezdő programozók számára is könnyen érthető. A dokumentációban minden fontos tudnivaló le van írva a keret-rendszerről, megismerhetjük az adatbázis kezelést, a migrációk működését és leírásokat kapunk a beépített osztályok és metódusok használatáról, illetve működéséről is.

A Laravel keretrendszer hivatalos dokumentációját a következő linkről érhetjük el: <https://laravel.com/docs/10.x>

#### 4.2.5. CSS

A CSS, vagyis Cascading Style Sheets (magyarul: "lépcsőzetes stíluslapok") egy stílusle-író nyelv, amely főként HTML dokumentumok megjelenését írja le. Ennek segítségével megadhatjuk a betűk színét, a háttérszínt, betűméretet, képek méretét és sok egyéb beállítást a kinézetre vonatkozóan.

#### 4.3. kód. Listaelem kijelölése sötét nézetben

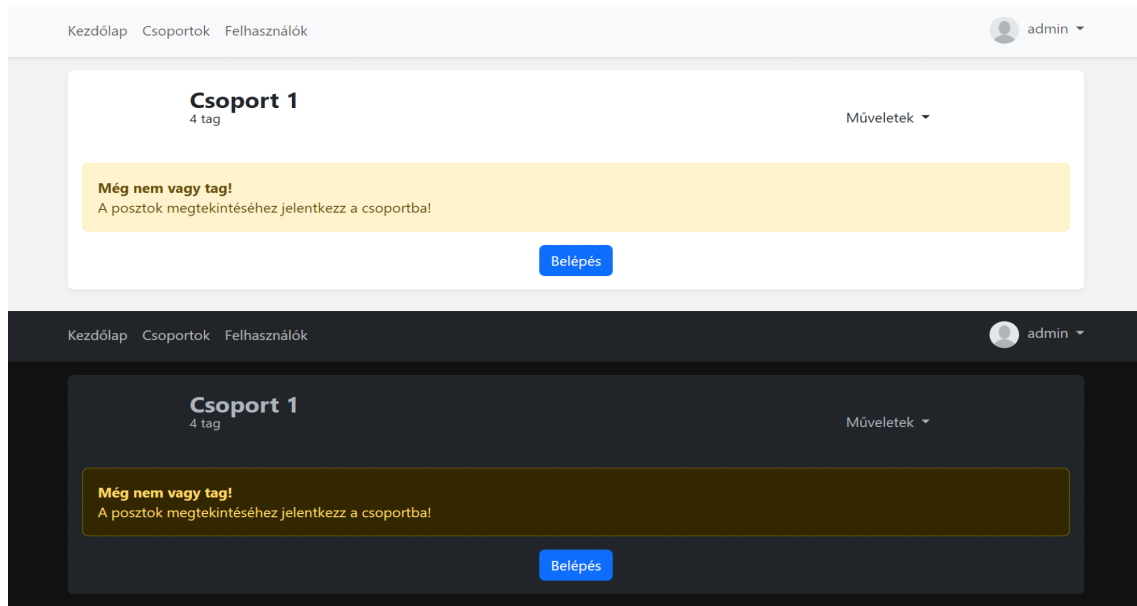
```
1 [data-bs-theme=dark] .element :hover td {
2     background-color: rgba(255,255,255,.02);
3     color: white;
4 }
5
6 [data-bs-theme=dark] .element :hover span {
7     filter: brightness(105%);
8 }
```

Az alkalmazásom több ilyen fájlt használ, hogy a listákat és menüket megfelelően

megjelenítse. Ezekben a fájlokban van leírva, hogy ha az egérmutatót egy listaelem fölé viszem, akkor annak háttere legyen egy kicsit világosabb, vagy sötétebb a megadott témától függően. (Sötét nézetben világosabb, míg világos nézetben sötétebb lesz) Külön blokkban van leírva a világos és sötét nézetre vonatkozó stílus. Ezek a stílusok vonatkoznak a listákra, és a legördülő menükben található listaelemekre is. A css fájlokban kívül a blade fájlokban is meghatároztam stílusokat. Ezek csak a kijelölt HTML elemre fognak vonatkozni. Itt a 'class' tag-en belül meg kell adni egy egyedi nevet, a CSS fájlban pedig erre tudunk hivatkozni.

## 4.2.6. Bootstrap

A bootstrap egy frontend fejlesztőeszköz csomag, amely segítségével könnyen készíthetünk grafikai elemeket, például gombokat és űrlapokat weboldalunkhoz. HTML, CSS és JavaScript bővítményeket használ.



4.1. ábra. Bootstrap-el készült alert világos és sötét témával

Ha megnyitjuk az alkalmazást, az alapértelmezetten sötét lesz, de van lehetőségünk kiválasztani, hogy világos vagy sötét témával szeretnék-e használni. A beállításunkat menteni fogja a rendszer a böngésző sütije között, ezért az addig érvényes marad, amíg meg nem változtatjuk.

Ezen kívül van egy "Eszköz Témája" lehetőség is, amelyet, ha kiválasztunk, a rendszer automatikusan azt a témát fogja kiválasztani, amelyik a számítógépünkön alapértelmezett témaként van beállítva.

#### 4.4. kód. A világos nézetet beállító JavaScript kód

```
1 <script>
2 function setlight() {
3     document.documentElement.setAttribute(
4         'data-bs-theme', 'light');
5     localStorage.setItem('theme', 'light');
6 }
7 </script>
```

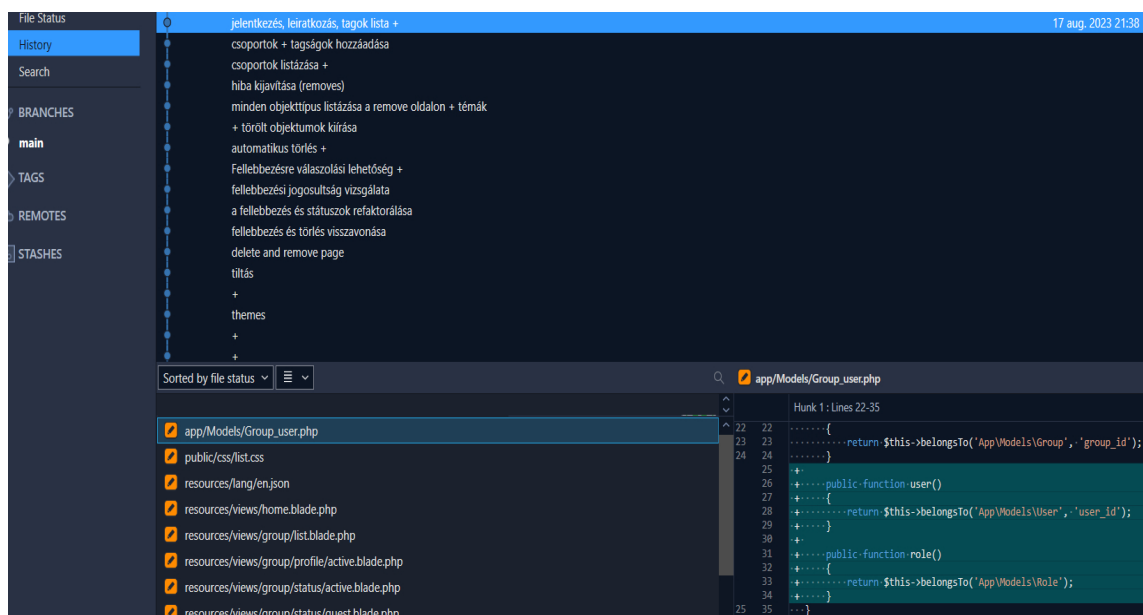
Ez a kód először a 'data-bs-theme' stílus osztály értékét világosra állítja, majd létrehoz egy 'theme' nevű változót a böngészőnk lokális tárhelyében, amit 'light' értékre állít.

Ha újratöltjük az oldalt, a data-bs-theme változó értékét a böngészőnkben elmentett értékre fogja állítani, amennyiben már van ilyen. Ha még nem állítottunk be stílust, a program a sötét témát fogja választani.

## 4.3. Egyéb szoftverek és alkalmazások

### 4.3.1. SourceTree

A SourceTree egy ingyenes Git kliens Windows és Mac rendszerekhez. A clone parancs segítségével feltölthetünk a számítógépünkön lévő projektet egy megadott GitHub repository-ba. Minden hozzáadott sort zölddel, míg a törölt sorokat pirossal fogja jelezni.



4.2. ábra. A SourceTree alkalmazás

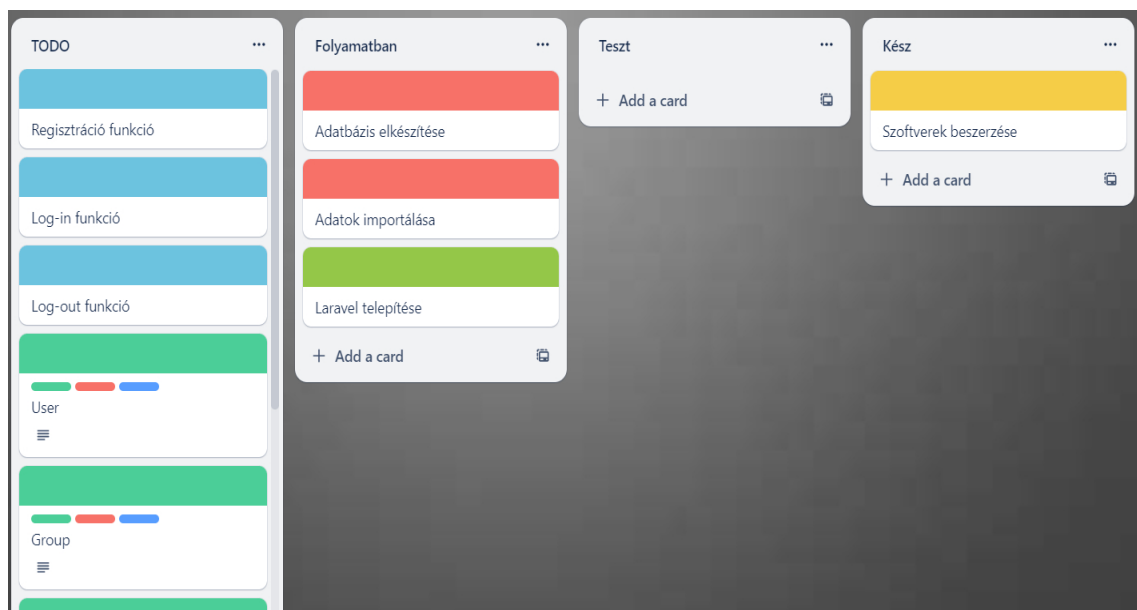
A **'commit'** paranccsal a módosított és elmentett, de még fel nem töltött fájlok listáját tudjuk megtekinteni. Innen kiválaszthatjuk, hogy mely fájlokat szeretnénk végül feltölteni a GitHub-ra. A **'pull'** paranccsal a repository-ból tudjuk letölteni a fájlokat a lokális gépre, míg a **'push'** gomb megnyomásával a változtatások fognak felkerülni a repository-ra.

A **'branch'**-el egy új ágot hozhatunk létre, ami hasznos, főleg ha többen dolgozunk egy projekten. Így a módosítások nem azonnal a main branch-re kerülnek, hanem egy saját ágra, amit a **'merge'** paranccsal vezethetünk vissza a fő ágba.

### 4.3.2. Trello

A trello egy kanban stílusú lista készítő webalkalmazás, ami a csoportos munkákat segíti elő. A felhasználók létrehozhatnak kártyákat, melyek egy-egy feladatot szimbolizálnak. Ezekhez hozzárendelhetnek egy felhasználót azaz azt a személyt, aki a feladatot kapta, és egy dátumot, ameddig azt teljesítenie kell.

Segítségével a csapatok vizuálisan is láthatják, hogy éppen hol tart a projekt, melyek azok a feladatok, amelyek éppen folyamatban vannak.


























4.3. ábra. A trello projekt oldala

Tervezés után itt hoztam létre a követelménylistában szereplő feladatokat. Kezdetben nagyobb story-kat hoztam létre, melyeket később kisebb feladatokra (task-ekre) bontottam. Először a "TO-DO" (teendő) listából kiválasztottam egy feladatot, amelyet áthelyeztem a "Folyamatban" listára. Elkezdtem a rész kidolgozását, majd áthelyeztem

a kártyát a "Teszt" oszlopba. Csak miután teljesen elkészítettem és leteszteltem egy részt, tettem a hozzá tartozó kártyát a "Kész" listába.

### 4.3.3. Docker

A Docker az egyik legnépszerűbb konténerizáló szoftver, amely képes operációs rendszer szintű virtualizációra. Ezzel kisebb szolgáltatásokat tudunk egy konténerbe csomagolni, egymástól elkülönítve.

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 <a href="#">szakdolgozat_hdmij3_2023</a>		Running (4/4)	0%		18 seconds ago	  
<input type="checkbox"/>	 <a href="#">node-1</a>	<a href="#">szakdolgozat_hdmij3_2023-node</a>	Running	0%		1 minute ago	  
<input type="checkbox"/>	 <a href="#">db-1</a>	<a href="#">mysql:latest</a>	Running	0%	<a href="#">3306:3306</a> 	27 seconds ago	  
<input type="checkbox"/>	 <a href="#">phpmyadmin-1</a>	<a href="#">phpmyadmin/phpmyadmin</a>	Running	0%	<a href="#">8090:80</a> 	18 seconds ago	  
<input type="checkbox"/>	 <a href="#">php-1</a>	<a href="#">szakdolgozat_hdmij3_2023-php</a>	Running	0%	<a href="#">8000:8000</a> 	19 seconds ago	  

4.4. ábra. A docker konténer a futó szolgáltatásokkal

Az én webalkalmazásomban 4 ilyen elkülönített mikroszervíz van. A PHP, a MySQL, a Node és a PhpMyAdmin. A PHP szolgáltatáson keresztül fut a webszerver, a "db"-nek elnevezett MySQL-en pedig létrehozza az adatbázist a 3306-os porton. Az adatbázist a 8090-es porton lévő PhpMyAdmin szolgáltatással tudjuk elérni, illetve módosítani felhasználónév és jelszó megadását követően.

## 5. fejezet

# Megvalósított funkciók

### 5.1. Bevezetés

Ebben a fejezetben bemutatom az alkalmazásom főbb funkcióit és a működésüket. Ezeket a funkciókat a követelmény specifikációban található követelménylista alapján készítettem el. Minden metódus tartalmaz egy jogosultság ellenőrzést, hogy a felhasználók csak azokat a funkciókat használhassák, amelyek elérhetőek számukra. Ezután jön az adatok lekérdezése az adatbázisból. Végül a metódus megnyitja a megfelelő nézetet, vagy valamilyen adatbázis művelet után visszairányít minket az előző oldalra.

### 5.2. Regisztráció és Bejelentkezés

A Laravel egyik sajátossága, hogy a keretrendszer telepítésekor az autentikációs és regisztrációs funkciókat is legenerálja, ezért nekem csak néhány módosítást kellett ezeken végrehajtanom.

5.1. kód. A regisztrációs adatok validálása

```
1 protected function validator(array $data)
2     {
3         return Validator::make($data, [
4
5             'username' => ['required', 'string',
6                 'min:5', 'max:255', 'unique:users'],
7
8             'password' => ['required', 'string',
9                 'min:5', 'confirmed'],
10        ]);
11    }
```

Regisztráció során egy legalább 5, maximum 255 karakter hosszú egyedi string-et kell



megadni felhasználónévként, továbbá meg kell adnunk egy jelszót 2-szer, annak érdekében, hogy ne zárjuk ki magunkat egy elgépelés miatt. Ezeket a 'validator' nevű metódus ellenőrzi, és egy piros hibaüzenetet jelenít meg az űrlapon, ha valamely feltétel nem teljesül. A blade-ben minden input mezőhöz tartozik egy **@error** blokk, amiben egy alapértelmezett hibaüzenet fog megjelenni. Ezt módosítottam úgy, hogy a hibákat magyarul írja ki. Ehhez a 'resources' mappában található validation.php fájlt kellett módosítanom. Ennek segítségével akár több nyelvet is hozzáadhatunk a projektünkhöz.

A bejelentkezés is hasonlóan működik. Ehhez meg kell adnunk a felhasználónevünket és a jelszavunkat. A rendszer megkeresi, hogy létezik-e felhasználó a megadott névvel az adatbázisban, majd ellenőrzi, hogy helyes jelszót adtunk-e meg.

5.2. kód. Az auth metódus

```
1 public function auth($prop)
2     {
3         if (Auth()->user()) {
4             return Auth()->user()->{$prop};
5         } else {
6             return null;
7         }
8     }
```

Az 'auth' metódus a bejelentkezett felhasználó egy paraméterben megadott tulajdonságával tér vissza. Például, ha a bejelentkezett felhasználó azonosítóját szeretnénk lekérdezni, a \$prop paraméternek 'id' értéket kell megadnunk.

## 5.3. Profil oldalak

A profil oldalak mindig egy azonosítót várnak, és egy nézettel térnek vissza, amennyiben van jogosultságunk annak megtekintéséhez. Nézzük meg mi történik, ha megnyitom a 'user/1' linket. Ilyenkor a UserController-ben lefut a profile metódus, amely 1-es értéket fog paraméterül kapni. Jogosultság ellenőrzés után ellenőrzi, hogy létezik-e 1-es azonosítóval rendelkező felhasználó, és ha igen, akkor betölti a user.profile.active nézetet. (ha a felhasználó aktív)

## 5.4. Létrehozás

A létrehozáshoz szükségünk van egy 'create' és egy 'create\_form' metódusra. Néhány esetben meg kell adnunk egy paramétert is, például poszt létrehozásánál egy csoport

azonosítót. Erre azért van szükség, hogy tudjuk, a posztot mely csoportban szeretnénk megosztani.

A `'create_form'` metódus egy űrlap nézetet fog megnyitni, ahol megadhatjuk az új objektum adatait. A kész gomb megnyomása után lefut a `'create'` metódus, amely az adatokat egy `INSERT sql` utasítással adja hozzá az adatbázishoz.

## 5.5. Módosítás

A módosításhoz 2 külön metódus kell. Egy (`edit`) ami megnyitja az űrlapot, ahol módosítani tudjuk az adatokat. Ezek `GET` módszert alkalmaznak. Ez azt jelenti, hogy az adatokat az URL-en keresztül adja át a metódusnak. Itt nekünk csak egy azonosítót (például: `'user/edit/1'`) kell megadnunk, ezzel jelezve, hogy melyik adatot szeretnénk módosítani.

A másik metódus (`update`) fogja végrehajtani a módosításokat. Ez `POST` küldést használ és akkor fut le, amint megnyomjuk a `'Szerkesztés'` gombot. A módosításokat egy **Request** típusú példányban adja át az `'update'` metódusnak. Természetesen a kapott adatokat először validálni kell. Ha mindent rendben találtunk, lefut az `UPDATE sql` utasítás a megadott adatokkal.

## 5.6. Törlés és Tiltás

A törléshez és tiltáshoz meg kell adnunk a törölni kívánt objektum azonosítóját az URL-ben. A metódus ellenőrzi, hogy a megadott adatot van-e jogunk törölni. Törlés esetén azt vizsgálja, hogy a kijelölt objektum `'user_id'` mezőjében a bejelentkezett felhasználó azonosítója szerepel-e. Tiltás esetén azt ellenőrzi, hogy a bejelentkezett felhasználónak van-e Admin vagy Moderátor jogosultsága.

A törléshez és tiltáshoz is két-két külön metódust használtam. Egyet az űrlaphoz, és egyet az adatok módosításához. Az űrlapon meg lehet adni a törlés okát és módját. A `'submit'` gomb megnyomása után a `'delete'` metódusban valójában nem egy törlés, hanem egy létrehozás és egy módosítás megy végbe. Először létrehoz egy Remove rekordot, majd módosítja a törölni kívánt objektum státuszát töröltre.

### 5.6.1. Automatikus törlés

Az automatikus törlés a Remove táblában található `delete_time` alapján történik. Először kilistázza az összes olyan rekordot, amely törlési ideje kisebb, mint a jelen időpont. Ezután Stratégia minta segítségével kiválasztja a megfelelő törlés metódust. Ezt a fellebbezések állapota alapján teszi meg. Abban az esetben, ha már küldtünk fellebbezést,

és lejárt az idő anélkül, hogy választ kaptunk volna, az `Appeal_sent.php` fájlban található `auto_delete` metódus fog lefutni. Ilyenkor a tartalmunk automatikusan helyreáll, és a fellebbezés sikeresnek minősül.

### 5.6.2. Törlési opciók

Törlés létrehozásánál kiválaszthatjuk, hogy csak a kijelölt, vagy minden kapcsolódó tartalmat törölni szeretnénk-e. Ha csak a kijelölt objektumot töröljük, akkor a `$depth` változó értéke 1 lesz. Ez határozza meg, hogy milyen mélységig fogjuk a tartalmakat törölni. Ha minden kapcsolódó tartalmat törölünk, akkor a `'delete'` metódus rekurzívan fog lefutni, egészen addig, amíg el nem éri a megadott mélységet. Miután kigyűjtöttük a kapcsolódó tartalmakat, ugyanez a törlés metódus fog lefutni egyesével az összes ilyen objektumon, azzal a különbséggel, hogy a mélységet növeljük 1-el.

## 5.7. Feliratkozás, leiratkozás

A csoportokkal kapcsolatos főbb funkciók a Feliratkozás, Leiratkozás és Meghívás.

Csatlakozásnál először azt ellenőrizzük, hogy a felhasználó tagja-e már a csoportnak. Ha már tagja, akkor ne legyen lehetősége újra csatlakozni. Ezután megnézzük, hogy a csoport adminisztrátora engedélyezi-e a jelentkezést. Ha a `'new_member_status'` mező értéke `NULL`, vagy a csoport láthatósága privát, akkor nincs lehetőségünk csatlakozni. Ha minden ellenőrzésen túljutottunk, akkor a felhasználó bekerül a csoportba, vagy felkerül a várólistára az admin beállításaitól függően.

Ha várólistára kerültünk, meg kell várnunk az `'admiss'` metódus futását. Ezt csak egy csoportadmin, vagy moderátor tudja meghívni. Ez a metódus egy várólistás tagot, aktív tagra fog módosítani.

A meghívásnál csak egy űrlapot kell kitöltenie az adminnak, ahol megad egy olyan felhasználót, aki még nem tagja a csoportnak. Ezt a meghívott, az `'accept'` metódus futtatásával tudja elfogadni. Ez a metódus akkor fut le, ha a csoport főoldalán megnyomjuk az **Elfogad** gombot.

5.3. kód. A csatlakozás metódus

```
1 public function join($id)
2     {
3         if (!(
4             $this->isLoggedIn() &&
5             $this->isActive()
6         )) return redirect()->back();
```

```

7
8     $current_membership = Group_user::where('group_id', $id)
9     ->where('user_id', $this->auth('id'))->first();
10
11     $group = Group::where('id', $id) -> first();
12
13     $new_member_status = $group -> new_member_status_id;
14
15     if ($current_membership != null)
16     return redirect()->back();
17
18     if ($new_member_status == null)
19     return redirect()->back();
20
21     if ($group -> visibility -> name == "private")
22     return redirect()->back();
23
24     Group_user::create([
25         'group_id' => $id,
26         'user_id' => $this->auth('id'),
27         'role_id' => 3,
28         'status_id' => $new_member_status
29     ]);
30
31     return redirect()->to('/group/'.$id);
32 }

```

## 5.8. Jogosultságok kezelése

Sok olyan metódus került az alkalmazásba, amit csak adminisztrátor, vagy moderátor tud használni. Ezek előtt a hasRight nevű metódus ellenőrzi, hogy van-e jogunk az adott művelethez. Ez a metódus mindig igazgal tér vissza, ha adminként, vagy a csoport tulajdonosaként vagyunk bejelentkezve. Ha ezek közül egyik sem igaz, akkor először összehasonlítja a szerepkörünket annak a felhasználónak a szerepkörével, aki birtokolja a megadott objektumot. Végül megnézi, hogy a Right táblában létezik-e a jogosultság és szerepkörünk elég magas-e a művelet elvégzéséhez. Ha magasabb rangúak vagyunk, mint a szerkeszteni kívánt objektum tulajdonosa és a rendszer megtalálta a jogot a Right táblában, a metódus TRUE-val tér vissza és szabadon szerkeszthetjük a kijelölt tartalmat.

A metódusnak meg kell adnunk egy jogosultságot, és egy objektum típust és azonosítót. A típus és azonosító segítségével fogja megtalálni azt a tartalmat, amit módosítani,

esetleg törölni szeretnénk. A jogosultsággként egy szöveget adunk át, ezt fogja megke-  
resni a 'Right' táblában.

## 5.9. Fellebbezés és helyreállítás

Az alkalmazásom lehetőséget biztosít a felhasználóknak, hogy törölt tartalmaikat hely-  
reállítsák. Ehhez egy űrlapot kell kitölteni, ahol megadják, hogy miért gondolják a  
tiltást tévesnek. Ez az űrlap egy POST küldéssel elmenti a fellebbezést az adatbázis-  
ban és a fellebbezés állapotát 'elküldött'-re módosítja. Ezen kívül az egy hetes vissza-  
számlálást újraindítja. Azt, hogy kinek legyen lehetősége fellebbezni, vagy fellebbezést  
elfogadni, az 'isDefendant' és 'isClaimant' metódusok fogják meghatározni. Az előbbi  
azt vizsgálja, hogy a bejelentkezett felhasználó tartalma lett-e törölve. Ha igen, akkor  
a kezdőoldalon megjelenik egy új listaelem, amely tartalmazza az eltávolított tartal-  
mat, a fellebbezés állapotát és a végleges törlés idejét. Erre kattintva érjük el az űrlapot.

Az isClaimant metódus fogja meghatározni, hogy ki az aki válaszolhat a fellebbezé-  
sünkre. Erre az 'admin' szerepkörű felhasználóknak van lehetőségük, feltéve, hogy a  
letiltott tartalom az ő csoportjukban lett megosztva.

Mindkét metódus egy Remove rekord azonosítóját várja paraméterül. Ez az objektum  
típus és objektum azonosító mezők alapján visszakeresi a törölt objektumot, majd az  
isDefendant metódus esetén a 'user\_id'-t összehasonlítja a bejelentkezett felhasználó-  
val, az isClaimant esetén pedig a 'group\_id' segítségével ellenőrzi, hogy a bejelentkezett  
felhasználónak van-e admin jogosultsága a csoportban.

## 6. fejezet

# Továbbfejlesztési lehetőségek

### 6.1. Bevezetés

Egy alkalmazás sosem kész, mert állandóan felmerülnek új felhasználói igények. Fontos, hogy bővíthető és rugalmas kódot írjunk, hogy ezeket az új igényeket könnyedén ki lehessen elégíteni. Ebben a fejezetben néhány olyan ötletet írok le, amivel az alkalmazásomat bővíteni lehet.

### 6.2. Néhány bővítési lehetőség

**Videók megosztása:** Az egyik ilyen lehetőség, hogy ne csak képeket, hanem videókat, vagy egyéb fájlokat is meg lehessen osztani. Ezzel a felhasználók több lehetőséget kapnak arra, hogy tartalmaikat megosszák másokkal.

**Csoport szabályok:** Jelenleg egy admin eldöntheti, hogy az új tagok csak jóváhagyással csatlakozzanak. Ezt a funkciót bővíteni lehetne, hogy posztokra is vonatkozzon. Így a poszt privát maradna mindaddig, amíg azt egy admin jóvá nem hagyja.

**Egyedi szerepkörök:** A csoporttulajdonosok kioszthatnának egyedi szerepköröket a meglévő admin, moderátor, tag és vendég szerepkörökön kívül, melyeknek megszabhatnák a jogait. Ezzel bővítenénk az adminok lehetőségeit.

**Linkek, értesítések:** Poszt létrehozásánál a szövegben elhelyezhetnénk kattintható linkeket. Megjelölhetnék felhasználókat, akik értesítést is kapnak, vagy megoszthatnánk mások csoportjait. Ezzel a felhasználók könnyebben megtalálhatnák egymást.

**Szövegek formázása:** A posztok és kommentek szövegét akár formázni is lehetne, ezzel kiemelve a fontosabb részeket. Hozzáadhatnánk kattintható linkeket, színezési le-

hetőséget, illetve félkövér és dőlt formázási lehetőségeket is.

**Albumok:** Jelenleg csak egyesével tudunk képeket megosztani és ha új képet szeretnénk létrehozni, ahhoz egy külön posztot kell csinálnunk. Továbbfejlesztési lehetőségként egy posztban akár több képet is megoszthatnánk. Ezeket egy külön 'images' nevű táblában tárolnánk és a poszt azonosítójával hivatkoznánk a posztra. Több képhez is tartozhatna ugyanaz a poszt azonosító.

**Menü megjelenítése a profil oldalakon:** A profil oldalakon megjelenhetne minden olyan tartalom, melyet az adott felhasználó hozott létre. Egy menüből elérhetnénk az adott felhasználó által létrehozott csoportokat, posztokat és hozzászólásokat.

# Összegzés

Végezetül összegezzük, hogyan is készült el az alkalmazásom. Az ötletelés után megírtam a fejlesztői dokumentációt, kezdve a követelmény és funkcionális specifikációval. Majd létrejött a rendszerterv, ahol megterveztem az adatbázist és kiválogattam azokat a technológiákat, amelyeket használtam a fejlesztés során. Ezek után következett a keretrendszer telepítése, majd a programozás elkezdése. Létrehoztam a tervezett funkciókat a követelménylista alapján. A fejlesztés közben gyakran teszteltem az alkalmazást, hogy biztos legyek abban, hogy mindent jól csináltam. Miután elkészült a projekt, a dokumentáció alapján még egyszer végigjártam a tesztlépéseket, és következhetek az utolsó simítások.

Az első fejezetben ismertettem az adatbázis alapú webes alkalmazások főbb tulajdonságait. Olvashattunk a YouTube és a Facebook funkcióiról és az internet veszélyeiről is. A második fejezetben láthattuk a dokumentációt, amely alapján elkészült az alkalmazás. Betekintést nyertünk az adatbázisba és láthattuk az alkalmazásom képernyőterveit és a megvalósítandó követelmények listáját is. A harmadik fejezetben szó volt az alkalmazás telepítéséről. Lépésről-lépésre végighaladtunk a XAMPP-os és Docker-es telepítésen is. A negyedik fejezetben részleteztem a felhasznált technológiákat, kezdve a MySQL-től, a Laravel-en át egészen a Docker-ig. Ezt követően összegeztem az alkalmazásom főbb funkcióit és azok működését az ötödik fejezetben. Olvashattunk az automatikus törlésről és jogosultság ellenőrzésről is. Végül felsoroltam néhány továbbfejlesztési lehetőséget.

Összességében ez a projekt sokat segített nekem abban, hogy jobban megismerkedjek a weboldalak fejlesztésével. Sokat tanulhattam a keretrendszerek, illetve az adatbázisok használatáról. Tapasztalatot szereztem a frontend fejlesztésben és dizájnolásban is a bootstrap-nek köszönhetően. Bele láthattam egy projekt menedzselésébe a Trello-val, és megismerkedtem a Docker konténerizációval is.



# Köszönetnyilvánítás

Szeretném köszönetet mondani az egyetemi tanárainknak, elsősorban Dr. Tajti Tibor Gábor tanár úrnak, a sok hasznos tanácsért, a támogatásért és oktatásért, amely hozzájárult a szakdolgozatom létrejöttéhez és szakmai fejlődésemhez. Továbbá köszönetet mondok hallgatótársaimnak is, akik gyakran ötletekkel és útmutatással láttak el egyetemi éveim alatt.

# Irodalomjegyzék

- [1] DR. KUSPER GÁBOR ÉS DR. RADVÁNYI TIBOR: *Jegyzet a projekt labor című tárgyhoz* , Eszterházy Károly Katolikus Egyetem, Eger, 2012.
- [2] DR. RADVÁNYI TIBOR: *Adatbázisrendszerek* , 2014.
- [3] THE WORLD BANK: *Az internethasználók százalékos aránya*
- [4] WIKIPEDIA: *Webalkalmazás*
- [5] WIKIPEDIA: *Adatbázis*
- [6] WIKIPEDIA: *Közösségi média*
- [7] WIKIPEDIA: *Facebook*
- [8] WIKIPEDIA: *YouTube*
- [9] WIKIPEDIA: *Twitter*
- [10] WIKIPEDIA: *XAMPP*
- [11] APACHE FRIENDS: *A XAMPP hivatalos dokumentációja*
- [12] WIKIPEDIA: *MySQL*
- [13] MYSQL: *A MySQL hivatalos dokumentációja*
- [14] WIKIPEDIA: *PhpMyAdmin*
- [15] PHPMYADMIN: *A PhpMyAdmin hivatalos dokumentációja*
- [16] WIKIPEDIA: *Laravel*
- [17] LARAVEL: *A Laravel hivatalos dokumentációja*
- [18] WIKIPEDIA: *CSS*
- [19] W3SHOOLS: *A CSS hivatalos dokumentációja*
- [20] COMPOSER: *A Composer hivatalos dokumentációja*

- [21] WIKIPEDIA: *Bootstrap (front-end framework)*
- [22] BOOTSTRAP: *A Bootstrap hivatalos dokumentációja*
- [23] ATlassian DOCUMENTATION: *A SourceTree hivatalos dokumentációja*
- [24] WIKIPEDIA: *Trello*
- [25] TRELLO: *A Trello hivatalos dokumentációja*
- [26] WIKIPEDIA: *Kanban (szoftverfejlesztés)*
- [27] DOCKER DOCS: *A Docker hivatalos dokumentációja*
- [28] WIKIPEDIA: *Docker (szoftver)*
- [29] DOCKER HUB: *Docker Hub*
- [30] KOCZKA FERENC: *Operációs Rendszerek, 16. Konténerek*
- [31] YOUTUBE - EMAD ZAAMOUT: *Laravel Docker Course / Complete Laravel Dockerization*
- [32] CODESOURCE: *Complete Laravel 10 Image upload Tutorial with an example*
- [33] VISUAL PARADIGM: *Use case diagram software*
- [34] DBDIAGRAM.IO: *Database Relationship Diagrams Design Tool*
- [35] TABLES GENERATOR: *HTML & LaTeX table generator*
- [36] PIXLR X: *Picture Editor*
- [37] FELHASZNÁLT BETŰTÍPUS - GOOGLE FONT: *Nunito*
- [38] FELHASZNÁLT BOOTSTRAP STÍLUS: *bootstrap min 5.3.0 stylesheet*
- [39] FELHASZNÁLT BOOTSTRAP STÍLUS: *bootstrap icons 1.3.0 stylesheet*
- [40] DR. TÓMÁCS TIBOR: *LaTeX – Számítógépes szöveg- és kiadványszerkesztés*

## NYILATKOZAT

Alulírott ..... Lázár Benedek ..... büntetőjogi felelősségem tudatában kijelentem, hogy az általam benyújtott, ..... Adatbázis alapú web alkalmazás fejlesztése ..... című szakdolgozat (diplomamunka) önálló szellemi termékem. Amennyiben mások munkáját felhasználtam, azokra megfelelően hivatkozom, beleértve a nyomtatott és az internetes forrásokat is.

Tudomásul veszem, hogy a szakdolgozat elektronikus példánya a véde után az Eszterházy Károly Katolikus Egyetem könyvtárába kerül elhelyezésre, ahol a könyvtár olvasói hozzájuthatnak.

Kelt: ..... Eger ..... 2023 év November hó 14 nap.

..... Lázár Benedek .....

aláírás