

# **ZÁRÓDOLGOZAT**

**Egersdorfer Dominik**

**5/13 SZOFT**

**BAJAI SZC TÜRRI ISTVÁN TECHNIKUM**

**SZOFTVERFEJLESZTŐ**

# **ZÁRÓDOLGOZAT**

**TIER LIST**

**Egersdorfer Dominik**

**2024**

## NYILATKOZAT A ZÁRÓDOLGOZATRÓL

Alulírott ..... (név) tanuló

### **kijelentem, hogy**

..... című  
záródolgozatomat (nyomtatott és elektronikus formában) a Bajai SZC Türr  
István Technikumának oktatói és tanulói:

- felhasználhatják (pl. hivatkozás alapjául, olvasótermi használatra) későbbi munkájukhoz a szerzői jogok tiszteletben tartása mellett.
- nem használhatják fel (titoktartási nyilatkozat csatolása mellett).

Ugyanakkor kijelentem, hogy a záródolgozat *saját munkám eredménye*.

Baja, 2024. ....

.....  
aláírás

# Tartalomjegyzék

1.	Bevezetés .....	1
1.1.	Témaválasztás .....	1
1.2.	Az oldalról .....	1
1.3.	Github .....	1
2.	Felhasználói dokumentáció .....	2
2.1.	Főoldal .....	2
2.1.1.	Regisztráció .....	3
2.1.2.	Bejelentkezés .....	4
2.2.	Listák .....	4
2.2.1.	Új lista.....	5
2.2.2.	Jogosultságok.....	6
2.3.	Lista készítő .....	6
2.3.1.	Karakterek.....	8
2.3.2.	Karakter hozzáadása .....	8
2.3.3.	Karakter módosítás .....	9
2.3.4.	Kategóriák.....	10
2.3.5.	Kategória hozzáadása .....	10
2.3.6.	Kategória módosítás .....	11
2.4.	Profil .....	11
2.4.1.	Statisztikák.....	12
2.4.2.	Adatok módosítása.....	12
2.5.	E-mail hitelesítés.....	14
2.6.	Navigáció .....	16
3.	Fejlesztői dokumentáció .....	17
3.1.	Telepítés.....	17
3.1.1.	Npm .....	17

3.1.2.	Bun.....	17
3.1.3.	Telepítési parancsok .....	18
3.1.4.	Docker.....	18
3.2.	Fejlesztés során használt szoftverek .....	19
3.2.1.	Visual Studio Code (^1.88.0).....	19
3.2.2.	XAMPP (^3.3.0) .....	20
3.2.3.	Git & Github .....	20
3.2.4.	Atlassian Jira.....	20
3.2.5.	Figma .....	21
3.2.6.	Firefox (^124.0.2) .....	21
3.3.	Frontend .....	21
3.3.1.	Függőségek .....	21
3.3.2.	Vite.....	22
3.3.3.	React .....	22
3.3.4.	Tailwind.....	22
3.3.5.	dnd kit .....	23
3.3.6.	FontAwesome .....	23
3.3.7.	Jikan API.....	23
3.4.	Backend .....	23
3.4.1.	Függőségek .....	23
3.4.2.	Környezeti változók (.env) .....	24
3.4.3.	Adatbázis .....	25
3.4.4.	Útvonalak.....	27
3.5.	Fájlrendszer.....	28
3.5.1.	Backend mappák.....	29
3.5.2.	Frontend mappák .....	29
3.6.	Forráskód .....	30

3.6.1.	backend/libs/database.js.....	30
3.6.2.	backend/libs/logger.js .....	31
3.6.3.	backend/socket.js .....	31
3.6.4.	backend/libs/resent.js .....	32
3.6.5.	backend/libs/errors.js .....	33
3.7.	Továbbfejlesztési lehetőségek .....	35
3.7.1.	Főoldal Demo .....	35
3.7.2.	Beállítások .....	35
3.7.3.	Téma mentes Tier List .....	35
3.7.4.	Barátlista, csevegő .....	35
3.7.5.	Elfelejtettem a jelszavam .....	36
3.7.6.	Karakter keresés anime alapján .....	36
3.7.7.	Törlés megerősítés .....	36
4.	Tesztelés .....	37
5.	Összegzés .....	39
6.	Ábrajegyzék.....	40
7.	Felhasznált források.....	42

# 1. Bevezetés

A szakdolgozatom egy anime témára specializálódott Tier List weboldal, ahol karaktereket lehet rangsorolni.

A Tier List röviden egy olyan lista, amiben rangsorolni tudunk valamilyen elemeket (például szereplőket), valamilyen szempontok alapján (például erősség). A kiválasztott elemeket különböző kategóriákba, szintekbe, azaz „tier”-ekbe sorolhatjuk és ezekben általában felül szerepelnek a „jobb”, alul pedig a „rosszabb” elemek. A témát és az abban szereplő elemeket, szinteket mi választjuk, és azt is, hogy mi alapján szeretnénk elrendezni. Az én oldalamon a téma adott, az elemek pedig az anime karakterek, de a besorolási szempontok a felhasználó kreativitására vannak bízva.

## 1.1. Témaválasztás

A témaválasztásom azért erre esett, mivel az általam ismert oldalakon sok bővítési lehetőséget láttam és a legtöbb oldalon egy képen kívül semmilyen információt nem kaphattunk az adott elemekről. Kevés anime témára specializálódott oldalt találtam, ezért egy jól átgondolt, hasznos funkciókkal kibővített felület volt az elképzelés, ahol a felhasználó szabadon létrehozhatja saját listáját.

## 1.2. Az oldalról

Az egyik leghasznosabb funkció az anime Tier List-ek elkészítéséhez, egy karakter kereső, ami a Jikan API segítségével egyszerűen megvalósíthattam. A cél az, hogy ha valaki tudja kit szeretne felhasználni listájában, akkor egyszerűen csak rákeres és már helyezheti is kedve szerint. Bármikor módosíthatja a már bent lévő karaktereket, vagy ha már valamiért nincs rá szüksége, akkor törölheti azokat. A kategóriákat is szabadon testre szabhatjuk, attól függően mi lesz a lista témája.

A közös listák készítése se okoz problémát, mivel minden – a lista oldalán tartózkodó – felhasználó látja az adott pillanatban történő módosításokat, tehát ha egy karakter áthelyezésre kerül, vagy valamilyen adata szerkesztve lett, akkor arról mindenki tudni fog. Az oldalon jogosultságokat lehet társítani a felhasználóknak a listáikhoz, hogy korlátozzuk vagy bővítsük az elvégezhető műveletek számát.

## 1.3. Github

A projekt Github oldala: <https://github.com/keitajs/tier-list>

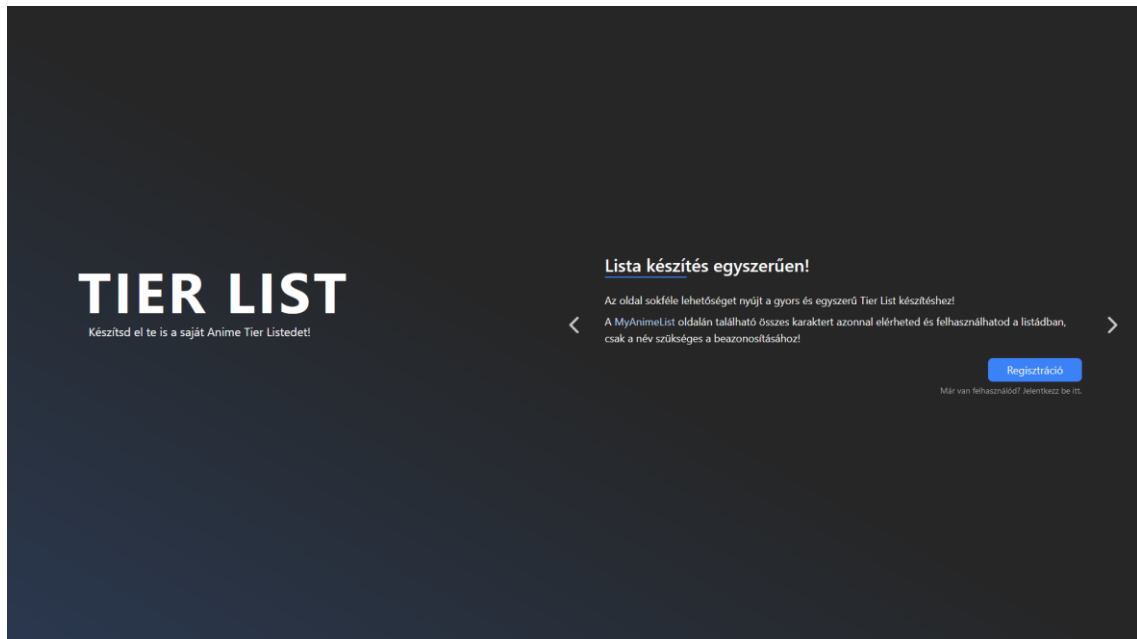
## 2. Felhasználói dokumentáció

Az oldal Anime karakterekből álló Tier List-ek létrehozásának ad egy gyors és egyszerű felhasználói felületet, ahol szabadon elkészítheti a saját listáit.

A Tier List egy olyan listát jelent, amelyben adott témák karaktereit, szereplőit, elemeit rangsorolják valamilyen szempontok, például hatékonyság, hasznosság vagy erő alapján. Legtöbb esetben – mint ezen az oldalon is – szintekre („tier”-ekbe) sorolják az elemeket, ebben a legjobbak a legfelső, míg a kevésbé „jó” az alsóbb kategóriákban helyezkednek el. A Tier List segít az embereknek eligazodni az adott témában, jelen esetben akár egy anime vagy animék közötti világokban, hogy ezáltal több információt szerezhessenek egy-egy karakterről. Ezen a területen több különböző gondolat is kialakulhat, így ezek kifejezhetnek véleményt is.

### 2.1. Főoldal

A főoldalunkon találhatunk néhány információt az oldalról. Elérhetjük a regisztrációt és a bejelentkezést is.



1. ábra: Főoldal

Az oldal használatához kötelező a regisztráció, e nélkül semmilyen funkciót nem érhetünk el.



### 2.1.1. Regisztráció

Regisztráció során három adatot kell megadnunk:

- **Felhasználónév:** Bármilyen szöveg, ami a felhasználó neveként fog szolgálni az oldalon. Kétféle hibaüzenetet kaphatunk:

Üres mezőnél: „Adj meg egy felhasználónevet!”

Foglalt név esetén: „A megadott felhasználónév már foglalt!”

- **E-mail:** Egy létező e-mail cím, ami segítségével könnyedén beazonosítható, hogy valóban hozzánk tartozik a felhasználó. Háromféle hibaüzenetet kaphatunk:

Üres mezőnél: „Adj meg egy emailt!”

Hibás formátumú e-mail címnél: „Az email cím nem megfelelő!”

Foglalt e-mail címnél: „A megadott email cím már foglalt!”

- **Jelszó:** Legalább 8 karakter hosszú szöveg, ajánlott számokat és speciális karaktereket is használni, de ezt a rendszer nem követeli meg. Kétféle hibaüzenetet kaphatunk:

Üres mezőnél: „Adj meg egy jelszót!”

Rövid jelszónál: „A jelszavadnak legalább 8 karakter hosszúnak kell lennie!”

- **Jelszó újra:** Ez az előző mezőben megadott jelszó megismétlése, csökkentve az elgépelés esélyét. Kétféle hibaüzenetet kaphatunk:

Üres mezőnél: „Add meg újra a jelszót!”

Ha a két jelszó nem ugyanaz: „A két jelszó nem egyezik!”

The image shows a mobile app registration screen with a dark background. The title 'REGISZTRÁCIÓ' is at the top in white. Below it are four input fields, each with a red 'X' icon on the right and a red error message below it. The fields are: 'Felhasználónév' (username), 'Email', 'Jelszó' (password), and 'Jelszó újra' (password repeat). The 'Jelszó' and 'Jelszó újra' fields have an eye icon to toggle visibility. At the bottom is a blue button labeled 'Regisztráció'.

2. ábra: Regisztráció

### 2.1.2. Bejelentkezés

Bejelentkezés során a már regisztrált felhasználónk nevét és jelszavát szükséges megadnunk. Az e-mailünkben kapott hitelesítést csak azután tudjuk elvégezni, miután belépett a weboldal felületére.

Sikeres belépés után az oldal azonnal átirányít a listáinkhoz.

#### - Felhasználónév lehetséges hibaüzenetei:

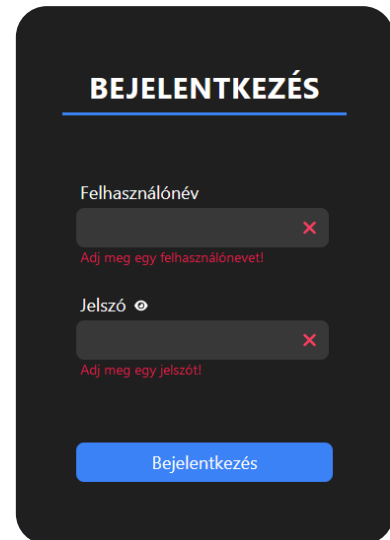
Üres mezőnél: „Adj meg egy felhasználónevet!”

Hibás felhasználónévnél: „A felhasználó nem található!”

#### - Jelszó lehetséges hibaüzenetei:

Üres mezőnél: „Adj meg egy jelszót!”

Helytelen jelszónál: „Hibás jelszó!”



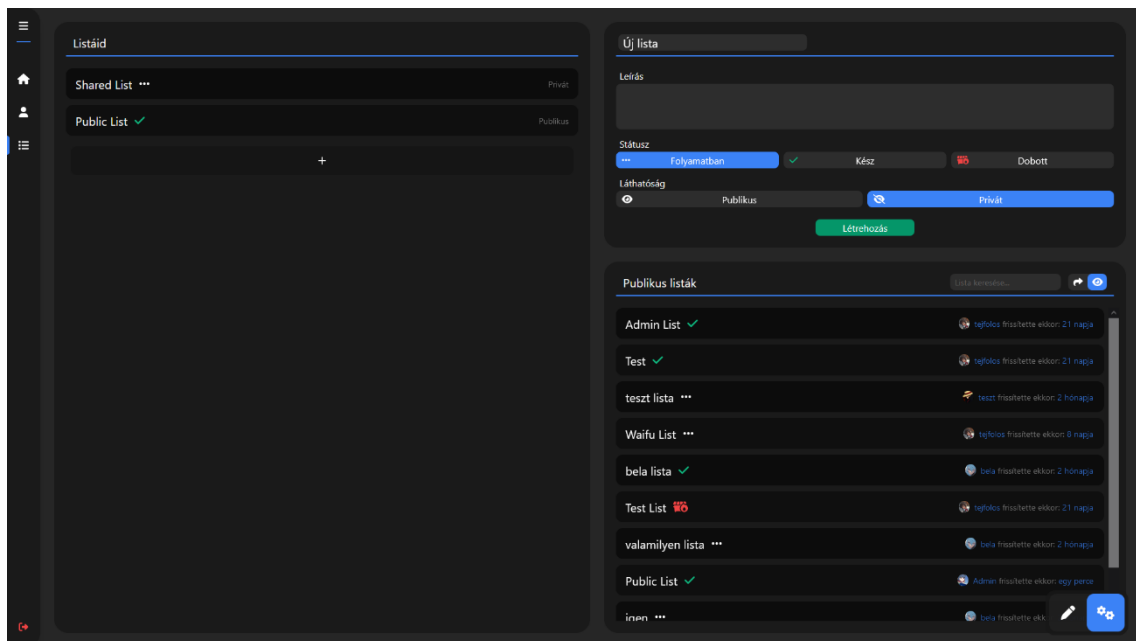
3. ábra: Bejelentkezés

## 2.2. Listák

Ez az oldal ad felületet, hogy a felhasználó a listákat kezelje, létrehozzon újakat, módosítsa azokat vagy akár törölhessen belőlük. Minden listához külön-külön kapcsolódnak jogosultságok, amiknél felhasználók kaphatnak engedélyt a kiválasztott lista elérésére.

Amennyiben már megnyitott valaki egy listát, a jobb alsó sarokban egy navigáció jelenik meg, ahol váltani lehet a jelenlegi és a lista készítő oldal között.

A bal oldali részen megtalálhatóak a már létrehozott listák, ezekre kattintva módosíthatja, megnyithatja vagy törölheti azokat.



4. ábra: Listák

Alapértelmezetten az új lista létrehozás és a lista kereső jelenik meg.

### 2.2.1. Új lista

Egy listának az alábbi adatokat lehet megadni:

- **Név:** A lista megjelenő neve, ez alapján lehet beazonosítani. Ez egy kötelező mező.
- **Leírás:** Rövid információ. Nem kötelező mező.
- **Státusz:** Ezzel a funkcióval megjelölheti a jelenlegi állapotot, de hatása nincs a listára.
- **Láthatóság:** A publikus listákra bárki rákereshet és megnézheti, a privátot kizárólag a jogosultsággal rendelkező felhasználók tekinthetik meg.

5. ábra: Új lista

Az új lista létrehozás alatt találhatunk egy lista böngészőt, amiknél láthatjuk az összes velünk megosztott és publikus listákat. Látható, hogy ki és mikor módosította utoljára az adott listát, valamint, hogy a tulajdonosa milyen státuszba sorolja azt.

Egy listára kattintva megtekinthetjük annak összes adatát és a már hozzáadott felhasználók jogosultságait.

### 2.2.2. Jogosultságok

A jogosultságoknál az alábbi adatokat lehet megadni:

- **Felhasználó:** Ide a pontos felhasználónevet kell megadni.

#### Lehetséges hibaüzenetek:

Üres mező: „Írj be egy felhasználónevet!”

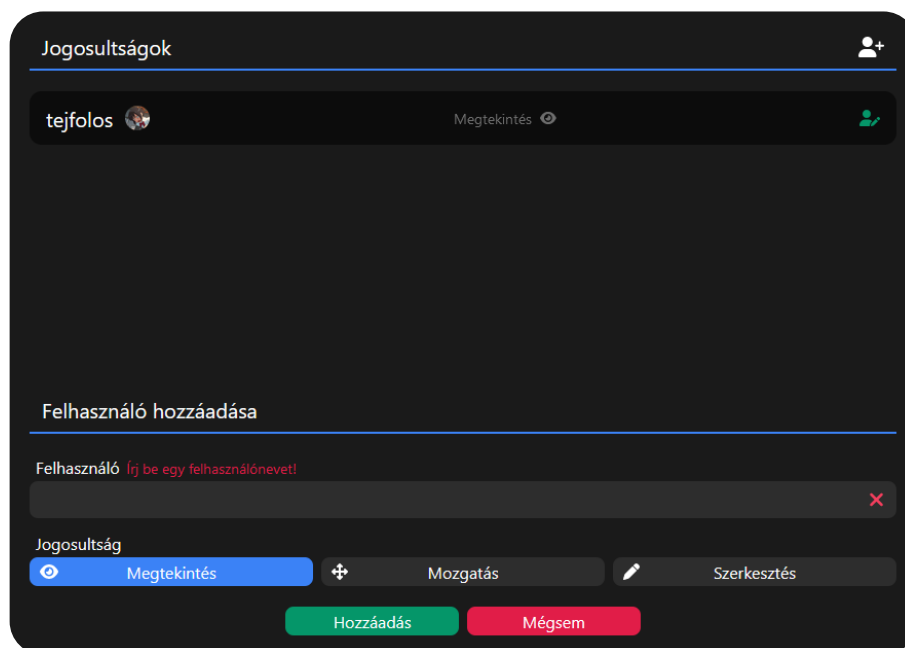
Helytelen felhasználónév: „Nem található felhasználó!”

- **Jogosultság:** A felhasználóknak három különböző jogosultságot lehet adni:

**Megtekintés:** Meg tudják tekinteni. Privát listánál van jelentősége.

**Mozgatás:** Karakterek és kategóriák mozgatása.

**Szerkesztés:** Lista tulajdonságain kívül minden elérhető számára.



6. ábra: Jogosultságok

Új jogosultságot a jobb oldalon fent található fehér ikonra kattintva tudunk hozzáadni, míg módosítani a már meglévő felhasználókra kattintva lehetséges. Mindkettőnél ugyanott – alul – megjelenik az ehhez szükséges felület.

### 2.3. Lista készítő



Minden lista módosításához erre az oldalra kell eljutnia a felhasználónak, amit a listák kezeléséből, az oldalt található navigációs menüből vagy a profil oldalról nyithat meg.

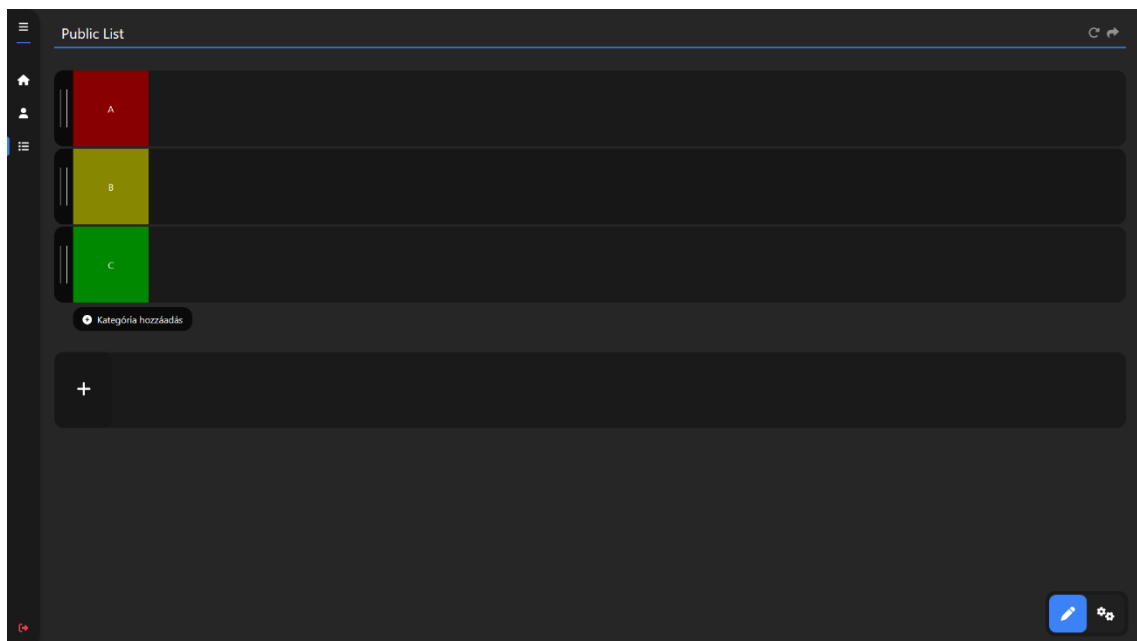
Itt mindenki a jogosultságaitól függően láthat gombokat és hajthat végre műveleteket. A „Kategória hozzáadás” és a karakter hozzáadást jelző gomb az alsó sávban nem látható, ha Szerkesztésnél gyengébb jogosultsággal rendelkezik.

A legalsó üres szint a nem használt karaktereknek ad helyet, ide tudunk felvenni újakat is. A felette található színes sávok már az általunk létrehozott kategóriák, amiket tudunk módosítani. Lista létrehozás után négy darab előre beállított szint kerül létrehozásra (A, B, C, D).

Az oldalon jelenlévő felhasználók azonnal láthatják egymás műveleteit, áthelyezett karaktereket, kategóriákat, módosított adatokat, törléseket. Ez nagyban elősegítheti a többszemélyes listakészítés végrehajtását vagy akár egy lista elkészülésének megfigyelését. A jelenlévő felhasználókat a lista címe alatt fogjuk látni.

Jobb felső sarokban két ikon található:

-  **Oldal újra töltése:** Ahhoz, hogy ne kelljen az egész oldalt újra betölteni, ezzel a gombbal kizárólag a megnyitott tier listet lehet frissíteni.
-  **Megosztás:** Erre kattintva egy linket fog kimásolni, amit mások megnyitva elérhetnek a listát. Csak akkor fog működni a megosztott link, ha a lista publikus és/vagy jogosultsága van a megtekintéshez az elküldött felhasználónak.



7. ábra: Lista készítés

### 2.3.1. Karakterek

A karakterek a szinteken megjelenő elemek, alapértelmezetten csak egy képként jelennek meg.

Két funkció elérhető:

- **Információ kinyitás:** Jobb kattintással kinyitható az információs rész, ahol a megadott adatok láthatóak, valamint itt érhető el a módosítása is. Bal oldalon alul található a karakter és anime URL ikonok, amire kattintva külön oldalon fogja megnyitni a megadott linket. Bezárni az „X”-el vagy újbóli jobb kattintással tudjuk.
- **Mozgatás:** Hosszan lenyomva a bal egér gombot tudjuk áthúzni a karaktert a kiválasztott helyre, akár szinten belül, akár másik kategóriába. Amint elengedjük, már az új helyére is fog kerülni.



8. ábra: Karakter

### 2.3.2. Karakter hozzáadása

Az üres kategóriában található „+” gombra kattintva tudjuk megnyitni az új karakter hozzáadása ablakot. Jobb felül kereshetünk karakterekre név alapján, amikből választva kitölti a mezőket. Karakter hozzáadáshoz öt adatot tudunk megadni:

- **Név:** Karakter neve, kötelező mező. Ha nem írunk semmit a mezőbe, akkor „Üres mező!” hibaüzenetet kapunk.
- **URL:** A karakterről egy weboldal URL, ahol leírás található, például MyAnimeList oldalról. Nem kötelező adat. Amennyiben adunk meg értéket, akkor hibás URL formátumnál „Nem megfelelő URL formátum!” hibaüzenetet kapunk.
- **Kép:** Karakter képe. Belinkelhető és feltölthető a kép. Kötelező mező, ha nem adjuk meg akkor „Üres mező!”, nem megfelelő belinkelt kép esetén „Az URL-en nincs elérhető kép!” hibaüzeneteket kaphatunk.

A következők a karakterhez tartozó anime adatai, amik nem kötelező mezők.

- **Cím:** Anime címe.

- **URL:** Az animéről egy weboldal URL-je, ahol leírás található, például a MyAnime-List oldalról. Hibás link megadásakor „Nem megfelelő URL formátum!” hibaüzenetet kapunk.

9. ábra: Új karakter

### 2.3.3. Karakter módosítás

A karakter információknál a jobb alsó sarokban tudjuk elérni a szerkesztést. Ekkor az adatok módosíthatóvá válnak. A karakter és a hozzá tartozó anime külön módosítható, ezek között jobb oldalon alul lehet váltogatni. Menteni és törölni a bal alsó sarokban található ikonokra kattintva tudunk. A karakterünk adatainak módosítása és törlése nem visszavonható. Ha mégsem szeretnénk módosítani, akkor a jobb felső sarokban található „X”-re kattintva visszatérhetünk az információk megtekintéséhez. Az itt lévő mezők megegyeznek a hozzáadásnál lévőkkel, de itt nem jelenik meg hibaüzenet. A hibás értékeknél egy piros „X” jelenik meg.

10. ábra: Karakter módosítás

### 2.3.4. Kategóriák

A kategóriák a listákban található szintek, amikbe kerülnek a karakterek. Három fő részből áll:

- **Mozgatás:** A bal oldali két csíkos sáv segítségével áthúzható a kategória. Legalább mozgatósi jogosultság kell hozzá, hogy használható legyen.
- **Cím:** Egy négyzet, aminek háttérszíne a kategória színe, rajta pedig a címe. A szöveg alkalmazkodik a háttérhez, így az mindig látható és olvasható.
- **Karakterek:** Az összes behúzott karakter itt lesz.



11. ábra: Kategória

### 2.3.5. Kategória hozzáadása

A kategóriák alatt található „Kategória hozzáadás” gombra kattintva megjelenik a létrehozó ablak, ahol két adatot tudunk megadni:

- **Cím:** Kategória címe, ez fog megjelenni a létrejött szinten. Kötelező mező.

#### Lehetséges hibaüzenetek:

Nem megadott érték esetén: „Üres mező!”

- **Szín:** Kategória színe, tetszés szerint állítható. Szöveg színe alkalmazkodik a háttérhez.

Az új kategória az összes többi alatt fog megjelenni, ahonnan bármikor áthelyezhetjük. Az előnézetben látni fogjuk a beírt címet is.

12. ábra: Új kategória



### 2.3.6. Kategória módosítás

A kategóriára ráhúzva az egerünket megjelenik a szerkesztés ikon (✎) a kategória címénél. Erre kattintva tudjuk módosítani a szintünk címét és színét. Hibaüzenetek a karakter módosításhoz hasonlóan itt se jelennek meg, egyedüli hiba lehetőség az „Üres mező!”. Menteni a bal alsó sarokban lévő zöld mentés ikonnal (💾), törölni a mellette található piros szemetesre (🗑️) kattintva tudunk. Az „X”-re kattintva a módosítások nem hajtnak végre, visszaáll az eredeti állapotába.

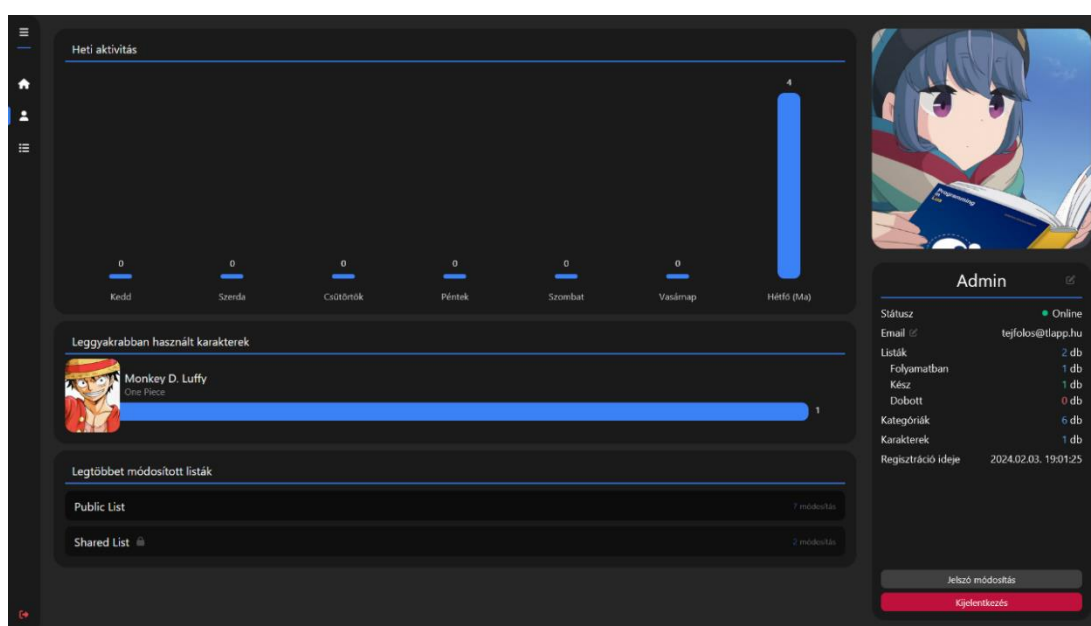
A törlés esetén a benne található összes karakter is elvész, ezért érdemes előtte áthelyezni azokat, amikre később még szükségünk lehet, mivel ez a művelet nem visszavonható.



13. ábra: Kategória módosítás

## 2.4. Profil

A profilunkon megtalálhatunk mindenféle információt, statisztikát, valamint itt módosíthatjuk a felhasználónevünket, e-mail címünket, jelszavunkat és profilképünket. Ezt az oldalt a bal oldali navigációs sávból érhetjük el. Amikor más profiljára kerülünk, akkor az e-mail címe rejtve van, a státusznál megjeleníti, hogy elérhető (Online) vagy nem (Offline). Az oldalon más felhasználónak az adatait nem módosíthatjuk, a hozzá tartozó szerkesztési opciók sem jelennek meg, valamint a privát listái sem érhetők el.



14. ábra: Profil

### 2.4.1. Statisztikák

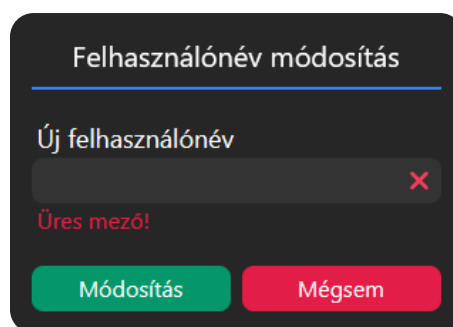
Három féle statisztikai adat jelenik meg az oldalon:

- **Heti aktivitás:** Látható rajta az elmúlt egy heti aktivitásunk. Aktivitásnak számít, amikor létrehozunk, mozdítunk, módosítunk vagy törölünk valamit a listában. Például, ha hozzáadunk 10 karaktert és 3 kategóriát, amik közül 5 karaktert áthelyezünk, akkor az 18 aktivitásnak számít arra a napra. Az egeret egy aktivitási oszlopra ráhúzva kiírja, hogy az adott napon melyik listában mennyi módosítás történt.
- **Leggyakrabban használt karakterek:** Azon karakterek listája, amelyeket a legtöbbször használtunk. Ez a létrehozott és létező karakterek száma, amit töröltünk az nem kerül bele a számításba. A képre kattintva elérünk egy képnézegetőt, amivel közelebbről is megvizsgálhatjuk a karakterünk kinézetét. A top 10 leghasználtabb fog itt megjelenni.
- **Legtöbbet módosított listák:** A listához jogosult felhasználók által elvégzett összes aktivitás alapján számolt legtöbbet módosított top 10 lista. Ezekre rákattintva – ha a saját profilunkon vagyunk vagy nem privát a lista – megnyithatjuk.

### 2.4.2. Adatok módosítása

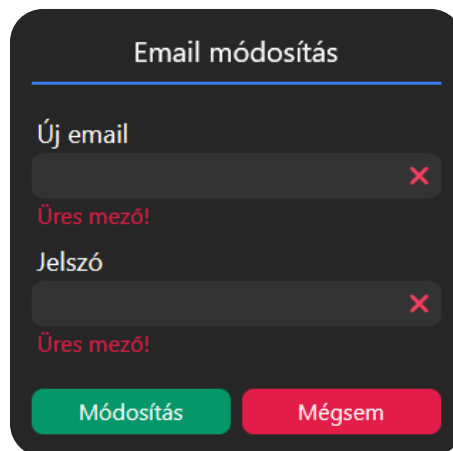
Az adatok módosítása felugró ablakokban történik, minden mező megadása kötelező. Amennyiben mégsem szeretnénk szerkeszteni az adatot, a „Mégsem” gombra kattintva bezárhatjuk.

- **Felhasználónév:** A felhasználónevünk mellett található szerkesztés ikonra (✎) kattintva érjük el. Már létező felhasználók neveit nem adhatjuk meg.



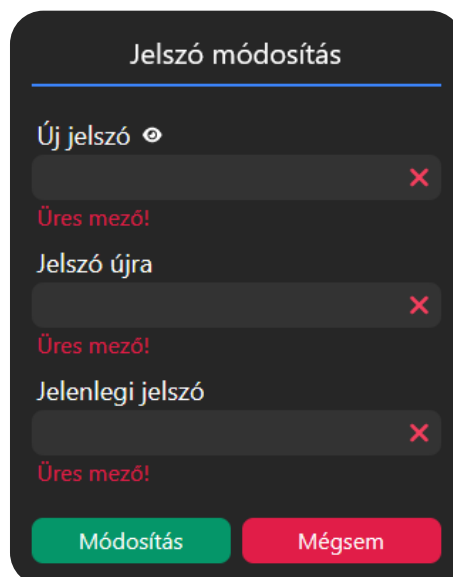
15. ábra: Felhasználónév módosítás

- **E-mail:** Az e-mail címünk melletti ikonnal szerkeszthető. Miután megváltoztattuk, hitelesíteni kell az új e-mailt, erre kapunk is egy üzenetet a megadott címre. Eredeti jelszó megadása kötelező.



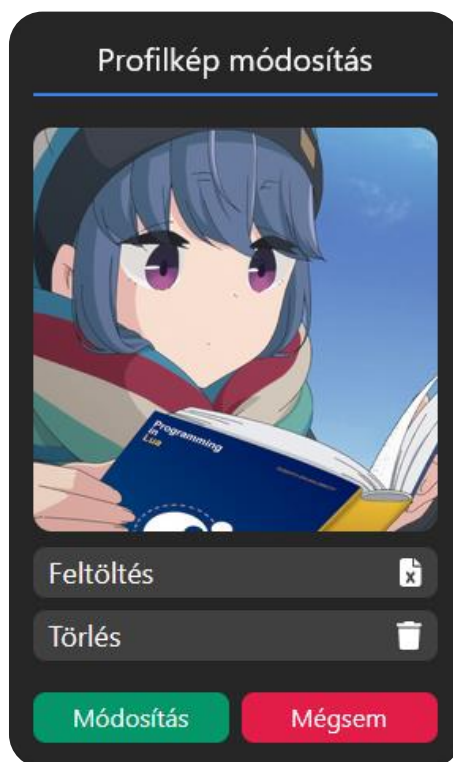
16. ábra: E-mail módosítás

- **Jelszó:** Az új jelszónak a már regisztrációkor is meglévő feltételeknek kell megfelelnie: legalább 8 karakter hosszúság kötelező, ajánlott speciális karaktereket és számokat beleírni. Eredeti jelszó megadása kötelező.



17. ábra: Jelszó módosítás

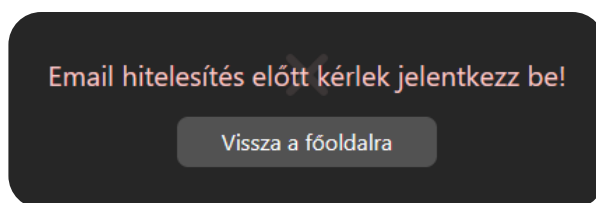
- **Profilkép:** A profilképünkre ráhúzva az egeret megjelenik a „Profilkép módosítása” felirat. Lehetőségünk van feltölteni egy képfájlt, amit ezt követően láthatunk előnézetben. A kiválasztott kép képaránya és felbontása nincs meghatározva, a négyzetet teljesen kitöltve jelenik meg. Amennyiben másik fájlt szeretnénk választani, kattintunk újra a „Feltöltés” gombra, ezzel törölve az előző fájlkunkat. A „Törlés” gomb azonnal eltávolítja a profilképünket.



18. ábra: Profilkép módosítás

## 2.5. E-mail hitelesítés

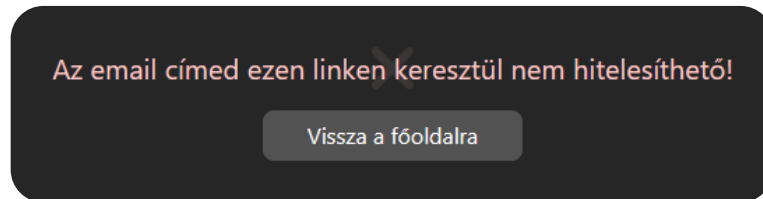
A megadott e-mail címre kapott üzenetben találunk egy hitelesítő linket, amit megnyitva – ha már bejelentkeztünk az oldalra – tudjuk hitelesíteni. Amennyiben nem vagyunk bejelentkezve, az oldal jelezni fogja számodra.



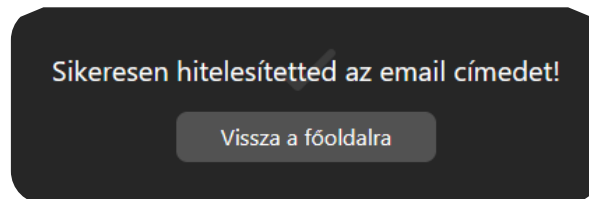
19. ábra: E-mail hitelesítés - Nincs bejelentkezve

E-mail módosítás után új hitelesítő linket kap, az előzővel már nem fog működni a hitelesítés. Megfelelő linket megnyitva, bejelentkezve sikeresen hitelesítheti az e-mail címét.

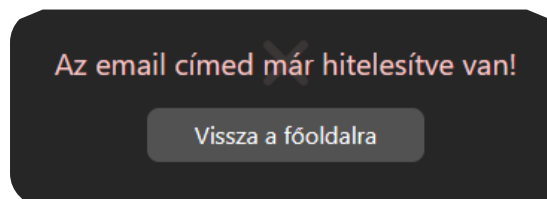
Ekkor már semmi egyéb teendő nincs ezzel kapcsolatban, amíg meg nem újítja azt. Amennyiben újra próbálná hitelesíteni az e-mail címét, azt az oldal jelezni fogja.



20. ábra: E-mail hitelesítés - Hibás link

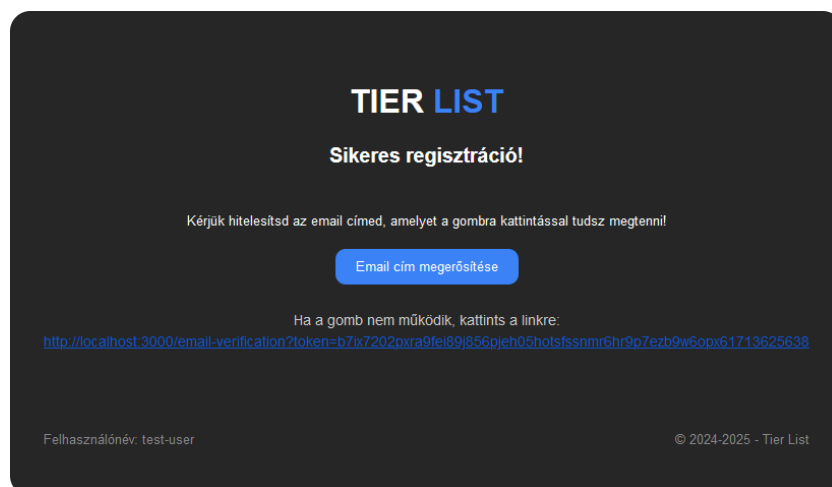


21. ábra: E-mail hitelesítés - Sikeres



22. ábra: E-mail hitelesítés - Már hitelesítve van

Az üzenet, amit kapunk a regisztráció és e-mail módosítás után is hasonlóan néz ki, különbség a címben lehet: „Sikeres regisztráció” a regisztrációnk után, „Email módosítás” az e-mail megváltoztatása után. Bal oldalon alul megtalálhatjuk a felhasználónevünket, ezzel megerősítve, hogy tényleg az általunk készült felhasználó hitelesítő üzenetét olvasuk. Az „Email cím megerősítése” gombra – vagy ha a gomb nem működik, akkor a linkre – kattintva kerülünk a feljebb már említett oldalra.



23. ábra: E-mail hitelesítő üzenet

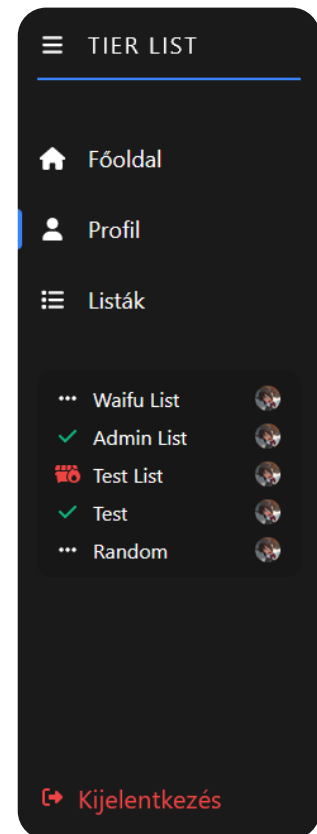
## 2.6. Navigáció

A navigációs sáv egy Sidebar, ami a bal oldalon található. Amikor ráhúzzuk az egeret, kinyitódik és láthatjuk a menüpontokat. Az öt legutóbb frissített lista is megjelenik, mellette a módosító profilképe. Alul lehetőségünk van kijelentkezni.

A listák oldalán, már megnyitott listánál megjelenik jobb oldalon alul egy két lehetőséget kínáló navigációs doboz. Itt tudunk a lista szerkesztése (bal oldali) és a lista adatok módosítása (jobb oldali) oldalak között mozogni. Ezeket a gombokat addig nem látjuk, amíg meg nem nyitottunk egy listát.



25. ábra: Lista navigáció



24. ábra: Sidebar

### 3. Fejlesztői dokumentáció

A weboldal frontend része Vite környezetben, React pluginnal készült. A backendet egy Node JS szerver biztosítja. A projekt JavaScript nyelvben íródott, az e-mail sablonok HTML-ben készültek el. A frontend és backend egyaránt Node alapú, ezért mindkettő Node csomagokat használ.

#### 3.1. Telepítés

Az oldal mindkét része Node alapú, ezért hasonló indítási lehetőségekkel rendelkeznek. A projekt több csomagkezelővel is telepíthető, az általam használtak és teszteltek az Npm és a Bun. A végleges verzióban én a Bun-t használom. Futtatás előtt ajánlott a dotenv fájlt bekonfigurálni, mivel anélkül problémák keletkezhetnek, ennek leírása megtalálható a Backend, Környezeti változók cím alatt.

##### 3.1.1. Npm<sup>1 2</sup>

Az NPM (Node Package Manager) egy olyan csomagkezelő rendszer, amelyet a Node JS-hez fejlesztettek ki. Segítségével a fejlesztők könnyen telepíthetnek, frissíthetnek és kezelhetnek különböző függőségeket. Az összes NPM által letölthető csomag megtalálható a <https://www.npmjs.com/> oldalon, legtöbb esetben használati leírással és magyarázattal. Az NPM a Node JS telepítésével kerül a számítógépünkre, amit a <https://nodejs.org/en> oldalról szerezhetünk be.



26. ábra: NPM

##### 3.1.2. Bun<sup>3</sup>

A Bun egy JavaScript és TypeScript projektekhez kifejlesztett futási környezet és eszközkészlet. Függőségek telepítése, frissítése és kezelése is megvalósítható. Gyors és optimalizált fejlesztői élményt nyújt, teljes Node JS kompatibilitással. Célja a szerveroldali teljesítmények növelése. Telepítése a <https://bun.sh/> oldalon található paranccsal történik.



27. ábra: Bun

---

<sup>1</sup> <https://nodejs.org/en>

<sup>2</sup> <https://www.npmjs.com>

<sup>3</sup> <https://bun.sh>

### 3.1.3. Telepítési parancsok

#### Npm

```
git clone https://github.com/keitajs/tier-list.git
```

#### Bun

#### Frontend

```
cd ./frontend
```

```
npm install
```

#### Production:

```
npm run build
```

```
npm run serve
```

#### Development:

```
npm run dev
```

```
cd ./frontend
```

```
bun install
```

#### Production:

```
bun run build
```

```
bun run serve
```

#### Development:

```
bun run dev
```

#### Backend

```
cd ./backend
```

```
npm install
```

#### Production:

```
npm run start
```

#### Development:

```
npm run dev
```

```
cd ./backend
```

```
bun install
```

#### Production:

```
bun run start
```

#### Development:

```
bun run dev
```

### 3.1.4. Docker<sup>4</sup>

A Docker olyan eszköz, amely lehetővé teszi a szoftver- és webalkalmazások konténerizált futtatását. A konténerek olyan egységként működő csomagok, amelyek tartalmazzák az alkalmazások futtatásához szükséges fájlokat, kódokat, függőségeket és konfigurációkat. Nem telepít teljes operációs rendszert, csak a futtatáshoz szükséges elemeket.



28. ábra:  
Docker

A Tier List projekt tartalmazza a Docker konténerizált futtatáshoz szükséges fájlokat:

- **compose.yml:** Konténerek konfigurációja.
- **frontend/frontend.Dockerfile:** Frontend összeállításához szükséges utasítások.
- **backend/backend.Dockerfile:** Backend összeállításához szükséges utasítások.
- **backend/.dockerignore:** Azokat a fájlokat tartalmazza, amik nem kerülhetnek másolásra.

---

<sup>4</sup> <https://www.docker.com/>



- **.env.docker:** Dockerben használt környezeti változók hasonlóan, mint a normál .env-ben.























A fájlokat módosítani nem szükséges, a *compose.yml* fájl könyvtárában – tehát a projekt főkönyvtárában – nyissunk egy *cmd*-t (parancssort), amiben használjuk az alábbi parancsot:

```
docker compose up -d
```

Amint futtattuk a parancsot, a docker létrehozza nekünk a szükséges konténereket.

- **tl-frontend:** Tier List frontend. 3000-es porton lesz elérhető.
- **tl-backend:** Tier List backend. 2000-es porton lesz elérhető.
- **tl-database:** A projekthez tartozó adatbázis. 3306-os porton lesz elérhető.

A konténer nevére kattintva – például *tl-frontend* – megnyithatjuk a logokat, ahol látjuk a szerver által kapott információkat. A portra kattintva megnyitja a hozzá tartozó oldalt (ennek a frontend esetén van értelme, mivel ott érjük el az oldalunkat). Jobb oldalt az Actions résznél tudjuk elindítani, leállítani és ha már nem kell, akkor törölni.

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	 tier-list		Running (3/3)		0.01%	1 minute ago	  
<input type="checkbox"/>	 <i>tl-frontend</i> 08534737127d 	<a href="#">tier-list-frontend</a>	Running	<a href="#">3000:3000</a> 	0%	1 minute ago	  
<input type="checkbox"/>	 <i>tl-backend</i> db991cb57718 	<a href="#">tier-list-backend</a>	Running	<a href="#">2000:2000</a> 	0%	1 minute ago	  
<input type="checkbox"/>	 <i>tl-database</i> 1a6b5612656f 	<a href="#">mariadb:latest</a>	Running	<a href="#">3306:3306</a> 	0.01%	1 minute ago	  

29. ábra: Elkészült Docker Container

## 3.2. Fejlesztés során használt szoftverek

Fejlesztés során sok szoftvert használhatunk, ami könnyíti a folyamatot. Tervezéshez, kódíráshoz, feladatok követésére és tesztelésre is találhatunk szoftvereket, weboldalakat.

### 3.2.1. Visual Studio Code (^1.88.0)<sup>5</sup>

A Visual Studio Code egy ingyenes, nyílt forráskódú kódszerkesztő és fejlesztői környezet. Számos programozási nyelvhez és keretrendszerhez nyújt kódolási felületet. Rendelkezik beépített funkciókkal, például verziókezeléssel és bővítmények támogatásával. A Visual Studio Code széles kínálatú bővítményeket nyújt számunkra, amelyeket a felhasználók hozzáadhatnak a kódszerkesztőhöz, ezzel könnyítve és hatékonyabbá téve a munkájukat.

<sup>5</sup> <https://code.visualstudio.com/>

Az alábbi bővítményeket használtam a fejlesztés során:

- **ES7 React/Redux/GraphQL/React-Native snippets:** Egy hasznos eszköz a fejlesztők számára, akik React, Redux vagy React Native alkalmazásokat készítenek. Előre definiált rövidítéseket tartalmaz, amik segítenek a gyakori feladatok végrehajtásában, ezzel megkönnyítve a fejlesztést (például komponensek létrehozása).
- **Console Ninja:** A fejlesztők számára segít a hatékonyabb böngésző konzol használatában a webfejlesztés során. Kiírja a futás közben keletkező konzol tartalmakat, így lehetővé teszi a gyorsabb hibakeresést és az értékek vizsgálatát.
- **Tailwind CSS IntelliSense:** A Tailwind CSS keretrendszerrel történő munka során segíti a fejlesztőket. Automatikusan kiegészíti a Tailwind osztályneveket és segíti azok gyors kiválasztását.
- **Thunder Client:** A backend kérések tesztelését teszi lehetővé a Visual Studio Code kódszerkesztőn belül. Könnyen használható felületet biztosít a különféle HTTP műveletek végrehajtására. Gyorsan és hatékonyan tesztelhető az API anélkül, hogy egyből az oldalon tesztelnénk.

### 3.2.2. XAMPP (^3.3.0)<sup>6</sup>

Az XAMPP egy ingyenes, nyílt forráskódú és platformfüggetlen webszerver-szoftvercsomag. Biztosít Apache HTTP szerver, MySQL adatbáziskezelőt és PHP futtatásához szükséges környezetet. A webfejlesztők helyileg fejleszthetik és tesztelhetik az webalkalmazásokat anélkül, hogy távoli szervert használnának. Ebben a projektben a MySQL adatbáziskezelőt használtam.

### 3.2.3. Git & Github<sup>7 8</sup>

Git egy verziókezelő rendszer, amely segít a fejlesztőknek nyomon követni és kezelni a kód változásait. A Github egy távoli kódtároló platform, amely lehetővé teszi a projektmenedzsmentet. A csapatmunkát és a projektek hatékony kezelését is könnyebbé teszi.

### 3.2.4. Atlassian Jira<sup>9</sup>

Az Atlassian Jira egy projektmenedzsment és feladatkezelő szoftver, amelyben lehetőségünk van a projektünkben lévő feladatok nyomon követésére és hibajegyek létrehozására. Összeköthetjük egy Github repositoryval, ezáltal a feladatokhoz tartozó módosításokat

---

<sup>6</sup> <https://www.apachefriends.org/hu/index.html>

<sup>7</sup> <https://git-scm.com/>

<sup>8</sup> <https://github.com/>

<sup>9</sup> <https://www.atlassian.com/software/jira>

figyelhetjük a Jira oldalán is. A weboldalam feladatainak követésére használtam, hogy ne maradjon ki egyetlen funkció sem.

### 3.2.5. Figma<sup>10</sup>

A Figma egy online tervező szolgáltatás, amely lehetővé teszi a felhasználók számára, hogy együtt dolgozzanak a projekt grafikai megtervezésén. Széleskörű tervezőeszközökkel rendelkezik, amelyekkel könnyedén lehet létrehozni és megosztani a felhasználói felületeket. Az oldal összes design terve ebben készült, amik megtalálhatóak a documents/tervezés mappában.

### 3.2.6. Firefox (^124.0.2)<sup>11</sup>

A Firefox egy böngésző, amit a weboldal tesztelésére használtam. A Firefox Developer Tools sokféle eszközkészletet kínál, amely lehetővé teszi a webfejlesztők számára a hibakeresést és elemzést. A böngésző rendelkezik még Responsive Design Mode funkcióval, amely segít a reszponzív weboldalak létrehozásában és tesztelésében.

## 3.3. Frontend

A frontend Vite környezetben készült React plugin felhasználásával. A Vite egy gyors fejlesztői környezet, amely sokféle projektet támogat. A React egy JavaScript könyvtár, ami lehetővé teszi, hogy az oldalon megjelenő tartalom újra töltés nélkül látható legyen.

### 3.3.1. Függőségek

A felsorolt csomagok elérhetőek az <https://www.npmjs.com/> oldalon.

Node verzió: v20.10.0

Npm verzió: v10.2.3

Bun verzió: v1.1.3

- **@dnd-kit/core (^6.1.0):** Drag and drop React könyvtár
- **@dnd-kit/modifiers (^7.0.0):** Dnd-kit mozgatót módosító eszközök
- **@dnd-kit/sortable (^8.0.0):** Sorbarendező dnd-kit interfészek
- **@dnd-kit/utilities (^3.2.2):** Segédeszközök a dnd-kithez
- **@fortawesome/free-solid-svg-icons (^6.5.2):** FontAwesome ikon csomag
- **@fortawesome/react-fontawesome (^0.2.0):** React FontAwesomeIcon, az ikonok használatához

---

<sup>10</sup> <https://www.figma.com/>

<sup>11</sup> <https://www.mozilla.org/hu/firefox/new/>

- **axios (^1.6.8):** Backend-frontend kapcsolat létrehozása
- **moment (^2.30.1):** Idő formátumok és műveletek
- **react (^18.2.0):** React könyvtár a keretrendszerhez
- **react-device-detect (^2.2.3):** Lekérhető, hogy a felhasználó milyen eszközről használja a weboldalt (telefon, számítógép)
- **react-dom (^18.2.0):** Reacthoz szükséges függőség
- **react-router-dom (^6.22.3):** React útvonalak kezelése
- **react-tooltip (^5.26.3):** Információ, eszköztipp jelenik meg egy-egy elemre ráhúzva az egeret
- **socket.io-client (^4.7.5):** Kétirányú eseményalapú kommunikációk kezelése kliens oldalon
- **use-debounce (^10.0.0):** Gépelés befejezésének megvárására használt csomag
- **postcss (^8.4.38):** Tailwindhez szükséges, JavaScript segítségével lehet módosítani a CSS-t
- **tailwindcss (^3.4.3):** Az oldal felületének designjáért felelős CSS keretrendszer
- **vite (^5.2.0):** Gyors fejlesztői környezet

### 3.3.2. Vite<sup>12</sup>

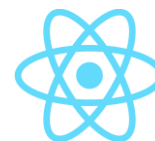
A Vite egy gyors és modern webfejlesztési környezetet biztosít a fejlesztők számára. Lehetőséget nyújt sokféle keretrendszer és könyvtár használatához, például Vue, Preact, és a projektem által használt React is. A Vite használható npm és bun csomagkezelővel egyaránt.



30. ábra: Vite

### 3.3.3. React<sup>13</sup>

A React egy komponens alapú JavaScript frontend könyvtár. Webes és natív interfészeket is létrehozhatunk vele. Interaktívvá tehetőek vele az oldalunk található elemek.



31. ábra: React

### 3.3.4. Tailwind<sup>14</sup>

A Tailwind egy CSS keretrendszer, ami lehetővé teszi, hogy egyszerűen designolhassuk az oldalunkat. Osztályneveket megadva alakíthatjuk a teljes kinézetét az oldalnak.



32. ábra: Tailwind

<sup>12</sup> <https://vitejs.dev/>

<sup>13</sup> <https://react.dev/>

<sup>14</sup> <https://tailwindcss.com/>

### 3.3.5. dnd kit<sup>15</sup>

Reacthoz készült könnyű, moduláris, nagy teljesítményű, hozzáférhető, bővíthető és könnyen szerkeszthető drag and drop eszközkészlet.



33. ábra:  
dnd kit

### 3.3.6. FontAwesome<sup>16</sup>

A FontAwesome egy internetes ikon könyvtár. Mindenféle kategóriában találhatunk itt ikonokat, bármilyen oldalra beépíthetőek és elérhető sokféle keretrendszerben is.



34. ábra:  
FontAwesome

### 3.3.7. Jikan API<sup>17</sup>

A Jikan API egy ingyenes, nyílt forráskódú PHP & REST API a MyAnimeList weboldalhoz, ami lehetővé teszi az anime adatok és információk egyszerű elérését és lekérdezését. A fejlesztők széleskörű funkcionalitást kaphatnak az anime adatok integrálásához alkalmazásaikba vagy weboldalaikba.



35. ábra:  
Jikan API

## 3.4. Backend

A backend egy Node JS szerveren fut. A legtöbb útvonal védve van a nem bejelentkezett felhasználók ellen, a főoldal az egyetlen regisztrációt nem igénylő oldal. A többfelhasználós használathoz tartozó socket szerver is itt fut.

### 3.4.1. Függőségek

A felsorolt csomagok elérhetőek az <https://www.npmjs.com/> oldalon.

Node verzió: v20.10.0

Npm verzió: v10.2.3

Bun verzió: v1.1.3

- **bcrypt (^5.1.1):** Jelszavak titkosítása
- **chalk (^5.3.0):** Console log színezés, jobban értelmezhető konzol elérése
- **cookie-parser (^1.4.6):** Sütik kezelése
- **cors (^2.8.5):** Cross Origin kérések engedélyezése
- **dotenv (^16.3.1):** Környezeti változók kezelése
- **express (^4.18.2):** HTTP szerver

---

<sup>15</sup> <https://dndkit.com/>

<sup>16</sup> <https://fontawesome.com/>

<sup>17</sup> <https://jikan.moe/>

- **jsonwebtoken (^9.0.2):** Session és bejelentkezés kezelés
- **moment (^2.30.1):** Idő formátumok és műveletek
- **multer (^1.4.5):** Fájlfeltöltések kezelése
- **mysql2 (^3.6.5):** Adatbázis-kezelés, sequelizehoz kapcsolódó csomag
- **resend (^3.2.0):** E-mail küldések kezelése
- **rndstr (^1.0.0):** Random string generátor
- **sequelize (^6.35.2):** Adatbázis kezelése
- **socket.io (^4.7.4):** Kétirányú eseményalapú kommunikációk kezelése
- **nodemon (^3.0.2):** Fejlesztői futtatáshoz tartozó fejlesztői függőség

### 3.4.2. Környezeti változók (.env)

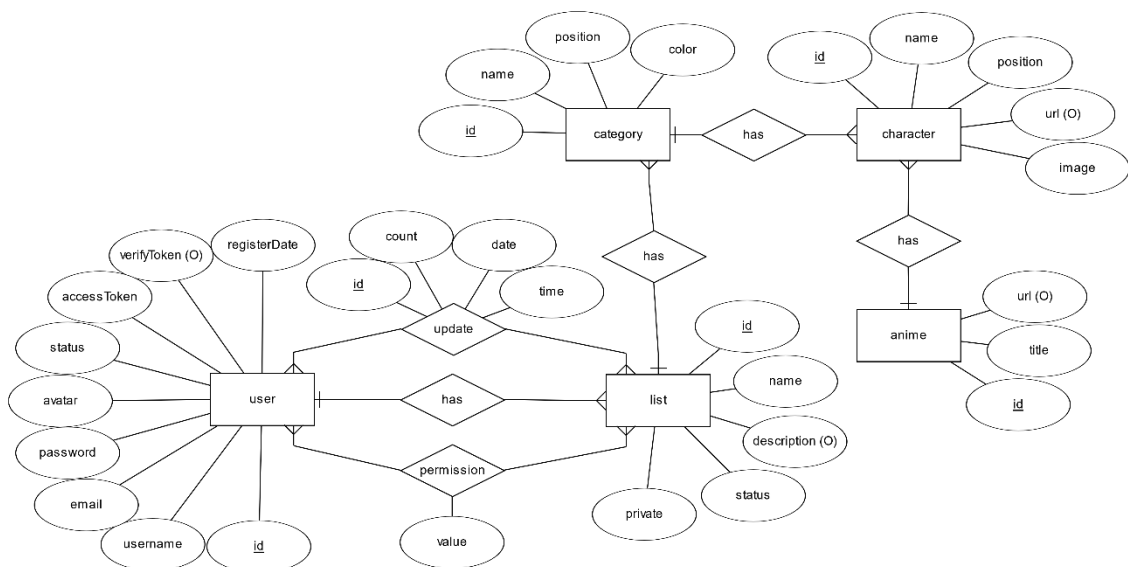
A backendhez tartozó környezeti változókat kötelező definiálni, mivel enélkül a program nem futtatható, hibákba fog ütközni. A működő adatokat tartalmazó dotenv megtalálható a backend mappájában **.env** vagy **.env.example** néven. Amennyiben a Github törölné a .env fájlt, akkor a .env.example-t csak át kell nevezni .env-re.

- **PORT:** A backend ezen a porton lesz elérhető futtatás után. Ajánlott az alapértelmezett portot használni, mivel a frontend is azt fogja keresni. Alapértelmezett: 2000
- **DB\_HOST:** Adatbázist elérő URL. Alapértelmezetten: localhost
- **DB\_USER:** Adatbázist elérő felhasználó neve. Alapértelmezetten: root
- **DB\_PASS:** Adatbázist elérő felhasználó jelszava. Alapértelmezetten üres, mivel localhoston a root felhasználóhoz nincs jelszó.
- **DB\_NAME:** Adatbázis neve. Alapértelmezetten: tier-list
- **ACCESS\_SECRET:** Egy véletlenszerűen generált token. Üres nem lehet.
- **RESEND\_KEY:** Resend mailer kulcs. Ehhez a projekthez van egy fenttartott kulcsom: re\_4DRKtUsP\_Hma4jULpQXoCay5kwRjy8NMm

### 3.4.3. Adatbázis

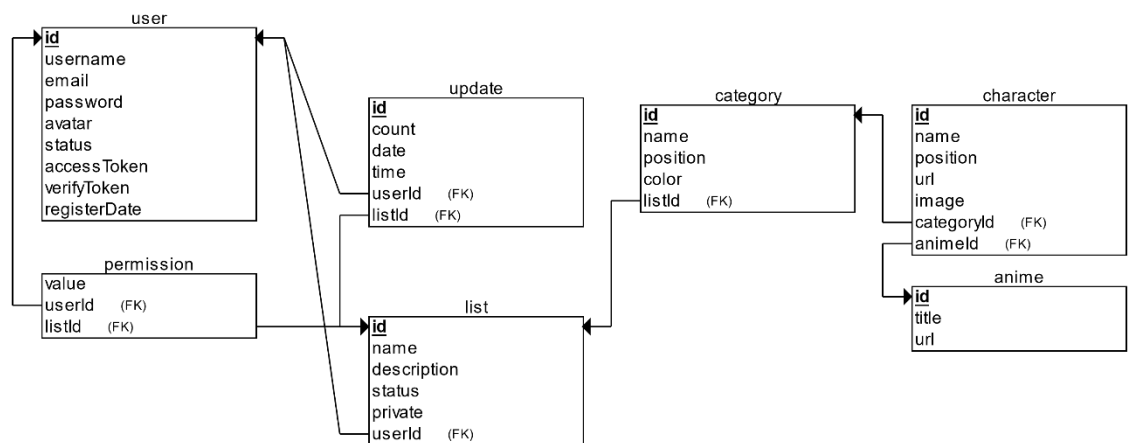
Az adatbázishoz az XAMPP v3.0.0 és a hozzá tartozó phpMyAdmin oldalát használtam, a MySQL verzió 8.2.12. Az egész adatbázist, táblákat és kapcsolatokat a Sequelize adatbáziskezelő csomag létrehozta, ezért azzal külön foglalkozni nem szükséges.

#### ER modell:



36. ábra: ER Modell

#### Adatbázis tábla kapcsolatok és relációs séma:



37. ábra: Relációs séma

A táblázatban a vastag betűs sorok az elsődleges kulcsokat, a dőlték az idegen kulcsokat jelölik. Oszlopok tartalma sorban: mező neve, típusa (zárójelben a hossza) és egy rövid leírás.

- **users:** Felhasználó adatai

id	Szám	Felhasználó azonosító
username	Szöveg (128)	Felhasználónév
email	Szöveg (128)	E-mail cím
password	Szöveg (128)	Titkosított jelszó
avatar	Szöveg (256)	Profilkép útvonal
status	Szám	Online / Offline állapot
accessToken	Szöveg (256)	Bejelentkezési token
verifyToken	Szöveg (64)	E-mail hitelesítő token
registerDate	Dátum	Regisztráció dátuma

- **lists:** Felhasználó listái

id	Szám	Lista azonosító
name	Szöveg (128)	Lista címe / neve
description	Szöveg (256)	Leírás
status	Szám (1)	Státusz
private	Logikai	Privát-e a lista
<i>userId</i>	<i>Szám</i>	<i>Felhasználó azonosító</i>

- **permissions:** Felhasználók jogosultságai a listákhoz

value	Szám (1)	Jogosultság fajtája
<i>userId</i>	<i>Szám</i>	<i>Felhasználó azonosító</i>
<i>listId</i>	<i>Szám</i>	<i>Lista azonosító</i>

- **updates:** Lista aktivitások

id	Szám	Azonosító
count	Szám	Módosítás darabszáma
date	Dátum	Módosítás dátuma
time	Idő	Módosítás ideje
<i>userId</i>	<i>Szám</i>	<i>Felhasználó azonosító</i>
<i>listId</i>	<i>Szám</i>	<i>Lista azonosító</i>



- **categories:** Lista kategóriák

id	Szám	Kategória azonosító
name	Szöveg (32)	Kategória neve
position	Szám	Listán belüli pozíció
color	Szöveg (32)	Színkód
<i>listId</i>	<i>Szám</i>	<i>Lista azonosító</i>

- **characters:** Kategórián belüli karakterek

id	Szám	Karakter azonosító
name	Szöveg (256)	Karakter neve
position	Szám	Kategórián belüli pozíció
url	Szöveg (512)	Karakter URL
image	Szöveg (512)	Kép URL
<i>categoryId</i>	<i>Szám</i>	<i>Kategória azonosító</i>
<i>animeId</i>	<i>Szám</i>	<i>Anime azonosító</i>

- **animes:** Karakterhez tartozó anime

id	Szám	Anime azonosító
title	Szöveg (256)	Anime címe
url	Szöveg (512)	Anime URL

### 3.4.4. Útvonalak

- GET / – Gyökér útvonal

#### Felhasználói útvonalak

- GET /logged – Bejelentkezési állapot lekérés
- POST /register – Regisztráció
- POST /login – Bejelentkezés
- DELETE /logout – Kijelentkezés
- GET /user/data – Felhasználói adatok lekérése
- GET /user/token/refresh – Token frissítés
- GET /user/lists – Felhasználó listáinak lekérése
- GET /user/lists/:id – Lista, kategóriái és karaktereinek lekérdezése
- PATCH /user/username – Felhasználónév módosítás
- PATCH /user/avatar – Profilkép módosítás
- DELETE /user/avatar – Profilkép törlése

- PATCH /user/email – E-mail módosítás
- PATCH /user/password – Password módosítás
- POST /user/email/verify – E-mail hitelesítés

#### **Lista útvonalak**

- GET /lists/sidebar – Navigációs menüben megjelenő listák
- GET /lists/shared – Megosztott listák
- GET /lists/public – Véletlenszerűen kiválasztott 10 publikus lista
- POST /lists/create – Lista létrehozás
- PATCH /lists/:id/update – Lista módosítás
- DELETE /lists/:id/remove – Lista törlés

#### **Lista jogosultság útvonalak**

- POST /lists/:id/permissions/create – Jogosultság létrehozás
- PATCH /lists/:id/permissions/update/:userId – Jogosultság módosítás
- DELETE /lists/:id/permissions/remove/:userId – Jogosultság törlés

#### **Lista kategória útvonalak**

- POST /lists/:id/categories/create – Kategória létrehozás
- PATCH /lists/:id/categories/:categoryId/move – Kategória mozgatás
- PATCH /lists/:id/categories/:categoryId/update – Kategória módosítás
- DELETE /lists/:id/categories/:categoryId/remove – Kategória törlés

#### **Lista karakter útvonalak**

- POST /lists/:id/characters/create – Karakter létrehozás
- PATCH /lists/:id/characters/:characterId/move – Karakter mozgatás
- PATCH /lists/:id/characters/:characterId/update – Karakter módosítás
- DELETE /lists/:id/characters/:characterId/remove – Karakter törlés

#### **Fájl lekérés útvonalak**

- GET /user/images/:filename – Profilkép lekérése
- GET /characters/images/:filename – Karakter kép lekérése

### **3.5. Fájrendszer**

A projekt három fő mappából áll: backend, frontend és documents. A documents mappában található az összes dokumentáció, tervezések és minden hozzá tartozó kép és egyéb fájl. A benne található gitignore azokat a fájlokat tartalmazza, amik csak ideiglenesen készültek el, valahol felhasználásra kerültek és nincs bennük fontos adat.

### 3.5.1. Backend mappák

- **controllers:** Minden, amit valahogy kezelni kell, annak a függvénye itt található valamely fájlban. A controller.js és socketController.js tartalmazza az összes útvonalakhoz és socket szerverhez tartozó funkciókat. Két fájlban csak middleware található, a checkPermission.js-ben a jogosultságokat tesztelem, míg a verifyToken.js a felhasználói bejelentkezéseket ellenőrzi.
- **images:** Profilképek és karakter képek. Amit a felhasználók feltöltenek, azok ide kerülnek.
- **libs:** Saját könyvtárak, amik elősegítik az egyszerűbb kódolást.
- **models:** Adatbázis tábla modellek. Az összes táblának megtalálható a sequelize-hoz szükséges modellje.
- **node\_modules:** Ez a mappa csak a függőségek telepítése után fog megjelenni, tartalmazza az összes node csomagot. (npm és bun parancsos telepítéssel is meg fog jelenni)
- **routes:** Backend útvonalakat tartalmazza.
- **sql:** Előre elkészített adatbázisok, nem szükséges beimportálni, kivéve, ha nem bíznunk a sequelize adatbázis létrehozásában vagy előre beállított adatokra van szükségünk.
- **Egyéb fájlok:** Itt található a .env, .env.example a környezeti változókhoz. Az index.js és socket.js fájlok tartalmazzák a szerverek alapbeállításait.

### 3.5.2. Frontend mappák

- **node\_modules:** Hasonlóan a backendhez, csak akkor fog megjelenni, ha letöltöttük a függőségeket.
- **public:** Publikus fájlok, ikonok.
- **src:** A weboldal összes aloldala, komponense és stílusai.
  - **components:** Apróbb komponensek az oldalhoz.
  - **css:** App.css, tailwind és animációk stílusai.
  - **pages:** Az oldalak fő fájljai.
- **Egyéb fájlok:** Index.html, PostCSS, TailwindCSS és Vite konfigurációs fájlok.

## 3.6. Forráskód

### 3.6.1. backend/libs/database.js

Az adatbázisért a Sequelize csomag felelős, amivel egyszerűen létrehozhatunk táblákat, kezelhetjük az adatainkat JavaScript nyelvben anélkül, hogy SQL parancsokat kellene írunk. Az egyetlen hiányossága az adatbázis létrehozás, amihez egy saját függvényt írtam.

```
const checkDatabase = async () => {
  try {
    const connection = await mysql.createConnection({
      host: process.env.DB_HOST,
      user: process.env.DB_USER,
      password: process.env.DB_PASS
    })
    await connection.query(`
      CREATE DATABASE IF NOT EXISTS \`${process.env.DB_NAME}\`
      DEFAULT CHARSET \`utf8\`
      COLLATE \`utf8_general_ci\`;`
    )
    logger.db('Database successfully created (if not exists).')
  } catch {
    logger.error('Something went wrong
      when trying to connect and create the database.')
    logger.error('Please start the database if it is not already running.')
  }
}
```

A mysql2 – Sequelize<sup>18</sup>-hoz kötelező csomag – használatával a program megpróbál kapcsolatot létrehozni és elkészíteni az adatbázist. Ha sikeresen lefut a parancs, azt a console-on látni fogjuk. Amennyiben probléma történne – általában az, hogy nem tudjuk elérni az adatbázist – akkor azt is jelezni fogja számunkra. Ebben a fájlban található még a Sequelize kapcsolat létrehozása is.

---

<sup>18</sup> <https://sequelize.org/>

### 3.6.2. backend/libs/logger.js

A console-on megjelenő logok az általam elkészített logger objektummal kerülnek kiíratásra. A bővítése egyszerű, amennyiben szükségünk lenne még egy kategóriára, hasonlóképpen elkészíthetjük, mint a többi függvényt. A színekért a chalk csomag felel, megadhatunk bármilyen – akár saját hexadecimális kódú – színt, amivel feldobhatjuk a konzolunkat, egyértelműbbé téve az épp kiírt információkat.

```
import chalk from "chalk"

const logger = {
  db: (msg) => {
    console.log(`${chalk.bgCyan.bold(' SEQUELIZE ')} ${msg}`)
  },
  socket: (msg) => {
    console.log(`${chalk.bgHex('#FA5').bold(' SOCKET ')} ${msg}`)
  },
  server: (msg) => {
    console.log(`${chalk.bgYellow.bold(' SERVER ')} ${msg}`)
  },
  error: (msg) => {
    console.log(`${chalk.bgRed.bold(' ERROR ')} ${msg}`)
  }
}

export default logger
```

Használata is egyszerű, először is be kell importálnunk:

```
import logger from './logger.js'
```

Ezután válasszuk ki a nekünk kellő logolást (jelen esetben db, socket, server vagy error) és használjuk a következő példához hasonlóan:

```
logger.server(`Running on http://localhost:${port}`)
```

### 3.6.3. backend/socket.js

A socket.io-ban nincs alapértelmezett süti átvittele, ezért – az express szerverhez hasonlóan – alkalmazni kell a cookie-parser middlewareet. Mivel itt máshogy működnek a middlewareben adott és kapott adatok, ezért valamivel többet kell írni. A süti azért kellene a socket.io-hoz, hogy ellenőrizni tudjuk – a backend útvonalakhoz hasonlóan – a felhasználó hitelességét, jogosultságait az adott művelethez. Enélkül bárki csatlakozhatna olyan socket szobákhoz, amikhez nem kapott engedélyt.

```
const wrap = middleware =>
  (socket, next) => middleware(socket.request, {}, next)
io.use(wrap(cp()))
io.use(verifyTokenSocket)
```

A „wrap” függvény lehetővé teszi, hogy a cookie-parser-nek egy olyan adatszerkezetet adjunk át, amire szüksége van. A verifyTokenSocket a már említett felhasználó jogosultság ellenőrzéshez szükséges, gyakorlatilag a verifyToken socket.io megfelelője, amiket a backend/controllers/verifyToken.js fájlban találunk.

#### 3.6.4. backend/libs/resend.js

Az e-mail küldésre Resend<sup>19</sup>-et használok. A Resend lehetővé teszi az egyszerű e-mailek küldését, külön cím regisztráció nélkül, saját domain-en keresztül. Az elküldendő üzenetet HTML fájllokból kéri ki a „resend\_templates” mappából.

```
import { Resend } from 'resend'
import fs from 'fs'

const resend = new Resend(process.env.RESEND_KEY)

export const sendMail = async (to, subject, template, values) => {
  let html = fs.readFileSync(template).toString()

  for (const [key, value] of Object.entries(values))
    html = html.replaceAll('{ ' + key + ' }', value)

  await resend.emails.send({
    from: 'noreply@tejfolos.sbcraft.hu',
    to,
    subject,
    html
  })
}
```

A sendMail függvénynek át kell adnunk a címzett e-mail címét, a tárgyat, a séma fájl útvonalát és a hozzá tartozó értékek objektumát. A kód beolvassa a kapott útvonalon található fájlt, majd kicseréli a benne található kapcsos zárójelekkel jelzett értékeket a kapott objektumban lévő adatokra. Ezután csak elküldi az üzenetet a megadott címre. A kód lefuttatásához szükségünk van egy RESEND\_KEY-re a környezeti változóknak.

---

<sup>19</sup> <https://resend.com/>

### 3.6.5. backend/libs/errors.js

A komplexebb hibák kezelésére egy saját készítésű osztályom van, amivel egyszerűen ellenőrizhetem az üres, e-mail és jelszó mezőket.

```
class Errors {
  constructor() {
    this.errors = {}
    this.check = false
  }

  get = (field) => {
    if (!field) return this.errors
    return this.errors[field]
  }

  push = (field, message) => {
    this.check = true
    if (!this.errors[field])
      this.errors[field] = message
  }

  count = () => {
    return Object.keys(this.errors).length
  }

  empty = (fields, message) => {
    for (const [key, val] of Object.entries(fields))
      if (EmptyTest(val))
        this.push(key, message)
  }

  email = (emails, message) => {
    for (const [key, val] of Object.entries(emails))
      if (!EmailTest(val))
        this.push(key, message)
  }

  password = (passwords, message) => {
    for (const [key, val] of Object.entries(passwords))
      if (!PasswordTest(val))
        this.push(key, message)
  }
}
```

Használatához először be kell importálnunk, majd létrehozni egy új változók.

```
import { Errors } from '../libs/errors.js'
const errors = new Errors()
```

Ezután az „errors” változóban tároljuk a hibákat. Az adatainkat tesztelni objektumonként átadva tudjuk, az objektumban lévő nevekhez fognak társulni az adott hibaüzenetek, amiket második paraméterként megadhatunk. Erre példa:

```
errors.empty(
  { username, email, password },
  'Üres mező!'
)

errors.email(
  { email },
  'Nem megfelelő formátum!'
)

errors.password(
  { password },
  'A jelszavadnak legalább 8 karakter hosszúnak kell lenni!'
)
```

Hibaüzenetet lekérni a get metódussal tudunk. Amennyiben nem adunk meg mezőnevet, akkor visszakapjuk az összes értéket.

```
// Összes hibaüzenet Lekérése
errors.get()
// Felhasználónév hibaüzenetének Lekérése
errors.get('username')
```

Hibaüzenetet manuálisan hozzáadni a push metódussal lehet, megadva a mező nevét és az üzenetet.

```
errors.push('username', 'A megadott felhasználónév már foglalt!')
```

A check változó tartalmazza, hogy van-e már hibaüzenet. Amikor hibát talál vagy adunk hozzá, akkor automatikusan igazra változik az értéke. Mielőtt visszaküldenénk a hibákat, le tudjuk ellenőrizni, hogy egyáltalán vannak-e:

```
// Ha vannak hibás adatok, akkor visszaküldi a felhasználónak
if (errors.check) return res.status(400).send({ errors: errors.get() })
```



### **3.7. Továbbfejlesztési lehetőségek**

A projekt elkészítése közben az is egy cél volt, hogy a későbbiekben egyszerűen továbbfejleszthető legyen. Az oldalak komponensekre vannak bontva, így fájl megnyitás nélkül megtalálhatunk egyes részeket. A kód a fontosabb részekben leírásokat – kommenteket – tartalmaz, hogy keresés közben tudjuk tájékozódni.

#### **3.7.1. Főoldal Demo**

A főoldalon jelenleg nem sok információ található az oldalról, valamint kevés a motivációs tényező a regisztrációra. Lejjebb görgetés esetén egy kevesebb funkciós tier list teszt verzió feldobhatná az oldalt, kedvet adhatna a felhasználóknak a regisztrációra.

#### **3.7.2. Beállítások**

Legfontosabb beállítások, mint az e-mail megjelenés, profilon megjelenő adatok, privát profilra váltás, felhasználók letiltása és bejelentkezett eszközök kezelése kerülne ide elsősorban. Ez a lehetőséget adna az oldalt használóknak, hogy testre szabhassák a profilukon megjelenő tartalmakat.

#### **3.7.3. Téma mentes Tier List**

Az oldal jelenleg kizárólag anime témában ad lehetőséget tier list elkészítésére. Szélesebb körű felhasználóbázis elérése érdekében lehetne több témát és egy téma mentes lista készítési opciót is biztosítani a felhasználók számára. Ezáltal mindenki azt készíthet, amit tényleg szeretne.

A téma mentes listák testre szabhatóak lennének azáltal, hogy beállíthatnánk a képek arányát – például 16:9, 4:3 vagy 3:4 –, megadhatnák a kötelező adatokat – például autó márka, típus – és a lehetséges linkeket. A saját mezők limitált számban lennének beállíthatóak, hogy azok elférjenek a karakter módosító területén.

#### **3.7.4. Barátlista, csevegő**

Az oldalon jelenleg nincs lehetőség a felhasználóknak egymással kommunikálni. Lista készítés közben egy helyi csevegő jelenhetne meg a jobb alsó navigációs menü opciói között, amit ki lehetne nyitni.

Barátokat a profil oldalakon lehetne hozzáadni, amiket a közös lista készítés során vagy egy felhasználó keresőben érhetnének el. Két felhasználó akkor lenne barát, ha a másik fél is elfogadta a barátkérelmet. Közöttük lehetnének privát csevegők, amik hasonlóan

működnének, mint a listánál, azzal a különbséggel, hogy ebben kizárólag két felhasználó tartózkodna.

A felhasználók letilthatnák a másikat, ezzel elérve, hogy ne jelölhessék be barátoknak, egyből elvehessék az összes jogosultságukat a listákból és tiltsák az összes csevegési lehetőséget.

#### **3.7.5. Elfelejtettem a jelszavam**

A bejelentkezésnél egy „Elfelejtettem a jelszavam” opció hasznos lenne, hogy aki tényleg ilyen problémával küzd, az meg tudja változtatni a jelszavát és ezáltal elérhesse a profilját. E-mail címére kapott jelszó változtató linkre kattintva, eljutna a weboldalra, ahol megadhatná az új jelszavát. Hasonlóan működne, mint az e-mail hitelesítés, a kapott üzenetben lévő link tartalmazná a tokent és azt felhasználva kapna engedélyt a felhasználó a jelszó módosítására.

#### **3.7.6. Karakter keresés anime alapján**

A karakterekre keresés név alapján már így is egy hasznos funkció lehet a felhasználók számára, de egy anime címét időnként egyszerűbb megjegyezni, mint egy karakter nevét, így plusz funkcióként egy anime kereső megegyszerűsíthetné a dolgukat. Rákereshetnének egy-egy anime címére, amiből a karaktereket felsorolná nekünk és onnan választhatnánk ki. Egyik megoldásként a karakter hozzáadásnál választhatnánk, hogy egy anime címére vagy karakter nevére keresünk vagy – a felhasználók számára egyszerűbb, de programozhatóságban nehezebb módszerként – a keresőbe beírt és az API-tól kapott adatok alapján döntené el a weboldal, hogy milyen találatok írjon ki.

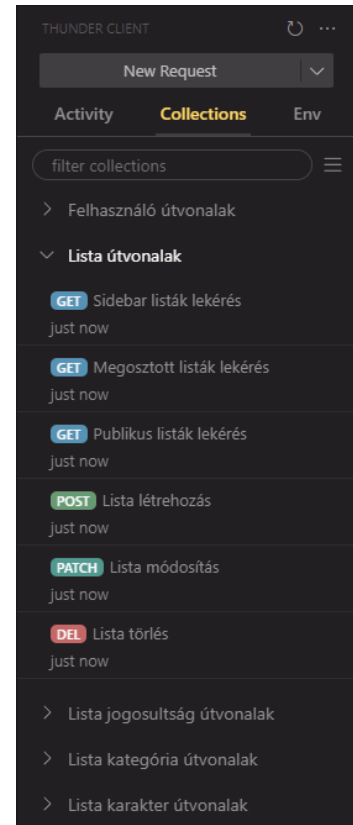
#### **3.7.7. Törlés megerősítés**

Törölni lehet, karaktert, kategóriát és profilképet is. Ezekre nincs semmilyen felugró üzenet, amiben megerősíti a törlési folyamatot. Ez zavaró lehet, ha véletlenül kattintunk a törlés gomb valamelyikére.

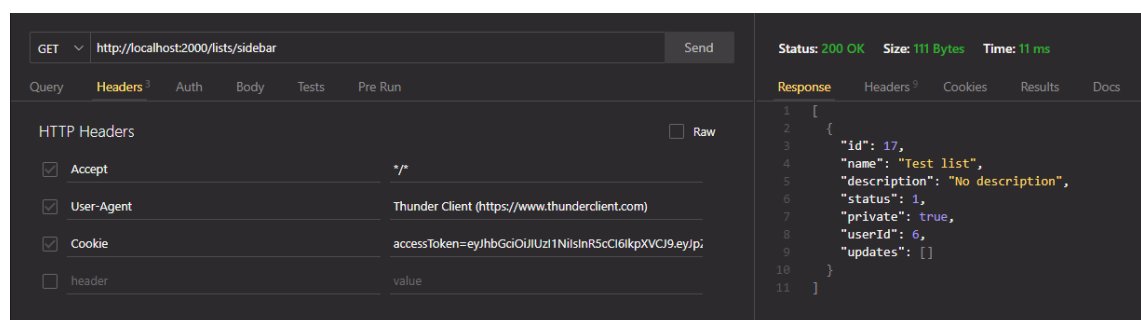
## 4. Tesztelés

A projekt elkészítése közben számos tesztelés történt. A funkciók, kódrészek egyből a fejlesztés közben kipróbálásra kerültek, hogy később ne ütközzenek váratlanul nagyobb hibákba.

A kész backendet a Thunder Client bővítménnyel teszteltem a Visual Studio Code-ban. A fejlesztés során frontenddel összekapcsolva is sokszor lett tesztelve, ezzel minimalizálva a lehetséges hibaforrásokat. Az összes útvonalhoz tartozó tesztet megtalálható a documents/tesztelés/backend mappában, több fájlként, szempontokra bontva (például felhasználó, lista útvonalak). Az újbóli lefuttatáshoz, helyenként át kell írni az URL-ben található paramétert, Headers menüben lévő Cookie accessToken-t, a Body JSON tartalmát vagy a Form-ban található képet. Ezek azért szükségesek, mivel az adatok nem fixek és felhasználónként változhatnak, például az accessToken minden bejelentkezésnél más, az általam feltöltött kép nem feltétlenül található meg azon a helyen, ahol nálam. A tesztesetek lefuttatásához szükséges lehet némi előismeret a Thunder Client bővítményhez.



38. ábra: Thunder Client Collections



39. ábra: Thunder Client Sidebar listák lekérdezése

A frontend teszteléséhez nem használtam semmilyen segédprogramot vagy bővítményt, kézzel lett elvégezve. A fejlesztés során már rengeteg ellenőrzés megtörtént, hogy ne ütközzenek később nagyobb hibákba, de az egyes nagyobb funkciók elkészülte után is történtek nagyobb tesztek. Találkoztam néhány hibával, a legnagyobbak a karakter és kategória áthelyezésnél voltak, mivel az általam kitalált metódus hiányos volt, nem minden esetben

működött. A mozgatáskor csak az új pozíciót küldöm el, a többi elem az alapján mozdul, hogy hova került átrakásra. Például, ha van 8 karakter, és én a hetediket áthelyezem az első helyre, akkor az addig 1 és 6 pozíciók közötti karaktereket eggyel „nagyobb” helyre, azaz jobbra vittem. A socket.io beépítése közben is találkoztam problémákkal, mivel a Tier List kategóriáknak egy olyan azonosítót kellett adnom – a dnd kit-nél használt komponensekben –, amik biztos nem egyeznek egyik karakterével sem, erre megoldásként a „category” kezdőkaraktereit, a „cat” kulcsszót helyeztem az adatbázisban szereplő azonosító után, ezt pedig egyszerűen leszedhetjük a parseInt JavaScript függvénnyel. A hiba ott keletkezett, hogy elmaradt a socket.io-tól kapott azonosító átírása, ezáltal a kategória megjelent, de a belehelyezett karakterek eltűntek, csak akkor jelentek meg, ha visszahelyezték egy – már oldal betöltésekor is – létező kategóriába vagy újratöltöttük az oldalt. A végső fázis felé közeledve tesztelés közben kiderült, hogy a felhasználó tudja módosítani más profil oldalán is az adatait, kezelni kellett a Jikan API által kapott adatokat, mivel nem biztos, hogy kaptunk tőle anime vagy manga szériát. A fejlesztés közben olyan kritikus hibába nem ütköztem, amivel nagyobb részeket kellett volna újra átgondolni, mivel rendszeresen teszteltem, így csak apróbb, azonnal javítható problémák keletkeztek.

## 5. Összegzés

A projekt elkészítése közben számos tapasztalatot szereztem backend és frontend téren is. A React<sup>20</sup> könyvtárat és a hozzá tartozó keretrendszereket, környezeteket a vizsgamunka elkészítése előtt kezdtem el részletesebben megismerni, ezért a weboldal megírásánál bővíthettem és megerősíthettem a tudásomat. Megismerkedhettem a dnd kit<sup>21</sup> eszköztárral, amivel drag and drop funkcionalitást vihettem az oldalba, emellett a React Tooltip<sup>22</sup> csomaggal is, amivel egérráhúzásra eszköztípek kiírását valósíthattam meg egyszerűen. A fejlesztés során eleinte CRA<sup>23</sup> (create-react-app) keretrendszert használtam, amiből áthelyeztem a projektet Vite<sup>24</sup> környezetbe – ami egy gyorsabb és modernebb lehetőséget biztosít a React fejlesztésben –, ezáltal tapasztalatot szerezhettem ezek használatában. A weboldal kezdeti fejlesztése során nem igazán ismertem a Bun<sup>25</sup> csomagkezelőt, de a kész állapot megközelítésénél már megfogott a gyorsasága, ezért kipróbáltam és elkezdtem használni. A fejlesztői böngészőm is módosult, az első konzultációmon még Opera GX<sup>26</sup>-ben dolgoztam, később viszont váltottam Firefox<sup>27</sup>-ra, mivel meglepően jobbak a benne található fejlesztői funkciók. A Bun, Firefox és Vite nem voltak nagy váltások a projekt elkészítésében, ugyanúgy megmaradtak a szokásos hibák, mint például elfelejtem átadni a props változót a komponenseknek Reactban, vagy nem importáltam be a functiont a backendben, de nagyban felgyorsították a fejlesztési folyamatot. Ami még problémákat okozott az elkészítés közben, az a socket.io és a hozzá tartozó események kezelése. Ez volt a második projektem, amiben Node JS<sup>28</sup> backendet és React frontendet használtam egy többfelhasználós funkció kialakításában, ezért már rendelkeztem valamennyi tapasztalattal, de még így is jól át kellett gondolni a megoldási folyamatot. Úgy vélem ez a feladat lehetőséget adott a tudásom megerősítésében és fejlesztésében, ami elősegítheti a jövőbeli munkáim elkészítését, mivel már rendelkezek tapasztalattal egy komplexebb projektben.

---

<sup>20</sup> <https://react.dev/>

<sup>21</sup> <https://dndkit.com/>

<sup>22</sup> <https://react-tooltip.com/>

<sup>23</sup> <https://create-react-app.dev/>

<sup>24</sup> <https://vitejs.dev/>

<sup>25</sup> <https://bun.sh/>

<sup>26</sup> <https://www.opera.com/hu/gx>

<sup>27</sup> <https://www.mozilla.org/hu/firefox/new/>

<sup>28</sup> <https://nodejs.org/en>

## 6. Ábrajegyzék

1. ábra: Főoldal .....	2
2. ábra: Regisztráció .....	3
3. ábra: Bejelentkezés .....	4
4. ábra: Listák .....	5
5. ábra: Új lista.....	5
6. ábra: Jogosultságok.....	6
7. ábra: Lista készítés.....	7
8. ábra: Karakter .....	8
9. ábra: Új karakter .....	9
10. ábra: Karakter módosítás .....	9
11. ábra: Kategória.....	10
12. ábra: Új kategória.....	10
13. ábra: Kategória módosítás .....	11
14. ábra: Profil .....	11
15. ábra: Felhasználónév módosítás .....	12
16. ábra: E-mail módosítás .....	13
17. ábra: Jelszó módosítás .....	13
18. ábra: Profilkép módosítás .....	14
19. ábra: E-mail hitelesítés - Nincs bejelentkezve.....	14
20. ábra: E-mail hitelesítés - Hibás link.....	15
21. ábra: E-mail hitelesítés - Sikeres .....	15
22. ábra: E-mail hitelesítés - Már hitelesítve van .....	15
23. ábra: E-mail hitelesítő üzenet .....	15
24. ábra: Sidebar .....	16
25. ábra: Lista navigáció .....	16
26. ábra: NPM.....	17
27. ábra: Bun.....	17
28. ábra: Docker.....	18
29. ábra: Elkészült Docker Container .....	19
30. ábra: Vite.....	22
31. ábra: React .....	22
32. ábra: Tailwind .....	22

33. ábra: dnd kit .....	23
34. ábra: FontAwesome .....	23
35. ábra: Jikan API.....	23
36. ábra: ER Modell.....	25
37. ábra: Relációs séma .....	25
38. ábra: Thunder Client Collections .....	37
39. ábra: Thunder Client Sidebar listák lekérdezése.....	37

## 7. Felhasznált források

*Stack Overflow – Where Developers Learn, Share, & Build Careers*

<https://stackoverflow.com/> [letöltés: 2024.03.09.]

*MDN Web Docs*

<https://developer.mozilla.org/en-US/> [letöltés: 2024.03.10.]

*Node.js – Run JavaScript Everywhere*

<https://nodejs.org/en> [letöltés: 2024.03.19.]

*npm | Home*

<https://www.npmjs.com/> [letöltés: 2024.03.20.]

*Bun – A fast all-in one JavaScript runtime*

<https://bun.sh/> [letöltés: 2024.03.20.]

*Welcome | React Tooltip*

<https://react-tooltip.com/> [letöltés: 2024.03.30.]

*Vite | Next Generation Frontend Tooling*

<https://vitejs.dev/> [letöltés: 2024.03.30.]

*React*

<https://react.dev/> [letöltés: 2024.03.30.]

*Tailwind CSS – Rapidly build modern websites without ever leaving your HTML.*

<https://tailwindcss.com/> [letöltés: 2024.03.30.]

*dnd kit – a modern drag and drop toolkit for React*

<https://dndkit.com/> [letöltés: 2024.03.30.]

*Font Awesome*

<https://fontawesome.com/> [letöltés: 2024.04.01.]

*Jikan – Unofficial MyAnimeList API*

<https://jikan.moe/> [letöltés: 2024.04.02.]

*Resend*

<https://resend.com/home> [letöltés: 2024.04.03.]



*Git*

<https://git-scm.com/> [letöltés: 2024.04.05.]

*GitHub*

<https://github.com/> [letöltés: 2024.04.05.]

*Jira | Issue & Project Tracking Software | Atlassian*

<https://www.atlassian.com/software/jira> [letöltés: 2024.04.06.]

*Figma: The Collaborative Interface Design Tool*

<https://www.figma.com/> [letöltés: 2024.04.06.]

*Töltsd le az asztali Firefoxot – a Mozillától*

<https://www.mozilla.org/hu/firefox/new/> [letöltés: 2024.04.06.]

*Visual Studio Code – Code Editing, Redefined*

<https://code.visualstudio.com/> [letöltés: 2024.04.06.]

*XAMPP Installers and Downloads for Apache Friends*

<https://www.apachefriends.org/hu/index.html> [letöltés: 2024.04.13.]

*Docker: Accelerated Container Application Development*

<https://www.docker.com/> [letöltés: 2024.04.20.]