Mangadex:

A MangaDex a „Project AniDex”-nek egy része ami más scanlation csoportok által fordított és feltöltött mangákat tárolja és hosztolja a felhasználóknak. A „Project AniDex” alapítója Holo 2009-ben hozott létre egy anime felirat fordító csoportot, ami azóta több anime, manga és visual novellával foglalkozó oldalt készített, mint például a MangaDex-et.

A MangaDex-nek van egyedül nyilvános API-ja amivel sok különböző fordítású mangát lehet elérni. Más API („scraper”-ek) meg aggregátor oldalakról szedik le a mangákat, amik már vagy nem léteznek, vagy elavultak az API call-ok és ezek az API-k nem az adott oldal által fejlesztett. És ezek az API-ok eléggé korlátozottak funkciókban.

MangaDex:

* Manga keresés
* Manga fejezetek lekérdezése
* Manga részletek lekérdezése
* Statikus manga tag-ek lekérdezése(műfaj,korosztály,manga státusz)
* Manga fejezet feltöltése(MangaDex szerverre)
* Manga létrehozása(MangaDex szerverre)
* Manga borítók lekérdezése,feltöltése
* Fordító csoportok követése, keresése, létrehozása
* Szerzők keresése, lekérdezése

Más API-k:

* Manga keresés
* Manga fejezetek lekérdezése
* Manga részletek lekérdezése

De a legtöbb API („scraper”) már elavult és nem tartják karban.

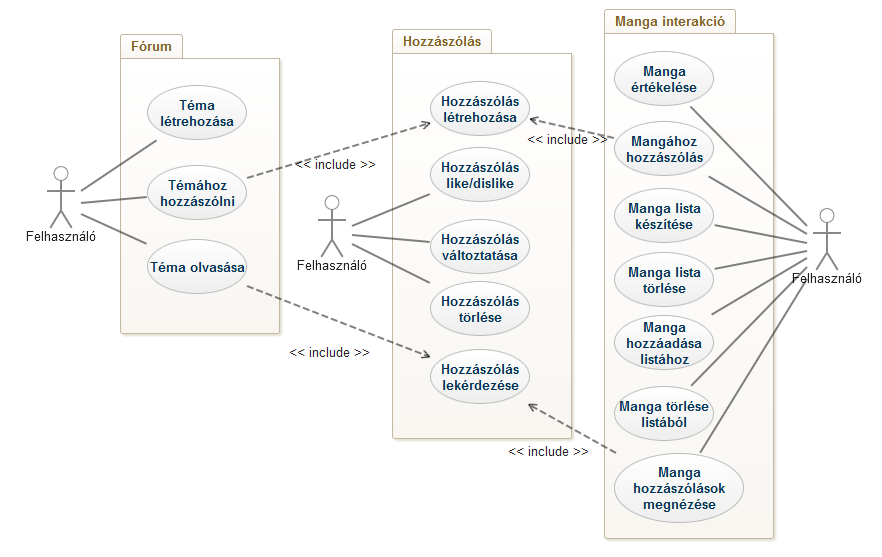
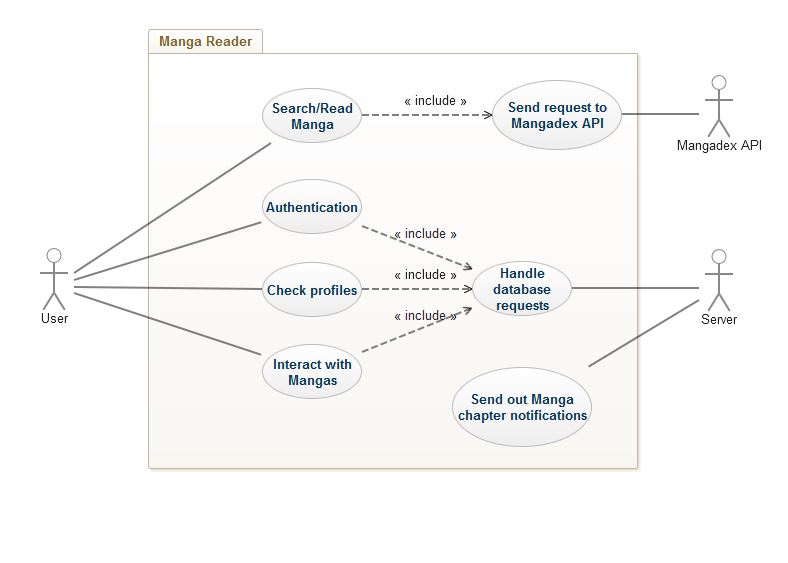
<https://github.com/febryardiansyah/manga-api>

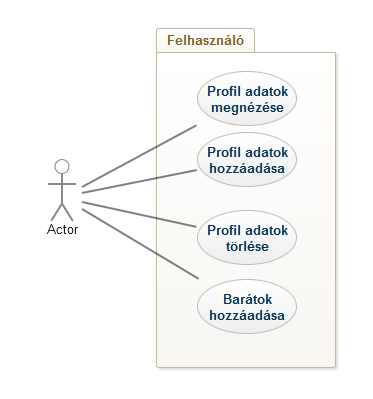
Maga az oldal ahonnan a mangákat szedné, kevés mangát tartalmaz, meg az API-sem csinál jól call-okat

<https://github.com/russellhill/mangaapi>

A kódban használt url-en már nem létezik maga az oldal

Use Case:





Search/Read Manga: A felhasználó a kliensen keresztül küld http requesteket a Mangadex API-ra ami vissza adja a kért adatokat, pl.: Manga keresés, Manga adatai megnézése, Manga fejezetek olvasása

Authentication: A felhasználó tud regisztrálni, bejelentkezni.

Check Profiles: Más felhasználók profiljának megnézése ami tartalmaz pl.: Kedvenc mangák, írt kommentek, Manga listák

Interact with Mangas: pl.: Kommentek írása Manga fejezetek alá, Mangák könyvjelzőbe tétele

**Más manga olvasók funkciója:**

MangaDex:

* Manga keresés, olvasás
* Felhasználó regisztráció,login
* Manga követése
* Manga lista készítése
* Manga fejezet feltöltése
* Kommentelés fejezetek alá
* Manga létrehozása
* Manga borítók feltöltése, nézése
* Fordító csoportok követése, keresése, létrehozása
* Szerzők keresése, lekérdezése

Más manga oldalak:

* Manga keresés, olvasás
* Felhasználó regisztráció,login
* Manga lista készítése
* Kommentelés fejezetek alá
* Egyéni feature: Fórum threadek amikben lehet adott témáról beszélgetni

Saját manga reader funkciói:

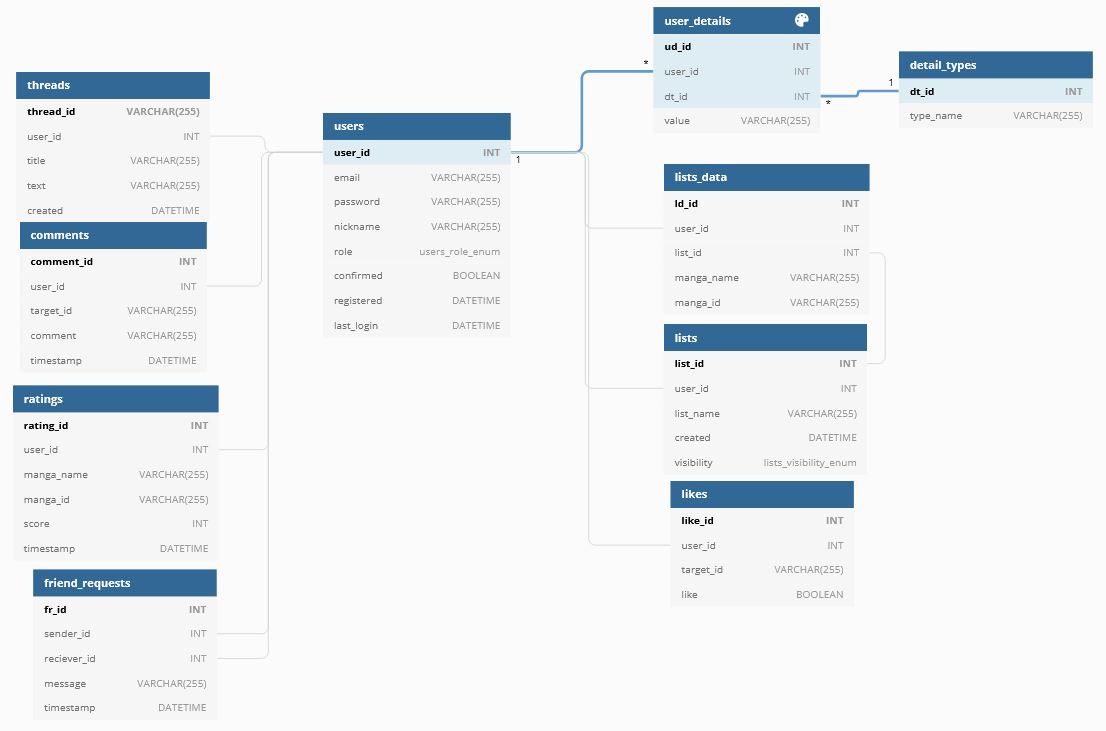
Biztos funkciók:

* Manga keresés, olvasás
* Felhasználó regisztráció,login
* Manga lista készítése
* Kommentelés fejezetek alá

Remélhetőleg meg lesz funkciók:

* Manga ajánlás
* Emailben értesítés küldése új fejezetről
* Fórum/beszélgetős thread ek
* + Közösségi dolog(chat,barátlista…)

Adatbázis felépítése:



API fájl struktúra/felépítés

* API
  + Index.js
  + Src
    - Routes
      * Auth.js
      * Comments.js
      * Lists.js
      * Threads.js
      * User.js
      * Votes.js
    - Helpers
      * Queries
      * Validations
      * checkIfFriends.js
      * checkIfMangaExists.js
      * getVisibility.js
      * isEmpty.js
      * verify.js
    - Database
      * Db.js

API fájl/mappa magyarázat

Index.js:

* Szerver indítása, express route-ok beállítása

Src/routes:

* Azokat az endpointokat tartalmazza amikre érkezhet valamilyen http request, itt feldolgozza a kérést és végrehajt valamit, a megadott queries, és validációt felhasználva

Src/Helpers

* Queries: SQL lekérdezéseket tartalmaz a különböző routokhoz
* Validations: Validáció sémákat, funkciókat tartalmaz mindegyik fájl különböző route-hoz tartalmaz validációkat
* checkIfFriends.js: Megnézi hogy a bemenetben megadott 2 user\_id barátok-e ha igen igazzal tér vissza, ha nem hamissal
* checkIfMangaExists.js: Azt fogja ellenőrizni hogy létezik-e a paraméterben megadott manga
* getVisibility.js: paraméterben megadott int-re switchel megadja a lista láthatóságát szövegesen
* isEmpty.js: Megnézi hogy a paraméterben megadott objekt/string üres-e
* verify.js: http requesteknél használjuk arra hogy ellenőrizzük a header-ben beállított Authorization token létezését, érvényességét

Routeok:

Auth.js

* localhost:3001/api/auth/login
  + [POST]
  + Body:{email, password}
  + Visszaad egy JWT-t ha van ilyen email, password az adatbázisban
* localhost:3001/api/auth/register
  + [POST]
  + Body:{email, nickname, password, password2}
  + Regisztrál egy felhasználót ha nincs ilyen email, nickname az adatbázisban

Comments.js

* localhost:3001/api/comments/:target\_id
  + [GET]
  + Body:{}
  + Visszaadja a paraméterben megadott target\_id-nak a kommentjeit
* localhost:3001/api/comments/:target\_id
  + [POST]
  + Body:{message}
  + A paraméterben megadott target\_id-hoz insertel egy kommentet
* localhost:3001/api/comments/:c\_id
  + [DELETE]
  + Body:{}
  + A paraméterben megadott c\_id-jú kommentet törli, ha a komment tulajdonosa a requester

Lists.js

* localhost:3001/api/lists
  + [POST]
  + Body:{list\_name, visibility}
  + Létrehoz egy manga listát a megadott list\_name-el, visibility vel ha még nincs ilyen a requester-nek
* localhost:3001/api/lists
  + [GET]
  + Body:{}
  + Vissza adja a requesternek az összes létrehozott listáját
* localhost:3001/api/lists/user/:user\_id
  + [GET]
  + Body:{}
  + Vissza adja a paraméterben megadott id-jú usernek a publikus listáit, és a barátoknak látható listáit ha barátok
* localhost:3001/api/lists/:list\_id
  + [GET]
  + Body:{}
  + Vissza adja a paraméterben megadott list\_id-jú listát, ha publikus, ha a lista tulaj és a requester barátok és a lista friendly visibility, ha a lista tulaja a requester és a lista privát
* localhost:3001/api/lists/:list\_id
  + [PUT]
  + Body:{list\_name, visibility}
  + Megváltoztatja az adott list\_id-jú lista nevét, láthatóságát, ha a requester a lista tulajdonosa
* localhost:3001/api/lists/:list\_id
  + [POST]
  + Body:{manga\_name, manga\_id}
  + Hozzáadja a body-ban megadott mangát a paraméterben megadott listához ha a requester a lista tulajdonosa
* localhost:3001/api/lists/:list\_id
  + [DELETE]
  + Body:{}
  + Kitörli a paraméterben megadott listát ha a requester a lista tulajdonosa
* localhost:3001/api/lists/entry/:list\_id/:ld\_id
  + [DELETE]
  + Body:{}
  + Kitörli a paraméterben megadott listának a megadott id-jú adattagját ha a requester a lista tulajdonosa

Threads.js

* localhost:3001/api/threads
  + [POST]
  + Body:{title, text}
  + Létrehoz egy thread-et a bodyban megadott cím,és text-el
* localhost:3001/api/threads
  + [GET]
  + Body:{}
  + Vissza adja az összes thread-et
* localhost:3001/api/threads/:t\_id
  + [GET]
  + Body:{}
  + Vissza adja a paraméterben lévő thread-et
* localhost:3001/api/threads/:t\_id
  + [DELETE]
  + Body:{}
  + Kitörli a paraméterben lévő thread-et ha a requester a tulajdonosa
* localhost:3001/api/threads/own/:u\_id
  + [GET]
  + Body:{}
  + Vissza adja a paraméterben megadott user\_id összes thread jét

User.js

* localhost:3001/api/user
  + [GET]
  + Body:{}
  + Vissza adja a JWT tulajdonosának az adatait
* localhost:3001/api/user/:u\_id
  + [GET]
  + Body:{}
  + Vissza adja a paraméterben megadott user nek az adatait
* localhost:3001/api/user/details
  + [POST]
  + Body:{dt\_id,value}
  + Az authorization be lévő usernek hozzáad egy új user\_detailt a bodyban lévő input alapján
* localhost:3001/api/user/details/:ud\_id
  + [DELETE]
  + Body:{}
  + Kitörli a paraméterben megadott user detail-t
* localhost:3001/api/user/details/:ud\_id
  + [PUT]
  + Body:{value}
  + Megváltoztatja a meglévő user\_detail ját a bodyban megadott input ra
* localhost:3001/api/user/friend-request
  + [POST]
  + Body:{reciever\_id, message}
  + Küld egy friend requestet a bodyban megadott felhasználónak egy üzenettel
* localhost:3001/api/user/friend-request/accept
  + [POST]
  + Body:{sender\_id}
  + Elfogadja a friend requestet a body-ban lévő user től
* localhost:3001/api/user/friend-request
  + [DELETE]
  + Body:{other\_id}
  + Elutasítja/türli a friend requestet a body-ban lévő user től
* localhost:3001/api/user/get/friend-request
  + [GET]
  + Body:{}
  + Vissza adja az összes elküldött,kapott friend requesteket
* localhost:3001/api/user/friend
  + [DELETE]
  + Body:{friend\_id}
  + Törli a bodyban megadott user-t a barát listából

Votes.js

* localhost:3001/api/votes/like
  + [POST]
  + Body:{target\_id, like}
  + Like/dislike-olja a target\_id-ben megadott dolgot
* localhost:3001/api/votes/like/:target\_id
  + [GET]
  + Body:{}
  + Megkapja a target\_id-ban megadott dolognak a like, dislike jait
* localhost:3001/api/votes/rate/:m\_id
  + [GET]
  + Body:{}
  + Megkapja a paraméterben megadott mangának az értékelését
* localhost:3001/api/votes/rate/:m\_id
  + [POST]
  + Body:{manga\_name, score}
  + Értékeli a paraméterben megadott mangát a bodyban megadott pont-al
* localhost:3001/api/votes/rate/:m\_id
  + [PUT]
  + Body:{manga\_name, score}
  + Megváltoztatja a felhasználó által előzőleg megadott értékelést
* localhost:3001/api/votes/rate/:m\_id
  + [DELETE]
  + Body:{manga\_name, score}
  + Törli a felhasználó áltat megadott értékelést