Pannon Egyetem

Műszaki Informatikai Kar

[Tanszék neve]

[Szak neve]

**SZAK-/DIPLOMADOLGOZAT**

**[Dolgozat címe]**

**[Név]**

Témavezető: [Témavezető neve]

Külső/belső konzulens: [Konzulens neve]

[évszám]

Témakiírás

A szkennelt formában megkapott témakiírás beillesztése a dolgozatba.



Hallgatói nyilatkozat

Alulírott <<hallgató neve>> hallgató kijelentem, hogy a dolgozatot a Pannon Egyetem <<tanszék neve>>én készítettem a <<végzettség>> végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Dátum: Veszprém, [év hónap nap]

*<<hallgató neve>>*

Témavezetői nyilatkozat

Alulírott <<témavezető neve>> témavezető kijelentem, hogy a dolgozatot *<<*hallgató neve*>>* a Pannon Egyetem <<tanszék neve>>én készítette <<végzettség>> végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Dátum: Veszprém, [év hónap nap]

*<<témavezető neve>>*

Köszönetnyilvánítás

A hallgató köszönetet nyilvánít mindazoknak, akiktől (elméleti, gyakorlati, erkölcsi stb.) segítséget kapott.

Tartalmi összefoglaló

Tartalmi összefoglaló magyarul. Az összefoglalónak tartalmaznia kell (rövid, velős és összefüggő megfogalmazásban) a következőket:

* téma megnevezése,
* megoldott feladat megfogalmazása,
* megoldási mód,
* elért eredmények,
* kulcsszavak (4-6 darab)
* terjedelme nem lehet több 1 A4-es oldalnál.

Az összefoglalót magyar és angol nyelven kell készíteni. Sorrendben a dolgozat nyelvével megegyező kerül előrébb. A cím Title stílusú, formázása: Times New Roman, nagybetű, 14 pt, félkövér, középre igazított; az összefoglaló Normál stílusú, formázása: Times New Roman, 12 pt, sorkizárt, 1.5-ös sortávolság.

**Kulcsszavak:** [4-6 kulcsszó felsorolása, vesszővel elválasztva]

Abstract

Abstract in English

**Keywords:** [list 4-6 keywords]

Tartalomjegyzék

[Jelölésjegyzék 9](#_Toc99977148)

[1. Bevezetés 10](#_Toc99977149)

[2. Versenytárs elemzés 10](#_Toc99977150)

[3. Saját megoldás ismertetése 10](#_Toc99977151)

[3.1. Backend 10](#_Toc99977152)

[3.2. Frontend 10](#_Toc99977153)

[3.3. Proxy 10](#_Toc99977154)

[4. Felhasznált technológiák 10](#_Toc99977155)

[4.1. Mangadex API(Külső API) 10](#_Toc99977156)

[4.2. Backend 10](#_Toc99977157)

[4.2.1. NodeJS 10](#_Toc99977158)

[4.2.2. Express 10](#_Toc99977159)

[4.2.3. MariaDB 10](#_Toc99977160)

[4.2.4. Json Web Token (JWT) 10](#_Toc99977161)

[4.2.5. Socket Io 10](#_Toc99977162)

[4.2.6. Nodemailer 11](#_Toc99977163)

[4.2.7. Bcrypt 11](#_Toc99977164)

[4.2.8. Nodecron 11](#_Toc99977165)

[4.2.9. Uuid 11](#_Toc99977166)

[4.3. Frontend 11](#_Toc99977167)

[4.3.1. React 11](#_Toc99977168)

[4.3.2. Redux 11](#_Toc99977169)

[4.3.3. Axios 11](#_Toc99977170)

[4.3.4. Socket Io Client 11](#_Toc99977171)

[4.3.5. Typescript 11](#_Toc99977172)

[4.3.6. Classnames 11](#_Toc99977173)

[4.4. Proxy 11](#_Toc99977174)

[4.4.1. http-proxy-middleware 11](#_Toc99977175)

[5. Specifikáció 11](#_Toc99977176)

[5.1. Funkcionális követelmények 11](#_Toc99977177)

[5.2. Nem funkcionális követelmények 12](#_Toc99977178)

[5.3. Rendszer architektúra 12](#_Toc99977179)

[5.3.1. Kommunikációk rétegek között 12](#_Toc99977180)

[5.3.2. MangaDex API 12](#_Toc99977181)

[5.3.3. Frontend 12](#_Toc99977182)

[5.3.4. Backend 12](#_Toc99977183)

[5.3.5. Adatbázis 12](#_Toc99977184)

[5.3.6. Proxy 12](#_Toc99977185)

[6. Tesztek 12](#_Toc99977186)

[6.1. Backend 12](#_Toc99977187)

[6.2. Frontend 12](#_Toc99977188)

[7. Továbbfejlesztési lehetőségek 12](#_Toc99977189)

[8. Telepítési utmutató 12](#_Toc99977190)

[9. Felhasználói dokumentáció 12](#_Toc99977191)

[Irodalomjegyzék 14](#_Toc99977192)

[Mellékletek 15](#_Toc99977193)

[Ábrajegyzék 16](#_Toc99977194)

[Táblázatjegyzék 17](#_Toc99977195)

Jelölésjegyzék

|  |  |
| --- | --- |
| AI: | Artificial Intelligence (Mesterséges Intelligencia) |
| GPU: | Graphical Processing Unit (Grafikus Processzor / Grafikus Feldolgozó Egység) |
| API: | Application Programming Interface (Alkalmazásprogramozási Felület) |
| CPU: | Central Processing Unit (Központi Feldolgozó Egység / Processzor) |
| GUI: | Graphical User Interface (Grafikus Felhasználói Felület) |
| HCI: | Human Computer Interaction (Ember-gép kapcsolat) |
| CIS: | Cognitive Information System (Kognitív információs rendszer) |

# Bevezetés

Azért választottam ezt a témát mert szakdolgozat téma választás idejében találtam rá a MangaDex API-ra ami lehetővé teszi hogy saját manga olvasó programot/weboldalt hozhassunk létre, és mivel szeretem a mangákat meg a webfejlesztés témaköre is érdekelt, úgy gondoltam hogy tökéletes szakdolgozat téma lenne egy saját közösségi weboldalt készíteni ami a mangák körül forog.

A weboldal főleg olyan embereknek készült akik nem csak mangákat szeretnének olvasni, hanem más, manga iránt érdeklődő emberekkel szeretnének találkozni, beszélgetni.

# Versenytárs elemzés

Nagyjából 3 fajta manga olvasó weboldal létezik az interneten

* Kiadó által üzemeltetett oldalak
* Fordító oldalak
* Gyűjtő oldalak

A kiadó által működtetett, és a fordító oldalakkal nem fogunk most foglalkozni mert bár lehet rajtuk mangákat olvasni, vannak hátrányaik mint például a hiányzó közösségi funkciók(Ez egy univerzális hiányosság a manga olvasó oldalak többségénél), a fordítói oldalaknál a kevés kínálat, a kiadói oldalaknál az előfizetés és néhány esetben a kevés kínálat.

A gyűjtő oldalak többsége nagy kínálatot biztosít és legtöbbször reklámokkal/adományokkal biztosítják fennmaradásukat. Ezek az oldalak az esetek túlnyomó részében web scraper megoldással különböző oldalakról(Fordító oldalakról, más manga kiszolgáló oldalakról) szedik össze a mangákat és adják át az olvasóknak.

A MangaDex(az az oldal ami a szakdolgozatban elkészített weboldalhoz biztosítja a manga api-t) is gyűjtő oldalnak mondható de ők a manga szolgáltatáshoz nem web scraper megoldással szerzik a mangákat, hanem saját szerverükre töltik fel a mangákat amiket a fordítók nyújtanak így is kimélik a fordító oldalakat a fölösleges hotlinkingtől.

A manga olvasó weboldalak többsége a fejezetek alá való kommentelésen(gyakoribb), fórumokon(ritkább) kívül nem rendelkezik közösségi funkciókkal és inkább más platformokon(Discord vagy IRC-ken) folytatnak aktív beszélgetéseket. De a közösségi funkciókon kívül szinte az összes manga olvasó weboldalon lehet ugye mangákat olvasni, saját listákat létrehozni ahová gyűjthetünk például olyan mangákat amik érdekelnek minket, így könnyebben lehet emlékezni hogy miket olvasunk.

# Saját megoldás ismertetése

## Backend

Backend szerverhez Node.js-t használom ami egy javascript futási környezetet biztosít, szóval lehet javascript kódokat lefuttatni a böngészőn kívül is.

A projetkhez használt backendnek fő feladata a közösségi funkciók szolgáltatása a frontendnek, de 2 fajta módon szolgálja ki a klienst:

* REST API
* Valós idejű szerver-kliens kommunikációval

A valós idejű szerver-kliens kommunikáció biztosítja a barátok közötti beszélgetés lehetőségét, míg a REST API http végpontokkal biztosítja az adatok szolgáltatását és változtatásának a lehetőségét.

Az api végpontjainak kezelésére Express.js-t használok, a szerver-kliens valós kommunikációhoz pedig socket.io-t.

## Adatbázis

Az adatok tárolására MariaDB-t használok ami egy relációs adatbázis. A backend és az adatbázis egy Node.js package(mysql) segítségével kommunikál.

## Frontend

A weboldalhoz React-ot használok mert az a fajta modularitás amit a React nyújt, átlátható és könnyen kezelhető kódot biztosít komponensek formájában.

A frontenden adatok tárolására React Redux ot használok, ami egy globális állapotban(Store) tud tárolni adatokat, amiket a program bármelyik komponenséből elérhetünk, és könnyen lehet követni a Store-ban történő változásokat, ami elősegíti az debugolást.

## Proxy

Magához a weboldalhoz még használok egy proxy szervert is ami ugyan úgy Node.js-ben fut Expressjs és egy Node.js package(http-proxy-middleware) segítségével. Ez a proxy a frontendről érkező azon kéréseket kapja, amik a külső api-al akarnak interaktálni.

Hogy miért használok proxyt? Elsősorban azért hogy a MangaDex-en lévő képeket legyen az Manga borítókép vagy a manga fejezetek paneljeit, letöltse saját tárhelyre hogy elkerüljük a hotlinkelést, szóval ha valaki mangát olvas a mi oldalunkon, akkor a képeket nem a MangaDex szerveréről fogja megkapni, hanem saját szerverről így csökkentjük a MangaDex szerverén a terhelést. A másik indok amiért proxyt használok, hogy elrejtsem az api útvonalát. Míg én most csak útvonal elrejtésre használom, a proxy használata tökéletes arra hogy biztonságos helyen(saját szerveren) tároljuk külső api-khoz szükséges hitelesítő kulcsokat, így nem áll fennt annak a veszélye hogy kikerül a nagyvilágba.

# Felhasznált technológiák

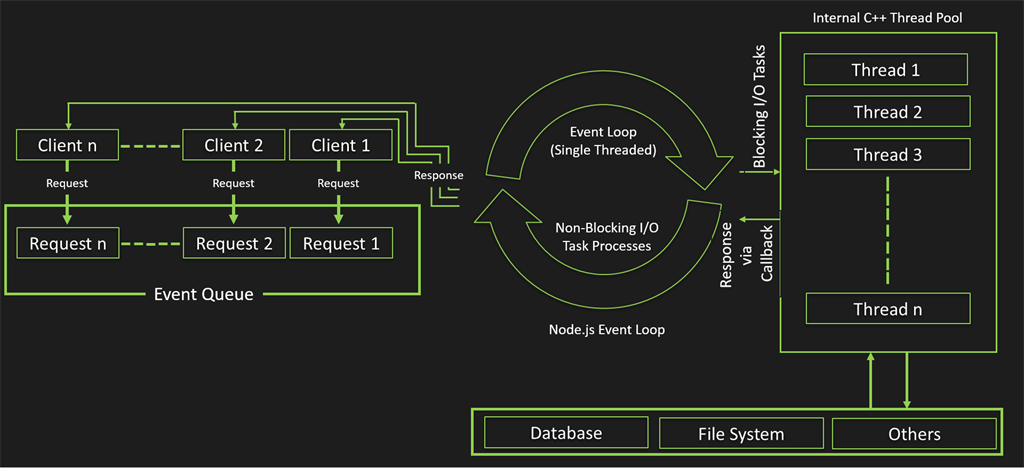
## Mangadex API(Külső API)

A MangaDex API lehetőséget biztosít arra, hogy a MangaDex hatalmas manga tárházával interaktálni tudjunk, legyen az manga információk lekérdezése, új manga feltöltése, fordító csapatok létrehozása és még sok más. Az adatokat RESTful API módon szolgáltatják, tehát URL címekre történő http requestekkel lehet lekérni, módosítani, feltölteni és törölni a kívánt adatokat.

## Backend

### NodeJS

A Node.js egy cross-platform futási környezetet biztosít ami lehetővé teszi a jól skálázható szerverek építését Javascript nyelvben. Node a kapcsolatokat „elméletben” 1 szálon dolgozza fel aszinkron módon, gyakorlatban a fő szál(Event Loop) más szálaknak(Workers) osztja ki a feladatokat egy event sorból amiket ők elvégeznek a megfelelő C++ megvalósítással párhuzamosan vagy aszinkron módon és vissza küldik az eredményt a fő szálnak, amit vissza adhat a kérést kezdeményezett kapcsolatnak



<https://www.c-sharpcorner.com/article/node-js-event-loop/>

A Node.js legnagyobb előnye a hatékony Input/Output kezelés mint például a fájlrendszerre való írás/olvasás vagy a hálozatról érkező adatok olvasása, a nehézkes szálak kezelésével. De mivel már elég népszerű lett ez a futási környezet ezért az interneten megtalálható nagyon sok segítség, és mivel javascriptet használ, ezért a frontend fejlesztők is könnyen beléphetnek a backend világába. Előnyhőz lehetne sorolni a Node.js csomagkezelőjét(Package manager) a Node Package Managert(npm), mert ennek segítségével nagyon hamar adhatunk hozzá a programunkhoz olyan kódokat amik megkönnyítik munkákat mert nem kell saját kódot írnunk hanem más, az esetek 99%-ban jobban megírt megvalósítását használhatjuk.

Viszont egyik hátránya a Node.js nek hogy a CPU intenzív feladatokkal nem tud olyan jól megbírkózni, mint a fennt említett I/O feladatokkal, mert a azokat a műveleteket amik a CPU-t veszik igénybe szinkron módon futnak, míg az I/O műveletek aszinkron módon.

### Express

Express.js egy framework(Nodejs package) amivel könnyen és gyorsan lehet NodeJS web applikációt, API-t létrehozni. A NodeJS alap http/https moduljait használja fel. Maga az Express sok „opcionális” middlewarel rendelkezik amiket külön lehet telepíteni, vagy saját middlewaret is írhatunk amik kibővítik az API vagy web applikációnak a funkcióit, és mivel ezeket a middlewareket külön lehet megszerezni és nem az Express alap csomagjában jönnek elég lightweight és könnyen testreszabható a web applikációnk vagy API-unk.

### MySQL(Node package)

Ez a Node Package egy driver-t biztosít amivel lehet csatlakozni egy MySQL adatbázishoz(Mi esetünkben egy MariaDB-hez de a MariaDB és a MySQL szinte azonosak), és egy olyan interfacet biztosít amivel könnyen lehet az adatbázis felé lekérdezéseket indítani. Ez nem egy ORM, vagyis az SQL utasításokat nekünk kell teljes egészében megírni és lefuttatni.

Ez a Driver lehetőséget ad több egy idejű csatlakozás biztosítására, hiba kezelésre, adat escapelésre hogy megvédjük szerverünket SQL injectionöktől, és az adatbázisban felmerülő típusokat SELECT query esetén Javascript típusokba kasztolja és egyéni kasztolásra is van lehetőség

### JSON Web Token (JWT)

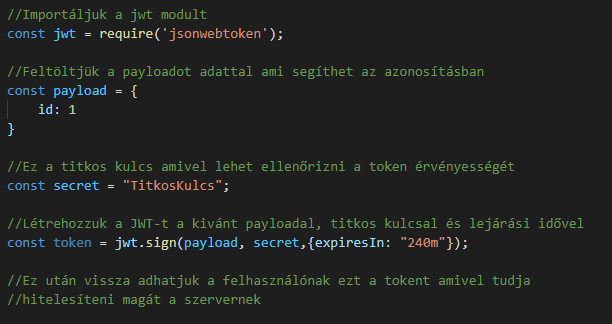
#### JSON Web Token

A JWT módot biztosít arra hogy kettő vagy több fél között biztonságosan tudjon adatokat küldeni. Ez a JWT egy hosszú string ami egy JSON-t tartalmaz amibe bármi féle adat lehet, és lelehet ellenőrizni ennek a tokennek a sértetlenségét, valóságát. Ezt a tokent tovább lehet titkosítani hogy a tartalmát is teljesen biztonságba tudjuk szállítani, mert alapból a JWT csak azt tudja biztosítani hogy ha a tokent megváltoztatták, akkor érvénytelen legyen, de attól még ha valaki megszerzi a tokent, dekódolhatja és a benne lévő adatokat megszerezheti, ezért ha nem használunk további titkosítást csak olyan adatokat küldjünk amikkel a támadó nem tud mit kezdeni.

Nagyon jól lehet használni hitelesítéshez a JWT-t, például ha a felhasználó bejelentkezik, átadunk neki egy ilyen tokent amit a kliens eltárol, és az elkövetkezendő requestekhez hozzá teszi ezt a tokent és a szerveren megvizsgáljuk a tokent hogy érvényes-e, és ha az, akkor a szerver végrehajtja a requestet.

#### Jsonwebtoken(Node Package)

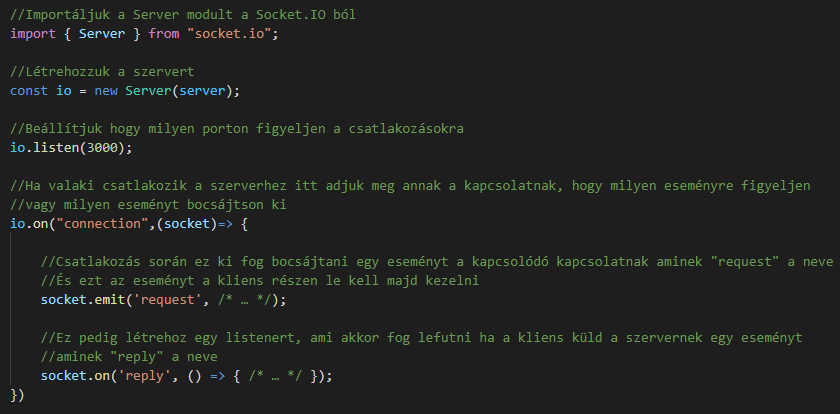
Ez a node package implementálja ezt a JWT technológiát, szóval lehet Javascriptben JWT-ket létrehozni, ellenőrizni, dekódolni. Könnyen belehet állítani a létrehozásnál hogy milyen algoritmust használjon a titkosításra, mikor járjon le az érvényessége, ki állította ki és még mást is.



### Socket Io

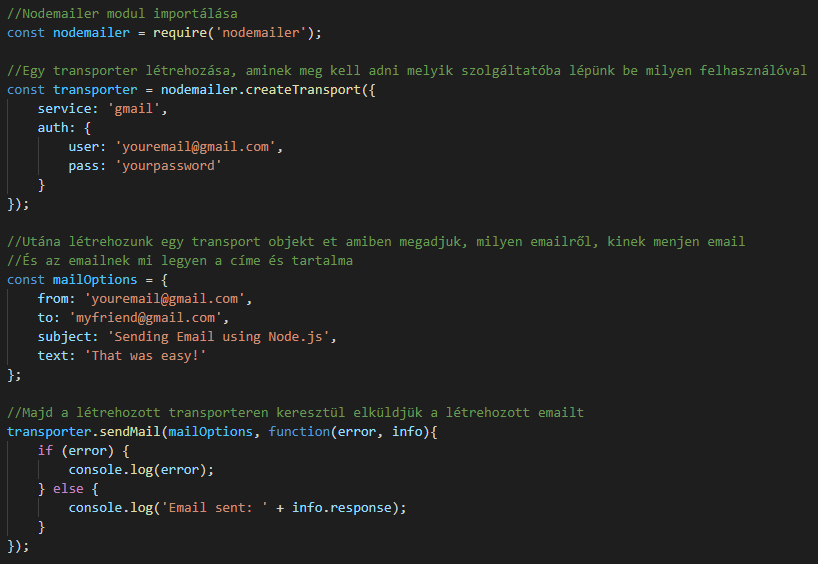
Socket.IO egy alacsony késleltetésű, kétirányú esemény alapú kommunikációs könyvtár ami lehetővé teszi a kliens és szerver közötti valós idejű kommunikációt.

A Socket.IO esemény alapú amit annyit jelent, hogy a szerver és a kliens létrehozott eseményekre figyel, vagy eseményeket bocsájt ki



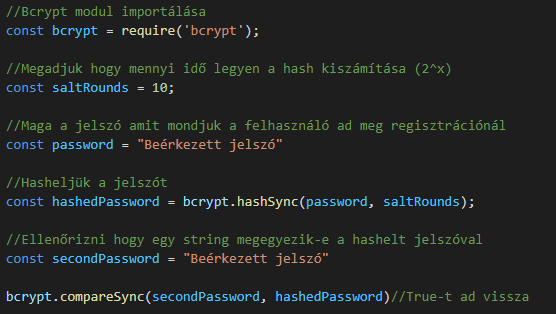
### Nodemailer

A Nodemailer egy olyan modul ami lehetővé teszi az e-mail-ek küldését a szerverről. Lehet saját email szerverről(Simple Mail Transfer Procotol – SMTP-vel) és más email szolgáltatásokkal(például: Gmail) emailt küldeni.



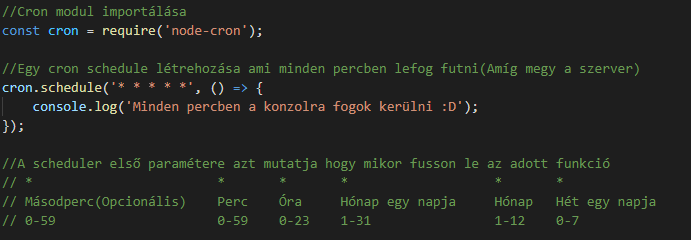
### Bcrypt(Node Package)

Maga a Bcrypt az egy jelszó hashing funkció ami egy elég megbízható titkosítást biztosít a jelszavaknak. Én a bcrypt Javascript implementálását használom.



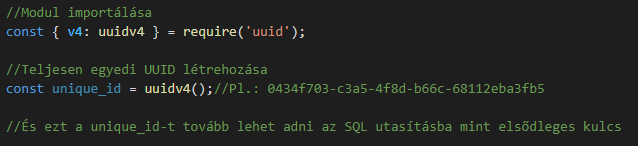
### Node Cron

Node Cron lehetővé teszi hogy bizonyos időpontokban lefuttassunk autómatikusan funkciókat. A modul szintaktikája a GNU operációs rendszer crontabján alapszik, ami mint a Node Cron, lehetővé teszi hogy bizonyos időpontban parancsok, fájlok fussanak le.



### Uuid

A Universal Unique Identifier(UUID) egy nagy valószínűséggel teljesen egyedi azonosító kódot jelent, amit sok különböző tényezőből például a létrehozásnál lévő időpont, hálózati cím segítségével hoz létre egy algoritmus. Én ennek az algoritmusnak a Node.Js beli implementációját használom(uuid Ami a node package nek a neve) az adatbázisban tárolt pár adatnak az azonosításához.



## Adatbázis

### MariaDB

## Frontend

### React

### Redux

### Axios

### Socket Io Client

### Typescript

### Classnames

## Proxy

### http-proxy-middleware

# Specifikáció

## Funkcionális követelmények

## Nem funkcionális követelmények

## Rendszer architektúra

### Kommunikációk rétegek között

### MangaDex API

### Frontend

### Backend

### Adatbázis

### Proxy

# Tesztek

## Backend

## Frontend

# Továbbfejlesztési lehetőségek

# Telepítési útmutató

# Felhasználói dokumentáció

Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | T. De Smedt and W. Daelemans, “Pattern for python.,” *The Journal of Machine Learning Research,* vol. 13, no. 1, pp. 2063-2067, 2012. |
| [2] | „Záróvizsga információk,” [Online]. Available: https://mik.uni-pannon.hu/index.php/hu/oktatas/zarovizsga.html. [Hozzáférés dátuma: 04 03 2022]. |
| [3] | D. J. Wetherall és A. S. Tanenbaum, Computer networks, Pearson Education, 2013. |

Mellékletek

Mappaszerkezet

+chatbot

| backen.bat

| backend.py

| files.doc

| fixedlinks.json

| ipcheck.py

| linkek.json

| linkfix.py

| log.txt

| sqlwriter.py

| textprocessor.py

|

+---backend

| pyvenv.cfg

+---lara

| | entities.py

| | nlp.py

| | parser.py

| | stemmer.py

+---static

| +---css

| | chat.css

| |

| \---js

| chat.js

|

+---templates

| index.html

|

[PÉLDA!!! Megjegyzés: A Python csomagkezelője által telepített fájlok, illetve a különböző cache fájlok a fenti listából kimaradtak, mivel ezekkel indokolatlanul és aránytalanul hosszú lenne a fenti felsorolás. A beadott fájlok között azonban a teljesség kedvéért szerepelnek ezek a fájlok is.]

Ábrajegyzék

[1. ábra: ábrafelirat (ha szükséges, akkor a forrás megjelölésével) [1] 1](#_Toc97890941)

Táblázatjegyzék

[1. táblázat (forrás megjelölésével) [2] 1](#_Toc97890942)