

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Hanicz Tamás

Fájlmegosztó keretrendszer fejlesztése RASPBERRY PI eszközre

Konzulens

Dr. Asztalos Márk

BUDAPEST, 2017

Tartalomjegyzék

[Összefoglaló 5](#_Toc497727978)

[Abstract 6](#_Toc497727979)

[1 Bevezetés 7](#_Toc497727980)

[1.1 Téma ismertetése 7](#_Toc497727981)

[1.2 A rendszerrel szemben támasztott elvárások 7](#_Toc497727982)

[2 Technológiák 9](#_Toc497727983)

[2.1 Frontend 9](#_Toc497727984)

[2.1.1 Angular 9](#_Toc497727985)

[2.1.2 TypeScript 9](#_Toc497727986)

[2.2 Backend 9](#_Toc497727987)

[2.2.1 Flask 9](#_Toc497727988)

[2.2.2 Passlib 9](#_Toc497727989)

[2.2.3 SQLAlchemy 9](#_Toc497727990)

[2.2.4 Pyjwt 10](#_Toc497727991)

[2.2.5 SQLite 10](#_Toc497727992)

[2.2.6 Dropbox 10](#_Toc497727993)

[2.3 Raspberry PI 10](#_Toc497727994)

[2.3.1 OS 10](#_Toc497727995)

[3 Specifikáció 11](#_Toc497727996)

[3.1 Felhasználás 11](#_Toc497727997)

[3.2 Use Case 11](#_Toc497727998)

[4 Tervezés és fejlesztés 12](#_Toc497727999)

[4.1 Fejlesztőkörnyezet bemutatása 12](#_Toc497728000)

[4.2 Adatbázis 12](#_Toc497728001)

[4.2.1 User 13](#_Toc497728002)

[4.2.2 File 14](#_Toc497728003)

[4.2.3 Folder 14](#_Toc497728004)

[4.2.4 File share 15](#_Toc497728005)

[4.2.5 Role 15](#_Toc497728006)

[4.2.6 Credential store 15](#_Toc497728007)

[4.2.7 Log 16](#_Toc497728008)

[4.3 REST API 16](#_Toc497728009)

[4.3.1 Autentikáció 17](#_Toc497728010)

[4.3.2 Adatok ellenőrzése 17](#_Toc497728011)

[4.3.3 UsersAPI 18](#_Toc497728012)

[4.3.4 RolesAPI 18](#_Toc497728013)

[4.3.5 LogsAPI 18](#_Toc497728014)

[4.3.6 NotesAPI 19](#_Toc497728015)

[4.3.7 DropboxAPI 20](#_Toc497728016)

[4.3.8 FilesharesAPI 20](#_Toc497728017)

[4.3.9 FilesAPI 21](#_Toc497728018)

[4.4 Modell 23](#_Toc497728019)

[4.4.1 DropboxModel 23](#_Toc497728020)

[4.4.2 CredentialstoreModel 24](#_Toc497728021)

[4.4.3 UsersModel 24](#_Toc497728022)

[4.4.4 RolesModel 25](#_Toc497728023)

[4.4.5 NotesModel 25](#_Toc497728024)

[4.4.6 LogsModel 26](#_Toc497728025)

[4.4.7 FilesModel 26](#_Toc497728026)

[4.4.8 FilesharesModel 27](#_Toc497728027)

[4.4.9 TokensModel 27](#_Toc497728028)

[4.4.10 Logolás 28](#_Toc497728029)

[4.5 UI 28](#_Toc497728030)

[4.5.1 Bejelentkezés 28](#_Toc497728031)

[4.5.2 Regisztráció 29](#_Toc497728032)

[4.5.3 Menü 30](#_Toc497728033)

[5 Mérések 31](#_Toc497728034)

[6 Összefoglalás 32](#_Toc497728035)

[6.1 Továbbfejlesztés 32](#_Toc497728036)

[7 Irodalomjegyzék 33](#_Toc497728037)

Hallgatói nyilatkozat

Alulírott **Hanicz Tamás**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző, cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2017. 04. 29

...…………………………………………….

Hanicz Tamás

Összefoglaló

Abstract

# Bevezetés

## Téma ismertetése

## A rendszerrel szemben támasztott elvárások

A feladat egy olyan alkalmazás elkészítése, mely a felhasználók által feltöltött fájlokat tárolja és lehetőséget biztosít azok megosztására. A felhasználók bármikor letölthetik, törölhetik és felülírhatják az általuk feltöltött fájlokat. A fájlokhoz a tulajdonosuk engedélye nélkül senki sem férhet hozzá.

A felhasználók a böngészőjük segítségével érhetik el az alkalmazást. Először egy login oldal köszönti őket, ahol a korábban beregisztrált adataikkal tudnak belépni. A még nem regisztrált felhasználóknak itt van lehetőségük regisztrálni. Regisztráció után a megadott e-mail címre érkezik egy megerősítő e-mail, benne egy aktiváló linkkel. A felhasználó accountja csak ezen link meglátogatása után használható. Bejelentkezés során a jelszót háromszor lehet elrontani, utána a rendszer kizárja a felhasználót és csak e-mail-en keresztül lehet jelszót változtatni.

Belépés után a felhasználónak lehetősége nyílik az általa korábban feltöltött fájlokat letölteni, azokat felülírni vagy esetleg újakat feltölteni. Továbbá lehet feljegyzéseket, szövegeket elmenteni az oldalon keresztül egy beépített jegyzettömb segítségével. A fájlok fájlrendszerbe rendezhetők tetszőleges módon. Törlés után 14 napig minden fájl visszaállítható, utána azonban már nincs rá lehetőség. A felhasználók limitált tárolási területtel rendelkeznek. Ez jelenleg 1 GB, ami a későbbiekben, a rendszer bővítését követően nőni fog. Az oldalon elérhető egy logfile, amiben az elmúlt napok történéseit lehet megtekinteni. Minden felhasználó csak a hozzá tartozó fájlokról lát logbejegyzéseket. A fájlok között keresni is lehet.

Egy adott fájl megosztásáról a tulajdonosa rendelkezhet. Tetszés szerint megoszthatja bárkivel egy publikus link, vagy adott felhasználókkal név vagy email segítségével. Valamennyi fájl megosztásához többféle jogosultsági szint rendelhető:

* Read
* Write
* Delete (magában foglalja az előző kettőt is)

Lehetőség van a fájlok „verziókövetett” tárolására is, ami azt jelenti, hogy egy fájl korábbi verziói is elérhetőek, ezt a funkciót azonban külön be kell kapcsolni fájlonként.

A rendszerhez egy felhasználó típus tartozik, ami az egyszerű felhasználót jelöli. Admin jogosultságra nincs szükség, mivel más adataihoz alapvetően nem lehet hozzáférni.

Az alkalmazás képes egy másik, nagyobb fájlmegosztóval szinkronizáltan működni, ami azt jelenti, hogy bizonyos fájlok a felhasználó kérésére feltöltődnek oda is vagy onnan letöltésre kerülnek.

# Technológiák

Ebben a fejezetben a rendszer elkészítéséhez felhasznált technológiák kerülnek bemutatásra, valamint, hogy az egyes könyvtárak, kiegészítők milyen pluszt adtak hozzá a szoftverhez.

## Frontend

### Angular

### TypeScript

## Backend

Az alkalmazás backend-je Python programozási nyelv felhasználásával készült. Az alábbiakban a fejlesztéshez használt framework-ök bemutatása olvasható.

### Flask

A backend alapja egy REST API, mely a Flask felhasználásval készült el. A Flask egy Python alapú webes keretrendszer. Könyvtárakat, technológiákat nyújt ahhoz, hogy minél egyszerűbben és hatékonyabban lehessen webes alkalmazásokat elkészíteni. Önmagában nem nyújt adatbázis absztrakciós réteget, így azt a fejlesztőnek kell megvalósítania, vagy felhasználnia valamilyen kiegészítőt.

### Passlib

A Passlib egy jelszó „hashelő” könyvtár Pythonhoz. Ennek segítségével „hashelem” a jelszavakat és tárolom el a kapott értéket az adatbázisban. A könyvtárat a belépés és regisztráció funkcionalitásoknál használtam fel, ezek a folyamatok a későbbiekben kerülnek bemutatásra.

### SQLAlchemy

Az SQLAlchemy egy Python SQL eszköz, mely ORM tulajdonsággal rendelkezik, nagy segítséget nyújtva ezzel a fejlesztéshez.

### Pyjwt

A Pyjwt egy könyvtár, ami kódol és dekódol JSON web tokeneket. A könyvtár az autentikáció során kerül felhasználásra. Belépéskor a kliens kap egy kódolt token-t, melyet minden kérésnél el kell küldenie a szerver felé.

### SQLite

Az alkalmazás működéséhez elengedhetetlen valamilyen adatbázis használata a felhasználói adatok, logok és a felhasználókhoz tartozó fájlok elérésének tárolásához. A választásom az SQLite-ra esett annak gyorsasága és egyszerűsége miatt.

### Dropbox

A Dropbox lett az általam kiválasztott fájlmegosztó portál, amivel összekötöm az alkalmazást. Az oldal ingyenes verzió esetén 2GB helyet biztosít a felhasználó számára.

## Raspberry PI

Az alkalmazás egy Raspberry Pi 3 B modellen fut. Mivel az erőforrásai végesek, olyan megvalósításra kell törekedni, mely számításba veszi az alábbiakban felsorolt paramétereket.

Az eszköz legfontosabb fizikai paraméterei:

* Quad Core 1.2GHz BCM2837 CPU - 64bit
* 1GB SDRAM
* 16GB-os C10-es microSD kártya (későbbiekben bővíthető külső HDD-vel)
* Raspbian OS

### OS

A Raspberry PI eszközön Raspbian operációs rendszer fut, ami Debian alapú és az eszközre lett optimalizálva [1]. Az OS sok előtelepített szoftverrel rendelkezik és tartozik hozzá grafikus UI.

# Specifikáció

Egy érdekes kérdés ami felmerült, hogy hogyan legyen leképezve a felhasználó által elképzelt mappa rendszer. Az egyik megoldás az, hogy a felhasználónak van egy fő mappája amibe aztán bekerül az összes almappa, de nem hierarchia rendszerben, hanem egyszintű lenne és az adatbázisban tárolt adatok alapján lehetne megállapítani a pontos hierachiát úgy, hogy ebből a felhasználó nem venne észre semmit. Ennek az előnye az, hogy gyorsabb az egyes mappák megtalálása és jobban átlátható, viszont amikor fizikai műveletekre kerül sor, mint például a törlés vagy áthelyezés, akkor nehezebben megvalósítható. A másik megoldás az, hogy pontosan leképezem a hierachiát, így nehezebbé és hosszabbá téve a mappák megtalálását, de ha egy olyat kell törölnöm melynek sok gyereke van, akkor az egy művelettel megoldható fizikailag. Végül a második megoldás mellett döntöttem, mert később, ha a rendszert szeretném továbbfejleszteni akkor ebből még előnyöm származhat.

Ahogy az a technológiáknál látszott a Dropbox a kiválasztott portál amivel szinkronizáltan működik az alkalmazás. Jól dokumentált és egyszerű API-val rendelkezik, ezért esett rá a választásom [2].

## Felhasználás

## Use Case

# Tervezés és fejlesztés

## Fejlesztőkörnyezet bemutatása

## Adatbázis

A rendszer helyes működéséhez elengedhetetlen az adatok perzisztálása. A rendzser adatbázis struktúrája, annak kialakítása, továbbá az egyes táblák szerepe és a közöttük lévő kapcsolatok kerülnek bemutatásra az alábbi fejezetben. Ahogy azt a technológiák fejezetben leírtam, a program SQLite adatbázist használ. Az SQLAlchemy könyvtár felhasználásával elég csak megadnom az adatbázis típusát illetve annak elérését, így később, ha szeretnék más adatbázisra váltani, akkor elég csak az alábbi kódrészletet átírnom. Más módosítást nem igényel.

engine = create\_engine(**'sqlite:///test.db'**, echo=**True**)

A program induláskor a definiált osztályok alapján megnézi, hogy a hozzájuk asszociált táblák valóban léteznek-e az adatbázisban. Amennyiben van olyan tábla amelyik nem létezik, akkor azt létrehozza.



### User

A felhasználók adatait tárolja, hogy a rendszer azonosítani tudja őket felhasználónév – jelszó páros segítségével. Emellett segít megkülönböztetni a még nem aktivált felhasználókat az aktiváltaktól és felfüggesztettektől, továbbá tárolja a rontott jelszavak számát. A jelszó „hashelve” kerül eltárolásra, növelve ezzel a rendszer biztonságát.

Atttribútumok:

* id: felhasználó egyedi azonosítója
* name: választott név
* email: e-mail cím, amire az aktiváló e-mail megy
* password\_hash: jelszó „hashelve”
* activation\_link: a token, amivel a felhasználó aktiválni tudja a regisztrációját.
* created: a létrehozás pontos dátuma.
* failed\_attempts: hibás jelszó próbálkozások
* dropbox\_auth: amennyiben összekötötte dropbox-szal a fiókját, a token itt kerül eltárolásra
* main\_folder: a felhasználó fő mappájának egyedi azonosítója; külső kulcs a folder táblából

### File

A File tábla a felhasználók által feltöltött fájlokról tárol információkat.

Atttribútumok:

* id: a fájl egyedi azonosítója
* user\_id: a fájl tulajdonosa, külső kulcs a User táblából
* file\_name: a felhasználó által választott név
* system\_file\_name: a név ahogy a rendszeren van eltárolva
* created: a fájl létrehozásának dátuma
* public\_link: publikus fájl esetén egy egyedi token
* content: ha ez a fájl egy note akkor van kitöltve
* folder\_id: a fájlt tartalmazó mappa, külső kulcs a Folder mappából
* delete\_date: ha a fájlt kijelölte a tulajdonos törlésre, akkor ide bekerül a törlés ideje
* version: verzikövetés esetén a fájl verziószámát tárolja

### Folder

A mappákat reprezentálja és tárol róluk információt azok azonosítása és megtalálása érdekében.

Atttribútumok:

* id: a mappa egyedi azonosítója
* user\_id: a mappa tulajdonosa, külső kulcs a User táblából
* parent\_folder: a mappa szülő mappájának azonosítója
* folder\_name: a felhasználó által választott mappa név
* path: a mappa fizikai elérésének útvonala
* created: a mappa létrehozásának dátuma
* delete\_date: ha a mappát kijelölte a tulajdonos törlésre, akkor ide bekerül a törlés ideje

### File share

A fájlok megosztásának tárolásáért felelős tábla, melynek minden rekordja egy fájl hozzárendelésre egy új felhasználóhoz valamilyen jogosultsággal.

Atttribútumok:

* id: a megosztás egyedi azonosítója
* file\_id: a fájl azonosítója, külső kulcs a File táblából
* user\_id: a felhasználó azonosítója aki kapja a jogosultságot a fájlra, külső kulcs a User táblából
* role\_id: a jogosultság azonosítója, külső kulcs a Role táblából
* created: a megosztás létrehozásának dátuma

### Role

A rendszerben definiált jogokat tárolja, amelyek segítségével fájlokhoz és note-okhoz hozzá lehet rendelni más felhasználókat. Ahogy a specifikációban említettem három különböző jogosultságot különböztetünk meg : READ, WRITE, DELETE. Ezek a jogok az alkalmazás telepítésekor létre kell, hogy jöjjenek az adatbázisban a gondtalan működés elősegítésére.

Atttribútumok:

* id: egyedi azonosító
* name: a jogosultság szöveges neve
* priority: egy olyan érték, mely meghatározza az adott jogosultság erősségét a későbbi adatbázis műveletek könnyítésére

### Credential store

A program zavartalan működéséhez elengedhetetlen jelszavak tárolására lett létrehozva. A jelszavak az alkalmazás telepítésekor be kell, hogy kerüljenek a táblába, mert addig nem tud elindulni. A működéshez négy jelszóra van szükség:

1. SECRET\_KEY: a token kódolásához és dekódolásához szükséges, ezt az applikációnak is külön át kell adni
2. MAIL: az aktiváló- és reset email küldéséhez szükséges adatokat tartalmazza
3. DROPBOX\_KEY: az applikációhoz tartozó dropbox kulcs
4. DROPBOX\_SECRET: az applikációhoz tartozó dropbox secret

Atttribútumok:

* id: egyedi azonosító
* environment: szöveges név, ami leírja melyik ez a jelszó pontosan
* code: a jelszó maga

### Log

A felhasználói logokat tároló tábla, minden rekordja egy bejegyzés valamilyen felhasználó tevékenységről, fájlműveletről vagy mappaműveletről.

Atttribútumok:

* id: a log bejegyzés egyedi azonosítója
* user\_id: a felhasználó akihez a log tartozik, külső kulcsa a User táblából
* file\_id: a fájl amin műveletet végzett, külső kulcs a File táblából, ha az értéke nem null akkor tudjuk, hogy fájlművelet volt
* folder\_id: a mappa amin műveletet végzett, külső kulcs a Folder táblából, ha az értéke nem null akkor tudjuk, hogy mappát is érintett a művelet
* created: az esemény dátuma
* message: a szöveges bejegyzés

## REST API

A REST API úgynevezett route-okat vagy elérési pontokat tartalmaz, melyek az alkalmazás működéséshez szükséges funkciók elérését biztosítják. Kevés implementációs logikát tartalmaznak, fő feladatuk a felhasználóktól kapott adatok ellenőrzése, autentikáció és a modell megfelelő metódusainak meghívása. Az alkalmazás alapvetően négy külöböző típusú HTTP metódust használ:

* GET : adatok lekérdezése
* POST : új entitás létrehozása
* PUT : entitás módosítása
* DELETE : adatok törlése

### Autentikáció

Sikeres bejelentkezés esetén a rendszer visszaküld egy tokent, ami a kliens oldalon a sütik között kerül eltárolásra, majd később minden kéréssel a szerver felé kerül továbbításra a felhasználó azonosítása céljából.

Néhány route-ot leszámítva, a szerver minden kérés előtt autentikálja a felhasználót, akitől a kérés érkezik. A kérésben a sütik között elküldésre kerül a token is, amit bejelentkezéskor a szerver visszaküldött a felhasználó felé. A szerver először megnézi, hogy ez a token szerepel-e a szerveroldalon tárolt tokenek között, ha igen akkor dekódolja és ellenőrzi érvényességét. Ha minden rendben van, akkor továbbítja a kérést a megfelelő route-ra, ellenkező esetben a válaszban jelzi, hogy problémát talált.

Az felhasználó azonosítása a token segítségével történik, mivel dekódolás után kiolvasható az egyedi azonosító.

### Adatok ellenőrzése

Minden route-ban specifikálva van, hogy milyen adatokat vár a felhasználótól. Egy példa a regisztrációt segítő elérési pontból:

{**'username'**: **None**,

**'password'**: **None**,

**'email'**: **"^[a-zA-Z0-9.\_-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}$"**}

Vár egy felhasználónevet és egy jelszót, amikre nincs megkötés, illetve egy e-mail címet, amire egy reguláris kifejezéssel megadott megkötés van. A felhasználótól kapott bemenetet és ezt a listát átadja egy validációs logikának, ami ellenőrzi, hogy minden paramétert átadott-e a felhasználó és a reguláris kifejezéseknek is megfelel-e. Azt is ellenőrzi, hogy a kapott érték ne legyen üres.

### UsersAPI

Olyan elérési pontokat tartalmaz, ami a felhasználókkal kapcsolatos műveleteket teszi lehetővé.

* /users/login: A felhasználók ezen keresztül tudnak bejelentkezni a rendszerbe. A body-ba belekerül a felhasználónév és jelszó.
* /users/register: Új fehasználókat lehet létrehozni ezen a route-on keresztül. A body-ba a felhasználónév, jelszó és email kerül elküldésre a szerver felé. Sikeres regisztráció esetén kap a felhasználó egy aktiváló emailt a megadott címre.
* /users/activate/<token>: A felhasználó az aktivációs emailben kapott token segítségével itt tudja aktiválni a regisztrációját.
* /users/reset: Ha a felhasználó háromszor elrontja a jelszavát, akkor a rendszer kizárja és küld a megadott email címre egy linket, amely segítségével új jelszót lehet megadni. A body-ba az emailben kapott token és az új jelszó szerepel kétszer. Siker esetén a felhasználó jelszava megváltozik a megadottra.
* /users/delete: A felhasználók ezen keresztül tudják törölni a regisztrációjukat. Ez a törlés nem visszaállítható, minden korábban feltöltött fájl és adat törlődik a felhasználóval együtt.
* /users/change: Email cím és jelszó megváltoztatására van lehetőség ezen a route-on keresztül. A body-ba az új email cím, a jelenlegi és az új jelszó kerül. Az aktuális jelszót mindenképpen meg kell adni, míg az új jelszó és email opcionális és ennek emgfelelően fog változni.
* /users/logout: Lehetőség van kijelentkezni ezen a route-on. A felhasználó egyedi token-e törlődik a sütik közül, valamint a szerveroldalon is.

### RolesAPI

A tárolt fájlok és note-ok megosztásához szükséges jogok kérdezhetők le.

* /roles: Ezen a route-on keresztül kérhetőek le a rendszerben található jogok.

### LogsAPI

Ezeken a route-okon keresztül lehet lekérni a felhasználókhoz tartozó logokat. Van lehetőség csak egy adott fájlra vagy mappára vonatkozó bejegyzéseket lekérni.

* /logs: Ezen a route-on keresztül tudják a felhasználók lekérni a hozzájuk tartozó bejegyzéseket. Minden olyan logot, ami tartalmazza a felhasználó azonosítóját visszaküldi a válaszban. Ezek lehetnek fájlokkal, mappákkal vagy a felhasználó tevékenységeivel (pl.: bejelentkezés) kapcsolatos műveletek.
* /logs/file/<file\_id>: A paraméterben megkapott fájlhoz tartozó összes bejegyzést elküldi a válaszban a felhasználónak. Az adott fájlhoz tartozó logok akkor kérhetőek le, ha a kérést küldő felhasználó a tulajdonos vagy legalább READ joggal rendelkezik a fájlra.
* /logs/folder/<folder\_id>: A fájlokhoz hasonló, de itt a mappákhoz tartozó logok kérhetőek le. Mivel a mappák nem megoszthatóak, ezért ezekhez csak a mappa tulajdonosa fér hozzá.

### NotesAPI

Az alkalmazás képes note-okat is tárolni, melyek nem a fájlrendszerben, hanem az adatbázisban vannak tárolva, hiszen csak szöveges tartalomból állnak. Ezek a note-ok később törölhetőek, módosíthatóak és megoszthatóak is a route-okon keresztül.

* /notes: Az összes olyan note-ot visszaadja amit a felhasználó hozott létre. A válaszba a note-ok tartalma is belekerül. Egy példa válasz JSON-ben:

{

"content": "test",

"created": "Wed, 01 Nov 2017 11:48:29 GMT",

"deleted": null,

"fileName": "test",

"folder": 1,

"id": 9,

"publicLink": null,

"version": 0

}

* /notes/note: Ezen a route-on keresztül tudnak a felhasználók új note-okat létrehozni. A kérésben el kell küldeni az új note nevét és tartalmát. Egy ilyen note maximum 300 karakter hosszú lehet, melyet a szerver a kérés fogadásakor ellenőriz.
* /notes/update: Lehetőség van a note-ok módosítására is, amelyet ez az elérési pont tesz lehetővé. A kérésben el kell küldeni a note azonosítóját valamint az új nevet és tartalmat. Egy note-ot a tulajdonosa vagy legalább a fájlhoz WRITE joggal hozzáférő felhasználó módosíthat. A karakterszámra itt is figyelni kell.
* /notes/<note\_id>: A felhasználók törölni is tudják a már nem használt note-okat. Ezt tudják megtenni ezen a route-on keresztül. Egy note-ot csak a tulajdonosa vagy legalább DELETE joggal rendelkező felhasználó tud törölni. Ha egy olyan felhasználó törli a note-ot, aki nem a tulajdonosa csak hozzáférése van, akkor a tulajdonostól is törlődik.
* /notes/shared: Itt kérhetőek le azok a note-ok egy adott felhasználóhoz, amikhez csak hozzáféréssel rendelkezik de nem ő a tulajdonosa.

### DropboxAPI

Ezeken a route-okon keresztül lehet műveleteket végezni a felhasználó dropbox fiókjával, valamint a fiókok összekapcsolásához szükséges teendők is itt végezhetőek el.

* /dropbox (GET): Visszaadja az autentikációs URL-t amire, a felhasználónak fel kell navigálnia és engedélyeznie kell az applikáció számára a hozzáférést a fiókjához. Sikeres engedélyezés után a felhasználó kap egy hozzáférési tokent.
* /dropbox (POST): A kapott hozzáférési tokent a felhasználó visszaküldi a szerver felé erre a route-ra.
* /dropbox/upload/<file\_id>: Ezen a route-on keresztül a paraméterben átadott fájlt a szerver felölti a felhasználó dropbox fiókjára, amennyiben korábban összekapcsolta az applikációval. A fájl méretére nincs semmilyen megkötés. A fájlt mindig a felhasználó fő mappájába tölti fel.
* /dropbox/download: A felhasználó dropbox fiókjáról letölti a body-ban megadott fájlt.

### FilesharesAPI

A fájlok megosztásához szükséges funkcionalitás érhető el ezeken a route-okon. A hozzáféréseket törölni is lehet utólag.

* /shares/share: Ezen az elérési ponton keresztül lehet jogokat adni más felhasználók számára. A kérés body-ban meg kell adni, hogy melyik fájlt, kihez és milyen hozzáféréssel szeretnénk hozzárendelni. Azt, hogy kinek ad hozzáférést egy email címmel kell azonosítani.
* /shares/public/<file\_id>: Lehetőséget nyújt a paraméterben átadott fájl publikus megosztására. Ezek után a fájl bárki számára elérhető lesz a link ismeretében. Csak a fájl tulajdonosa publikálhatja a fájlt.
* /shares/private/<file\_id>: Az előző pontban bemutatott megosztást törli. Sikeres művelet után a fájl nem lesz többé elérhető a linken keresztül. Csak a fájl tulajdonosa tudja törölni a megosztást.
* /shares/revoke/<id>: Törli a paraméterben megadott megosztást. Ha sikeres a törlés akkor utána már nem férhet hozzá a fájlhoz vagy note-hoz a felhasználó, aki számára meg volt osztva. Csak a fájl tulajdonosa törölheti.
* /shares/<file\_id>: A paraméterben megadott fájlhoz tartozó összes megosztást vissza adja. Csak a fájl tulajdonosa kérdezheti le.

### FilesAPI

A fájlokkal kapcsolatos műveletek végezhetőek el az alábbiakban bemutatott route-okon.

* /files/download/<file\_id>: Ezen az elérési ponton keresztül lehet letölteni a paraméterben megadott fájlt. A fájlokhoz csak akkor férhet hozzá a felhasználó, ha ő a tulajdonosa vagy legalább READ joggal rendelkezik.
* /files/file/<folder\_id> (GET): Visszaadja a paraméterben megadott mappa tartalmát, tehát, hogy fájlokat tartalmaz. A mappa tulajdonosán kívül más nem tudja lekérdezni.
* /files/folder/<folder\_id> (GET): Az előző elérési ponthoz hasonlóan egy a paraméterben megadott mappa tartalmát adja vissza, azonban itt a mappákat adja vissza.
* /files/file/deleted: A felhasználó törölt fájljait adja vissza, melyek még nem lettek törölve a szerverről csak meg vannak jelölve, hogy 14 nap letelte után törlődjenek.
* /files/folder/deleted: A felhasználó törölt mappáit adja vissza. Ugyanazok a feltételek igazak rá, mint a törölt fájlokra.
* /files/file/<file\_id> (DELETE): A paraméterben megadott fájlt törli. A fájlt csak akkor tudja törölni a felhasználó ha ő a tulajdonosa, vagy van legalább DELETE joga a fájlra.
* /files/folder/<folder\_id> (DELETE): Az előző route-hoz hasonlóan itt is törölni lehet, azonban itt egy mappának az azonosítóját kell átadni. Mivel mappát nem lehet megosztani így csak a tulajdonos tudja törölni.
* /files/file/<folder\_id> (POST): Ez a route új fájl feltöltésére szolgál. Paraméterben adja át a felhasználó az új fájl kívánt helyét. A kérés elején a szerver ellenőrzi, hogy a feltöltendő fájlnak mekkora a mérete, ami, ha meghaladná az 1 GB-ot akkor elutasítja a kérést. Azt is ellenőrzi, hogy a fájlt feltöltve meghaladja-e az engedélyezett felhasználói limitet, mely 1 GB-t.
* /files/createFolder: Ezen az elérési ponton keresztül lehet új mappákat létrehozni. A body-ba bekerül az új mappa neve, illetve a szülőjének az azonosítója.
* /files/getPublicFile/<link>: A publikussá tett fájlok ezen a route-on keresztül érhetőek el. A publikus fájlok akkor is letölthetőek, ha a kérést olyan személy indítja akinek nincs regisztrációja.
* /files/search/<file\_name>: A keresés funkció elérését teszi lehetővé. Paraméterben vár egy fájl nevet a felhasználótól, ha talál a feltételeknek megfelelő egy vagy több fájlt, akkor azokat visszaküldi.
* /files/file/move: Ezen a route-on lehet fájlokat mozgatni más mappákba. A kérés body-ba a fájl azonosítója és az új mappa azonosítója kell, hogy bekerüljön. Sikeres művelet esetén visszaküldi a mozgatott fájl adatait az új szülőmappával.
* /files/folder/move: A fájl mozgatáshoz hasonló, azonban itt mappákat lehet áthelyezni máshova. A body-ba az áthelyezni kívánt mappa és az új szülőmappa azonosítója kell, hogy kerüljön. Sikere esetén visszaadja az módosított mappa adatait.
* /files/file/rename: A fájlok átnevezését teszi lehetővé. Body-ban várja a fájl azonosítóját és az új nevet. Sikeres művelet esetén visszatér az új adatokkal.
* /files/folder/rename: Mappa átnevezésére jó, annyiban tér el a fájl átnevezésétől, hogy itt a body-ban a mappa azonosítóját várja.
* /files/folder/list: Visszaadja a felhasználóhoz tartozó összes mappát. A fő mappa Main néven található meg.
* /files/file/restore/<file\_id>: Azok a fájlok amelyeket a tulajdonos vagy valamelyik DELETE joggal rendelkező kijelölt törlésre 14 napig vissza lehet állítani. Ez a route erre ad lehetőséget. Paraméterben kell átadni a fájl azonosítóját, melyet csak a tulajdonos tud visszaállítani.
* /files/folder/restore/<folder\_id>:
* /files/shared: A felhasználó számára lehetőséget nyújt lekérni azokat a fájlokat amikhez van valamilyen jogosultsága. A saját fájljait nem adja vissza.

## Modell

A modell tartalmazza az adatbázis műveleteket és a tényleges logikát.

### DropboxModel

Kéréseket küld a Dropbox hivatalos API-ja felé, melyek segítségével fájlokat lehet le és feltölteni, illetve segít összekapcsolni a felhasználó fiókjait. Az alábbi képen a két fiók összekötésének menete látható:



* auth\_url(): Visszaadja az URL-t, amelyre navigálva a felhasználó engedélyezheti az applikáció számára, hogy hozzáférjen a fájlokhoz.
* auth\_finish(token, user\_id): A felhasználótól kapott token-t ellenőrízve összeköti a két regisztrációt és lementi adatbázisba a hozzáféréshez szükséges kódot, hogy azt ne kelljen minden alkalommal elkérni a felhasználótól.
* get\_access\_token(user\_id): A paraméterben kapott felhasználóhoz tartozó hozzáférési tokent adja vissza.
* upload\_file\_to\_dbx(user\_id, file\_id): Az adott felhasználóhoz tartozó fájlt feltölti dropbox-ra is. Amennyiben van már ilyen nevű fájl akkor azt felülírja. A limitált erőforrások miatt a fájl 100MB-os részletekben kerül feltöltésre.
* download\_from\_dbx(user\_id, input\_dictionary): Letölti dropbox-ról a felhasználó által megadott fájlt a kiválasztott mappába.

### CredentialstoreModel

Hozzáférést biztosít a credential\_store adatbázis táblához. A pogram indulásakor a táblában található összes jelszót betölti környezeti változóként a rendszer, hogy ne kelljen minden alkalommal adatbázis műveletet végezni amikor, pl.: dekódol egy felhasználói tokent.

* get\_code(environment): A paraméterként átadott névhez tartozó jelszót adja vissza.

### UsersModel

A felhasználókkal kapcsolatos műveletek logikáját tartalmazza és az adatbázis elérést.

* login\_user(username, password, ip): Megkeresi az adatbázisban a kapott felhasználónévhez tartozó rekordot. Amennyiben létezik, dekódolja a tárolt jelszót és összehasonlítja a felhasználótól kapottal. A művelet sikerességét logolja, majd visszatér egy új tokennel, ha a megadott jelszó jó volt.
* increment\_bad\_password(user): Amennyiben a felhasználó által megadott jelszó rossz volt, az adatbázisban növeli a hibás próbálkozások számát. Ha ez a szám eléri a hármat, akkor kizárja a rendszerből és a megadott e-mail címre küld egy levelet, aminek a segítségével új jelszót lehet beállítani.
* register\_user(username, user\_password, email): Készít egy új rekordot a User táblában a megadott adatokkal. A jelszót először hash-eli, majd a kapott értéket tárolja el.
* activate\_user(token): A paraméterben kapott token alapján megkeresi az adatbázisban a rekordot és aktiválja azt az activation\_link attribútum null értékbe állításával. Készít egy új mappát az aktivált felhasználó számára ahova a fájljai majd feltöltésre kerülnek.
* reset\_user(token, password): A kapott token alapján azonosítja a rekordot, ahova az új jelszót be kell állítania.
* delete\_user(user\_id): Törli a paraméterben megkapott azonosítójú felhasználót az adatbázisból. Törlés után nem lehet már visszállítani. A felhasználó valamennyi fájlja törlődik.
* change\_user\_data(user\_id, input\_dictionary): Lehetőséget nyújt a felhasználók számára, hogy megváltoztassák az e-mail címüket és a jelszavukat. A paraméterben kapott map-ben megnézi, hogy van-e új jelszó vagy e-mail cím és ha van, akkor azt frissíti az adatbázisban.
* get\_user\_data(user\_id): Visszaad egy user objektumot a paraméterben kapott azonosító alapján.

### RolesModel

A jogosultságok lekérdezéséhez szükséges metódus megvalósításást tartalmazza. Új jogosultság felvételére nincs lehetőség, mivel az alkalmazás működése nem igényli, hisz nincs admin szerepkör.

* get\_all\_roles(): Visszaadja az adatbázisban található összes jogosultságot.

### NotesModel

A felhasználói note-okon végzett műveletek találhatóak benne.

* create\_note(user\_id, input\_dictionary): Létrehoz a paraméterben kapott felhasználónak egy új note-ot. A note címe és tartalma az input\_dictionary-ben található.
* delete\_note(user\_id, note\_id): Törli a paraméterben meghatározott note-ot, amennyiben a user\_id olyan felhasználót jelöl, aki vagy a tulajdonosa a note-nak vagy pedig van rá legalább DELETE jogosultsága.
* update\_note(user\_id, input\_dictionary): Egy note címét és tartalmát lehet módosítani a segítségével, ha a paraméterben átadott felhasználónak legalább WRITE jogosultságva van vagy ő a tulajdonos.
* get\_all\_notes(user\_id): Visszaadja a felhasználóhoz tartozó valamennyi note-ot.
* get\_shared\_with\_me\_notes(user\_id): A felhasználóval megosztott note-okat lehet lekérdezi ebben a metódusban. A függvény visszaad egy note-ot akármilyen jogosultsága is van rá.

### LogsModel

A logoláshoz szükséges műveleteket foglalja magába.

* create\_log\_entry(user\_id, message, file\_id, folder\_id, session):
* get\_user\_entries(user\_id):
* get\_file\_entries(user\_id, file\_id):
* get\_folder\_entries(user\_id, folder\_id):

### FilesModel

A fájlokkal végzett műveletek, letöltés és feltöltés megvalósítása.

* allowed\_file(filename):
* get\_all\_files(user\_id, folder\_id):
* get\_all\_folders(user\_id, folder\_id):
* get\_all\_deleted\_files(user\_id):
* get\_all\_deleted\_folders(user\_id):
* search\_user\_file(user\_id, file\_name):
* upload\_file(user, folder\_id):
* create\_file(user\_id, filename, sys\_fname, folder\_id):
* remove\_file(user\_id, file\_id):
* remove\_folder(user\_id, folder\_id):
* delete\_shares(user\_id, file\_id):
* crt\_folder(user\_id, input\_dictionary):
* rename\_file(user\_id, input\_dictionary):
* rename\_folder(user\_id, input\_dictionary):
* move\_folder(user\_id, input\_dictionary):
* move\_file(user\_id, input\_dictionary):
* delete\_job():
* get\_file\_data(user\_id, file\_id):
* get\_public\_file(public\_link):
* get\_folder\_list(user\_id, file\_id):
* restore\_file(user\_id, file\_id):
* get\_parent\_folder(user\_id, folder\_id):
* restore\_folder(user\_id, folder\_id):
* get\_shared\_with\_user\_files(user\_id):

### FilesharesModel

A fájlok megosztásával kapcsolatos függvények találhatóak benne.

* public\_file(user\_id, file\_id):
* revoke\_public(user\_id, file\_id):
* share\_file(user\_id, input\_dictionary):
* delete\_share(user\_id, share\_id):
* get\_shares(user\_id, file\_id):

### TokensModel

Az azonosítást elősegítő tokenen végzett műveletek megvalósítását tartalmazza.

* encode\_token(id): Készít egy új tokent, amit az alábbi adatok kódolásával hoz létre:

payload = {  
 **'exp'**: datetime.datetime.utcnow() + datetime.timedelta(minutes=60),  
 **'user'**: id  
}

Belekódolja a felhasználó egyedi azonsoítóját, illetve, hogy meddig érvényes.

* decode\_token(token): Dekódolja a kapott tokent és megállapítja érvényességét.
* login\_required(f): Megvizsgálja, hogy melyik route-ra érkezett a kérés és amennyiben autentikáció szükséges, akkor a tokent átadja a fentebb említett függvénynek.

### Logolás

Még nem esett szó a szerver oldalon történő hibák logolásáról, amik a fejlesztés során és a későbbi használat alatt is rendkívül hasznosak tudnak lenni. Mivel az alkalmazás nem igényel komplexebb logolást ezért amellett döntöttem, hogy saját magam által megírt loggert fogok használni. Bevezettem egy osztályt, ami három szintet tartalmaz magában:

1. Info
2. Warning
3. Error

A Flask könyvtár lehetőséget biztosít arra, hogy az alkalmazásban feliratkozzak nem elkapott hibákra vagy, ha a szerver 500-as HTTP kóddal tér vissza, tehát valamilyen hiba történt a kérés kiszolgálása közben. Ebben az esetben as elkapott esemény során a hibát logolom egy szöveges fájlba.

## UI

### Bejelentkezés

A helyes URL beírása után a felhasználók erre az oldalra jutnak először. Ha már van regisztrációjuk korábbról, akkor a felhasználónevük és a jelszavuk megadásával tudnak belépni a rendszerbe. A kérést csak akkor küldi el a szerver felé a kliens, ha mindkét mező ki lett töltve, hiányzó adat esetén az oldal figyelmezteti a felhasználót. A folyamat az Enter billentyű lenyomásával vagy a „Login” gombra kattintva indítható.



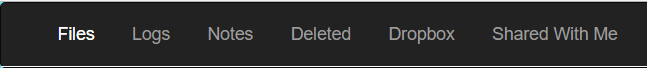
### Regisztráció

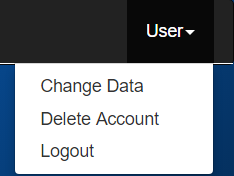
Ha a felhasználónak még nincs regisztrációja akkor a bejelentkező oldalon a „Create an account” feliratra kattintva átnavigálhatnak a regisztrációs oldalra. Itt egy egyedi felhasználónevet és e-mail címet kell megadniuk, továbbá egy választott jelszót. Az oldal ellenőrzi,hogy minden mező ki lett-e töltve, ha igen akkor validálja a bemenetet és ha mindent rendben talál, akkor küldi csak el a szerver felé a kérést. Hiány vagy helytelen adat esetén az oldal figyelmezteti a felhasználót. A folyamat az Enter billentyű lenyomásával vagy a „Register” gombra kattintva is elindítható.



### Menü

Belépés után a menüsor segítségével tud a felhasználó navigálni az alkalamzáson belül.





# Mérések

# Összefoglalás

## Továbbfejlesztés

Ebben a fejezetben azok a még meg nem valósított funkciók kerülnek bemutatásra, amelyek az elkészült alkalmazást még élvezhetőbbé és kényelmesebbé tehetik.

* Cache menedzsment:
* Mappa megosztás: A rendszerben jelenleg csak fájlokat lehet megosztani, azonban a felhasználói élményt nagyban javítaná, ha hasonló feltételekkel mappákat is meg lehetne.

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | „Raspbian OS,” [Online]. Available: http://www.raspbian.org/. [Hozzáférés dátuma: 6 November 2017]. |
| [2] | „Dropbox Python API,” [Online]. Available: http://dropbox-sdk-python.readthedocs.io/en/latest/. [Hozzáférés dátuma: 6 november 2017]. |