

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Hanicz Tamás

Fájlmegosztó keretrendszer fejlesztése RASPBERRY PI eszközre

Konzulens

Dr. Asztalos Márk

BUDAPEST, 2017

Tartalomjegyzék

[Összefoglaló 5](#_Toc497309813)

[Abstract 6](#_Toc497309814)

[1 Bevezetés 7](#_Toc497309815)

[1.1 Téma ismertetése 7](#_Toc497309816)

[1.2 A rendszerrel szemben támasztott elvárások 7](#_Toc497309817)

[2 Specifikáció 9](#_Toc497309818)

[2.1 Felhasználás 9](#_Toc497309819)

[3 Technológiák 10](#_Toc497309820)

[3.1 Frontend 10](#_Toc497309821)

[3.1.1 Angular 10](#_Toc497309822)

[3.1.2 TypeScript 10](#_Toc497309823)

[3.2 Backend 10](#_Toc497309824)

[3.2.1 Flask 10](#_Toc497309825)

[3.2.2 Passlib 10](#_Toc497309826)

[3.2.3 SQLAlchemy 10](#_Toc497309827)

[3.2.4 Pyjwt 11](#_Toc497309828)

[3.2.5 SQLite 11](#_Toc497309829)

[3.2.6 Dropbox 11](#_Toc497309830)

[3.3 Raspberry PI 11](#_Toc497309831)

[3.3.1 OS 11](#_Toc497309832)

[4 Tervezés és fejlesztés 12](#_Toc497309833)

[4.1 Use Case 12](#_Toc497309834)

[4.2 Fejlesztőkörnyezet bemutatása 12](#_Toc497309835)

[4.3 Adatbázis 12](#_Toc497309836)

[4.3.1 User 13](#_Toc497309837)

[4.3.2 File 14](#_Toc497309838)

[4.3.3 Folder 14](#_Toc497309839)

[4.3.4 File share 14](#_Toc497309840)

[4.3.5 Role 14](#_Toc497309841)

[4.3.6 Credential store 14](#_Toc497309842)

[4.3.7 Log 14](#_Toc497309843)

[4.4 REST API 14](#_Toc497309844)

[4.4.1 Autentikáció 15](#_Toc497309845)

[4.4.2 Adatok ellenőrzése 15](#_Toc497309846)

[4.4.3 UsersAPI 15](#_Toc497309847)

[4.4.4 RolesAPI 16](#_Toc497309848)

[4.4.5 LogsAPI 16](#_Toc497309849)

[4.4.6 NotesAPI 17](#_Toc497309850)

[4.4.7 DropboxAPI 18](#_Toc497309851)

[4.4.8 FilesharesAPI 18](#_Toc497309852)

[4.4.9 FilesAPI 19](#_Toc497309853)

[4.5 Modell 19](#_Toc497309854)

[4.5.1 DropboxModel 19](#_Toc497309855)

[4.6 UI 20](#_Toc497309856)

[5 Mérések 21](#_Toc497309857)

[6 Továbbfejlesztés 22](#_Toc497309858)

[7 Irodalomjegyzék 23](#_Toc497309859)

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 megosztja lehetőség szerint. A felhasználók bármikor letölthetik, törölhetik 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 email címre érkezik egy megerősítő email benne egy aktivációs linkkel. A felhasználó accountja csak ezen link meglátogatása után használható. Bejelentkezés során a jelszót háromszor lehet elrontani, utána rendszer kizárja a felhasználót és csak email-en keresztül lehet jelszót változtatni.

Belépés után a felületen az eddig a felhasználó által feltöltött fájlokat lehet letölteni, felülírni vagy esetleg újat feltölteni. Továbbá lehet feljegyzéseket, szövegeket elmenteni az oldalon keresztül egy beépített jegyzettömb segítségével. A fájlok fájlrendszerbe rendezhetők tetszőleges módon. Törlés után 14 napig még minden fájl visszaállítható utána azonban már nincs rá lehetőség. A felhasználók limitált tárolási területtel rendelkeznek. Ez jelenleg 1 GB, a későbbiekben az rendszer bővítése után 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 segítségével vagy pedig adott felhasználókkal név vagy email segítségével. A jegyzettömb segítségével megosztott fájlokat több módon is meg lehet osztani:

* Read
* Write
* Delete (Magában foglalja az előző kettőt)

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 az felhasználó kérésére feltöltődnek oda is vagy pedig onnan letöltésre kerülnek.

# Specifikáció

## Felhasználás

# 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 felhasználásal 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 webes keretrendszer, mely Python alapú. 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 hash-elő könyvtár Pythonhoz. Ennek a segítségével hash-elem 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 az felhasználói adatok, logolás és a felhasználókhoz tartozó fájlok elérésének tárolásához. A választásaom az SQLite-ra esett annak gyorsasága és egyszerűsége miatt.

### Dropbox

## Raspberry PI

Az alkalmazás egy Raspberry Pi 3 B modellen fog futni. Mivel az erőforrásai végesek ezért 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

# Tervezés és fejlesztés

## Use Case

## Fejlesztőkörnyezet bemutatása

## Adatbázis

A rendszer helyes működéséhez elengedhetetlen az adatok perzisztálása. A rendzser adatbázis struktúrája és annak kialakítása kerül bemutatásra az alábbi fejezetben. Továbbá az egyes táblák szerepe és a közöttük lévő kapcsolatok. 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 indulásakor 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 megkülönböztatni a még nem aktivált felhasználókat az aktiváltaktól és felfüggesztettektől, továbbá tárolja a rontott jelszavak számát. A jelszó hash-elve kerül eltárolásra növelve ezzel a rendszer biztonságát.

Atttribútumok:

* id: Felhasználó egyedi azonosítója
* name: Választott név
* email: Email cím amire az aktiváló emial 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-al 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:
* user\_id:
* file\_name:
* system\_file\_name:
* created:
* public\_link:
* content:
* folder\_id:
* delete\_date:
* version:

### Folder

Atttribútumok:

* id:
* user\_id:
* parent\_folder:
* folder\_name:
* path:
* created:
* delete\_date:

### File share

Atttribútumok:

* id:
* file\_id:
* user\_id:
* role\_id:
* created:

### Role

A rendszerben definiált jogokat tárolja, amelyek segítségével fájlokhoz és note-okhoz hozzá lehet rendelni más felhasználókat. Ahogy a specifikációban említve volt 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

Atttribútumok:

* id:
* user\_id:
* file\_id:
* folder\_id:
* created:
* message:

## 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 lesz eltárolva és később minden kéréssel a szerver felé lesz továbbítva a felhasználó azonosítása végett.

Néhány route-ot leszámítva, minden kérés előtt a szerver 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 email 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 registráció 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>:

/files/file/move:

/files/folder/move:

/files/file/rename:

/files/folder/rename:

/files/folder/list:

/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

### 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.

## UI

# Mérések

# Továbbfejlesztés

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

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | http://dropbox-sdk-python.readthedocs.io/en/latest/ |
| [2] |  |
| [3] |  |