



TANSZÉKVEZETŐ

## DIPLOMATERVEZÉSI FELADAT

**Hanicz Tamás**

Mérnökinformatikus hallgató részére

### Fájlmegosztó keretrendszer fejlesztése Raspberry PI rendszerre

A modernizált világban az emberek a dokumentumaikat digitalizált formában tárolják, vagy a saját gépükön vagy valamilyen fájlmosztó oldalon. Az ilyen fájlmosztók nagyon megkönnyíthetik az emberek életét és növelhetik is a biztonságot. Segít abban, hogy a fontos dokumentumok bárhol és bármikor elérhetőek legyenek akár egy mobiltelefon segítségével.

A hallgató feladata egy olyan fájlmosztó keretrendszer tervezése és fejlesztése, mely lehetővé teszi a felhasználók számára a különböző fájlok tárolását és mosztását. A rendszer legyen képes a fájlok verziókövetésére is. A felhasználók egymás között is meg tudják osztani a fájlokat tetszésük szerint.

Az egész rendszer egy Raspberry PI eszközön fog futni Raspbian operációs rendszeren. Az eszköz fizikai tulajdonságai miatt erőforrás takarékos megoldásra kell törekedni. Az alkalmazás szerveroldali logikája Python nyelvben fog elkészülni, míg a felhasználói felület Angular2 alapú lesz.

A hallgató feladatának a következőkre kell kiterjednie:

- Tervezze meg és készítse el a rendszer szerveroldali logikáját úgy, hogy a rendszer Raspberry PI-n legyen képes futni.
- Készítsen egy Angular2 alapú webes kliens alkalmazást.
- Készítsen egy olyan scriptet, mely egy frissen telepített Raspberry-t beüzemel és telepíti az alkalmazást.
- Az alkalmazás tudjon fájlokat tárolni, mosztani és verziókövetést biztosítani.
- Az alkalmazás legyen képes együttműködni valamelyik fájlmosztó portállal.

**Tanszéki konzulens:** Dr. Asztalos Márk

Budapest, 2017. március 05.

Dr. Charaf Hassan  
egyetemi tanár  
tanszékvezető





**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

<b>Összefoglaló .....</b>	<b>6</b>
<b>Abstract.....</b>	<b>7</b>
<b>1 Bevezetés .....</b>	<b>8</b>
<b>2 Technológiák .....</b>	<b>10</b>
2.1 Frontend .....	10
2.2 Backend .....	10
2.3 Dropbox .....	11
2.4 Raspberry PI .....	12
2.4.1 Raspbian.....	14
<b>3 Specifikáció.....</b>	<b>15</b>
3.1 Felhasználási esetek .....	17
<b>4 Tervezés és fejlesztés.....</b>	<b>20</b>
4.1 Architektúra .....	20
4.2 Fejlesztőkörnyezet bemutatása .....	21
4.3 Telepítő script .....	21
4.4 Adatbázis .....	23
4.4.1 User.....	24
4.4.2 File .....	25
4.4.3 Folder .....	26
4.4.4 File share.....	26
4.4.5 Role.....	27
4.4.6 Credential store .....	27
4.4.7 Log .....	28
4.5 Végpontok.....	29
4.5.1 Autentikáció.....	29
4.5.2 Adatok ellenőrzése.....	29
4.5.3 UsersAPI.....	30
4.5.4 RolesAPI.....	31
4.5.5 LogsAPI.....	31
4.5.6 NotesAPI.....	31
4.5.7 DropboxAPI.....	32

4.5.8 FilesharesAPI.....	33
4.5.9 FilesAPI .....	34
4.6 Modell.....	36
4.6.1 DropboxModel.....	36
4.6.2 CredentialstoreModel.....	37
4.6.3 UsersModel.....	37
4.6.4 RolesModel.....	38
4.6.5 NotesModel.....	38
4.6.6 LogsModel.....	39
4.6.7 FilesModel .....	39
4.6.8 FilesharesModel.....	42
4.6.9 TokensModel .....	42
4.6.10 EmailHandler .....	43
4.6.11 Felhasználói limitek.....	43
4.6.12 Logolás.....	44
4.7 UI .....	45
4.7.1 Művelet sikeressége.....	45
4.7.2 Bejelentkezés .....	45
4.7.3 Regisztráció .....	46
4.7.4 Files menü.....	46
4.7.5 Notes menü .....	51
4.7.6 User menü .....	53
<b>5 Mérések.....</b>	<b>56</b>
<b>6 Összefoglalás.....</b>	<b>60</b>
6.1 Továbbfejlesztés .....	60
<b>7 Irodalomjegyzék.....</b>	<b>63</b>

# 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. 12. 08.

.....  
Hanicz Tamás

# Összefoglaló

Manapság nagyon elterjedtek a különböző fájlmegosztó portálok, rengeteg funkcióval rendelkeznek, nagy tárhely áll a felhasználó rendelkezésére és kényelmesen lehet kezelni a feltöltött fájlokat. Ezek a szolgáltatások legtöbbször ingyen is igénybevehetőek, de néhány esetben adott limit feletti tárhelyért fizetni kell. Egy ilyen rendszer követelményeihez tartozik, hogy bármikor elérhető legyen egy egyszerű böngészővel, kényelmesen lehessen használni és a felhasználónak csak regisztrálnia kelljen, ha használni szeretné a szolgáltatást.

Dipomamunkám keretein belül egy olyan fájlmegosztót valósítottam meg, amely valamennyi követelménynek megfelel, egy egyszerű felhasználó összes igényét kielégíti és egyszerűen lehet használni. Fontos megjegyezni azonban, hogy az általam készített fájlmegosztó annyiban eltér az ismertebb portáloktól, hogy elég egy egyszerű és olcsó eszköz az üzemeltetéséhez, továbbá egy privátabb megoldást kínál.

Megismerkedtem új technológiákkal, melyekkel korábban még nem találkoztam és ezeket sikeresen felhasználtam a projekt előrehaladtával. Készítettem egy felhasználói felületet az Angular keretrendszert felhasználva, kiegészítve TypeScript-el. Fejlesztettem hozzá egy REST API-t Python nyelvben, melynek segítségével az adatbázisban található adatok lekérdezhetők és módosíthatók.

Az alkalmazás egy Raspberry PI eszközön fut, így a fejlesztés során figyelembe kellett vennem, hogy limitált erőforrások állnak rendelkezésre a szoftver futása közben.

A dolgozatban leírom az alkalmazás funkcióit, bemutatom a felhasznált technológiákat, részletesen szemléltetem az alkalmazás felépítését, kitérve az adatbázis struktúrájára is.

# Abstract

Nowadays filesharing portals are very popular, they offer a big variety of functions, the users have almost unlimited storage and these portals are easy to use and manage the uploaded files on them. In most cases this service is free, although sometimes it requires some payment if the user wants to have a bigger storage. A filesharing portal has various requirements, it has to be available anytime using only a basic browser, it has to be comfortable to use and the users shouldn't have to do more than a normal registration to use the services of the portal.

In my thesis I created a filesharing portal that fulfills all of the requirements mentioned above and statisfies all the needs of a simple user, furthermore it is easy to use. It is important to note that the filesharing application that I've developed is different from the more popular portals because to host it a cheap and simple hardware is enough, furthermore it offers a more private solution.

During my work I became familiar with new technologies that I haven't used before and I was able to successfully use these technologies in my project. I created a user interface using Angular with the help of TypeScript. I developed a REST API in Python to be able to manipulate and query the data from the database.

The application is running on a Raspberry PI tool so during the development phase I had to pay attention to the fact that there are limited resources when the software is in use.

In my thesis I write about the functions of the application, I introduce the technologies I used, I go into details about the architecture of the application including the structure of the database.

# 1 Bevezetés

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

A felhasználók a böngészőjük segítségével érhetik el és használhatják az alkalmazás valamennyi szolgáltatását, tudnak új regisztrációt készíteni vagy korábban létrehozott fiókjukba belépni. Belépés után a felhasználónak lehetősége nyílik az általa korábban feltöltött fájlokat letölteni, törölni vagy újakat feltölteni, továbbá lehet feljegyzéseket, szövegeket elmenteni az oldalon keresztül egy beépített jegyzetkönyv segítségével. A fájlok tetszőleges módon fájlrendszerbe rendezhetők, valamint keresni is lehet közöttük, illetve a felhasználó saját tevékenységeiről lát logbejegyzéseket.

Egy adott fájl megosztásáról a tulajdonosa rendelkezhet. Tetszés szerint megoszthatja bárkivel egy publikus linkkel, vagy adott felhasználókkal e-mail segítségével. Valamennyi fájl megosztásához többféle jogosultsági szint rendelhető. Lehetőség van a fájlok „verziókövetett” tárolására is, ami azt jelenti, hogy egy fájl korábbi verziói is elérhetőek, ezt a rendszer automatikusan elvégzi minden fájl esetén.

A cél az volt, hogy az alkalmazás képes legyen egy másik, nagyobb fájlmegosztóval szinkronizáltan működni, ami azt jelenti, hogy bizonyos fájlok a felhasználó kérésére feltöltődnek oda is vagy onnan letöltésre kerülnek. Ez azért jó, mert a felhasználó számára fontos fájlok több helyen is elérhetőek lesznek, így csökkentve a veszélyét annak, hogy elveszik a fájl, továbbá a két rendszer közti összekapcsolás kényelmet ad a kliensnek.

Az alkalmazás egy Raspberry PI eszközön fut, úgy, hogy a kérések kiszolgálásáért felelős logikát Python-ban írtam, míg az adatok megjelenítéséért felelős kódrészlethez Angulart használtam fel.

Felmerülhet a kérdés, hogy miért pont Raspberry PI-ra készült a szoftver, és hogy pontosan milyen célt is szolgál. Az ötlet az volt, hogy készítsék egy olyan szoftvert, melynek segítségével személyes fájljaim, feljegyzéseim bárhol is elérhetőek legyenek anélkül, hogy azokat feltölteném egy fájlmegosztóra. Mindenképpen olyan hardvert kerestem, ami bár kis teljesítményű, de egész nap hozzáfér a hálózathoz, képes



folytonosan üzemelni és nem igényel sok erőforrást ehhez. Mivel valamennyi felsorolt kritériumnak megfelel, választásom a Raspberry PI eszközre esett. A szoftver is olyan igényekkel készült, hogy nem kell sok párhuzamos felhasználót egyidejűleg kiszolgálnia, hiszen csak kis számú felhasználója lesz. Az előnye ennek az, hogy a hálózatra kötve az eszköz viszonylag gyors működés mellett nem vesz el sokat a rendelkezésre álló sáv szélességből és természetesen fizikailag sem igényel nagy helyet és csak kevés karbantartást. Az eszköz olcsó és egyszerű, ezek miatt bárki számára elérhető, és bár a fájlok bárholnan letölthetőek, mégis egy privát megoldást kínál. Az alkalmazás pedig otthonról is futtatható személyes felhasználásra, esetleg egy kisebb csoport számára. A hátránya az, hogy egy nagy fájlmegeosztó protállal nem tudja felvenni a versenyt és limitek vannak a rendszerben.

## 2 Technológiák

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

### 2.1 Frontend

A frontend npm, Angular, TypeScript és Bootstrap felhasználásával készült.

npm: Az npm [1] egy csomag menedzselő JavaScripthez. A projektben megadott függőségeket letölti vagy frissíti, így ezekkel a fejlesztés során már nem kell foglalkozni, csak kiadni a megfelelő parancsokat.

TypeScript: A TypeScript [2] ugyanazt a szemantikát és szintaxist használja, mint a JavaScript. A TypeScriptben megírt kód egyszerű JavaScript kódra fordul, ami képes futni bármelyik böngészőben, s többek között lehetőséget nyújt arra, hogy osztályokat és interfészeket használjanak a fejlesztők, sok IDE már kódolás során jelzi a hibákat.

Angular: Az Angular [3] egy keretrendszer, amivel kliens alkalmazásokat lehet létrehozni HTML-ben, felhasználva hozzá JavaScript-et vagy TypeScript-et. A keretrendszer több könyvtárból áll. Segítségével jól átlátható, karbantartható és újrahasznosítható komponenseket lehet létrehozni.

Bootstrap: A Bootstrap [4] egy eszköz, melynek segítségével gyorsan lehet HTML, CSS és JS fejlesztéseket végezni. Nagyban megkönnyíti a munkát, és minimális kóddal lehet szép és tartalmas felhasználói felületet készíteni. Elég csak a fő komponens `<head>` tag-jében megadni a stylesheet és a script elérését, majd ezek segítségével az oldal betöltésekor ezeket is letölti. Létezik hozzá rengeteg példa és előredefiniált elem, amelyeket könnyű átemelni a saját projektekbe, ezzel is gyorsítva a fejlesztést. Természetesen ezek az elemek később személyre szabhatóak.

### 2.2 Backend

Az alkalmazás backend-je Python programozási nyelv felhasználásával készült, amelynek az alapja egy REST API. A REST szolgáltatás azt jelenti, hogy a rendszerben definiált erőforrásokhoz elérési pontokon keresztül lehet hozzáférni, azokat módosítani

vagy törölni. A kommunikáció JSON objektumok segítségével történik, ezek reprezentálják az egyes erőforrásokat.

Flask: A backend alapja tehát egy REST API, mely a Flask [5] felhasználásával 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.

SQLite: Az alkalmazás működéséhez elengedhetetlen valamilyen adatbázis használata a felhasználói adatok, logok és a felhasználókhoz tartozó fájlok elérésének tárolásához. A választásom az SQLite-ra [6] esett annak gyorsasága és egyszerűsége miatt.

A fejlesztés megkönnyítése érdekében felhasználtam külső könyvtárakat, amelyek valamilyen problémára nyújtanak megoldást, vagy egy kényesebb rész implementációját valósítják meg, így azt nekem nem kell leimplementálnom.

Passlib: A Passlib [7] egy jelszó „hashelő” könyvtár Pythonhoz. Ennek segítségével „hashelem” a jelszavakat és tárolom el a kapott értéket az adatbázisban. A könyvtárat a belépés és a regisztráció funkcióknál használtam fel, ezek a folyamatok a későbbiekben kerülnek bemutatásra.

SQLAlchemy: Az SQLAlchemy [8] egy Python SQL eszköz, mely ORM tulajdonsággal rendelkezik, nagy segítséget nyújtva ezzel a fejlesztéshez. Rengeteg adatbázis típushoz nyújt támogatást, nem igényel sok konfigurációt és a lekérdezések megírását is könnyíti.

Pyjwt: A Pyjwt [9] 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.

## 2.3 Dropbox

A Dropbox lett az általam kiválasztott fájlmegosztó portál, amivel összekötöm az alkalmazást, alapvető szolgáltatásokkal rendelkezik, könnyű kezelni és jól érthető API-val rendelkezik. Az oldal ingyenes verzió esetén 2 GB tárhelyet biztosít a felhasználó számára. Regisztrálás után lehet új alkalmazásokat létrehozni Dropbox-on belül, itt meg lehet adni, hogy milyen hozzáférést kérjen a felhasználóktól engedélyezéskor. Az

alkalmazást 500 db felhasználóhoz lehet hozzárendelni amíg fejlesztés állapotban van a szoftver, több beállítást nem igényel.

Python-hoz létezik egy Dropbox-os könyvtár is, amelyet be kell importálni a projektbe ahhoz, hogy fel lehessen használni. Ez a könyvtár előre definiált függvényekkel rendelkezik, amiket meghívva lehet felvenni a kapcsolatot a Dropbox szolgáltatásaival. A függvények leírása és felhasználásukhoz rengeteg információ van az API [10] hivatalos dokumentációjában, ugyanitt példák is találhatóak. Ha egy felhasználó engedélyezte a Dropbox-ához való hozzáférést, akkor egy hozzáférési token segítségével lehet őt azonosítani később, és a nevében kéréseket indítani. A könyvtár tartalmaz egy Dropbox osztályt, amely példányosítása során elkéri a token-t és ezek után, az objektum valamennyi függvénye arra a felhasználóra fog vonatkozni.

Az API természetesen támogatást nyújt fájlok kezelésére is, lehet vele le- és feltölteni is. Letöltéshez meg kell adni, hogy pontosan milyen útvonalon érhető el a fájl a felhasználó mappa rendszerében Dropbox-on. Míg feltöltés során be lehet állítani, hogy felülíródjon-e azonos nevű fájl, hova töltődjön vagy, hogy értesítse-e a felhasználót a változásokról.

## 2.4 Raspberry PI

Az alkalmazás egy Raspberry Pi [11] 3 B modellen fut, amely egy kis méretű számítógép, monitorra csatlakoztatva. Képes bármire, amire egy hétköznapi asztali számítógép vagy laptop képes, lehet vele például böngészők segítségével internetezni, filmeket nézni vagy videójátékokkal tölteni az időt. Ezek mellett még rengeteg lehetőséget nyújt egyedi ötletek megvalósításához, ilyen lehet például egy „okosotthon” megvalósítása. 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
- Raspbian OS



**Kép 1 Raspberry PI eszköz**



**Kép 2 Raspberry PI eszköz tokban**

### **2.4.1 Raspbian**

A Raspberry PI eszközön Raspbian operációs rendszer fut, ami Debian alapú, és az eszközre optimalizált [12]. Az OS több előtelepített szoftverrel is rendelkezik, és tartozik hozzá grafikus UI. Bár az operációs rendszeren van Python, azt frissíteni kell, mivel csak 3.6-os verzióval fut az alkalmazás helyesen. Továbbá fel kell telepíteni az npm package managert is, hogy a frontendhez tartozó komponensek is felkerüljenek a rendszerre.

### 3 Specifikáció

A cél az volt, hogy a szoftvert egy böngésző segítségével lehessen elérni, miután a felhasználó a böngészőbe beírja a helyes URL-t. Az alkalmazás a login oldalra kell dobja őt, és ha még nem rendelkezik korábbról regisztrációval, át kell tudjon navigálni a regisztrációs oldalra, ahol név, e-mail cím és jelszó megadásával tud új fiókot készíteni magának. Regisztráció után kapnia kell a megadott e-mail címre egy aktivációs levelet, benne egy linkkel. A linkre kattintva a rendszer aktiválja őt, és ezek után a login oldalon be tud jelentkezni a korábban választott névvel és jelszóval. A rendszerben a felhasználónév és e-mail cím egyedi kell legyen, csak ilyen feltételekkel tud regisztrálni. Bejelentkezéskor hibás jelszó kétszer adható meg, harmadik alkalommal az alkalmazás kizárja a felhasználót a rendszerből és küld egy e-mailt a tárolt címre, benne egy linkkel, amivel új jelszót tud beállítani. A bejelentkezést követően a felhasználó az oldal valamennyi funkcióját igénybe tudja venni.

A felhasználó képes feltölteni fájlokat, melyeknek a mérete kisebb mint 1 GB, amennyiben van még szabad tárhelye. Minden felhasználó 1 GB tárhellyel rendelkezik. Képes létrehozni új mappákat, ezáltal kialakítani olyan mappa hierarchiát, ami számára megfelelő. A feltöltött fájlok bármikor leölthetők, kereshetők a nevük alapján és megoszthatók más felhasználókkal, megadva az e-mail címüket. Egy fájl megosztásához többféle jogosultsági szint rendelhető:

- Read: olvasás jog
- Write: írás jog
- Delete (magában foglalja az előző kettőt is) : törlés jog

Ezek a megosztások bármikor visszavonhatóak, de csak a tulajdonos által. A fájlok publikussá is tehetőek, ekkor egy egyedi linken keresztül letölthetőek lesznek bárki számára, még a nem regisztráltaknak is, de ez a lehetőség is visszavonható. Lehet mappákat egyben is letölteni, ilyenkor a rendszer egy tömörített zip fájlt készít, és azt küldi el a felhasználó felé. Mappák megosztására azonban nincs lehetőség. A fájlok és mappák átnevezhetők, áthelyezhetők és törölhetők. Törlés esetén a fájl vagy mappa még 14 napig elérhető egy erre a célra létrehozott oldalon, lehetőséget biztosítva arra, hogy ezeket vissza lehessen állítani eredeti helyükre. Egy fájl csak akkor visszaállítható, ha a szülő mappája nincs törölve. Ugyanez vonatkozik a mappákra is, kiegészítve azzal, hogy

az eredeti helyén azóta nem lett létrehozva egy új mappa, azonos névvel. Ebben az esetben először az új mappát át kell nevezni. A felhasználó meg tudja tekinteni a vele megosztott fájlokat is egy külön oldalon, ahol végrehajthat rajtuk műveleteket, a kapott jogosultságnak megfelelően.

A feltöltött fájlokhoz tartozik egy verziószám is, ezt a rendszernek automatikusan követnie kell, mégpedig úgy, hogy ha egy ugyanolyan nevű fájlt feltölt a felhasználó ugyanabba a mappába, akkor a verziószáma eggyel nagyobb lesz, mint az aktuálisan legmagasabb. Átnevezés és áthelyezés esetén is figyelnie kell a rendszernek a verziókat, és ennek megfelelően módosítania azokat. Törlés esetén minden fájlnak csökken eggyel a verziója, amelyeknek a verziószáma nagyobb, mint a törölt fájlé. Ha egy törölt fájlt visszaállít a felhasználó, akkor nem az eredeti verzióját kapja vissza, hanem az aktuális legnagyobbat.

Az oldalon van még lehetőség feljegyzéseket létrehozni, ezek mérete maximum 300 karakter lehet, és tetszőleges szöveg tárolható bennük. Ezek a jegyzetek később módosíthatók, megoszthatók és törölhetők, ugyanolyan feltételekkel mint egy fájl.

A felhasználók szabadon tudják változtatni a jelszavukat vagy e-mail címüket, de továbbra is csak egyedi cím adható meg. A rendszer a felhasználó tevékenységeiről log bejegyzéseket készít, ezeket a logokat a felhasználó bármikor megtekintheti. Tudják törölni a regisztrációjukat, azonban ez nem visszaállítható, tehát törlés után valamennyi fájl véglegesen elveszik. Lehetőség van még kijelentkezésre, ekkor a login oldalra kell visszajutniuk.

Ahogy az a technológiáknál bemutattam, a Dropbox a kiválasztott portál, amivel szinkronizáltan kell együttműködnie az alkalmazásnak. Jól dokumentált és egyszerű API-val rendelkezik, ezért esett rá a választásom [10]. Ide feltölthető a felhasználó bármelyik fájlja, továbbá innen letölthető bármelyik, maximum 100 MB méretű fájl, ezekről a felhasználók szabadon tudnak rendelkezni. Feltöltéshez elég csak kiválasztani a fájlt és jelezni róla, hogy ezt szeretné feltölteni Dropboxra, letöltéshez pedig meg kell adni a fájl elérhetőségét a Dropboxos hierarchiában, valamint a fájl nevét. Ezekhez azonban a felhasználónak először össze kell kötnie a két fiókját, erre az alkalmazásnak kell lehetőséget biztosítania.

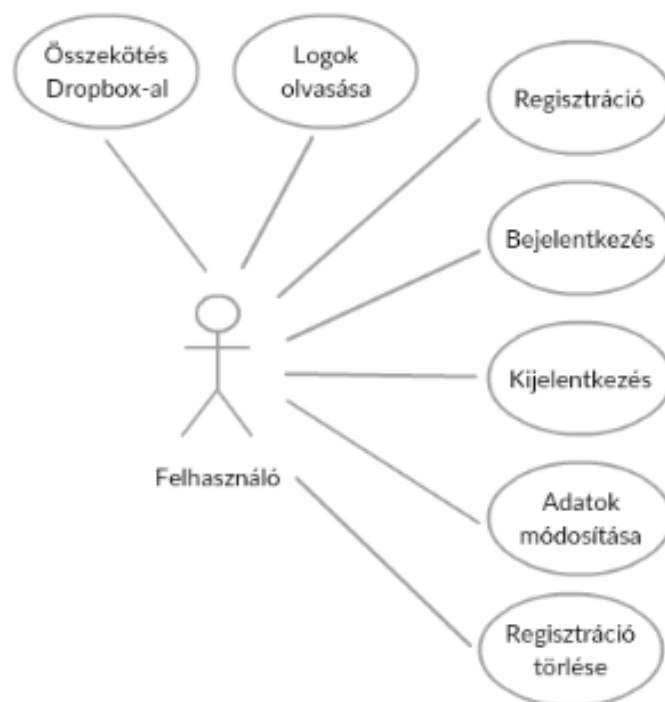


A szoftver mellett található egy telepítő script is, ez egy frissen telepített Raspbian operációs rendszert konfigurál fel úgy, hogy az alkalmazást utána rögtön indítani lehessen.

### 3.1 Felhasználási esetek

A fejezetben a korábban leírt funkciók tekinthetők meg Use Case diagramok formájában, melyek segítenek összefoglalni és vizualizálni azokat. A diagramokat szétbontottam tevékenység csoportokra, hogy átláthatóbb legyen. A rendszerben csak egy szerepkör található ez pedig a felhasználó. Más szerepkörre nincs szükség, hiszen az oldal indítás után semmilyen konfigurációt nem igényel. Az egyedüli tevékenység, amit nem a felhasználók végeznek az a fájlok végleges törlése a rendszerről, erre azonban egy automatizált folyamat van beütemezve.

**Felhasználót érintő tevékenységek:**



**Kép 3 Felhasználói tevékenységek Use Case**

Ide azok a tevékenységek kerültek, melyeket a felhasználó a saját fiókján tud végrehajtani, ezzel módosítva a regisztrációját, továbbá olyan műveletek, melyek a rendszerhez való hozzáférést befolyásolják.

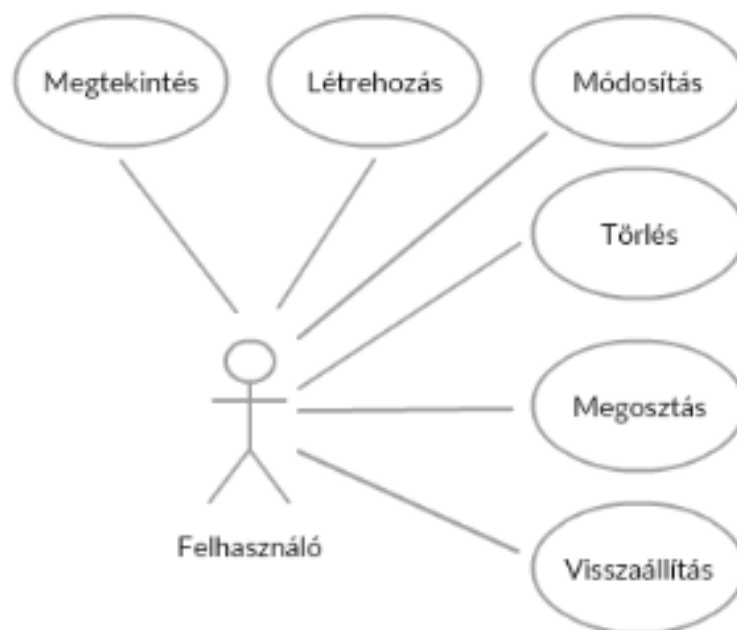
### Fájlokon végzett műveletek:



Kép 4 Fájlokon végzett műveletek Use Case

A diagram bemutatja azokat a tevékenységeket, amelyeket a felhasználó el tud végezni az általa feltöltött és a mások által vele megosztott fájlokon.

### Feljegyzéseken végzett műveletek:



Kép 5 Feljegyzéseken végzett műveletek Use Case

Az utolsó tevékenység csoport a feljegyzéseken elvégzett műveletekből áll. Összefoglalja az összes lehetőséget, amelyet a felhasználó végrehajthat saját, vagy vele megosztott szöveges tartalmokon.

## 4 Tervezés és fejlesztés

A fejezetben bemutatom az alkalmazás felépítését és a megvalósítás módszereit, továbbá a kész rendszer felhasználhatóságát képekkel illusztrálva.

### 4.1 Architektúra

A rendszer két fő kliens-szerver alapon működő részből épül fel, amelyek fejlesztés szempontjából jól elkülöníthetők, ez a két komponens pedig a frontend és backend. A felhasználó csak a felhasználói felületet látja a működés során, és nem tudja, hogy a háttérben pontosan milyen szolgáltatások futnak, valamint, hogy az egyes kérések pontosan hogyan is kerülnek kiszolgálásra.

Ahogy azt korábban említettem, a megjelenítésért felelős kódrész Angular és TypeScript felhasználásával készült. A kód komponensekre van felosztva az átláthatóság és újrafelhasználhatóság miatt. Minden komponens egy .ts fájlból, ami a TypeScript kódot tartamazza, egy .html fájlból, ami a HTML kódot tartalmazza, és egy .css fájlból épül fel, amely a HTML elemek kinézetéért felel. Minden komponensnek van egy külön mappája, amelyben csak a hozzátartozó fájlok vannak. A komponenseken kívül van egy entities mappa is, ahova azok az osztályok kerülnek, amelyek gyakran használt objektumokat definiálnak, továbbá egy services mappa, ahova a backend-del kommunikáló kódrészek kerültek.

A backend felépítése jól felosztható rétegekre, legalul található az adatbázis, ami az adatok tárolására szolgál, ezt a Pythonban írt logika közvetlenül eléri. Felette van egy absztrakciós szint, ami azt jelenti, hogy az adatbázisban található táblák le vannak képezve osztályokra, ez a könnyeb kezelés miatt fontos. Az üzleti logika megvalósítására van egy külön réteg, ahol az adatbáziselérés történik, továbbá ahol a szoftver logikája van implementálva. Az SQLAlchemy-nek köszönhetően az itt található lekérdezések és kódrészletek függetlenek az adatbázis típusától, tehát egy esetleges adatbázis típus váltás nem okozná az itt megvalósított részek változását. Az üzleti logika metódusait a REST API elérési pontjai hívják meg. Ezek az elérési pontok a kapcsolat a frontend és backend között, ellenőrzik a kapott bemenetet, meghívják a megfelelő metódusokat a kérés kiszolgálásához, és válaszolnak a kívánt objektumokkal.

A frontend futtatására Lite szerver [13] használok, amely a fejlesztéshez kiváló, indítás után automatikusan megnyitja a preferált böngészőt és változások esetén frissíti az oldalt. A backend futtatásához a Flask biztosít egy beépített web szerver.

## 4.2 Fejlesztőkörnyezet bemutatása

Az alkalmazást egy asztali számítógépen készítettem és teszteltem. Ezen a gépen Windows operációs rendszer fut, így fejlesztés során olyan rendszerhívásokat kellett használnom Pythonban, amelyek környezet függetlenül lefutnak. Tesztelés céljából fejlesztés közben áttöltöttem a Raspberry PI eszközre is, így ellenőrizni tudtam mindig, hogy a szoftver valóban a vártak megfelelően működik.

A backend fejlesztéséhez a Pycharm [14] nevű fejlesztőeszközt használtam, amely minden igényt kielégítve nyújt támogatást Pythonban írt projektek fejlesztésére. Könnyen kezelhető benne a projekt, a készített alkalmazást egy gombnyomással el lehet indítani, és indítás után láthatom a kimenetet, tehát nem igényel semmilyen parancssori beavatkozást. Az elkészült REST API-t Postman segítségével teszteltem, hogy megfelelően reagál-e a kapott bemenetekre. A Postman [15] egy könnyen kezelhető, átlátható felülettel rendelkező eszköz fejlesztők számára, amely segítségével tesztelni lehet különböző API-kat, minden igényt kielégítve. Könnyű sütiket beállítani, megadni a kéréshez szükséges adatokat.

Az SQLAlchemy egyik tulajdonsága, hogy fejlesztés során beállítható, hogy a végrehajtott lekérdezéseket kiírja a kimenetre, ezzel is gyorsítva a fejlesztés menetét, hiszen így könnyebb leellenőrizni, hogy az elkészített utasítások és lekérdezések valóban helyesek-e. Az adatbázis kezelésére letöltöttem a DB Browser for SQLite [16] nevű programot, így kényelmesen, jól áttekinthető felületen tudtam módosítani az adatokat.

A frontend fejlesztéséhez Visual Studio Code-ot [17] használtam, betölthető bele az egész projekt, ezért könnyű navigálni a fájlok között, és rengeteg plugin tartozik hozzá, ami a fejlesztést megkönnyíti. Emellett tartozik hozzá egy parancssor is, így el lehet indítani belőle az alkalmazást.

## 4.3 Telepítő script

A cél egy frissen telepített Raspbian operációs rendszer felkonfigurálása úgy, hogy az alkalmazás probléma nélkül induljon. Ennek eléréséhez egy Python-ban megírt scriptet készítettem, ami az alábbi parancsokat fogja végrehajtani (a script futása 20 - 30

percet is igénybe vehet a rendszer frissítése és az újabb Python telepítése miatt, szükség van internet elérésre is, továbbá a /home/pi mappából kell indítani, ami változtatható, de az a script átírásával jár):

```
1. sudo apt-get update  
sudo apt-get upgrade
```

Frissíti az operációs rendszerre telepített csomagokat.

```
2. git clone https://github.com/hanicz/dipterv1.git
```

Letölti a projektet Github-ról a git verziókezelő segítségével, mindig itt érhető el a legfrissebb verzió.

```
3. sudo apt-get install sqlite3
```

Feltelepíti az sqlite3 csomagot, hogy az adatbázis parancssoron keresztül is elérhető legyen. Ez nem szükséges a szoftver futtatásához, de kényelmesebb, ha később módosítani kell valamit.

```
4. sudo apt-get install nodejs npm
```

Feltelepíti az npm-et.

```
5. wget https://www.python.org/ftp/python/3.6.0/Python-3.6.0.tgz  
tar xzvf Python-3.6.0.tgz  
./configure --enable-loadable-sqlite-extension  
make  
sudo make install
```

Mivel a backend futásához Python\_3.6-ra van szükség, ezért a script ezt letölti a hivatalos honlapról, majd feltelepíti az sqlite-os kiegészítővel.

```
6. pip3.6 install -r /home/pi/dipterv1/backend/requirements.txt
```

A requirements.txt fájlban található korábban bemutatott könyvtárakat feltelepíti pip segítségével. A pip [18] egy Pythonhoz használt csomag menedzselő, amellyel gyorsan feltelepíthetők a kívánt könyvtárak.

```
7. curl -sL https://deb.nodesource.com/setup\_9.x  
sudo apt-get install nodejs
```

A fentebb látható linken egy script fájl érhető el, amelyet a telepítő lement egy fájlba, majd ezt a scriptet futtatva lefrissíti a nodejs-t, majd telepíti.

```
8. npm install
```

Feltelepíti a frontend-hez szükséges függőségeket, a parancsot a frontend projekt mappájában kell kiadni.

9. Végül a telepítő létrehozza az SQLite adatbázis fájlt, és abban a Role és Credential\_Store táblákat, amelyekbe beszúr pár rekordot, hogy az alkalmazás hiba nélkül tudjon indulni. A Role tábla az előre definiált jogosultságokat tartalmazza, míg a Credential\_Store tábla a rendszer működéséhez elengedhetetlen kódokat. Ezeket a script futása során kéri be a telepítő a scriptet futtató adminisztrátortól. Ezek a táblák az adatbázis fejezetben részletesebben be lesznek mutatva.

## 4.4 Adatbázis

A rendszer helyes működéséhez elengedhetetlen az adatok perzisztálása. A rendszer 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 a kód.

```
engine = create_engine('sqlite:///test.db', echo=True)
```

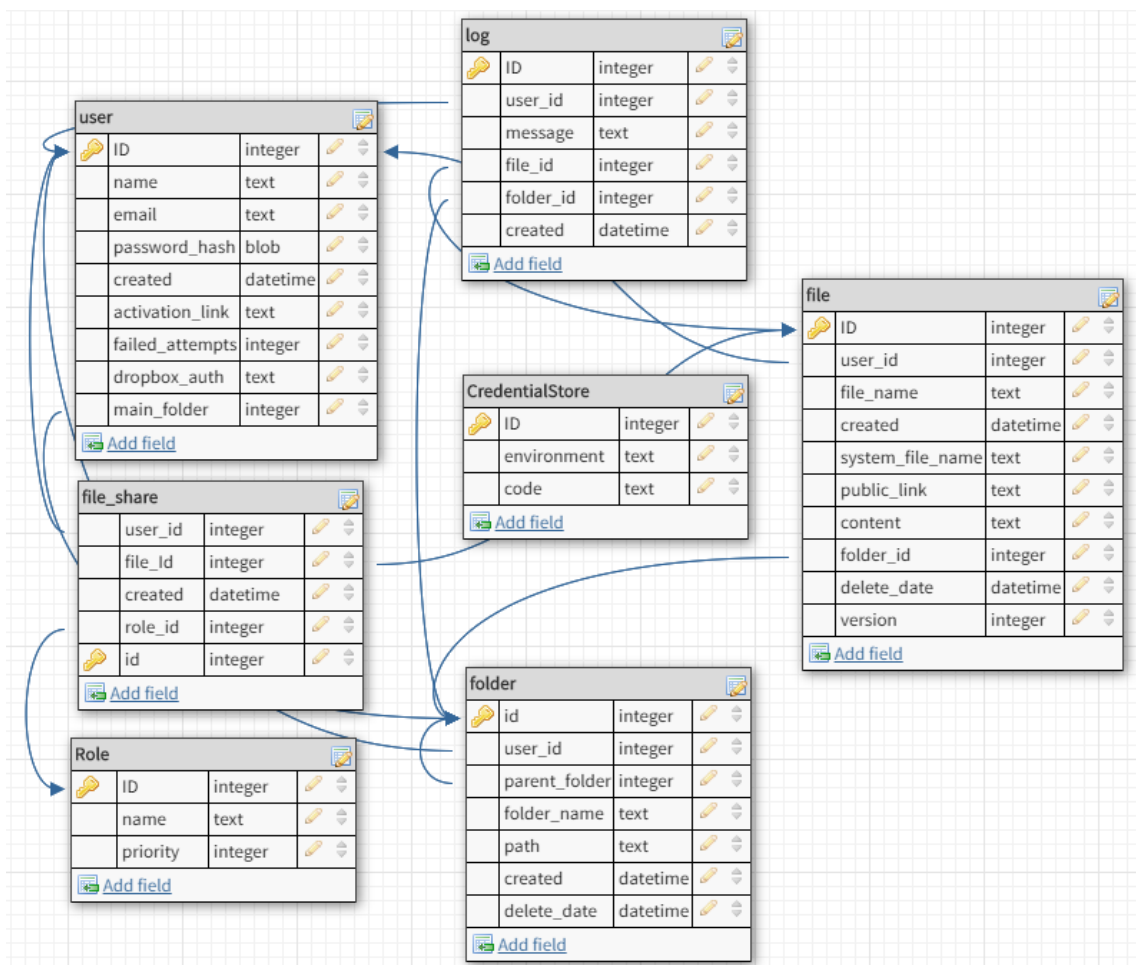
Az adatbázisban a felhasználó azonosításához szükséges adatokat tárolom, az egyes műveletekről logokat, a jogosultságokat, a fájlok eléréséhez szükséges információkat, a fájlmegosztásokat, a rendszer működéséhez szükséges kódokat és a feljegyzéseket.

A felhasználók által feltöltött fájlok fizikailag kerülnek tárolásra a fájlrendszerben, azonban ennek több megvalósítása is lehetséges. Az egyik megoldás az, hogy a felhasználónak van egy fő mappája, amibe bekerül az összes almappa, de nem hierarchia-, hanem egyszintű rendszerben, és az adatbázisban tárolt adatok alapján lehet megállapítani a pontos hierarchiát úgy, hogy ebből a felhasználó nem vesz észre semmit. Ennek az az előnye, 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 hierarchiát, így nehezebbé és hosszabbá téve a mappák megtalálását, de ha egy olyat kell törölnöm, melynek sok gyereke van, akkor az fizikailag egy művelettel megoldható.

Végül a második megoldás mellett döntöttem, mert később, ha a rendszert szeretném továbbfejleszteni, akkor ebből még előnyöm származhat.

A korábban említett feljegyzések könnyen megkülönböztethetők a fájloktól, mivel ezek az adatbázisban vannak eltárolva csak, fizikailag nem jönnek létre fájlok belőlük.

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



Kép 6 Adatbázis struktúra

#### 4.4.1 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.

Név	Leírás
<b>id</b>	felhasználó egyedi azonosítója
<b>name</b>	választott név
<b>email</b>	e-mail cím, amire az e-mailek mennek
<b>password_hash</b>	jelszó „hashelve”
<b>activation_link</b>	a token, amivel a felhasználó aktiválni tudja a regisztrációját
<b>created</b>	a létrehozás pontos dátuma
<b>failed_attempts</b>	hibás jelszó próbálkozások
<b>dropbox_auth</b>	amennyiben összekötötte dropbox-szal a fiókját, a token itt kerül eltárolásra
<b>main_folder</b>	a felhasználó fő mappájának egyedi azonosítója, külső kulcs a folder táblából

**Tábla 1 User tábla**

#### 4.4.2 File

A File tábla a felhasználók által feltöltött fájlokról tárol információkat, hogy könnyítse azok azonosítását, megtalálását és kezelését. Egy bejegyzés valódi fájlt jelöl, ha a content mező értéke üres, míg ha ki van töltve akkor egy feljegyzést.

Név	Leírás
<b>id</b>	fájl egyedi azonosítója
<b>user_id</b>	fájl tulajdonosa, külső kulcs a User táblából
<b>file_name</b>	a felhasználó által választott név
<b>system_file_name</b>	a név, ahogy a rendszeren van eltárolva

<b>created</b>	a fájl létrehozásának dátuma
<b>public_link</b>	publikus fájl esetén egy egyedi token
<b>content</b>	ha ez a fájl egy feljegyzés, akkor van benne érték
<b>folder_id</b>	a fájlt tartalmazó mappa, külső kulcs a Folder mappából
<b>delete_date</b>	ha a fájlt kijelölte a tulajdonos törlésre, akkor ide bekerül a törlés ideje
<b>version</b>	verzikövetés esetén a fájl verziószámát tárolja

**Tábla 2 File tábla**

#### 4.4.3 Folder

A mappákat reprezentálja és tárol róluk információt azok azonosítása és megtalálása érdekében. Fontos megjegyezni, hogy tárol saját magára egy idegen kulcsot is, amely a szülő mappáját jelöli.

Név	Leírás
<b>id</b>	a mappa egyedi azonosítója
<b>user_id</b>	a mappa tulajdonosa, külső kulcs a User táblából
<b>parent_folder</b>	szülő mappa azonosítója
<b>folder_name</b>	a felhasználó által választott mappa név
<b>path</b>	a mappa fizikai elérésének útvonala
<b>created</b>	a mappa létrehozásának dátuma
<b>delete_date</b>	ha a mappát kijelölte a tulajdonos törlésre, akkor ide bekerül a törlés ideje

**Tábla 3 Folder tábla**

#### 4.4.4 File share

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

Név	Leírás
<b>id</b>	a megosztás egyedi azonosítója
<b>file_id</b>	a fájl azonosítója, külső kulcs a File táblából
<b>user_id</b>	a felhasználó azonosítója, aki kapja a jogosultságot a fájlra, külső kulcs a User táblából
<b>role_id</b>	a jogosultság azonosítója, külső kulcs a Role táblából
<b>created</b>	a megosztás létrehozásának dátuma

**Tábla 4 File\_share tábla**

#### 4.4.5 Role

A rendszerben definiált jogokat tárolja, amelyek segítségével fájlokhoz és feljegyzésekhez 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.

Név	Leírás
<b>id</b>	egyedi azonosító
<b>name</b>	a jogosultság szöveges neve
<b>priority</b>	egy olyan érték, mely meghatározza az adott jogosultság erősségét

**Tábla 5 Role tábla**

#### 4.4.6 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 e-mail küldéséhez szükséges adatokat tartalmazza
3. DROPBOX\_KEY: az applikációhoz tartozó Dropbox kulcs
4. DROPBOX\_SECRET: az applikációhoz tartozó Dropbox secret

Név	Leírás
<b>id</b>	egyedi azonosító
<b>environment</b>	szöveges név, ami leírja melyik ez a jelszó pontosan
<b>code</b>	a jelszó maga

**Tábla 6 Credential\_store tábla**

#### 4.4.7 Log

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

Név	Leírás
<b>id</b>	a log bejegyzés egyedi azonosítója
<b>user_id</b>	a felhasználó akihez a log tartozik, külső kulcsa a User táblából
<b>file_id</b>	a fájl amin műveletet végzett, külső kulcs a File táblából, ha az értéke nem null, akkor tudjuk, hogy fájlművelet volt
<b>folder_id</b>	a mappa amin műveletet végzett, külső kulcs a Folder táblából, ha az értéke nem null, akkor tudjuk, hogy mappát is érintett a művelet
<b>created</b>	az esemény dátuma
<b>message</b>	a szöveges bejegyzés

**Tábla 7 Log tábla**

## 4.5 Végpontok

A webes réteg elérési pontokat tartalmaz, melyek az alkalmazás működéséhez szükséges funkciók elérését biztosítják. Kevés implementációs logikát tartalmaznak, fő feladatuk a felhasználóktól kapott adatok ellenőrzése, az autentikáció és a modell megfelelő metódusainak meghívása. A kommunikáció JSON objektumok segítségével történik. Az alkalmazás alapvetően négy különböző típusú HTTP metódust használ:

- GET : entitás lekérdezése
- POST : entitás létrehozása
- PUT : entitás módosítása
- DELETE : entitás törlése

### 4.5.1 Autentikáció

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

Néhány végpontot leszámítva a szerver minden kérés előtt autentikálja a felhasználót, akitől a kérés érkezik. A kérésben a süti 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 elérési pontra, ellenkező esetben a válaszban jelzi, hogy problémát talált.

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

### 4.5.2 Adatok ellenőrzése

Minden végpontban 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, melyekre 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, tehát a példa esetében a username és password mezőknek is kell értéket tartalmazniuk.

### 4.5.3 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. Sikeres bejelentkezés esetén visszaküld egy token a kliens felé, amivel később azonosítani tudja magát. Ezt a token-t a szerver is eltárolja, mégpedig a Flask által biztosított session objektum segítségével. Ennek a tartalma megmarad két újraindítás között is.
- /users/register: Új felhasználókat lehet létrehozni ezen a végponton keresztül. A body-ban a felhasználónév, jelszó és e-mail kerül elküldésre a szerver felé. Sikeres regisztráció esetén kap a felhasználó egy aktiváló e-mailt a megadott címre.
- /users/activate/<token>: A felhasználó az aktivációs e-mailben 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 e-mail címre egy linket, amely segítségével új jelszót lehet megadni. A body-ban az e-mailben kapott token és az új jelszó szerepel. Sikeres esetén a felhasználó jelszava a megadottra változik.
- /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: E-mail cím és jelszó megváltoztatására van lehetőség ezen a végponton keresztül. A body-ba az új e-mail cím, a jelenlegi és az új jelszó kerül.

Az aktuális jelszót mindenképpen meg kell adni, míg az új jelszó és e-mail opcionális és ennek megfelelően fog változni.

- /users/logout: Lehetőség van kijelentkezni ezen a elérési ponton. A felhasználó egyedi tokene törlődik a süti közül, valamint a szerveroldalon is a session objektumból.

#### 4.5.4 RolesAPI

A tárolt fájlok és feljegyzések megosztásához szükséges jogok kérdezhetők le.

- /roles: Ezen a végponton keresztül kérhetők le a rendszerben található jogok.

#### 4.5.5 LogsAPI

Ezek a elérési pontokon keresztül lehet lekérni a felhasználókhoz tartozó logokat. Van lehetőség csak egy adott fájlra vagy mappára vonatkozó bejegyzéseket lekérni.

- /logs: Ezen a végponton 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 kérhetők 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 kérhetők le. Mivel a mappák nem megoszthatóak, ezért ezekhez csak a mappa tulajdonosa fér hozzá.

#### 4.5.6 NotesAPI

Az alkalmazás képes feljegyzéseket is tárolni, melyek nem a fájlrendszerben, hanem az adatbázisban vannak tárolva, hiszen csak szöveges tartalomból állnak. Ezek a feljegyzések később törölhetők, módosíthatók és megoszthatók is a végpontokon keresztül.

- /notes: Az összes olyan feljegyzést visszaadja, amit a felhasználó hozott létre. A válaszba a jegyzetek 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 az elérési ponton keresztül tudnak a felhasználók új feljegyzéseket létrehozni. A kérésben el kell küldeni az új feljegyzés nevét és tartalmát. Egy ilyen jegyzet maximum 300 karakter hosszú lehet, melyet a szerver a kérés fogadásakor ellenőriz.
- /notes/update: Lehetőség van a feljegyzések módosítására is, amelyet ez az elérési pont tesz lehetővé. A kérésben el kell küldeni a jegyzet azonosítóját valamint az új nevet és tartalmat. Csak a tulajdonosa vagy legalább WRITE joggal hozzáférő felhasználó módosíthatja. A karakterszámra itt is figyelni kell.
- /notes/<note\_id>: A felhasználók törölni is tudják a már nem használt feljegyzéseket, ezt tudják megtenni ezen a végponton keresztül. Egy jegyzetet csak a tulajdonosa vagy legalább DELETE joggal rendelkező felhasználó tud törölni. Ha egy olyan felhasználó törli, aki nem a tulajdonosa csak hozzáférése van, a tulajdonostól akkor is törlődik.
- /notes/shared: Itt kérhetőek le azok a feljegyzések egy adott felhasználóhoz, amikhez csak hozzáféréssel rendelkeznek, de nem ő a tulajdonosa.

#### 4.5.7 DropboxAPI

Ezek az elérési pontokon 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 végpontra.
- /dropbox/upload/<file\_id>: Ezen az elérési ponton 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.

#### 4.5.8 FilesharesAPI

A fájlok megosztásához szükséges funkcionalitás érhető el ezeken a végpontokon. 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 e-mail 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 azt követően már nem férhet hozzá a fájlhoz vagy feljegyzéshez a felhasználó, aki számára meg volt osztva. Csak a fájl tulajdonosa törölheti.
- /shares/<file\_id>: A paraméterben megadott fájlhoz tartozó összes megosztást visszaadja. Csak a fájl tulajdonosa kérdezheti le.

## 4.5.9 FilesAPI

A fájlokkal kapcsolatos műveletek végezhetőek el az alábbiakban bemutatott végpontokon.

- /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/download/folder/<folder\_id>: A paraméterben kapott mappát lehet innen letölteni, zip-be összetömörítve.
- /files/file/<folder\_id> (GET): Visszaadja a paraméterben megadott mappa tartalmát, hogy mely fájlokat tartalmazza. 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, de nem a fájlokat, hanem az almappákat.
- /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 végponthoz 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 az elérési pont új fájl feltöltésére szolgál. Paraméterben adja át a felhasználó az új fájl kívánt helyét. A kérés elején a szerver ellenőrzi, hogy a feltöltendő fájlnek mekkora a mérete, ami, ha meghaladná az 1 GB-ot, akkor elutasítja a kérést. Azt is ellenőrzi, hogy a fájl feltöltve meghaladja-e az engedélyezett felhasználói limitet, mely 1 GB-t.

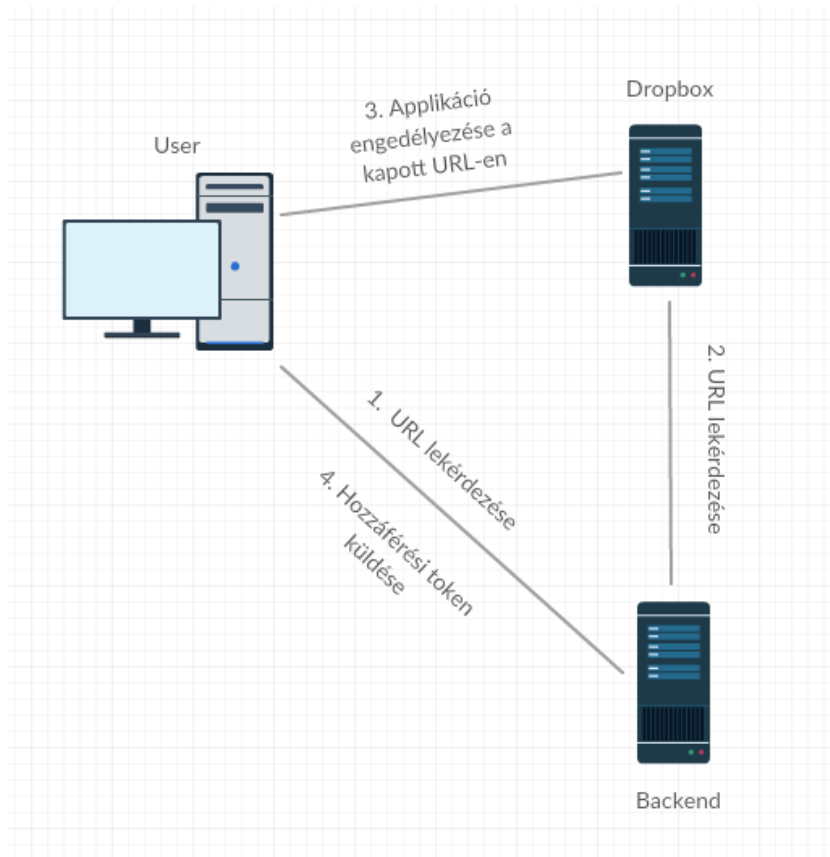
- /files/createFolder: Ezen az elérési ponton keresztül lehet új mappákat létrehozni. A body-ba bekerül az új mappa neve, illetve a szülőjének az azonosítója.
- /files/getPublicFile/<link>: A publikussá tett fájlok ezen a végponton keresztül érhetőek el. A publikus fájlok akkor is letölthetőek, ha a kérést olyan személy indítja, akinek nincs regisztrációja.
- /files/search/<file\_name>: A keresés funkció elérését teszi lehetővé. Paraméterben vár egy fájl nevet a felhasználotól, ha talál a feltételeknek megfelelő egy vagy több fájlt, akkor azokat visszaküldi.
- /files/file/move: Ezen az elérési ponton 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. Siker esetén visszaadja a 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>: Azokat a fájlokat, amelyeket a tulajdonos vagy valamelyik DELETE joggal rendelkező felhasználó kijelölt törlésre, 14 napig vissza lehet állítani. Ez a végpont erre ad lehetőséget. Paraméterben kell átadni a fájl azonosítóját, melyet csak a tulajdonos tud visszaállítani.
- /files/folder/restore/<folder\_id>: Visszaállítja a törlésre kijelölt mappát a tartalmával együtt. Paraméterben kapja meg, hogy melyik mappáról van szó.
- /files/shared: A felhasználó számára lehetőséget nyújt lekérni azokat a fájlokat, amelyekhez van valamilyen jogosultsága. A saját fájljait nem adja vissza.

## 4.6 Modell

A modell tartalmazza az adatbázis műveleteket és a tényleges logikát.

### 4.6.1 DropboxModel

A függvények kéréseket küldenek a Dropbox hivatalos API-já felé, melyek segítségével fájlokat lehet le- és feltölteni, illetve segít összekapcsolni a felhasználó fiókjait. Az alábbi képen a két fiók összekötésének menete látható:



Kép 7 Dropbox engedélyezés

- 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 ellenőrizve 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 a Dropbox-ról a felhasználó által megadott fájlt a kiválasztott mappába. Maximum 100MB-os fájl tölthető le, mert az API csak olyan végpontot biztosít ami a fájlt egyben tölti le. A fájl sikeres azonosítása után indít egy új szálat, ahol a fájlt elkezdheti letölteni, majd visszatér azzal, hogy a letöltés kezdetét vette.

#### 4.6.2 CredentialstoreModel

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

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

#### 4.6.3 UsersModel

A felhasználókkal kapcsolatos műveletek logikáját és az adatbázis elérést tartalmazza.

- login\_user(username, password, ip): Megkeresi az adatbázisban a kapott felhasználónévhez tartozó rekordot. Amennyiben létezik, dekódolja a tárolt jelszót a Passlib segítségével, és összehasonlítja a felhasználótól kapottal. A művelet sikerességét logolja, majd visszatér.
- 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, amelynek 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-el a Passlib könyvtár segítségével, 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. Az új jelszót is paraméterben kapja meg.
- delete\_user(user\_id): Törli a paraméterben megkapott azonosítójú felhasználót az adatbázisból. Törlés után nem lehet már visszaillesztani. 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.

#### 4.6.4 RolesModel

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

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

#### 4.6.5 NotesModel

A felhasználói feljegyzéseken végzett műveletek találhatóak benne.

- create\_note(user\_id, input\_dictionary): Létrehoz a paraméterben kapott felhasználónak egy új feljegyzést. A jegyzet címe és tartalma az input\_dictionary-ben található.
- delete\_note(user\_id, note\_id): Törli a paraméterben meghatározott feljegyzést, amennyiben a user\_id olyan felhasználót jelöl, aki vagy a tulajdonosa a jegyzetnek, vagy pedig van rá legalább DELETE jogosultsága.

- update\_note(user\_id, input\_dictionary): Egy feljegyzés címét és tartalmát lehet módosítani a segítségével, ha a paraméterben átadott felhasználónak legalább WRITE jogosultsága van, vagy ő a tulajdonos.
- get\_all\_notes(user\_id): Visszaadja a felhasználóhoz tartozó valamennyi feljegyzést.
- get\_shared\_with\_me\_notes(user\_id): A felhasználóval megosztott jegyzeteket lehet lekérdezi ebben a metódusban. A függvény visszaad egy feljegyzést, akármilyen jogosultsága is van rá.

#### 4.6.6 LogsModel

A logoláshoz szükséges műveleteket foglalja magába.

- create\_log\_entry(user\_id, message, file\_id, folder\_id, session): Létrehoz egy új bejegyzést a kapott adatokkal. Ha a fájl vagy mappa azonosító üres, akkor NULL érték kerül az adatbázisba.
- get\_user\_entries(user\_id): Visszaadja a paraméterben kapott felhasználóhoz tartozó valamennyi log bejegyzést.
- get\_file\_entries(user\_id, file\_id): Az előzőhöz hasonló, azonban itt egy konkrét fájlhoz tartozó bejegyzéseket ad csak vissza.
- get\_folder\_entries(user\_id, folder\_id): A paraméterben kapott mappa összes bejegyzését visszaadja.

#### 4.6.7 FilesModel

A fájlokkal végzett műveletek, letöltés és feltöltés megvalósítása.

- allowed\_file(filename): Ellenőrzi a fájlnev alapján, hogy a fájl a megengedett kiterjesztések között található-e.
- get\_all\_files(user\_id, folder\_id): A paraméterben kapott mappában található összes fájlt visszaadja, ha azok nincsenek törölve.
- get\_all\_folders(user\_id, folder\_id): Az előző függvényhez hasonló, azonban itt a paraméterben kapott mappában található mappákat adja vissza.
- get\_all\_deleted\_files(user\_id): A felhasználóhoz tartozó összes törölt fájlt visszaadja, hierarchiától függetlenül.

- get\_all\_deleted\_folders(user\_id): A törölt mappákat adja vissza, amelyek a felhasználóhoz tartoznak. A hierarchia itt nem számít.
- search\_user\_file(user\_id, file\_name): A kapott fájlnev részlet alapján lekérdezi a felhasználóhoz tartozó olyan fájlokat, amelyeknek nevében benne van a kapott név részlet. Kis és nagybetű között nincs különbség a keresés során.
- upload\_file(user, folder\_id): Egy új fájl feltöltésére szolgál, először megkeresi a mappát, ahova tölteni kell a fájlt, létrehoz egy teljesen véletlen karakterekből álló nevet a fájlnek, majd létrehoz egy stream-et, amelyen keresztül a fájl részletekben felöltésre kerül. Erre azért van szükség, hogy a Raspberry PI eszköz memória használata ne érje el a fizikai határait.
- create\_file(user\_id, filename, sys\_fname, folder\_id): Az adatbázisban létrehoz egy új rekordot az éppen feltöltésre váró fájlhoz. Ellenőrzi, hogy van-e már ilyen nevű fájl az adott mappában, ha igen akkor a verziószámot növeli eggyel.
- remove\_file(user\_id, file\_id): Kijelöli törlésre a paraméterben kapott fájlt, ami 14 napig még helyreállítható lesz. Törlődik azonban a fájlhoz tartozó összes megosztás, amelyek visszaállítás esetén már nem jönnek vissza. Ha van ugyanilyen nevű fájl nagyobb verziószámmal a mappában, akkor azoknak a verzióját csökkenti eggyel.
- remove\_folder(user\_id, folder\_id): Az adott mappát kijelöli törlésre úgy, hogy a tartalma is törlésre kerül, de 14 napig még visszaállítható. Mappák esetén nincs verziókövetés, továbbá a kijelölt fájlok verziójával nem kell törődni, hisz ugyanabban a mappában vannak.
- delete\_shares(user\_id, file\_id): Törli a fájlhoz tartozó összes megosztást, melyek nem visszaállíthatók.
- create\_folder(user\_id, input\_dictionary): Létrehoz egy új mappát, úgy, hogy előtte ellenőrzi a szülőmappa tartalmát, hogy elkerülje a névazonosságot. Az új mappa neve nem lehet „...” és üres sem. Fizikailag is létrejön egy új mappa a hierarchiának megfelelően.
- rename\_file(user\_id, input\_dictionary): Átnevezi a fájlt a paraméterben kapott adatok szerint, közben figyelve arra, hogy a fájl régi nevével azonos nevű fájlok verziói változzanak, a szóban forgó fájl pedig új verziót kapjon az új név alapján.



- rename\_folder(user\_id, input\_dictionary): Átnevezi a mappát fizikailag és az adatbázisban is, arra figyelve, hogy azonos testvér mappája ne legyen. Frissíteni kell a hierachiában alatta található összes mappa elérési útvonalát is.
- move\_folder(user\_id, input\_dictionary): Áthelyezi a paraméterben kapott mappát a kijelölt helyre, fizikailag és az adatbázisban is, ha ott nincs ilyen nevű mappa. Figyelni kell a hierachiában alatta álló mappák elérési útvonalának megváltoztatására.
- move\_file(user\_id, input\_dictionary): Áthelyezi az adott fájlt a kívánt helyre mind az adatbázisban, mind fizikailag, ügyelve arra, hogy a verziók az új- és régi mappán belül megfelelően változzanak.
- delete\_job(): Ez egy időzített művelet, amely külön szálon fut, úgy, hogy a fő szál nem várakoztatja futás közben. Alapvetően 24 óránként fut le, ez az érték azonban könnyen változtatható. Az adatbázisból lekérdezi az összes olyan mappát és fájlt, amelynek a törlési ideje legalább 14 nap, majd az adatbázisból és fizikailag is törli ezeket. Ez a művelet már nem visszafordítható, tehát az így törölt fájlok végleg elvesztek.
- get\_file\_data(user\_id, file\_id): A paraméterben kapott fájl letöltéséhez szükséges adatokat adja vissza, amelyek a fájl fizikai elérési útvonala, valamint a valódi- és a feltöltés során hozzárendelt fájlnev.
- get\_folder\_data(user\_id, folder\_id): Előkészíti a mappát letöltéshez, majd visszaadja az adatokat hozzá. Először átmásolja a mappát a zip könyvtárba, törli belőle a törlésre kijelölt mappákat és fájlokat, majd átnevezi a fájlokat, hogy azoknak a felhasználó által megadott nevük legyen. Ezek után tömöríti a mappát és törli az átmásolt mappát, hogy ne foglaljon több helyet. Visszatérési értéként átadja az adatokat, amelyek ahhoz kellenek, hogy a zip-et le lehessen tölteni.
- get\_public\_file(public\_link): A paraméterként kapott token alapján azonosítja a fájlt, majd visszaadja az adatokat, amelyek a letöltéshez szükségesek.
- get\_folder\_list(user\_id, file\_id): A felhasználóhoz tartozó valamennyi mappát visszaadja, viszont egy adott mappa nevében a teljes hierarchia benne lesz.

- restore\_file(user\_id, file\_id): Egy törlésre kijelölt fájlt visszaállít eredeti helyére, figyelve arra, hogy a verziószáma nagyobb legyen eggyel, mint az aktuális legnagyobb.
- get\_parent\_folder(user\_id, folder\_id): Visszaadja a paraméterben kapott mappához tartozó szülőmappa adatait.
- restore\_folder(user\_id, folder\_id): Visszaállítja a törlésre kijelölt mappát és tartalmát az eredeti helyére, ha azóta nem lett létrehozva ott azonos nevű mappa.
- get\_shared\_with\_user\_files(user\_id): Az adott felhasználóhoz valamilyen jogosultsággal hozzárendelt fájlok listáját adja vissza.

#### 4.6.8 FilesharesModel

A fájlok megosztásával kapcsolatos függvények találhatók benne.

- public\_file(user\_id, file\_id): A paraméterben megkapott azonosítójú fájlt publikussá teszi, úgy, hogy a fájlhoz tartozó összes megosztást törli először. A fájlhoz generál egy tokenet, amely egyedi azonosítóként fog szolgálni, és ennek a segítségével lehet majd letölteni.
- revoke\_public(user\_id, file\_id): Paraméterben kap egy fájlt, amely ha publikus, akkor azt újra priváttá teszi, ezek után a fájlt már nem lehet letölteni az addigi tokennel.
- share\_file(user\_id, input\_dictionary): Létrehoz egy új megosztást az input\_dictionary-ben definiált adatokkal. Meg kell adni a jogosultságot, a felhasználót, akit meghatalmazunk és a fájl azonosítóját. Egy fájl csak akkor osztható meg, ha nem publikus.
- delete\_share(user\_id, share\_id): Törli a paraméterben megadott megosztást.
- get\_shares(user\_id, file\_id): Visszaadja, hogy az adott fájlhoz milyen megosztások tartoznak. Csak a fájl tulajdonosa kérheti le.

#### 4.6.9 TokensModel

Az azonosítást elősegítő tokenen végzett műveletek megvalósítását tartalmazza.

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

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

Belekódolja a felhasználó egyedi azonosítóját, illetve, hogy meddig érvényes, ami ebben az esetben 60 perc.

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

#### 4.6.10 EmailHandler

Ahogy korábban említettem, aktiváláshoz és elfelejtett jelszó esetén új beállításához a felhasználó kap egy e-mailt az általa megadott címre. A szerveroldal az e-mailt egy e-mail szolgáltatás segítségével küldi. Az működés biztosításra egy gmail<sup>1</sup>-es cím áll rendelkezésre, amelyet a projekt miatt hoztam létre. Minden esetben erről az e-mail címről küldi a leveleket a felhasználóknak a szoftver.

- send\_activate\_email(to address, activate): A környezeti változók közül kiszedi az alkalmazáshoz tartozó gmail-es felhasználónevet és jelszót, majd a Gmail SMTP szerverét felhasználva küld egy aktivációs e-mailt a paraméterben kapott címre. Az e-mail tartalmazni fogja az aktiváláshoz szükséges linket.
- reset\_password\_email(to address, reset): Az előzőhöz hasonlóan megszerzi az e-mail-hez szükséges adatokat, majd küld egy e-mail-t benne egy linkkel, amivel új jelszót lehet beállítani.

#### 4.6.11 Felhasználói limitek

Korábban említettem, hogy a rendszer jelenleg limitált, köszönhetően annak, hogy kevés fizikai memóriával és tényleges tárhellyel rendelkezik a futtató eszköz.

- maximum 1GB-os fájlokat lehet feltölteni annak biztosítása érdekében, hogy több felhasználó is tudja használni az alkalmazást

---

<sup>1</sup> <https://www.google.com/gmail>

- a feljegyzések maximum 300 karakterből állhatnak
- Dropbox-ról maximum 100MB-os fájl tölthető le, mert csak egyben tölthető le a fájl

Ezek betartásáért létrehoztam két függvényt, amelyek a megfelelő végpontok meghívása előtt futnak majd le, és ellenőrzik a paramétereket:

- limit\_content\_length(max\_length): Megnézi a kérés méretét, és ha az nagyobb mint a paraméterben kapott maximális méret, akkor a kérést elutasítja.
- user\_file\_limit(): Ellenőrzi, hogy ha a kérésben kapott fájlt is feltölti a felhasználó, vagy létrehozza a feljegyzést, akkor nem lesz-e nagyobb az össz fájlmérete, mint a megengedett. Amennyiben igen, a kérést elutasítja.

#### 4.6.12 Logolás

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

1. Info
2. Warning
3. Error

Az egyes logüzenetek átadása során megadható annak szintje, ezzel jelezve a fontosságát. A rendszer indulása során be van állítva az a logolási szint, amely legalább szükséges ahhoz, hogy az üzenetet a rendszer kiírja a logfájlba.

A Flask könyvtár lehetőséget biztosít arra, hogy az alkalmazásban értesítést kérjek nem elkapott hibákról, vagy arról, ha a szerver 500-as HTTP kóddal tér vissza, tehát valamilyen hiba történt a kérés kiszolgálása közben. Ebben az esetben az elkapott eseményt Error szinttel adom át a logolást végző függvénynek.

## 4.7 UI

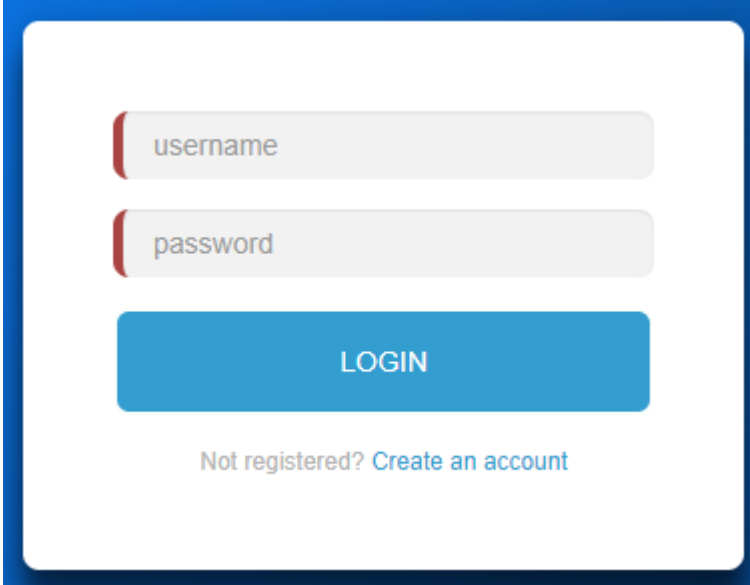
Ebben a fejezetben bemutatásra kerül a felhasználói felület felépítése és működése, képekkel illusztrálva.

### 4.7.1 Művelet sikeressége

A frontend normál működése során sok kérést küld a backend felé, s ezeknek a kéréseknek a sikerességéről tájékoztatni kell a felhasználót is, hogy tudja, az egyes műveletek valóban sikerültek-e. Egy adott művelet után a felhasználó a jobb felső sarokban egy felugró kis zöld ablakot láthat, benne egy rövid üzenettel, hogy sikerült a módosítás, míg sikertelen művelet esetén egy piros ablak jelenik meg, amely a művelet sikertelenségéről tájékoztat.

### 4.7.2 Bejelentkezés

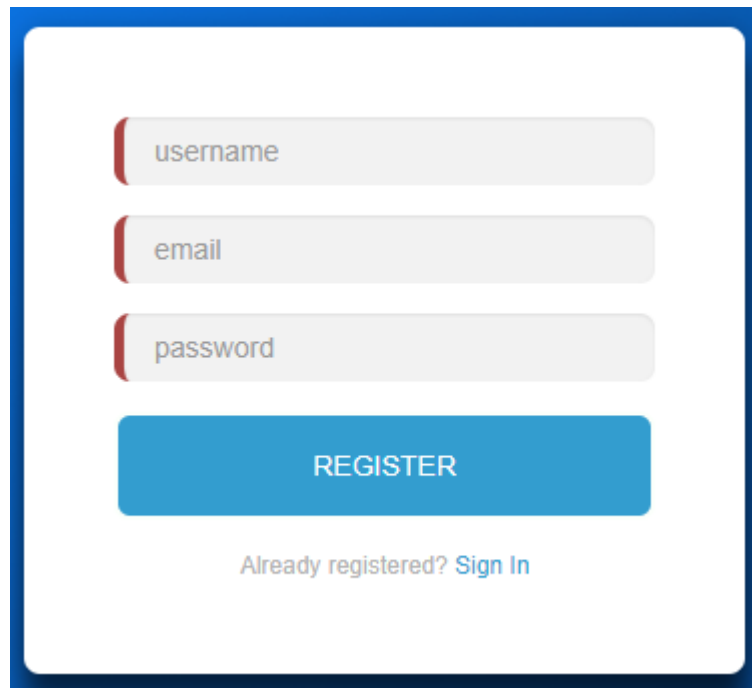
A helyes URL beírása után a felhasználók erre az oldalra jutnak először. Ha már van regisztrációjuk korábbról, akkor a felhasználónevük és a jelszavuk megadásával tudnak belépni a rendszerbe. A kérést csak akkor küldi el a szerver felé a kliens, ha mindkét mező ki lett töltve, hiányzó adat esetén az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

A login form is shown within a blue-bordered container. It features two input fields: the first is labeled 'username' and the second is labeled 'password'. Both fields have a red vertical bar on their left side. Below these fields is a blue button with the text 'LOGIN' in white. At the bottom of the form, there is a link that reads 'Not registered? Create an account'.

Kép 8 Bejelentkezés

### 4.7.3 Regisztráció

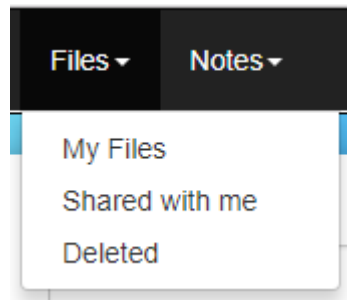
Ha a felhasználónak még nincs regisztrációja, akkor a bejelentkező oldalon a „Create an account” felíratra kattintva átnavigálhatnak a regisztrációs oldalra. Itt egy egyedi felhasználónevet és e-mail címet kell megadniuk, továbbá egy választott jelszót. Az oldal ellenőrzi, hogy minden mező ki lett-e 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.

A registration form interface with a blue border. It contains three input fields labeled 'username', 'email', and 'password' in a light gray font. Below these fields is a blue button with the text 'REGISTER' in white. At the bottom, there is a link that says 'Already registered? Sign In' in a light blue font.

Kép 9 Regisztráció

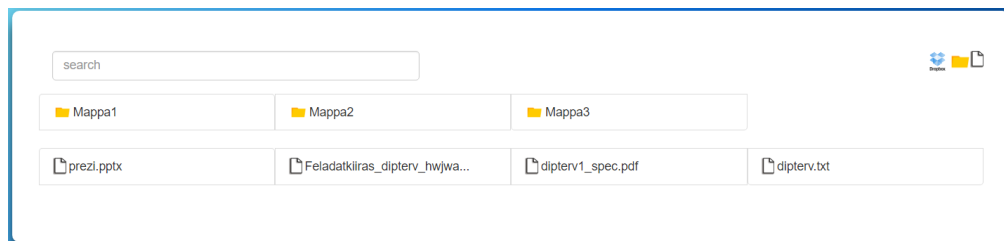
### 4.7.4 Files menü

Belépés után a menüsor segítségével tud a felhasználó navigálni az alkalmazáson belül. A menü három lenyíló ablakot tartalmaz, az első a „Files”, ahol a fájlokkal kapcsolatos dolgok érhetőek el.



**Kép 10 Files menü**

A „My Files” menüpontra kattintva a felhasználó a saját fájljait és mappáit tekintheti meg listába rendezve, ahol felül először a mappák, alattuk pedig a fájlok láthatóak. A hierarchiában lehetőség van lépkedni a kívánt mappára duplán kattintva, ha a felhasználó nem a legfelső fő mappájában van, akkor egy „...” mappa jelöli az előző mappába történő visszajutást. Fájlok kereséséhez a felső mezőbe kell beírni a keresendő szót. A kereső nem tesz különbséget kis és nagybetű között, és mindegy egyes változást érzékelve új kérést küld a szerver felé, tehát a keresés folytonosan történik.

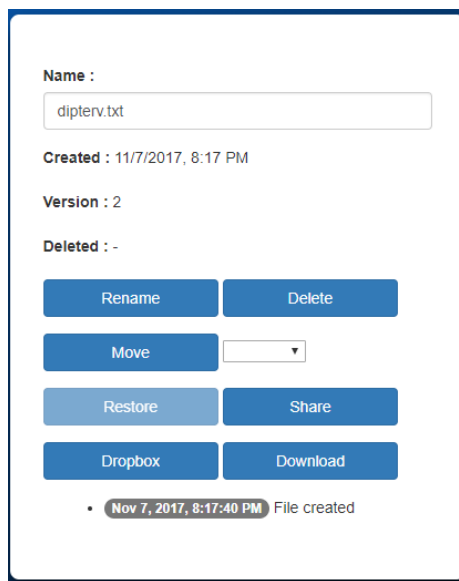


**Kép 11 Felhasználóhoz tartozó fájlok és mappák**

A listában található fájlok és mappák kijelölhetők, melynek segítségével megtekinthetők a részleteik. Egy fájlra kattintva a képernyő jobb oldalán felugrik egy ablak a következő információkkal: név, feltöltés dátuma, verziószám, törlés dátuma és a fájlra vonatkozó logok. Mivel a törölt fájlok oldalán is van lehetőség megtekinteni az ott található fájlok részleteit, ezért a két oldal ugyanazt a komponenst használja erre a funkcióra. A felugró ablakban lehetőség nyílik műveleteket végezni a fájlra:

- A név mező átírása után a „Rename” gombra kattintva a fájl átnevezhető.
- A „Delete” gombra kattintva a fájl törölhető, ha az még nincs törölve.
- A „Move” gomb mellett a lenyíló listából választható egy új mappa, majd a gombra kattintva a fájl áthelyezhető oda.

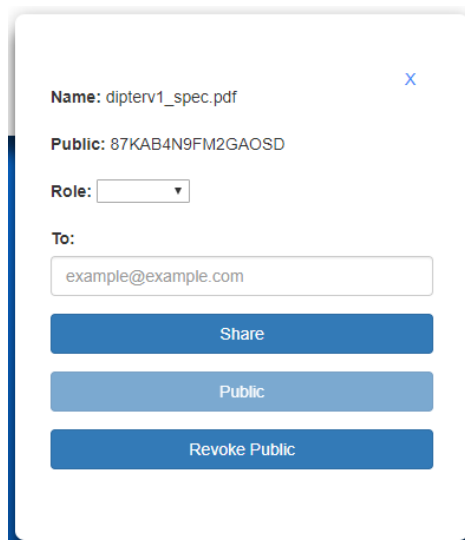
- A „Share” gombra kattintva felugrik egy új ablak, ahol más felhasználókkal lehet megosztani a fájlt.
- A „Restore” gomb segítségével vissza lehet állítani a fájlt, amennyiben az törölve volt.
- A „Download” gomb elindítja a fájl letöltését.
- A „Dropbox” gomb feltölti a felhasználó Dropbox fiókjára a fájlt, ha össze van kötve az alkalmazással.



**Kép 12 Fájl adatok**

Egy kiválasztott fájl megosztásait tehát a „Share” gomb megnyomásával tudjuk kezelni. Az így felugró ablakon láthatjuk a kiválasztott fájl nevét, és ha publikus a fájl, akkor a token-t, amelynek segítségével letölthető. Új jogok úgy adhatók hozzá, hogy először kiválasztunk egy jogosultságot, megadjuk a felhasználót, akinek szánjuk, majd rákattintunk a „Share” gombra az ablakon. Ha a fájl nem publikus, akkor a „Public” gombra kattintva tehetjük azzá, visszavonni pedig a „Revoke Public” gombbal tudjuk. Amennyiben a fájl megosztásra került legalább egy felhasználóval, a gombok alatt megjelenik egy lista, melyben a felhasználó e-mail címe és a jogosultság típusa látható, illetve egy kék „x” amely a jogosultság törlésére szolgál.





Name: dipterv1\_spec.pdf

Public: 87KAB4N9FM2GAOSD

Role:

To:

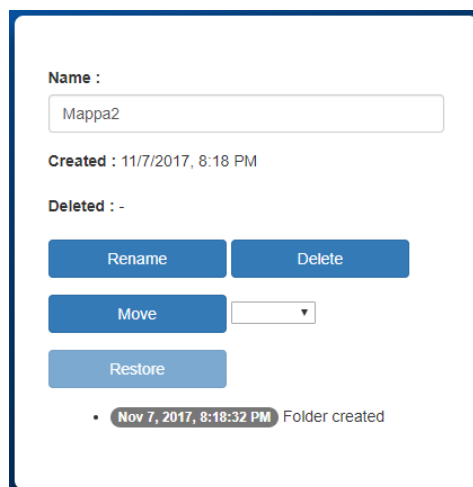
Share

Public

Revoke Public

**Kép 13 Megosztás**

A fájlokhoz hasonlóan tehát megtekinthetőek a mappák részletei is, mint a név, létrehozás- és a törlés dátuma, és a mappához tartozó logok. Itt is fennáll az, hogy ahol a törölt mappák vannak megjelenítve, az az oldal ugyanezt a komponenst használja és a gombok ennek megfelelően vannak engedélyezve és tiltva. Az itteni gombok ugyanazt a funkciót jelölik, mint a fájlokhoz tartozó komponensen.



Name :

Created : 11/7/2017, 8:18 PM

Deleted : -

Rename Delete

Move

Restore

• Nov 7, 2017, 8:18:32 PM Folder created

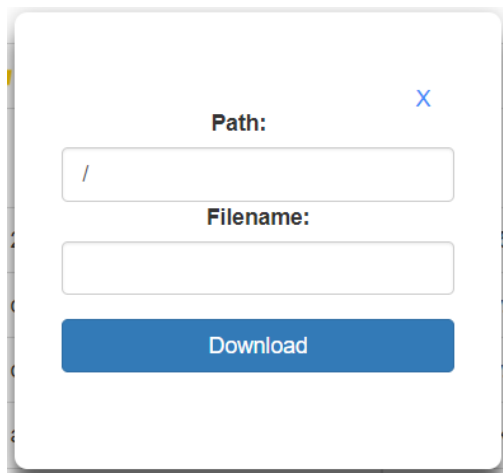
**Kép 14 Mappa adatok**

A keresést segítő mezőtől jobbra három ikon található, egy mappa, egy fájl és egy Dropbox ikon.



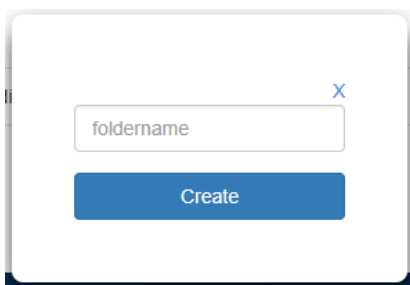
**Kép 15 Művelet ikonok**

A Dropbox ikonra kattintva egy felugró ablakon lehet megadni egy olyan Dropbox-on tárolt fájl adatait, amelyet szeretne a felhasználó letölteni onnan. A „Path” mezőbe kell megadni a fájl elérését, míg a „Filename” mezőbe a fájl nevét. Helyes adatok esetén a fájl átkerül ide is.

A screenshot of a small, white dialog box with a blue 'X' close button in the top right corner. It contains two text input fields. The first field is labeled 'Path:' and contains a single forward slash '/'. The second field is labeled 'Filename:' and is empty. Below these fields is a blue button with the text 'Download' in white.

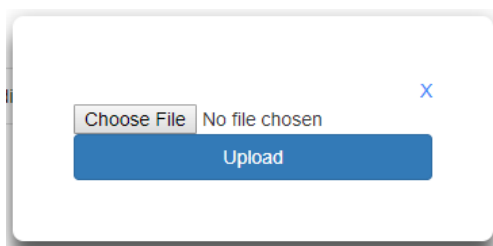
**Kép 16 Letöltés Dropbox-ról**

A mappa ikonra kattintva a felugró ablak segítségével egy új mappa hozható létre ott, ahol a felhasználó éppen tartózkodik.

A screenshot of a small, white dialog box with a blue 'X' close button in the top right corner. It contains a text input field with the placeholder text 'foldername'. Below the field is a blue button with the text 'Create' in white.

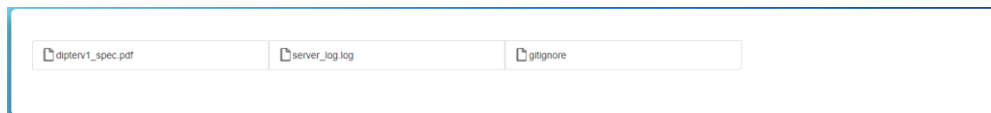
**Kép 17 Mappa létrehozása**

Míg a fájl ikonra kattintva lehet feltölteni új fájlokat a felugró ablakon keresztül, szintén oda, ahol a felhasználó éppen tartózkodik.

A screenshot of a small, white dialog box with a blue 'X' close button in the top right corner. It contains a text input field with the placeholder text 'Choose File' and 'No file chosen'. Below the field is a blue button with the text 'Upload' in white.

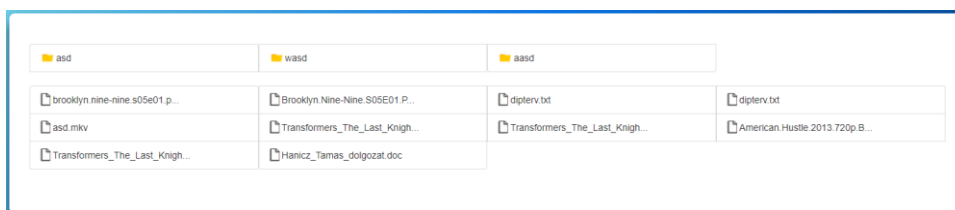
**Kép 18 Fájl feltöltése**

A „Files” menü második pontja a „Shared with me” menüpont, ahol a felhasználó azokat a fájlokat látja amelyeket vele megosztottak. Az egyes fájlok kiválaszthatók, és műveletek végezhetők rajtuk a jogosultságnak megfelelően. Törölni például csak akkor tudja a felhasználó, ha DELETE jogosultsággal rendelkezik.



**Kép 19 A felhasználóval megosztott fájlok**

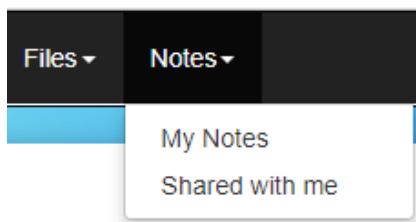
Az utolsó menüpont a lenyíló listában a „Deleted”, ide kerülnek azok a fájlok és mappák, amiket a felhasználó kijelölt törlésre, azonban a 14 nap még nem telt le. Az egyes elemek kijelölésekor megjelenik a korábban bemutatott ablak, amely a fájl részleteit tartalmazza, azonban most a törlést jelölő dátum kitöltött. A fájl vagy mappa a „Restore” gomb megnyomásával visszaállítható korábbi helyére. A 14 nap letelte után a fájl vagy mappa véglegesen törlődik a rendszerből, és nem állítható vissza.



**Kép 20 Törölt fájlok és mappák**

#### 4.7.5 Notes menü

A menüsor második eleme a „Notes”, ahova a feljegyzésekkel kapcsolatos menüpontok kerültek.



**Kép 21 Notes menü**

Az első menüpont a „My Notes”, ahol a felhasználó a saját feljegyzésein tud műveletek végezni, vagy újat tud létrehozni. Felül egy lista látható a feljegyzésekről, amelyben az egyes elemek kiválaszthatók. Kiválasztás után az adott elem adatai jelennek meg. A lehetséges műveletek:

- Ha még nincs kiválasztva egy feljegyzés sem, akkor a név és tartalom megadása után a „New” gombra kattintva létrejön az új jegyzet. Amennyiben ki van választva egy már létező feljegyzés, akkor a „New” gombra kattintva törlődik a kiválasztás, és meg lehet adni az adatait.
- Az „Update” gomb egy kiválasztott feljegyzés adatain módosít megnyomás után.
- A „Delete” gomb segítségével lehet törölni a jegyzeteket.
- A „Share” gomb megnyomásával pedig a már korábban bemutatott megosztást segítő ablak ugrik fel.

The image shows a web application interface for managing notes. At the top, there is a grid of five note thumbnails, each with a document icon and a label: 'note1', 'note2', 'note3', 'note4', and 'note5'. Below this grid is a form section. It starts with a label 'Name' followed by a text input field. To the right of the input field are four blue buttons: 'New', 'Update', 'Delete', and 'Share'. Below the form section is a large, empty rectangular text area for editing the content of a selected note.

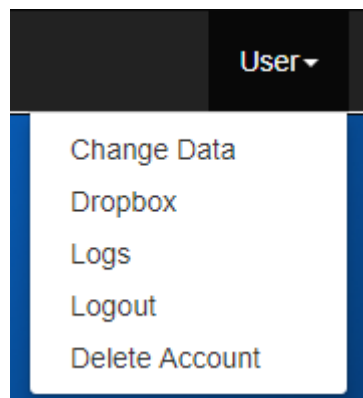
**Kép 22 Felhasználóhoz tartozó feljegyzések**

A második menüpont a „Shared with me” ahol a felhasználóval megosztott feljegyzések tekinthetők meg. Az oldal kialakítása hasonló az előzőhöz, azonban itt csak két műveletre van lehetőség. Az egyik a módosítás az „Update” gombbal, a másik pedig a törlés a „Delete” gombbal. A felhasználó az adott műveletet csak akkor tudja végrehajtani, ha van hozzá megfelelő jogosultsága.

**Kép 23 Felhasználóval megosztott jegyzetek**

#### 4.7.6 User menü

A menüsor utolsó eleme a „User” menü, itt a felhasználóval kapcsolatos műveletek végezhetők el.



**Kép 24 User menü**

Az első menüpont a „Change Data”, amelyre kattintva a felhasználó meg tudja változtatni az e-mail címét és/vagy a jelszavát. A sikeres művelethez mindenképpen meg kell adni a jelenlegi jelszót helyesen.

**Kép 25 Felhasználói adatok megváltoztatása**

A következő menüpont a „Dropbox”, ahol a felhasználó össze tudja kötni a Dropbox-os- és az alkalmazásban lévő fiókját. Az oldalon egy link látható, ahova átnavigálva be kell lépni a Dropbox-os regisztrációba, majd ott engedélyezni lehet az alkalmazás számára a hozzáférést. A művelet végén a felhasználó kap egy hozzáférési tokenet, amelyet az oldalra be kell másolnia, majd meg kell nyomni az „Authorize” gombot, melyek után a két regisztráció összekapcsolódik.

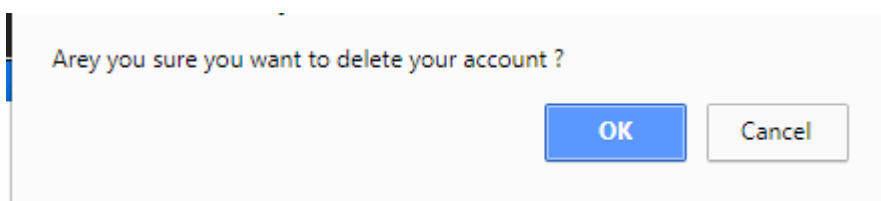
**Kép 26 Dropbox engedélyezés megvalósítása**

A harmadik menüpont a lenyíló listában a „Logs”, ahol az összes, felhasználóhoz kapcsolódó log bejegyzés megtekinthető. Itt lehetnek sikeres és sikertelen bejelentkezési kísérletek, vagy a felhasználóhoz tartozó mappákon és fájlokon végzett műveletek. A bejegyzések nem módosíthatók és nem törölhetők.

Nov 10, 2017, 10:18:09 PM	User logged in with IP: 127.0.0.1
Nov 9, 2017, 7:55:11 PM	gitignore READ role on file for: test@tesd1d21t.hu
Nov 9, 2017, 7:55:04 PM	server_log.log DELETE role on file for: test@tesd1d21t.hu
Nov 9, 2017, 7:54:30 PM	gitignore File created
Nov 9, 2017, 7:54:24 PM	server_log.log File created
Nov 9, 2017, 7:54:09 PM	User logged in with IP: 127.0.0.1
Nov 9, 2017, 7:48:30 PM	dipterv1_spec.pdf WRITE role on file for: test@tesd1d21t.hu
Nov 9, 2017, 7:48:07 PM	dipterv1_spec.pdf File not public anymore
Nov 9, 2017, 7:40:29 PM	dipterv1_spec.pdf File made public
Nov 9, 2017, 7:38:08 PM	dipterv1_spec.pdf WRITE role on file for: test@tesd1d21t.hu
Nov 9, 2017, 7:36:38 PM	dipterv1_spec.pdf File created
Nov 9, 2017, 7:36:28 PM	User logged in with IP: 127.0.0.1
Nov 7, 2017, 8:09:55 PM	tesawd DELETE role on file for: test@tesd1d21t.hu
Nov 7, 2017, 8:09:19 PM	New note created
Nov 7, 2017, 8:09:11 PM	User logged in with IP: 127.0.0.1
Nov 6, 2017, 10:06:45 PM	testqaw DELETE role on file for: test@tesd1d21t.hu
Nov 6, 2017, 10:06:31 PM	User logged in with IP: 127.0.0.1
Nov 6, 2017, 10:05:52 PM	testqaw WRITE role on file for: test@tesd1d21t.hu
Nov 6, 2017, 10:04:50 PM	User logged in with IP: 127.0.0.1
Nov 6, 2017, 9:58:10 PM	testqaw READ role on file for: test@tesd1d21t.hu
Nov 6, 2017, 9:57:46 PM	New note created
Nov 6, 2017, 9:57:39 PM	User logged in with IP: 127.0.0.1
Nov 6, 2017, 9:57:35 PM	User activated

**Kép 27 Felhasználói logok**

Az utolsó előtti menüpont a „Logout”, amelyre kattintva a felhasználó kijelentkezik a rendszerből és visszakerül a bejelentkező oldalra. Az utolsó pedig a „Delete Account”, ahol törölhető a regisztráció. Kattintás után felugrik egy megerősítő ablak, ahol az „OK”-ra kattintva a felhasználó fiókja véglegesen törlődik, a művelet nem visszaállítható.



**Kép 28 Regisztráció törlésének megerősítése**

## 5 Mérések

A fejezetben olyan mérések eredményét és összehasonlítását fogom bemutatni, amelyeket először elvégeztem az asztali gépen, ahol a fejlesztés történt, majd a Raspberry PI eszközön is. A kéréseket egy harmadik gépről indítottam. A cél bemutatni, hogy milyen teljesítmény csökkenéssel jár a kérések kiszolgálása a lassabb eszközön. Ahogy említettem az alkalmazás Windows és Raspbian operációs rendszeren is tud futni.

Az erősebb gép lényeges specifikációi:

- RAM: 16 GB
- Tárhely: SSD 256 GB, 1 TB HDD (Az adatbázis műveletek HDD-re íródnak, a fájlműveletek az SSD-n történnek)
- Processzor: i7 6700 CPU @ 3.40 GHz
- Internet sebessége: 1 Gbit (A Raspberry PI eszköz specifikáció szerint 100 Mbit-re képes Ethernet kábelén keresztül, mivel nem USB-t használok.)
- Windows 10 64-bit

Amiről még szót kell ejteni, hogy az RPI eszköznek melyek a pontos írási és olvasási limitei. Hálózat esetén tehát 100 Mbit-re képes, ami átszámítva 12.5 MB/s. Az SD kártya írási és olvasási sebességét az alábbi parancsokkal teszteltem:

Írás: `dd if=/dev/zero of=~/.test.tmp bs=500K count=1024`

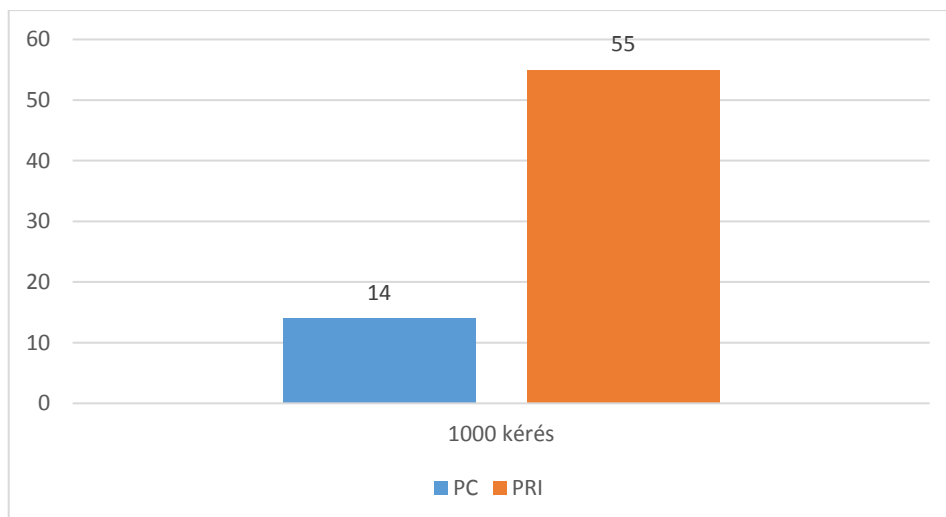
Olvasás: `dd if=~/.test.tmp of=/dev/null bs=500K count=1024`

A kapott érték írásra 16.8 MB/s, míg olvasásra 22.6 MB/s, tehát mindkét esetben a hálózat az, ami korlátoz pár fájlműveletet.

1. 1000 egymás után indított kérés:

A kiválasztott végpont `/files/folder/<folder_id>`, amelyet azért választottam, mert van benne adatbázis elérés, így jobb képet kapunk. Arra ügyeltem, hogy az adatbázisban található rekordok mindkét eszközön azonosak legyenek, tehát a két mappa tartalma megegyezik. A kapott eredmények átlagolva, másodpercenként:

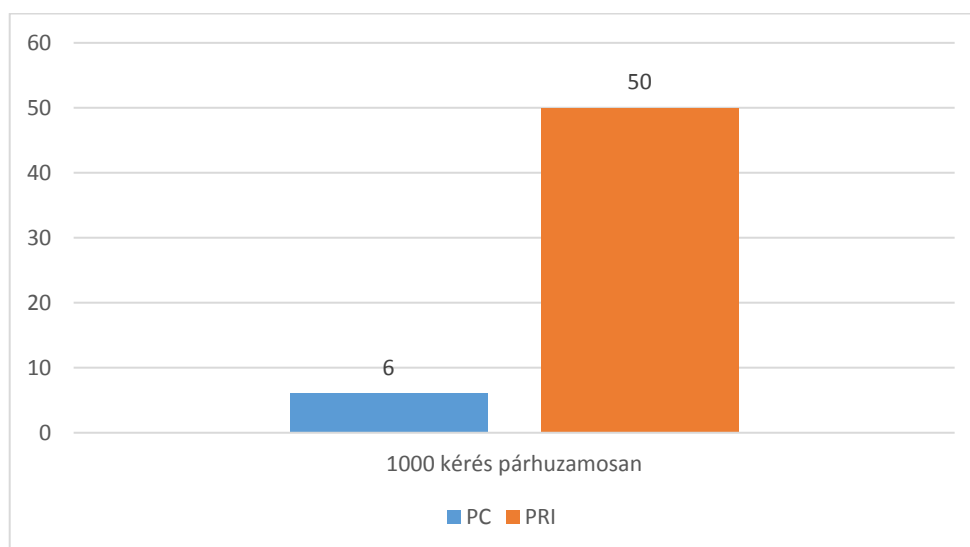




**Diagram 1 1000 kérés**

2. 1000 kérés párhuzamosítva:

A végpont ugyanaz, mint az előző mérés során. A változás annyi, hogy a kérések több szálon futnak a szerver felé. Összesen 3 szálát használtam, mindegyik szál pontosan 333 darab kérést küld.

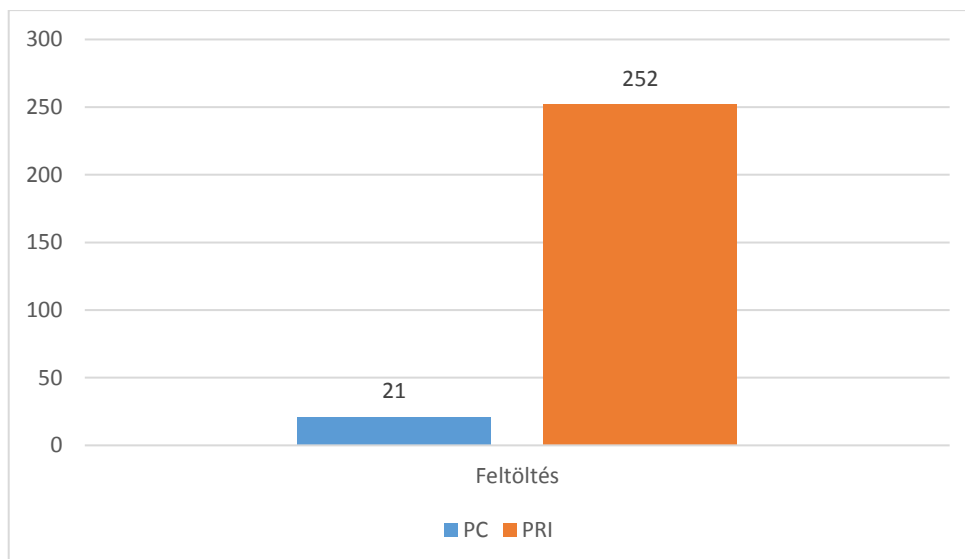


**Diagram 2 1000 kérés párhuzamosítva**

Jól látható javulást értünk el PC esetén, azonban az RPI csak minimális pozitív változást mutat.

3. 1 GB-os fájl feltöltése majd letöltése:

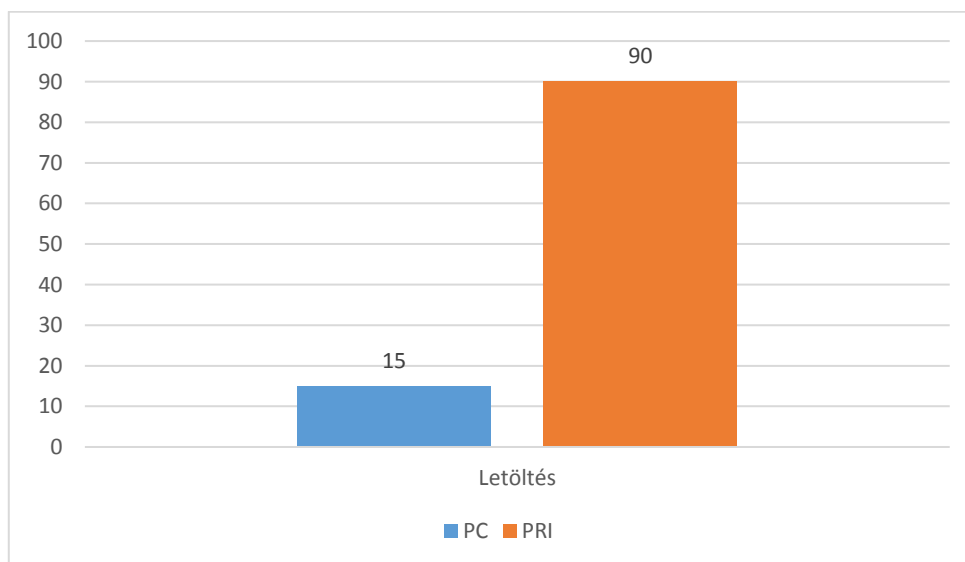
A fájl pontos mérete 951MB.



**Diagram 3 1GB-os fájl feltöltése**

A PC esetén 21 másodperc alatt töltődött fel a fájl, ami 45 MB/s-es feltöltési sebességet jelent. Az RPI rendszerre 252 másodperc alatt sikerült feltölteni. Ezt átszámolva 3.77 MB/s-es feltöltési sebességet kapunk, jól látható tehát, hogy nem közelíti meg a fizikai limitet.

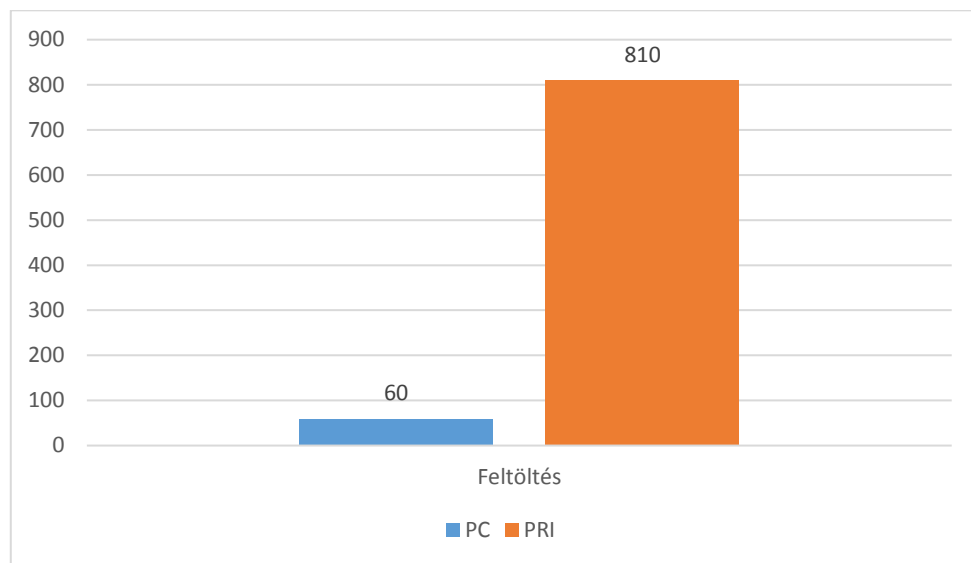
Ezt követően ugyanezt a fájlt letöltöttem.



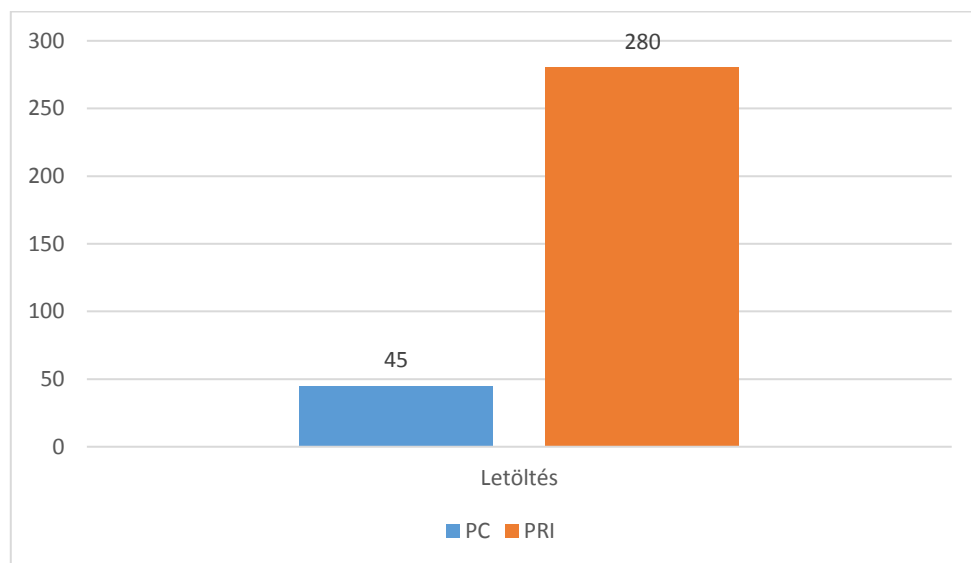
PC esetén 15 másodperc elég volt ahhoz, hogy letöltssem a fájlt, ez átszámítva 63 MB/s-es letöltési sebességet jelent. RPI esetén kerekén 90 mp kellett ahhoz, hogy le tudjam tölteni, ezt átszámítva 10.5 MB/s-es letöltési sebességet kapunk, amely elég jól megközelíti a felső limitet.

#### 4. 3 GB-os fájl feltöltése majd letöltése:

A fájl pontos mérete 2908 MB



PC esetén 60 másodperc kellett, hogy feltöltődjön a fájl, ami 48 MB/s-es sebességet jelent. RPI esetén 810 másodperc alatt tudtam feltölteni, ez 3.6 MB/s-es feltöltése sebesség. Az sebesség szinte azonos az előző teszttel.



Ugyanezt a fájlt le is töltöttem, először PC-ről, ezt 45 másodperc alatt sikerült végrehajtani ami 64 MB/s-es sebességet jelent. RPI-ről 280 másodperc alatt tudtam letölteni, ez 10.3 MB/s-es sebességet jelent. Az értékek itt is szinte megegyeznek az előző teszttel.

## 6 Összefoglalás

A projekt tervezése és fejlesztése alatt új technológiákkal ismerkedtem meg, így sokat tanultam, és fejlődtem szakmailag. A feladat megoldása során több problémába ütköztem, ahol mérlegelnem kellett több lehetséges megoldást, majd kiválasztani a szerintem legjobbat.

A projekt végére elkészítettem egy jól működő fájlmegosztó portált, amely egy egyszerű felhasználó számára minden igényt kielégítően alkalmas arra, hogy fájljait ott tárolja és megossza másokkal. A szoftvert összekötöttem egy népszerű fájlmegosztó szolgáltatással (Dropbox) úgy, hogy lehetőség nyíljon oda feltölteni, illetve onnan letölteni fájlokat.

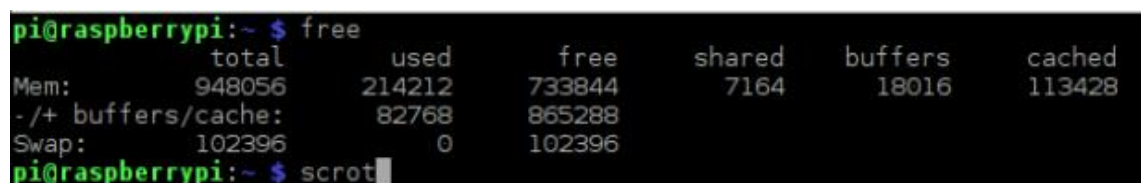
Meismerkedtem a Raspberry PI eszközzel, és sikeresen telepítettem rá az elkészült szoftvert. Méréseket végeztem, hogy átfogó képet kapjak arról, mekkora lassulással jár, ha az alkalmazás az RPI-n fut, nem pedig egy jóval erősebb asztali gépen.

### 6.1 Továbbfejlesztés

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

- Cache menedzsment: Az eszköz memória menedzsmentje jelenleg úgy működik, hogy ha egy fájl feltöltésre kerül a rendszerre, akkor azt a cache-ben eltárolja addig, amíg elegendő hely áll rendelkezésre. Ezt követően mindig fel kell szabadítani annyi helyet, hogy a következő beérkező részletet is be tudja tenni. Addig, amíg nincs szükség hely felszabadítására, a feltöltés gyorsabb, mint amikor már fel kell szabadítani helyet. Ez jól látható, ha a feltöltést monitorozom.

A memória állapota az operációs rendszer indítása után:



```
pi@raspberrypi:~ $ free
              total        used        free       shared    buffers     cached
Mem:      948056      214212      733844         7164       18016      113428
-/+ buffers/cache:      82768      865288
Swap:      102396           0      102396
```

Kép 29 Memória állapota boot után

A memória állapota a backend indítása után:

```
pi@raspberrypi:~ $ free
              total        used        free       shared    buffers     cached
Mem:          948056      282696      665360         7712       22392      136320
-/+ buffers/cache:      123984      824072
Swap:         102396           0       102396
pi@raspberrypi:~ $ scrot
```

Kép 30 Memória állapota backend indítása után

A memória állapota a frontend-es Lite server indítása után:

```
pi@raspberrypi:~ $ free
              total        used        free       shared    buffers     cached
Mem:          948056      603260      344796         7716       32868      220568
-/+ buffers/cache:      349824      598232
Swap:         102396           0       102396
pi@raspberrypi:~ $ scrot
```

Kép 31 Memória állapota frontend indítása után

A memória állapota, miközben egy 3 GB-os fájlt töltök fel:

```
pi@raspberrypi:~ $ free
              total        used        free       shared    buffers     cached
Mem:          948056      935848       12208         7720       32544      552308
-/+ buffers/cache:      350996      597060
Swap:         102396           0       102396
pi@raspberrypi:~ $ scrot
```

Kép 32 Memória állapota fájl feltöltése közben

A képen jól látható, hogy a tényleges szabad memória közel ~600 MB (második sor free oszlop), azonban a cached memória ~550 MB. A cél az lenne, hogy az eszköznek ne kelljen a cache-elt memóriából kijelölnie új területet, hanem a szoftver programozottan ürítse azt. Természetesen mérni kell majd, hogy milyen gyorsulás érhető el így.

- Kérés menedzsment: Ha az eszköznek sok kérést kell kiszolgálnia, például sok fájl feltöltése esetén, akkor előfordulhat, hogy ténylegesen elfogy a memória, ami a rendszer összeomlásához vezethet. A cél az lenne, hogy egy feltöltés, letöltés vagy Dropboxos művelet elkezdése előtt a szoftver ellenőrizze a szabad memóriát, és csak akkor engedélyezze, ha van legalább ~150 MB szabad memória.

- Mappa megosztás: A rendszerben jelenleg csak fájlokat lehet megosztani, azonban a felhasználói élményt nagyban javítaná, ha hasonló feltételekkel mappákat is meg lehetne osztani.
- Tárhelybővítés: Az eszköz jelenleg egy 16 GB-os microSD kártyával rendelkezik csak, amely sajnos kis mozgásteret ad. Szeretném kibővíteni, és megszüntetni a rendszerben található felhasználói limiteket.
- Nginx: Célom egy Nginx webszerver feltelepítése az eszközre, majd bekonfigurálása, hogy a be- és kimenő forgalom csak rajta mehessen keresztül.
- Https: Jelenleg nincs titkosítás a frontend és a backend kommunikációja között, ezért nem nevezhető biztonságosnak. Https bevezetésével szeretném ezt a problémát kiküszöbölni a későbbiekben.
- Fájl titkosítás: A feltöltött fájlok jelenleg nincsenek titkosítva a szerveroldalon. Biztonsági okokból viszont jó lenne ennek a megvalósítása.

## 7 Irodalomjegyzék

- [1] „npm,” [Online]. Available: <https://www.npmjs.com/>. [Hozzáférés dátuma: 25 november 2017].
- [2] „Typescript,” [Online]. Available: <https://www.typescriptlang.org/>. [Hozzáférés dátuma: 25 november 2017].
- [3] „Angular,” [Online]. Available: <https://angular.io/guide/architecture>. [Hozzáférés dátuma: 25 november 2017].
- [4] „Bootstrap,” [Online]. Available: <https://getbootstrap.com/>. [Hozzáférés dátuma: 25 november 2017].
- [5] „Flask,” [Online]. Available: <http://flask.pocoo.org/docs/0.12/>. [Hozzáférés dátuma: 21 november 2017].
- [6] „SQLite,” [Online]. Available: <https://www.sqlite.org/about.html>. [Hozzáférés dátuma: 24 november 2017].
- [7] „Passlib,” [Online]. Available: <https://passlib.readthedocs.io/en/stable/index.html>. [Hozzáférés dátuma: 24 november 2017].
- [8] „SQLAlchemy,” [Online]. Available: <https://www.sqlalchemy.org/>. [Hozzáférés dátuma: 24 november 2017].
- [9] „Pyjwt,” [Online]. Available: <https://pyjwt.readthedocs.io/en/latest/>. [Hozzáférés dátuma: 24 november 2017].
- [10] „Dropbox Python API,” [Online]. Available: <http://dropbox-sdk-python.readthedocs.io/en/latest/>. [Hozzáférés dátuma: 6 november 2017].
- [11] „RPI,” [Online]. Available: <https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>. [Hozzáférés dátuma: 22 november 2017].
- [12] „Raspbian OS,” [Online]. Available: <http://www.raspbian.org/>. [Hozzáférés dátuma: 6 November 2017].

- [13] „Lite server,” [Online]. Available: <https://www.npmjs.com/package/lite-server>.  
[Hozzáférés dátuma: 4 december 2017].
- [14] „Pycharm,” [Online]. Available: <https://www.jetbrains.com/pycharm/features/>.  
[Hozzáférés dátuma: 24 november 2017].
- [15] „Postman,” [Online]. Available: <https://www.getpostman.com/>. [Hozzáférés dátuma: 24 november 2017].
- [16] „DB Browser for SQLite,” [Online]. Available: <http://sqlitebrowser.org/>.  
[Hozzáférés dátuma: 24 november 2017].
- [17] „VS Code,” [Online]. Available: <https://code.visualstudio.com/>. [Hozzáférés dátuma: 24 november 2017].
- [18] „pip,” [Online]. Available: <https://pip.pypa.io/en/stable/installing/>. [Hozzáférés dátuma: 24 november 2017].
- [19] B. Alex és B. Everard, Learning Python with Raspberry Pi, 2014.
- [20] R. Leonard, M. Amundsen és S. Ruby, RESTful Web APIs, 2013.