

Tartalomjegyzék

1 Bevezetés.....	3
1.1 Motiváció.....	3
1.2 A feladat.....	4
2 Felhasználói dokumentáció.....	5
2.1 A program célja.....	5
2.2 Kiknek íródott ez a program?.....	5
2.2.1 Felhasználói előismeretek.....	5
2.3 Fontosabb eszközök a megvalósításhoz.....	5
2.4 Szükséges hardver és szoftver feltételek.....	6
2.4.1 Hardware.....	6
2.4.2 Szoftver.....	6
2.5 Letöltés githubról.....	7
2.6 Installálás.....	8
2.6.1 Függőségek.....	8
2.6.2 Előkészületek.....	8
2.6.3 Linux service létrehozás.....	9
2.7 Konfigurálás.....	11
2.7.1 Konfigurációs változók.....	11
2.7.2 Konfigurációs opciók.....	12
2.7.3 Példa.....	14
2.8 Használat.....	14
2.8.1 Szerver oldal.....	14
2.8.2 Kliens oldal.....	16
3 Fejlesztői dokumentáció.....	33
3.1 Adattárolás.....	33
3.1.1 Felhasználói adatok tárolása a szerveren.....	33
3.1.2 Fájlnemek.....	34
3.1.3 AESEncrypter formátuma.....	35
3.1.4 Txt fájl (.txt) formátuma.....	35

3.1.5 Telefonkönyv fájl (.phb) formátuma.....	35
3.2 Szerver oldal felépítése.....	36
3.2.1 Főbb modulok.....	36
3.2.2 RPCWrapper osztály.....	37
3.2.3 Komponensek kommunikációja.....	38
3.3 Kliens oldal felépítése.....	39
3.3.1 Komponensek kommunikációja.....	40
3.4 Megvalósítás.....	41
3.4.1 Fontosabb algoritmusok.....	41
3.4.2 Döntések a fejlesztés során.....	42
3.4.3 Szerver oldal mappa és fájl struktúrájának áttekintése.....	43
3.4.4 Kliens oldal mappa és fájl struktúrájának áttekintése.....	46
3.4.5 Fejlesztési lehetőségek.....	51
3.5 Tesztelési terv.....	51
3.5.1 Előkészületek.....	51
3.5.2 Eset 1: A szerver elindítása (black box).....	52
3.5.3 Eset 2: Felhasználók létrehozása konzolból (black box).....	54
3.5.4 Eset 3: Üresen hagyott mezők felhasználó létrehozása közben.....	55
3.5.5 Eset 4: Jelszó és jelszó mégegyszer nem egyezik (CLI).....	55
Elvárt eredmény.....	55
3.5.6 Eset 5: Létező felhasználó hozzáadása azonos jelszóval.....	55
3.5.7 Eset 6: Létező felhasználó hozzáadása más jelszóval (cli).....	56
3.5.8 Eset 7: Belépés hibás jelszóval (GUI).....	57
3.5.9 Eset 8: Bejelentkezés valós felhasználókkal (GUI).....	57
3.5.10 Eset 9: Felhasználó létrehozása (GUI).....	58
3.5.11 Eset10: Bejelentkezés nem létező felhasználóval.....	58
3.5.12 Eset11: Txt Fájl létrehozása.....	59
3.5.13 Eset 12: Telefonkönyv fájl létrehozása.....	60
3.5.14 Eset 13: Ellenőrizzük, hogy a szerveren tárolt adatok valóban titkosak!...61	
3.5.15 A teszt közben létrejött felhasználók és fájlok.....	63
4 Forrásjegyzet.....	65

1 Bevezetés

1.1 Motiváció

Jelen világunkban az információ érték, ezért gyakorlatilag mindenki visszaél vele.

Sajnos nem ellenőrizhetem azt, hogy az operációs rendszer, amit használok, mennyi információt gyűjt rólam, és mire használja fel. Ez nincs másképp az e-mail szolgáltatókkal, a különböző felhő alapú tárolókkal.

Nem tudom ellenőrizni, hogy ha egy felhőben tárolom az adataimat, akkor vajon a Google, a Microsoft, a Facebook, az Apple vagy más cégek, akik ilyeneket szolgáltatnak, felhasználják-e őket. Ilyen szempontból mondjuk a megnevezettek még korrektek, mert az általános szerződési feltételeikben megfogalmazzák, hogy az adatainkat felhasználják.

Sajnos arra jelenleg nincs erőforrásom, hogy egy saját operációs rendszert írjak, vagy leellenőrizzek egy Linuxot, Open/Free BSD-t hogy vajon visszaél-e az adataimmal, így ezeknek el kell hinnem, hogy nem teszik.

De tudok csinálni olyan programot, ami azt biztosítja számomra, hogy az adataimat a kliens oldalon titkosítva küldöm el a felhőbe, akkor a felhőben lévő cég nem tudhatja, hogy mik is voltak azok.

Így jött az ötlet, hogy először egy egyszerű szervert készítek, ahol fájlokat tudok tárolni, melyekben az egyszerűség kedvéért először csak szövegeket, vagy telefonkönyveket tudok tárolni.

Így hiába olvashatja az adott szerverszolgáltató az adataimat, nem fog hozzáférni az információhoz, mert az már a kliens oldalon titkosítva van.

Fontos volt számomra, hogy az általam készített program nyílt forráskódú legyen, hogy biztosítva legyen, hogy tényleg titkosít, és tényleg nem ment semmilyen adatot.

1.2 A feladat

Egy olyan program írása, ami szöveges adatokat, és telefonkönyv adatokat olyan titkosan tart, amennyire csak lehetséges.

A program két részből áll. Egy szerverből és egy kliensből.

A kliensnek négy feladata van:

1. Telefonkönyv fájl és txt fájl létrehozása, megjelenítése, módosítása
 1. A telefonfájl fájlnál figyelni kell arra, hogy a memóriában mindig maximum egy kontaktnak legyenek titkosítatlanul az adatai.
2. A fájlok titkosítása, és visszafejtése, még a nevüket is titkosítani kell
3. A felhasználói adatok titkosítása a szerver előtt (még a felhasználónevét is)
4. Kommunikáció a szerverrel (RPC segítségével)
 1. Autentikáció
 2. Fájlok letöltése, feltöltése, törlése.

A szerver oldalnak öt feladata van:

1. A felhasználók autentikációja
2. A felhasználók adatainak tárolása (fájlkiszolgáló)
 1. Minden felhasználónak létre kell hozni egy mappát, de figyelni kell rá, hogy a felhasználóról minél kevesebb információt tároljon, így hash-elést kell használni, hogy még a felhasználó neve se derülhessen ki.
 2. A fájlok eleve titkosan kell, hogy megérkezzenek, egy titkos névvel, és tartalommal
 3. Ezen fájlok manipulálása: átnevezés, létrehozás, felülírás, törlés.
3. A klienssel kommunikálás RPC-t használva.
4. A kliens statikus fájljainak kiszolgálása.
5. Biztosítani, hogy a kommunikáció is titkosan zajlik.
6. Felhasználói dokumentáció

2 Felhasználói dokumentáció

2.1 A program célja

Ez a program két részből áll, egy szerverből, és egy kliensből.

A szervernek két feladata van: a felhasználó azonosítása, és fájl kiszolgálás.

A kliens oldal telefonkönyvfájlok, és szöveges fájlok kezelésére, és titkosítására való.

Továbbá kommunikál a szerverrel.

2.2 Kiknek íródott ez a program?

A program célközönsége olyan felhasználók sokasága, akik nem bíznak a felhőszolgáltatókban, és biztosak szeretnének lenni abban, hogy az adataik biztonságban vannak.

2.2.1 Felhasználói előismeretek

A program két részből áll, egy kliens oldaliból, és egy szerver oldaliból, így van egy felhasználói oldala, és egy üzemeltetői oldala

2.2.1.1 Kliens oldali felhasználói előismeretek

Azoknak a felhasználóknak, akiknek csak a klienst kell használniuk, elegendő minimális informatikai ismerettel rendelkeznie. Csak a böngésző használata követelmény a számára

2.2.1.2 A szerver üzemeltetői előismeretek.

Igyekeztem minél egyszerűbben konfigurálható szervert létrehozni, és igyekeztem részletes telepítési, és üzemeltetési útmutatót adni. Mindemellett érdemes minimális szintű Linux ismeretekkel rendelkezni.

2.3 Fontosabb eszközök a megvalósításhoz

AES256 algoritmus: A fájlok, és a fájlok neveinek titkosításához

SHA256 algoritmus (sózva): a felhasználó nevek és jelszavak titkosításához, a felhasználói mappa létrehozásához, meg az AES kulcs generálásához.

HTTPS: Az adatok titkos továbbításához.

tornado: Python webservert, a statikus fájlok kiszolgálásáért, és az RPC legalsó rétegéért.

2.3 Fontosabb eszközök a megvalósításhoz

Jsonrpcserver, simple-jsonrpc-js: Az RPC kommunikációért (a **tornado** felett).

2.4 Szükséges hardver és szoftver feltételek

2.4.1 Hardware

RAM: 16GB (ez lehet, hogy kevés lesz sok felhasználóra.)

Processor: Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz

2.4.2 Szoftver

2.4.2.1 Operációs rendszer

Distributor ID: Ubuntu

Description: Ubuntu 18.04.4 LTS

Release: 18.04

Codename: bionic

2.4.2.2 Python

Python 3.6.9

Csomag	Verzió	Hivatalos weblap
tornado	6.0.3	https://www.tornadoweb.org/en/stable/ https://github.com/tornadoweb/tornado/ https://pypi.org/project/tornado/
jsonrpcserver	4.1.2	https://github.com/bcb/jsonrpcserver https://pypi.org/project/jsonrpcserver/
pandas	1.0.3	https://pandas.pydata.org/ https://github.com/pandas-dev/pandas https://pypi.org/project/pandas/
pyexcel-ods	0.5.6	https://github.com/pyexcel/pyexcel-ods https://pypi.org/project/pyexcel-ods/

2.4.2.3 Böngészők

Google Chrome 80.0.3987.149 <https://www.google.com/chrome/>

Mozilla Firefox 74.0 <https://www.mozilla.org/en-US/firefox/>

2.4 Szükséges hardver és szoftver feltételek

2.4.2.4 JavaScript

Ecmascript 6

Csomag	Verzió	Hivatalos weblap
aes-js	3.1.2	https://www.npmjs.com/package/aes-js https://cdn.rawgit.com/ricmoo/aes-js
bootstrap	4.4.1	https://getbootstrap.com/
jquery	3.4.1.slim	https://jquery.com/
js-sha256	0.9.0	https://www.npmjs.com/package/js-sha256
popper.js	1.16.0	https://popper.js.org/
simple-jsonrpc-js	1.0.0	https://github.com/jershell/simple-jsonrpc-js

2.5 Letöltés githubról

A programot a githubról lehet letölteni. Programozóknak a git-tel javaslok, felhasználóknak teljesen megfelelő a szimpla zip-es letöltés.

1. git-tel
 1. git installálás (ha nincs telepítve)
sudo apt install git
 2. klónozás
 1. **git clone** https://github.com/somla/real_private_data.git
vagy
 2. **git clone** [git@github.com:somla/real_private_data.git](https://github.com/somla/real_private_data.git)
vagy
 3. **Forokolod a saját repoid közé (fejlesztőknek)**
2. Letöltés githubról zip formátumban
 1. Egy böngészőben nyissuk meg ezt a linket:
https://github.com/somla/real_private_data
 2. Kattintsunk a **Clone or download** gombra
 3. Kattintsunk a **Download ZIP** gombra
 4. Tömörítsük ki
unzip real_private_data-master.zip

2.6 Installálás

2.6 Installálás

Ehhez egy VirtualBoxot használtam, arra feltelepítettem egy Ubuntut, így egy teljesen új linuxon van tesztelve, amin még nincs semmi.

2.6.1 Függőségek

Ezeket a függőségeket kell telepíteni, hogy a program teljes mértékben működhessen.

2.6.1.1 Iptables-persistent

Csak ha portforwardingolni akarunk

sudo apt-get install iptables-persistent

2.6.1.2 pip3

sudo apt install python3-pip

2.6.1.3 Python csomagok

sudo pip3 install tornado

sudo pip3 install pandas

sudo pip3 install pyexcel-ods

sudo pip3 install jsonrpcserver

2.6.2 Előkészületek

1. Menjünk abba a mappába, ahova letöltöttük a programot
cd ./real_private_data
2. hozzunk létre egy könyvtárat az adatoknak (nem muszáj itt, de akkor át kell állítani a config-ban lásd a Konfiguráció fejezetet)
mkdir data
3. hozzunk létre SSL-kulcsot, vagy ha van saját, akkor másoljuk be a .key mappába, vagy adjuk meg a helyét a config.json-ban (lásd a Konfiguráció fejezetet)

mkdir .key;

**openssl req -x509 -out rpd.crt -keyout rpd.key **

**-newkey rsa:2048 -nodes -sha256 **

**-subj '/CN=localhost' -extensions EXT -config <(**

printf "[dn]\nCN=localhost\n[req]\ndistinguished_name = dn\n[EXT]\n

subjectAltName=DNS:localhost\nkeyUsage=digitalSignature

2.6 Installálás

```
nextendedKeyUsage=serverAuth");
```

```
cd ..
```

4. Hozzunk létre egy mappát a generált javascript fájloknak

```
mkdir src/web/generated/
```

5. Másoljuk le a config.sample.json-t a config.json-ra

```
cd src/python
```

```
cp config.sample.json config.json
```

```
cd ../../
```

6. Hozz létre legalább egy felhasználót

```
./bin/server/rpd_create_user.sh
```

7. Ha minden jól sikerült, akkor el kell, hogy tudjuk indítani a szerver

```
./bin/server/rpd_server.sh
```

2.6.3 Linux service létrehozás

A Linux service automatikusan indul, amikor a linux elindul, és újraindul, ha a folyamat valamiért leáll. Én itt egy alapbeállítást mutatok be, további információért nézz utána a Linux folyamatoknak, és a **systemctl** parancsnak

Ehhez érdemes egy új felhasználót létrehozni, nálam ez “rpd-server” lesz

```
sudo adduser rpd-server
```

1. hozzunk létre egy új mappát az adatoknak

```
sudo mkdir -p /var/local/rpd/data
```

```
sudo chown rpd-server:rpd-server /var/local/rpd/data
```

2. Csináljunk egy kulcsot a szerverünknek (lásd feljebb: Előkészületek 3. lépés)
aminek az rpd-server a tulajdonosa

3. Csináljunk egy config fájlt a service-nek

- 1) Másoljuk le a sample config-ot

```
cd {project_dir}/src/python
```

```
cp config.sample.json config.service.json
```

- 2) Írjuk át a “config.service.json”-t

```
(1) data_dir="/var/local/rpd/data"
```

```
(2) secure_port:10443
```

```
(3) open_port:10080
```

2.6 Installálás

- (4) `crt_file:<crt fájl helye>`
- (5) `key_file:<key fájl helye>`
- 4. Csináljunk egy service fájlt:
 - 1) másoljuk le a sample-t
cd {project_dir}/src/service
cp rpd.sample.service rpd.service
 - 2) állítsuk be az "rpd.service"-t
 - (1) `ExecStart=/home/rpd-server/real_private_data/bin/server/rpd_server.sh`
`--configFile "[[dir_project]]/src/python/config.service.json"`
 - (2) `User=rpd-server`
- 5. Hozzunk létre felhasználót (felhasználókat)
su rpd-server
./bin/server/rpd_create_user.sh --configFile ./src/python/config.service.json
- 6. Másoljuk be a service fájlt a linux service könyvtárba
sudo cp rpd.service /etc/systemd/system/
systemctl daemon-reload
systemctl start rpd
systemctl enable rpd
- 7. Csináljunk port forwardingot, hogy a 80-as és a 443 portokról lehessen elérni a szervert
sudo iptables -t nat -A OUTPUT -o lo -p tcp --dport 80 -j REDIRECT --to-port 10080
sudo iptables -t nat -A OUTPUT -o lo -p tcp --dport 443 -j REDIRECT --to-port 10443
sudo iptables -i <interface> -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 10080
sudo iptables -i <interface> -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-port 10443
su
iptables-save > /etc/iptables/rules.v4
ip6tables-save > /etc/iptables/rules.v6

2.7 Konfigurálás

2.7 Konfigurálás

A konfigurálás két módon lehet. Vagy fájlból, vagy command line argumentumként megadva. A command line argumentumnak nagyobb prioritása van. Kötelező, hogy legyen config fájl. A konfigurációs paraméterek lehetnek publikusak, ez azt jelenti, hogy a kliens oldalon is láthatóak. A nem publikus konfigurációs beállítások csak a szerver oldalon láthatóak.

2.7.1 Konfigurációs változók

A konfigurációban vannak változók amiket [[változó név]]-ként érünk el.

Például: `./rpd_server.sh --logLevel /var/tmp/log[[now]].log`

`./rpd_server.sh --logLevel /var/tmp/log20200330_163019.log` lesz, vagy ehhez hasonló

Változó név	Leírás	Példa
[[dir_project]]	A projekt gyökér könyvtára	
[[dir_src]]	A projekt src könyvtára	
[[dir_web]]	A projektben lévő web könyvtára	
[[dir_python]]	A projektben lévő python fájlok könyvtára	
[[now]]	Az aktuális idő ÉvHóNap_ÓraPercMásodperc formában	20200330_163019
[[today]]	A mai nap ÉvHóNap formában	20200330

2.7 Konfigurálás

2.7.2 Konfigurációs opciók

Név	Leírás	Alapérték	Opcionális	Publikus
configFile	Config json, ez az a konfigurációs fájl, amiből a beállítások jönnek: config.json	./config.json	Igen	Nem
debug	Debug mód, ha be van kapcsolva, akkor több ellenőrzés van, több log van, de az a logLeveltől is függ.	False	Igen	Igen
logLevel	logLevel, értékei lehetnek CRITICAL - 50 ERROR - 40 WARNING - 30 INFO - 20 DEBUG -10 NOTSET – 0 Lásd: https://docs.python.org/3/library/logging.html	INFO	Igen	Nem
logFile	Log fájl, helye, ha nem töltjük ki, akkor nem logolunk fájlba, csak a consola.		Igen	Nem
logFormat	Log formátuma, ahogy a python várja lásd: https://docs.python.org/3/library/logging.html#logging.Formatter	[%(asctime)s] [% (levelname)s] %(message)s	Igen	Nem
show_rpc_message	Mutassuk-e az rpc üzeneteket	False	Igen	Nem
open_port	Indítunk egy http szerveret is, ami átirányít a https szerverre, ennek a portja	8080	Igen	Nem

2.7 Konfigurálás

Név	Leírás	Alapérték	Opcionális	Publikus
debug_openn_port	Debug módban indítunk egy http szerveret, ami nem titkos, ez segítheti a debuggolást, de nem biztonságos, így production rendszerbe nem fut	8081	Igen	Nem
secure_port	A szerver portja, https kapcsolat	8443	Igen	Nem
host	a host neve, átirányításnál fontos	localhost	Igen	Nem
crt_file	Certification fájl az SSL-hez	None	Nem	Nem
key_file	Key fájl az SSL-hez	None	Nem	Nem
web_root	a statikus fájlok könyvtára	None	Nem	Nem
data_dir	Az adatok mappája, ide lesznek elmentve a titkos fájljai a felhasználóknak	None	Igen	Nem
test_dir	Egy mappa a tesztekhez	/var/tmp/ real_private_data	Igen	Nem
salt	Egy hash "só" a kliens oldalra	My own Salt	Igen	Igen
server_salt	Egy hash "só" a szerver oldalra	Server salt	Igen	Nem
enable_create_user	Engedélyezzük, hogy felhasználók is létre tudjanak hozni új felhasználókat, ha nem, akkor csak a szerveren lehet új felhasználókat létrehozni commandline paranccsal.	False	Igen	Igen
defaultRpcClient	Az alapértelmezett RPC metódus neve. jelenleg SimpleJsonRpcWebSocketClientService vagy	SimpleJsonRpcWebSocketClientService	Igen	Igen
hideMessageTime	Az üzenetek elrejtése előtti idő ezredmásodpercben	5000	Igen	Igen

2.7 Konfigurálás

Név	Leírás	Alapérték	Opcionális	Publikus
show_encrypted_data	Mutassa a weblapon a titkosított adatot	False	Igen	Igen

2.7.3 Példa

my_config.json

```
{
  "debug":false,
  "logLevel":"INFO",
  "logFile":"/var/tmp/rpd_[[now]].log",
  "host":"localhost",
  "open_port":8080,
  "secure_port":8443,
  "crt_file": "[[dir_project]]/.key/rpd.crt",
  "key_file": "[[dir_project]]/.key/rpd.key",
  "web_root": "[[dir_project]]/src/web",
  "data_dir": "[[dir_project]]/data"
}
```

ha most meghívjuk a programot

```
./rpd_server.sh --configFile --logLevel /var/tmp/log[[now]].log --configFile
my_config.json --data_dir "[[dir_project]]/data2"
```

akkor a command line data dir fog érvényesülni.

2.8 Használat

2.8.1 Szerver oldal

Alapjában véve a **./bin** mappában vannak a futtatható fájlok, ott van egy **server** és egy **tools**

A **server** mappában vannak a szerverhez kellő dolgok, a **tools** mappában a fejlesztéshez szükséges toolok, ezért azokat majd a fejlesztői dokumentációban fogom részletezni.

2.8 Használat

2.8.1.1 `./bin/server/rpd_server.sh`

Lásd `./src/python/run_server.py`

2.8.1.2 `./bin/server/rpd_create_user.sh`

Lásd `./src/python/create_user.py`

2.8.1.3 `./src/python/run_server.py`

Maga a szerver, a beállításokat lásd a **Konfigurálás** fejezetet, alapértelmezettként a `./src/python/config.json` fájlt fogja betölteni `--configFile`

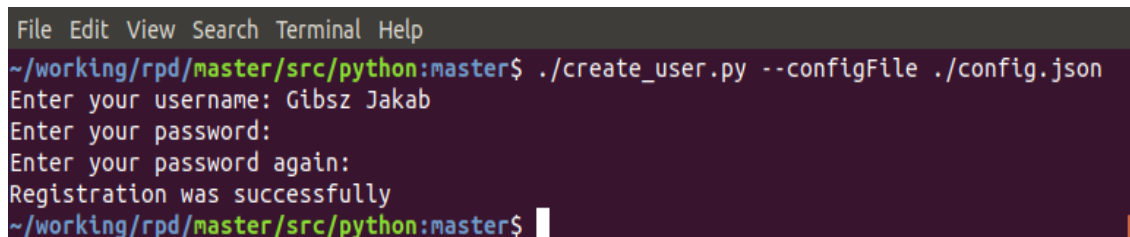
2.8.1.4 `./src/python/create_user.py`

Felhasználó létrehozása, érdemes beállítani a `--configFile`-t, alapértelmezettként a `./src/python/config.json` fájlt használja.

Enter your username: Írd be a felhasználónevet

Enter your password: Írd be a jelszót

Enter your password again: Írd be a jelszót megint

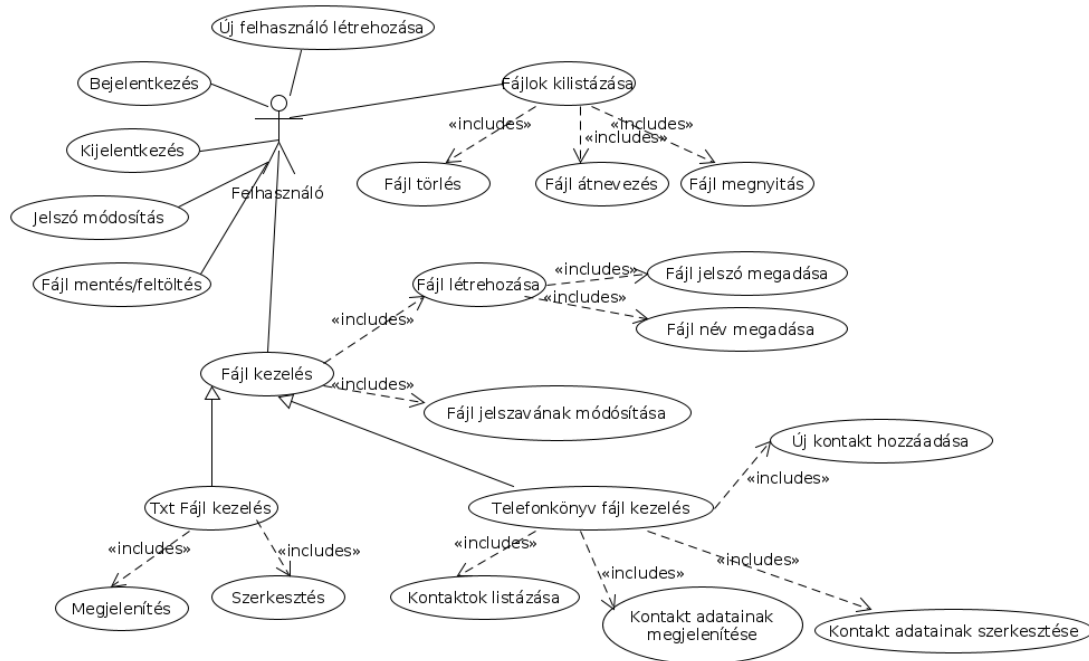


```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile ./config.json
Enter your username: Gibbs Jakab
Enter your password:
Enter your password again:
Registration was successfully
~/working/rpd/master/src/python:master$
```

1. ábra: Felhasználó létrehozása CLI felületről

2.8.2 Kliens oldal

2.8.2.1 Use case diagramm

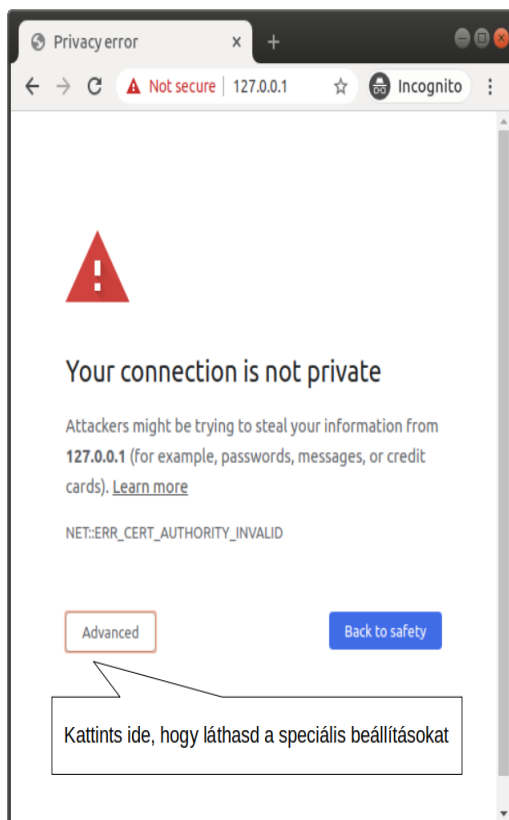


2. ábra: Az oldal Use case diagrammja

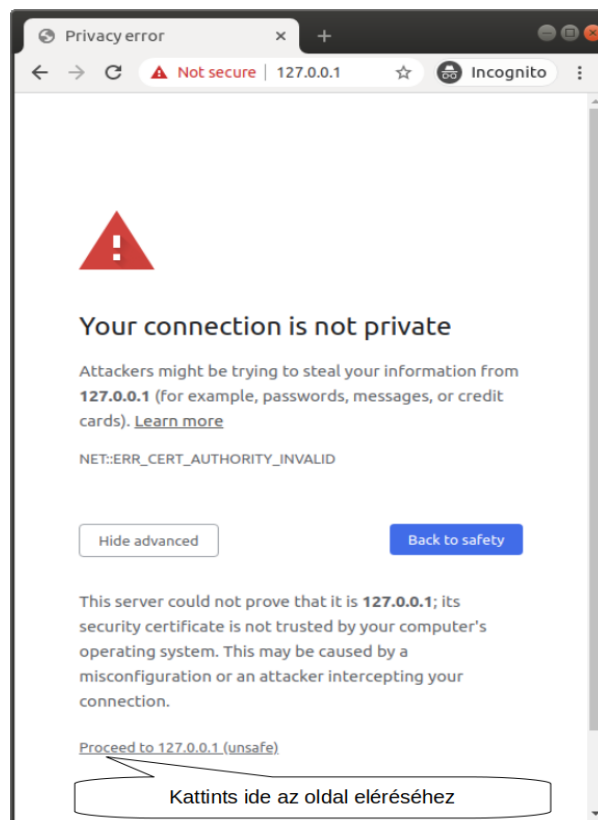
2.8 Használat

2.8.2.2 Az oldal elérése

Az oldalt az épp aktuális címén lehet elérni a böngészőben, de ha nem akarunk SSL hitelesítést venni, akkor sajnos a böngésző “nem biztonságos”-nak fogja látni az oldalunkat.



3. ábra: SSL hiba továbblépés (1)

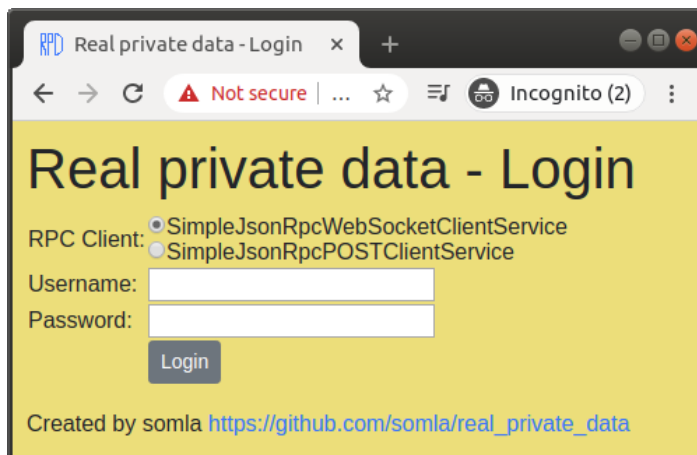


4. ábra: SSL hiba továbblépés (2)

2.8.2.3 Bejelentkezés

RPC Client: SimpleJsonRpcWebSocketClientService: a kommunikációhoz használjuk a SimpleJsonRpcWebSocketClient-et, ez egy websocket alapú kommunikáció. Előnye, hogy folyamatos kapcsolat van a szerver, és a kliens között, Hátránya, hogy így folyamatosan van kommunikációs forgalom, de csak elhanyagolható.

SimpleJsonRpcPOSTClientService: HTTP post alapú kommunikációt biztosít. Előnye, hogy csak akkor van kommunikáció, amikor szervertől kérünk valamit. Hátránya, hogy mindig új kapcsolatot kell létesíteni.



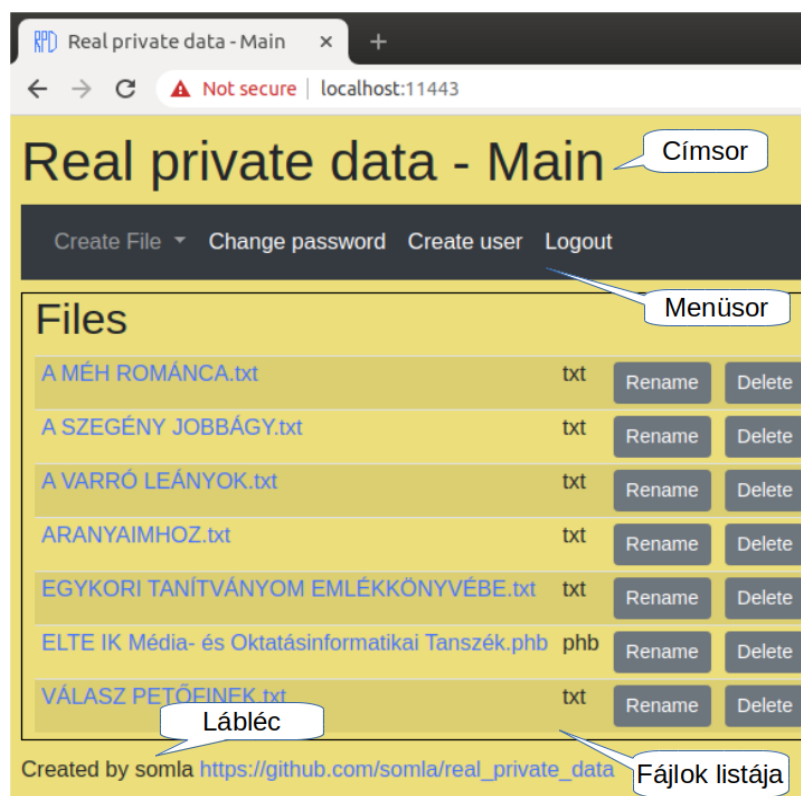
5. ábra: Bejelentkezés űrlap

Username: felhasználónév

Password: jelszó

Login: bejelentkezés gomb a bejelentkezéshez

2.8.2.4 A főoldal struktúrája



6. ábra: A főoldal struktúrája

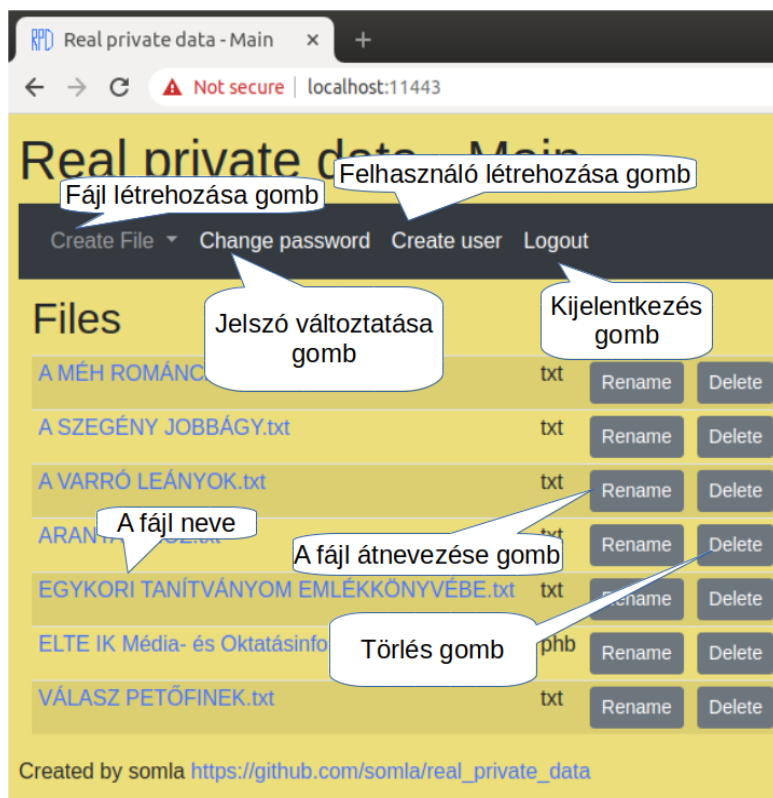
2.8 Használat

2.8.2.5 A főoldal gombjai

Felhasználó létrehozása gomb

Ez a gomb csak akkor jelenik meg, ha a `enable_create_user` opció `True` (lásd Konfigurálás fejezet)

A fájl neve



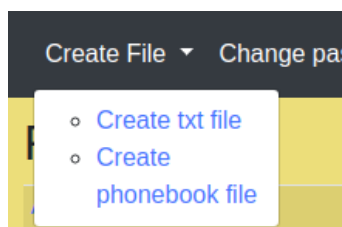
7. ábra: A főoldal gombjai

Ha rákattintasz, megnyitja a fájlt.

2.8.2.5.1 Fájl létrehozása menü

Create txt file: Létrehoz egy txt fájlt

Create phonebook file: Létrehoz egy telefonkönyv fájlt



8. ábra: Új fájl gombjai

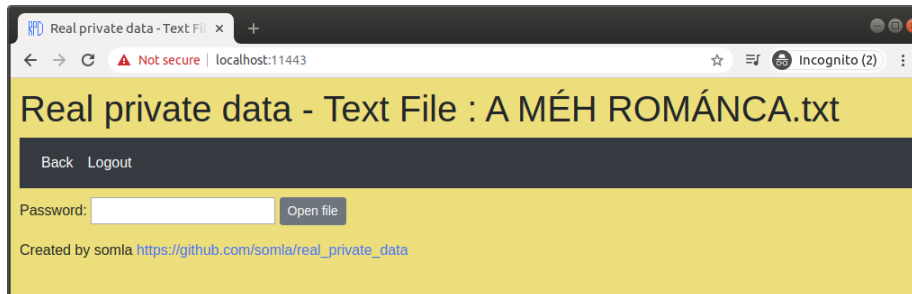
2.8 Használat

2.8.2.5.2 Fájl átnevezés

Fájl neve mező: Ide kell beírni az új nevét a fájlnak

Save gomb: Elmenti a névváltoztatást

Cancel gomb: Megszakítja a névváltoztatást



9. ábra: A fájl jelszavának megadása

2.8.2.6 Txt/Phonebook fájl megnyitás

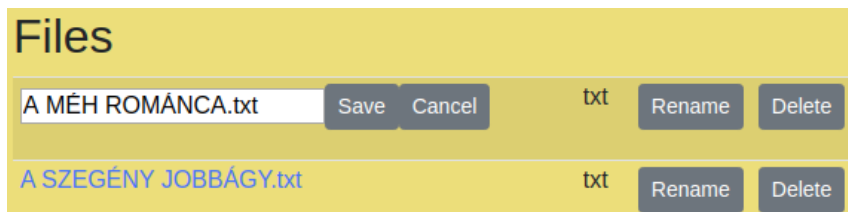
Ahhoz, hogy megnyissunk egy fájlt, ahhoz be kell írunk a fájl jelszavát

Back menü: Visszalép a főoldalra

Logout menü: Kilép

Password mező: Ide kell írni a fájl jelszavát, hogy megnyissuk

Open file gomb: Megnyitja a fájlt



10. ábra: Fájl átnevezése

2.8 Használat

2.8.2.7 Txt fájl oldal struktúrája

Save menü: Elmenti a txt fájlt

Change password menü: Megváltoztatja a fájl jelszavát

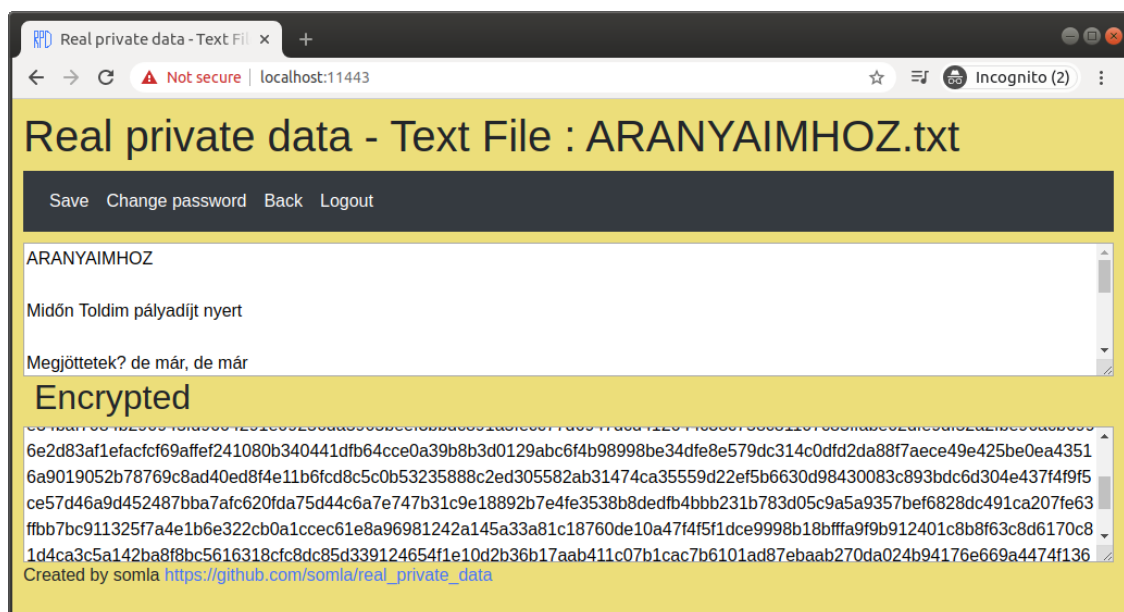
Back menü: Visszamegy a főoldalra

Logout menü: Kijelentkezik az oldalból

Txt mező: A txt fájl tartalma, ez szerkeszthető

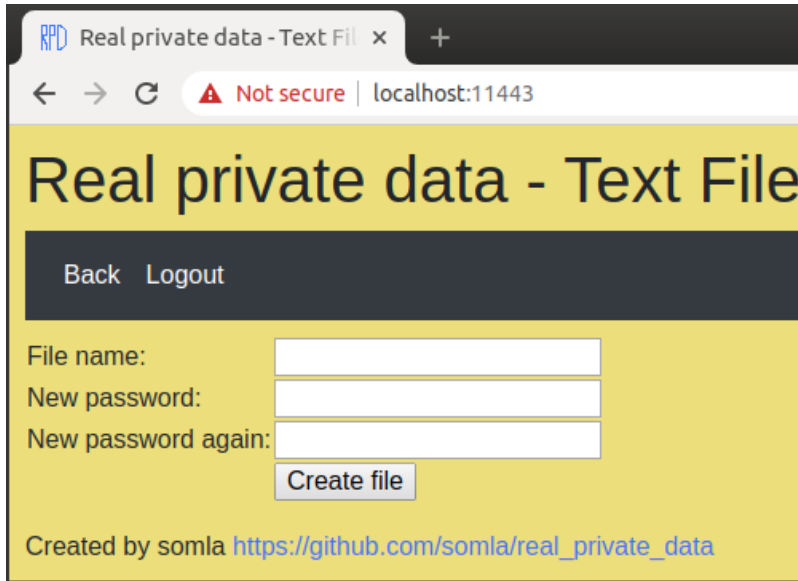
Encrypted mező: A txt fájl titkosítva, ez csak akkor látszik, ha a

show_encrypted_data konfiguráció **True** (Lásd **Konfiguráció** fejezet)



11. ábra: A text fájl oldal struktúrája

2.8.2.8 Txt fájl létrehozása



Real private data - Text File

Back Logout

File name:

New password:

New password again:

Create file

Created by somla https://github.com/somla/real_private_data

12. ábra: Txt fájl létrehozása form

2.8.2.8.1

Back menü: Visszalép a főoldalra

Logout menü: Kijelentkezés

File name mező: A fájl neve

New password mező: A fájl jelszava

New password again mező: A fájl jelszava még egyszer

Create file gomb: Ez a gomb hozza létre a fájlt, meg fog jelenni egy üres fájl.

2.8 Használat

2.8.2.8.2 A *create file* gomb megnyomása után

Real private data - Text File

Save Change password Back Logout

File name:

New password:

New password again:

test_file1

Encrypted

e8771c851655a9f7f9d5760746a99334e80ec7af98c18767c5db533ddbfaae
abfb23d592732b11fccdf6459abc3acfca5f5219f29f0f3c5fa5a8eb7c83ca9

Created by somla https://github.com/somla/real_private_data

13. ábra: A txt fájl készen áll a szerkesztésre, amikor már van neve és jelszava

Save menü: Elmenti a fájlt

Change password menü: Jelszóváltoztatás

Fontos, hogy a fájl csak a **Save** gomb lenyomásával lesz elmentve, ha a fájl létezik, akkor hibát ír. Apró hiba, hogy ezután ismét meg kell nyomni a **Create file** gombot, a fájl átnevezése után (**File name** mező), majd utána megint a **Save** gombot valamikor javítani fogom, hogy intuitívabb legyen.

2.8 Használat

2.8.2.9 Txt/Telefonkönyv fájl jelszóváltoztatás

Real private data - Text File

Not secure | localhost:11443

Real private data - Text

Save Change password Back Logout

Change password

Old password:

New password:

New password again:

Change password Cancel

ARANYAIMHOZ

Midőn Toldim pályadíjt nyert

Megjöttetek? de már, de már

Encrypted

02dfe9df52a2fbe96acb6996e2d83af1efacfcf69affef2410

14. ábra: Fájl jelszóváltoztatás form

Old password mező: Ide kell írni a fájl régi jelszavát

New password mező: Ide kell írni a fájl új jelszavát

New password again mező: Ide kell írni a fájl új jelszavát még egyszer

Change Password gomb: Elmenti az új jelszót

Cancel gomb: Megszakítja a jelszóváltoztatást

2.8.2.10 Telefonkönyv fájl oldal struktúrája



- 25

2.8 Használat

4. Encrypted layer 2: A kétrétegű titkosítás 1. rétegét mutatja (hexadecimális számok)

Ez a réteg van a memóriában, csak akkor dekódolja a második réteget, ha rákattintunk egy kontaktra, és akkor is csak annak a kontaktnak az információit csomagolja ki

Az **Encrypted layer 1** és az **Encrypted layer 2**, csak akkor látszik, ha beállítjuk a **show_encrypted_data** konfigurációt **True**-ra (Lásd Konfigurálás fejezet)

2.8.2.11 Telefonkönyv kontakt

Nickname: A kontakt beceneve

Fullname: A kontakt teljes neve

Address: A kontakt címe

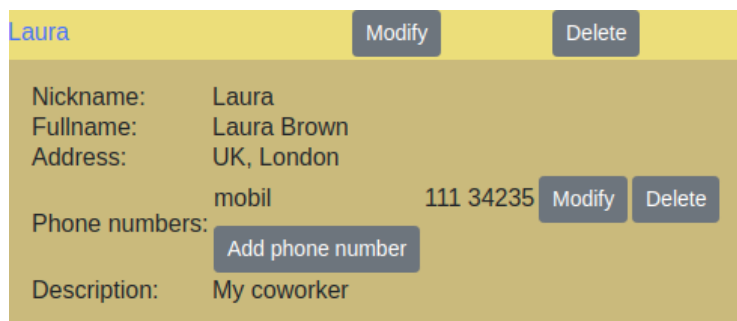
Phone numbers: A kontakt telefonszámai

Description: A kontaktról egy leírás

Modify gomb: A telefonszám módosítása

Delete gomb: A telefonszám törlése

Add phone number gomb: Telefonszám hozzáadása

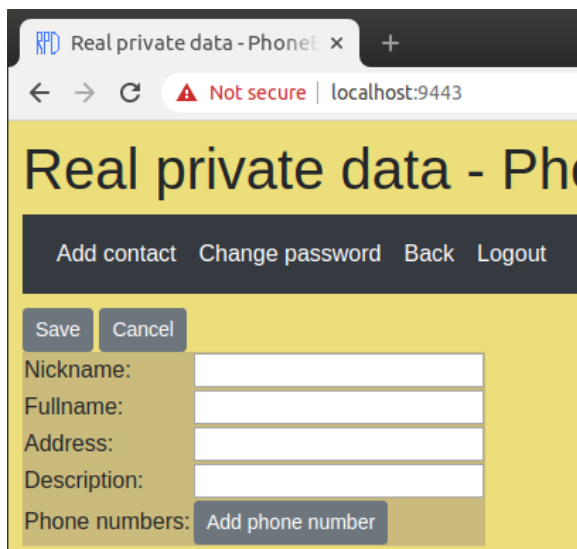


The screenshot shows a contact entry for 'Laura'. At the top, there is a yellow header bar with the name 'Laura' in blue text, and two grey buttons labeled 'Modify' and 'Delete'. Below the header, the contact details are listed: Nickname: Laura, Fullname: Laura Brown, Address: UK, London. Under 'Phone numbers:', there is a list with 'mobil' and '111 34235', each with its own 'Modify' and 'Delete' buttons. Below the phone numbers, there is an 'Add phone number' button. At the bottom, the 'Description' is 'My coworker'.

16. ábra: Telefonkönyv kontakt

2.8 Használat

2.8.2.12 Telefonkönyv – Új kontakt hozzáadása



The screenshot shows a web browser window with the title 'Real private data - Phone'. The address bar indicates 'localhost:9443' and a 'Not secure' warning. The page has a yellow header with the title 'Real private data - Phone'. Below the header is a dark navigation bar with links: 'Add contact', 'Change password', 'Back', and 'Logout'. The main content area is yellow and contains a form for adding a new contact. At the top of the form are 'Save' and 'Cancel' buttons. The form fields are: 'Nickname:' (text input), 'Fullname:' (text input), 'Address:' (text input), 'Description:' (text input), and 'Phone numbers:' (text input) with an 'Add phone number' button next to it.

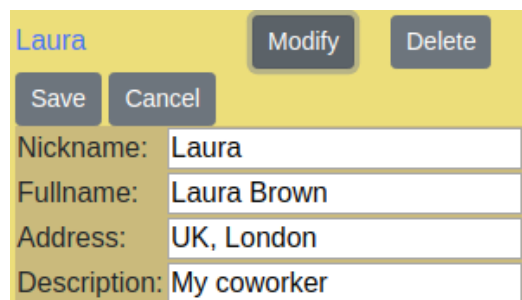
17. ábra: Új kontakt hozzáadása űrlap

Save gomb: Elmenti az új kontaktot

Cancel gomb: Megszakítja a kontakt hozzáadását

Nickname mező: A kontakt beceneve

2.8.2.13 Telefonkönyv kontakt – Módosítás



The screenshot shows the 'Modify' form for a contact named 'Laura'. The form has a yellow header with the name 'Laura' and buttons for 'Modify' and 'Delete'. Below the header are 'Save' and 'Cancel' buttons. The form fields are: 'Nickname:' (text input with 'Laura'), 'Fullname:' (text input with 'Laura Brown'), 'Address:' (text input with 'UK, London'), and 'Description:' (text input with 'My coworker').

18. ábra: Kontakt módosítás

Save gomb: Elmenti a módosítást

Cancel gomb: Megszakítja a módosítást

Nickname mező: A kontakt beceneve

Fullname mező: A kontakt teljes neve

Address mező: A kontakt címe

Description mező: A kontaktról egy leírás

2.8 Használat

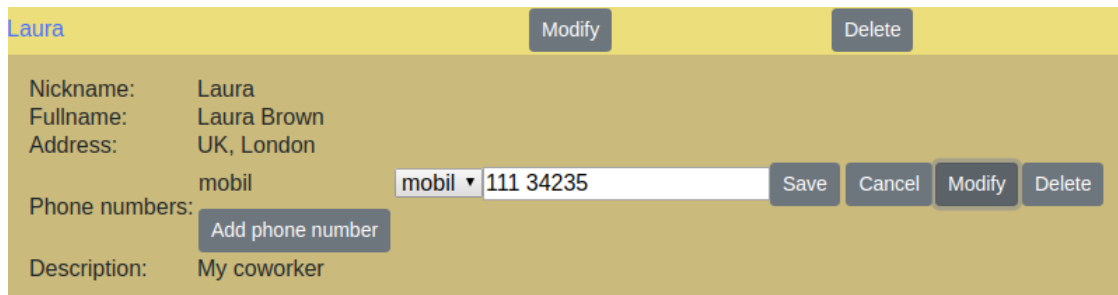
2.8.2.14 Telefonkönyv kontakt – Telefonszám hozzáadás

Típus mező: mobil, vagy office (irodai) vagy home (otthoni) lehet a telefon típusa

Szám mező: a telefonszám

Save gomb: Elmenti a módosítást

2.8.2.15 Telefonkönyv kontakt – Telefonszám módosítás



19. ábra: Telefonszám módosítás űrlap

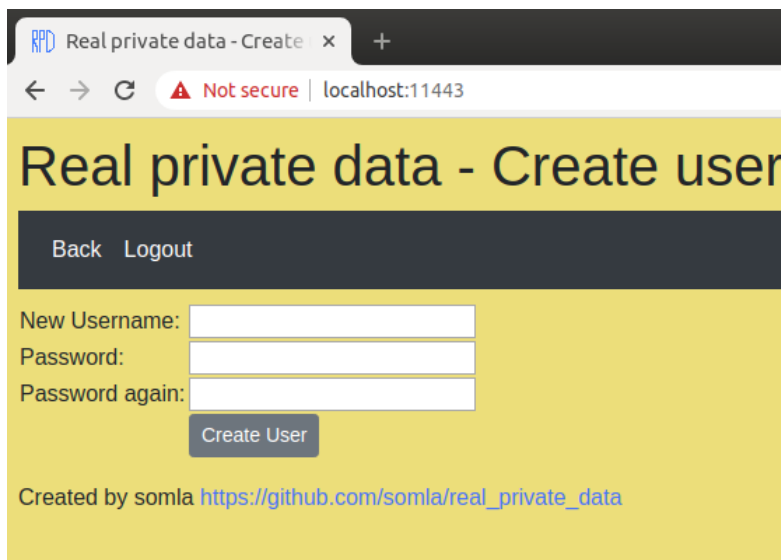
Típus mező: mobil, vagy office (irodai) vagy home (otthoni) lehet a telefon típusa

Szám mező: a telefonszám

Save gomb: Elmenti a módosítást

Cancel gomb: Megszakítja a módosítást

2.8.2.16 Felhasználó létrehozása



20. ábra: Új felhasználó létrehozása űrlap

New Username mező: Új felhasználó neve

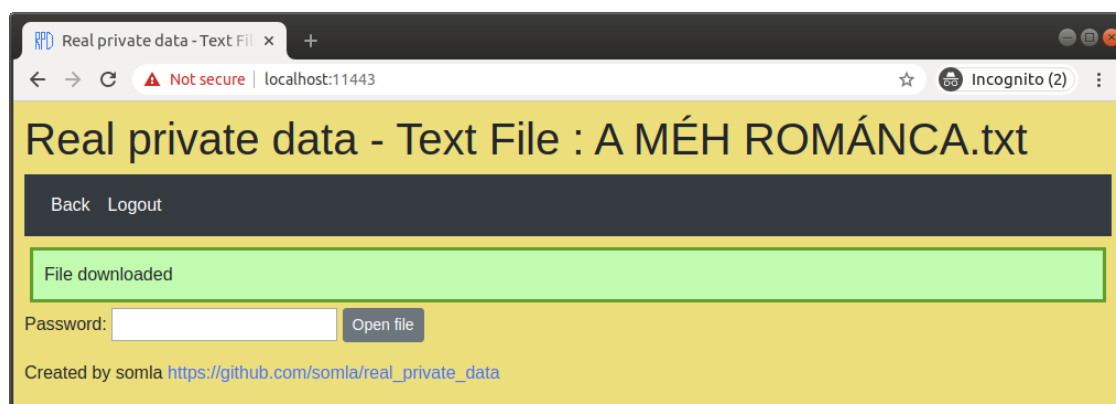
Password mező: Új felhasználó jelszava

Password again mező: Új felhasználó jelszava megint

2.8 Használat

Create User gomb: A felhasználó létrehozása

2.8.2.17 Üzenetek megjelenítése

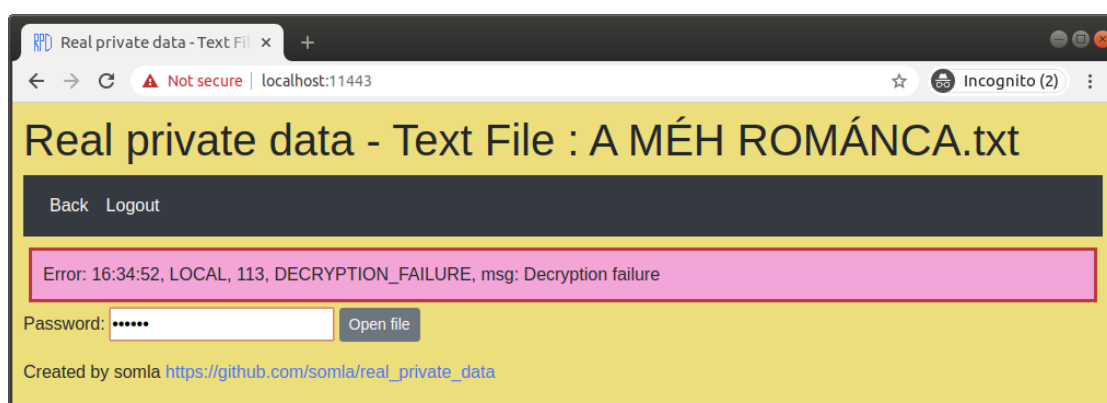


21. ábra: Üzenet megjelenítése (valami sikeres volt)

Az üzenetek a menüsor alatt jelennek meg. Ha valami sikeres volt, akkor zöld háttérük lesz, ha sikertelen, akkor piros. A hibaüzenetekről lásd a **Hibaüzenetek** fejezetet

2.8.2.18 Hibaüzenetek

Két fajta hiba lehet, lehet kliens oldali (**LOCAL**) és lehet szerver oldali (**REMOTE**) hiba. A kliens oldali hibák kódjai **1xx** a szerver oldaliak **2xx** alakúak. Egy hibának van kódja, hibaüzenete, és esetleg további adata (például távoli függvény hibánál a függvény hibaüzenete)



22. ábra: Hibaüzenet megjelenítése

2.8 Használat

2.8.2.18.1 Lokális, kliens oldali hibák (LOCAL)

Kód	Hibanév	Hibaüzenet (angolul)	Hibaüzenet (magyarul)
101	CONNECTION_ERROR	Connection error	Kapcsolat hiba
102	LOCAL_CALL_ERROR	Function call local error	Lokális függvényhiba
103	ALREADY_LOGEDIN	You are already logged in	Már be vagy jelentkezve
104	EMPTY_USERNAME_PASSWORD	Empty username and/or password and/or password again	Üres felhasználónév és/vagy jelszó és/vagy a jelszó még egyszer mező üres
105	PASSWORD_NOT_EQUAL_PASSWORD2	Password and password again is not equal	A jelszó és a jelszó még egyszer nem egyezik
106	PASSWORD_PASSWORD2_EMPTY	Password , and/or password again is empty	Üres jelszó mező
107	PASSWORD_PASSWORD2_OLD_PASSWORD_EMPTY	Password , and/or password again is empty and or oldPassword	A régi jelszó és/vagy az új jelszó és/vagy az új jelszó még egyszer üres
108	EMPTY_FILE_FIELD	File field is empty	A fájl mező üres
109	DOWNLOAD_ERROR	Download error	Hiba a letöltéskor
110	CONTACT_ALREADY_IN_LIST	The contact has been already in the contact list	A kapcsolat már a kapcsolat listában van
111	CONTACT_NOT_FOUND	Contact not found	A kapcsolat nem található
112	SUDDENLY_LOGGED_OUT	Suddenly logged out	Hirtelen kijelentkezés történt
113	DECRYPTION_FAILURE	Decryption failure	A visszafejtés sikertelen

2.8 Használat

2.8.2.18.2 Távoli, szerver oldali hibák (REMOTE)

Kód	Hibanév	Hibaüzenet (angolul)	Hibaüzenet (magyarul)
201	MISSING_USERNAME_PASSWORD	Missing username and/or password	Hiányzó felhasználónév és/vagy jelszó
202	BAD_USERNAME_PASSWORD	Bad username and/or password	Hibás felhasználónév és/vagy jelszó
203	REMOTE_FUNCTION_ERROR	Remote function error	Távoli függvény hiba
204	USER_REGISTRATED	User has been already registrated	A felhasználó már regisztrálva van
205	DISABLED_CREATE_USER	Disabled create new user	Nincs engedélyezve új felhasználó hozzáadása
206	FILE_EXIST	File has been already exist	A fájl már létezik
207	FILE_NOT_EXIST	File not exist	A fájl nem létezik

3 Fejlesztői dokumentáció

A projekt két részből áll. Egy szerverből, és egy kliensből. A szerver pythonban íródik, a kliens JavaScript-ben.

A szervernek két szerepe van.

1. a statikus (html, és JavaScript) fájlok kiszolgálása
2. a felhasználók fájljainak tárolása (fontos, hogy a szerver a felhasználókról minél kevesebbet tudjon, így minden, titkosan fog érkezni a szerverhez: felhasználó név, jelszó, fájlnev, fájl tartalom.)

3.1 Adattárolás

3.1.1 Felhasználói adatok tárolása a szerveren

A szerveren van egy mappa a felhasználóknak, ezt `adata_dir` konfigurációval állíthatjuk be, hogy hol legyen. Ebben a mappában minden felhasználónak létrehozunk egy új mappát, amiben a már titkos adatokat tároljuk. A felhasználó mappájába a már előre titkosított fájlok vannak titkosított névvel.

3.1.1.1 Felhasználó mappa generálása

Bemenet: felhasználónév, jelszó

1. `shaAlgoritmus = SHA256Salty(theConfig.server_salt)`
2. `felhasználó_hash = shaAlgoritmus(felhasználónév)`
3. `jelszó_hash = shaAlgoritmus(jelszó)`
4. `felhasználó_mappa = shaAlgoritmus(concat(felhasználó_hash, jelszó_hash))`

Megjegyzések

1. a 2. és 3. lépés jellemzően a kliens oldalon történik meg (csak a CLI regisztrálásnál történik szerver oldalon), így a szerver már a felhasználó nevét is titkosan kapja meg.
2. Az `SHA256Salty` algoritmus kicserélhető, és ki is kell cserélni hosszútávon valami lassabbra
3. A `theConfig.server_salt` konfigurálható, lásd a konfiguráció fejezetet
4. az `SHA256Salty` visszatérési értéke hexadecimális számrendszerben ábrázolt számok (00-ff)

3.1 Adattárolás

5. Az SHA256Salty algoritmust lásd lejjebb

3.1.2 Fájlnevek

A titkosítatlan fájlnevek tartalmazzák a fájlok kiterjesztését (jelenleg .txt vagy .phb (szöveges fájl, vagy telefonkönyv fájl)

A fájlnevek a következőképpen generálódnak (ez minden esetben a kliens oldalon történik)

Bemenet: jelszó, fájlnev

1. shaAlgoritmus = SHA256Salty(theConfig.server_salt)
2. fájl_név_hash = shaAlgoritmus(concat(jelszó, jelszó))
3. fájl_név = AESEncryptor(fájl_név_hash, fájl_név)

Megjegyzések:

1. Az shaAlgoritmus cserélhető, és érdemes is lesz valami lassabbra cserélni
2. AESEncryptor algoritmust lásd lejjebb. (ez is cserélhető lesz)
3. Jelenleg ugyanazt a jelszót használom a fájlok nevének titkosításához, mint a szerver eléréséhez, de másképp hash-elem le. (a szervernél szimplán a jelszót hashelem, itt meg a jelszót kétszer leírva hashelem, de ha a szerver oldali jelszót sikerül feltörni, akkor ezt is.

Titkosított fájl (SecretFile)

A titkos fájl két részből áll:

Van egy fájlnev (lásd feljebb), és egy fájl tartalom.

A Titkosított fájl tartalmának felépítése:

titkosítatlan_tartalom = concat (időbélyeg, "|", tartalom)

titkosított tartalom: AESEncryptor(fájl jelszó, titkosítatlan_tartalom)

Megjegyzések

1. az időbélyeg 1970.január 1. 0:00:00 másodpercétől eltelt másodpercek száma
2. Az AESEncryptor majd kicserélhető lesz

3.1 Adattárolás

3.1.3 AESEncryptor formátuma

1. hash = sha256(adat)
2. titkosítatlan = concat(hash, adat)
3. titkosított = AES256(titkosítatlan)

Megjegyzés

1. A hash egy ellenőrző összeg, és mindig 64 hosszú string (0-f), mert 16-os számrendszerben van ábrázolva
2. A kimenet byte sorozat.

3.1.4 Txt fájl (.txt) formátuma

A txt fájl 4 rétegből áll.

1. titkosítatlan szöveg (txt)
2. Minden txt fájl egy Titkosított fájl (<időbélyeg>|<txt>)
3. Minden Titkosított fájl AESEncryptor-t használ, így (<hash><időbélyeg>|<txt>) alakú
4. Az AES titkosított adat

3.1.5 Telefonkönyv fájl (.phb) formátuma

Három rétegből áll:

1. Titkosított adat (AESEncryptor-ral titkosítva)
2. Az első kicsomagolás után

```
{
  "becenév1":<titkosított kontakt adat>,
  "becenév2":<titkosított kontakt adat>
}
```
3. Titkosítatlan kontakt adat

```
{
  "full_name": "<teljes név>",
  "phone_numbers": [
    {
      "type": "<telefon típus>",
      "number": "<telefonszám>"
    }
  ]
}
```

3.1 Adattárolás

```
}  
]  
"address":"<cím>",  
"description":"<leírás>"  
}
```

Megjegyzések:

1. Az AESEncryptor kicserélhető lesz
2. A <titkosított becenévadat> is ugyanazzal az jelszóval, és ugyanazzal az encryptorral van jelenleg titkosítva.
3. Mivel a <titkosított kontakt adat> külön titkosítva van, így a memóriában mindig csak egy kontaktnak látszanak az adatai.

3.2 Szerver oldal felépítése

Fontos, hogy a szerver oldalt modulokból építsük fel, hogy könnyű legyen a modulokat cserélgetni.

3.2.1 Főbb modulok

config: Modul a konfigurációnak, fontos, hogy ezeket a konfigurációkat (vagyis ezek egy részét) a kliens oldalon is elérhetővé kell tenni.

A szervert lehessen json fájlból is és command line argumentumokkal is konfigurálni.

A konfigurációs argumentumoknál be lehessen állítani, hogy mi az alapértéke, láthassa-e a kliens és hogy kötelező-e megadni.

file_manager: A felhasználói mappák és fájlok kezeléséért felelős: létrehozás, törlés, átnevezés...

error_object: Központosított Error kezelés, a hibaüzeneteket és azok leírását kezeli, fontos, hogy kliens oldalon is elérhető legyen.

rpc_wrapper: A kliens oldalról elérhető függvények, az rpc protokolltól függetlenül.

Fontos szempont ezt külön kezelni mind a belső működéstől (**file_manager**) az autentikációtól (**auth_wrapper**), és az adatátvitel megvalósításától (**server**, **rpc_request_post_handler**, **rpc_request_ws_handler**)

server: Első körben **tornado** szervert használjunk, de gondoskodjunk arról, hogy ez cserélhető legyen.

3.2 Szerver oldal felépítése

Itt biztosítuk a kliens statikus fájljainak elküldését is, és adjunk lehetőséget RPC csatlakozásra is

create_user – legyen egy modulunk szerver oldalon is, ami hasonlóan titkosítja a felhasználó adatokat, mint kliens oldalon, hogy szerver oldalról is tudjunk létrehozni felhasználókat CLI felületről.

3.2.2 RPCWrapper osztály

Ez az osztály a külső interface.

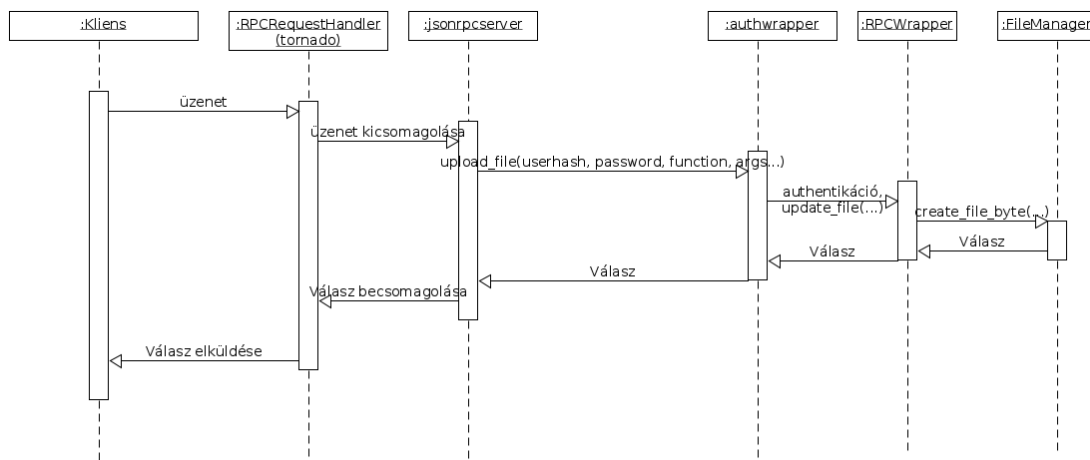
RPCWrapper
file_manager
<pre>add_other_name(x, y) change_password(old_user_hash, old_password_hash, new_user_hash, new_password_hash, files) create_user(new_userhash, new_passshare) del_file(file_name, __userhash__, __passshare__) download_file(encryptedName, __userhash__, __passshare__) list_dir(__userhash__, __passshare__) login(userhash, passshare) ping() rename_file(old_name, new_name, __userhash__, __passshare__) replay(message) upload_file(encryptedName, encryptedContent, is_new, __userhash__, __passshare__)</pre>

23. ábra: RPCWrapper class diagramm

A függvényeket az auth_wrapper fogja meghívni, és a **__userhash__** és **__passshare__** változókat ő fogja átadni.

3.2 Szerver oldal felépítése

3.2.3 Komponensek kommunikációja



24. ábra: Komponensek kommunikációja

Mint látható, a kommunikáció rétegekre van bontva, minden réteg cserélő. A példa egy fájl feltöltésének szekvenciáját mutatja be.

Az **RPCRequestHandler** a legelső réteg, ez felel a kommunikációért a szerver és a kliens között. Jelenleg két megvalósítás érhető el:

- **RPCRequestWSHandler:** A kommunikáció WebSocket protokollon keresztül történik
- **RPCRequestPOSTHandler:** A kommunikáció HTTP POST protokollon történik

A következő réteg a **jsonrpcserver** melynek a feladatai a következők:

1. A kientől érkezett kérések értelmezése
2. Kinyerni, az autentikációs adatokat
3. Kinyerni a felhasználó, melyik függvényt szeretné meghívni, milyen paraméterekkel
4. Átadni az autentikációs rétegnek
5. A visszakapott eredményt szerializálni
6. A szerializált adatot továbbítani a **RPCRequestHandler** rétegnek.

3.2 Szerver oldal felépítése

Ez a réteg is cserélhető lesz, de jelenleg csak egy implementációja van, ami JSON-t használ a szerializálásra.

Az **authwrapper** feladata az autentikáció biztosítása, egyes függvényeknek át kell adnia a felhasználó hash-t és jelszó hash-t.

Ha az autentikáció sikerült, akkor át kell adni az információkat az **RPCWrapper** rétegnek. Jelenleg csak egy fajta autentikáció létezik, minden alkalommal bekéri a hashelt felhasználónevet és jelszót, de ezt is ki lehet cserélni.

Az **RPCWrapper** a publikusan elérhető függvények listája, ez hívja meg az utolsó réteget a **FileManager** réteget.

A **FileManager** réteg a fájlok tényleges fájlműveleteket végzi. Ez is cserélhető, De jelenleg csak egy verzió érhető el, de lehetne csinálni valamilyen SQL adatbázis alapút, vagy egy tömörített zip fájlban is tárolhatnám az adatokat.

3.3 Kliens oldal felépítése

A kliens oldal MVC szerint íródott.

A View megvalósítása egyszerű html fájlokkal történik a **html/** mappában, ezt a **pageloader** componens kezeli.

A Controller a **js/components/controller** mappában lévő JavaScript fájlok

A Modell több modulból tevődik össze:

1. **UserManagerService** – A felhasználó kezelésért felelős
2. **DirManagerService** – A felhasználó mappájának kezelése
3. **FileFactory** – Secret fájl osztály létrehozásáért felelős
4. **SecretFile, PhoneBookNumber, TxtFile** – A fájlok viselkedéséért felel

A szerverrel való kommunikációért két réteg felel.

Az **RPCWrapperService**, ami a szerveren meghívható függvények gyűjteménye (lásd feljebb **RPCWrapper** osztály fejezet), ez egy Singleton, amit a **theRpcWrapper** globális változóba hozunk létre a program futásának indításakor ez az osztály a **theRpcClient call** függvényét hívja meg.

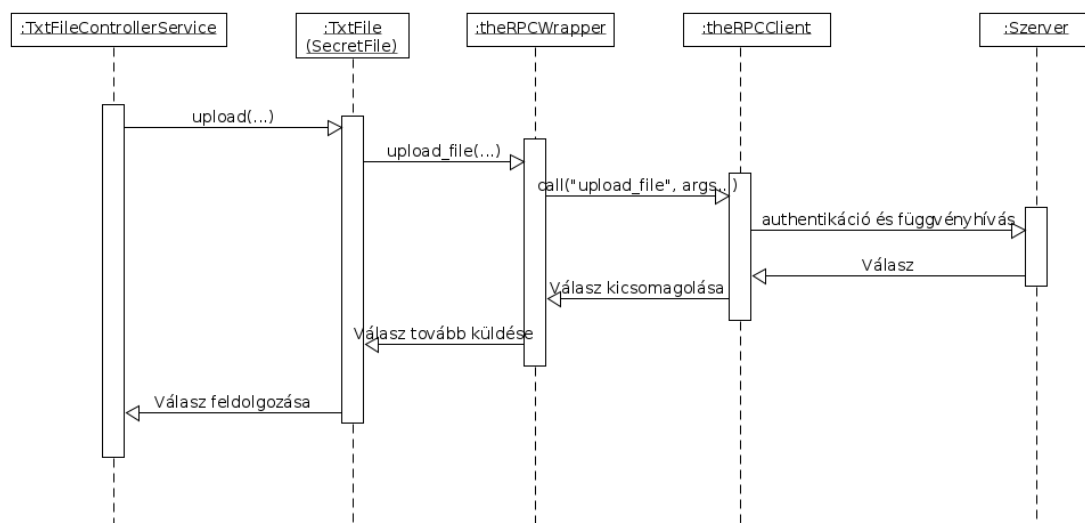
És az **IRPCClient**, ami a tényleges kommunikációért felel.

3.3 Kliens oldal felépítése

Az **IRPCClient** egyik leszármazottja egy Singleton, ami a **theRpcClient** globális változóba kerül bele.

A **SimpleJsonRpcWebSocketClientService** és a **SimpleJsonRpcPOSTClientService** a két megvalósítása az **IRPCClient**-nek, az előbbi WebSocket protokollt használ, a másik HTTP POST protokollt a kommunikációért. Ezen protokollok felett mindkettő a SimpleJsonRpc-t használja.

3.3.1 Komponensek kommunikációja



25. ábra: Kliens oldali komponensek kommunikációja

A fenti példa bemutatja a kliens oldalon a komponensek kommunikációjának szekvenciáját a fájlfeltöltésen keresztül.

Az első réteg a **Controller**, ennek a feladata a felhasználó kiszolgálása, és a modell réteg elérése (az **MVC** szerint.)

A következő réteg a **Model**, részei:

1. **SecretFile** – A fájlokért felel, titkosítás, módosítás, létrehozás, stb.
1. **TxtFile** – A szöveges fájlokért felel
2. **PhonebookFile** – A telefonkönyv fájlokért felel
2. **DirManager** – A felhasználómappa tartalmáért felel, listázás, törlés belőle, stb.

3.3 Kliens oldal felépítése

3. UserManager – A felhasználóért felel.

A következő réteg a **theRPCWrapper** – a szerveren meghívható függvények gyűjteménye, ennek a feladata az **theRPCClient** részére elküldeni a hívandó függvényt, és annak paramétereit.

Ezután jön a **theRPCClient** ami a szerverrel való kommunikációért, és az autentikációért felel.

3.4 Megvalósítás

A megvalósításhoz Visual Studio Code-ot használtam Ubuntu linux alatt.

3.4.1 Fontosabb algoritmusok

3.4.1.1 SHA256Salty

Leírás: Ez az algoritmus egy hash-t csinál, ami szóva van.

Bementet: str, salt_string

Algoritmus:

```
hexString(SHA256(preSalt(str,salt_string)))
```

Ahol preSalt(str,salt_string):

```
res = ""
```

```
amíg i = 0-tól (n-1)-ig:
```

```
saltChar = salt_string[i moduló hossz(salt_string)] || "\0"
```

```
res = res + fromCharCode((charCode(str[i]) + i * charCode(saltChar) ) % 256)
```

```
res += saltChar;
```

```
amíg vége
```

```
return res;
```

ahol fromCharCode egy számból adja vissza az ott elhelyezkedő ASCII karaktert, és charCode pedig egy ASCII karakterből adja vissza az ASCII kódját.

3.4.1.2 AESEncryptor

Leírás: Ez az algoritmus adatokat titkosít AES256 algoritmussal

Bemenet: data, jelszó_hash

3.4 Megvalósítás

Algoritmus:

```
ellenőrző_hash = sha256(data);  
data = ellenőrző_hash + data  
textBytes = utf8ToBytes(data)  
encryptedBytes = AESencrypt(textBytes)  
return encryptedBytes
```

3.4.2 Döntések a fejlesztés során

A szerver oldal programozási nyelveként négy jelöltem volt **NodeJS**, **PHP**, **Python** és **Java**.

A **Java** mellett az szólt, hogy az egyetemen azt tanultam, nagy tudású, gyors, de azért nem döntöttem mellette, mert most egy prototípust csinállok, ahol jó, ha menet közben tudok könnyen változtatni, a Java nem egy script nyelv, mindig le kell fordítani, ez a fejlesztés közben idő, és ráadásul memória igényes is.

A **NodeJS**, és a **PHP** mellett az szólt, hogy mindkettő elég elterjedt, szkript nyelv back end terén, és ezek is elég nagy tudásúak. Ellenük szólt, hogy egyikben sincs nagy tapasztalatom.

Így maradt végül a **Python**, amit a MorganStanleynél sokat kellett használnom.

Három esélyes könyvtáram volt a szerver megvalósítására: a **tornado**, a **Django** és a **Flask**, ezeket ajánlották ismerőseim, kinek melyik tetszett, de mivel a MorganStanleynél két évig kellett foglalkoznom a **tornadoval** és nagyon meg voltam vele elégedve, ezért választottam. Ami még fontos volt a **tornado** mellett, hogy natívan támogatja a WebSocket protokollt és a HTTPS-t, ellentétben a másik kettővel, amikhez külön WebSocket kiegészítőket kell illeszteni.

Az RPC kommunikációra azért választottam a **Simple JSON-RPC**-t mert elterjedt és egyszerű, itt gondolkodtam még, hogy a **Google Protobuf**-ját használom, de végül két okból nem azt választottam.

Egyrészt bonyolultabb, megpróbáltam összekötni a tornadoval, és nem triviális, másfelől a debuggolás is nehezebb, mert binárisan küldi az adatokat, így nehéz látni menet közben, hogy épp titkos-e az adat, vagy nem.

3.4 Megvalósítás

A **Simple JSON-RPC**nél ez sokkal egyszerűbb, mert JSON-okat küld, így nem kell bináris adatokat visszafejteni, de a **protobuf**-nak az az előnye, hogy sokkal kisebb forgalmat generál, így lehet, hogy hosszútávon ki lesz cserélve.

Igyekeztem a szervert és a klienst minél inkább modulárisan megírni, hogy minden részét könnyen ki lehessen cserélni, továbbá egy RPC interface-t csinálni, hogy akár magát a szerver és a kliens nyelvét is ki lehessen cserélni. Erre azért is gondoltam, mert majd szeretnék csinálni egy klienst mobiltelefonokra is.

A kliens oldalt először WEBre akartam megírni, hogy minél több eszközön lehessen futtatni. Itt három nyelv jött szóba: **TypeScript**, **JavaScript**, **WebAssembly**.

TypeScripttel próbálkoztam, de a debuggolás még nehezebb, mint JavaScriptben, meg ott is kell fordítani, amivel kevésbé rugalmas, mint a JavaScript.

A WebAssembly-nek két előnye van a másik kettőhöz képest. Egyfelől sokkal gyorsabb kódokat lehet vele generálni, másrésről a memória kezelése sokkal tisztább, abban a nyelvben nincs garbage collector, így biztos lehetek benne, hogy a felszabadított memória tényleg felszabadul, amivel jobban tudom garantálni a biztonságot.

Végül azért maradtam a JavaScript mellett, mert ez egyszerű, jól debuggolható, és minden gyorsan tesztelhető vele. Kíváló nyelv prototípus készítésére.

3.4.3 Szerver oldal mappa és fájl struktúrájának áttekintése

src/web

- | — [common](#) – A gyakran használt egyszerű függvények, osztályok helye
- | | — [dictgenerator.py](#) – JSON-t generál dictionaryból
- | | — [generatedby.py](#) – HTML és JavaScript fájlokba írja bele, melyik fájl, és mikor generálta (még fejleszteni kell)
- | | — [__init__.py](#) – Csomag init fájlja
- | | — [kill_log.py](#) – Ha valamiért leáll a program, loggoljuk ki
- | | — [singleton.py](#) – Singleton osztály
- | | — [the_project_paths.py](#) – A projekt fontosabb mappáit tartalmazza
- | — [config](#) – A projekt összes beállítását itt egyben kezelem
- | | — [arg.py](#) – A beállítások argumentum típusai
- | | — [config_base.py](#) – A the_config öse. Garantálja, hogy singleton legyen

3.4 Megvalósítás

- | |— [config_factory.py](#) – A config osztály létrehozója
- | |— [__init__.py](#) – A csomag inicializálója
- | |— [the_config.py](#) – A konfigurációs opciók (ide lehet új beállításokat rakni)
- | |— [the_config_variables.py](#) – A konfigurációs változók, amiket aztán fel lehet használni a konfiguráció írásakor
- | |— [tools.py](#) – Segéd eszközök a konfigurációhoz
- |— [data_manager](#) – A felhasználói adatok kezelése
- | |— [file_manager.py](#) – Fájlok (és mappák) kezelése
- | |— [__init__.py](#) – A csomag inicializálója
- |— [error_object](#) – Központosított hiba objektumok, a szerveren és a kliens oldalon is elérhetőek
- | |— [enum2.py](#) – Enum kibővítése
- | |— [error_object.py](#) – Egy Hiba objektum osztálya
- | |— [error_type_enum.py](#) – A hibák típusának enum-ja
- | |— [error_type.py](#) – Egy hiba típus osztálya
- | |— [error_types.py](#) – Az összes fajta hiba típus (új hibatípust ide kell felvenni)
- | |— [get_error_type_dict.py](#) – Egy json-t csinál a hibatípusokból
- | |— [__init__.py](#) – A projekt inicializálója
- |— [js_generator](#) – JavaScript-et generál
- | |— [__init__.py](#) – A projekt inicializálója
- | |— [js_generator.py](#) – JavaScript generátor
- |— [log](#) – Logokért felelős csomag
- | |— [__init__.py](#) – A csomag inicializálója
- | |— [log.py](#) – A python logging inicializálása
- |— [rpc_wrapper](#) – A szerver és a kliens közötti kommunikációért felelős osztály
“felső rétege”, a meghívható függvények gyűjteménye, és
autentikálás biztosítása
- | |— [auth_wrapper.py](#) – A hitelesítésért felelős wrapper függvény
- | |— [__init__.py](#) – A csomag inicializálása
- | |— [rpc_wrapper.py](#) – A függvények, amiket a kliens is elér
- | |— [web_method.py](#) – Jelzi az `rpc_wrapper`-ben, ha egy függvény elérhető a kliens

3.4 Megvalósítás

oldalon is

— [server](#) – Tornado webserver, mind a statikus adatok kiszolgálásáért, mind az RPC “alsó rétegéért felel”

| — [data_request_handler.py](#) – A generált adatokért: felel [generated/data.js](#)

| — [__init__.py](#) – A csomag inicializálása

| — [redirector_request_handler_factory.py](#) – HTTP szerver, ami átirányít a titkosított szerverre

| — [rpc_request_post_handler_factory.py](#) – JSON RPC hívások POST

protokollon keresztül

| — [rpc_request_ws_handler_factory.py](#) – JSON RPC hívások WebSocket protokollon keresztül

| — [rpc_wrapper_factory.py](#) – Az RPC szerver létrehozása

| — [web_request_handler_factory.py](#) – A statikus fájlok kiszolgálója

— [sha256Salty](#) – Hashelés SHA256 szózással csomagja

| — [__init__.py](#) – Csomag inicializálása

| — [sha256Salty2.py](#) – Az egyik szózott hash algoritmus

| — [sha256Salty.py](#) – A másik szózott hash algoritmus

— [config.json](#) – Config fájl (nem verziókövetett)

— [config.sample.json](#) – Config fájl példa verziókövetett

— [config.test.json](#) – Config fájl, a config.*.json fájlok nem verziókövetettek kivétel

a sample

— [create_user.py](#) – Felhasználó létrehozása

— [run_server.py](#) – szerver inicializálása, és futtatása

— [tools_create_config_csv.py](#) – A configokból csinál csv-t

— [tools_create_config_ods.py](#) – A configokból csinál ods-t

— [tools_create_html_file_dict.py](#) – A statikus html fájlok nevéből csinál egy JSON-

t

— [tools_getclasses.py](#) – Az összes html fájlból kinyeri a class mezőket

3.4 Megvalósítás

3.4.4 Kliens oldal mappa és fájl struktúrájának áttekintése

src/web

- | — [generated](#) – A generált fájlok mappája
- | | — [data.js](#) – Adatok (theConfig, errorObjects,...)
- | | — [theHtmlClasses.js](#) – HTML Classes
- | — [html](#) – A statikus html fájlok
- | | — [chgPassword.html](#) – Jelszóváltoztatáshoz a html fájl
- | | — [createUser.html](#) – Új felhasználó létrehozásához a html fájl
- | | — [filePassword.html](#) – A fájl jelszavának megadásához a html fájl
- | | — [login.html](#) – A bejelentkezéshez szükséges HTML fájl
- | | — [main.html](#) – A főoldal html fájlja
- | | — [newFilePassword.html](#) – Az fájl jelszavának megadásához HTML fájl
- | | — [phoneBookContact.html](#) - A Telefonkönyv kontaktjának HTML fájlja
- | | — [phoneBookFile.html](#) – A telefonkönyv HTML fájlja
- | | — [phoneBookModifyContact.html](#) – A telefonkönyv kontakt módosításának HTML fájlja
- | | — [txtFile.html](#) – A txt fájl HTML fájlja
- | — [js](#) – Az összes JavaScript
- | | — [common](#) – A gyakran használt JavaScript függvények, osztályok
- | | | — [js](#) – Alap JavaScript függvények, osztályo
- | | | — [AbstractClass.js](#) – Absztrakt osztály hack JavaScriptben
- | | | — [BigInttoJSONHack.js](#) – Nagy számok tárolása JSON-ban, mert a JavaScript alaptól nem támogatja
- | | | — [tools.js](#) – Pár hasznos függvény
- | | — [thirdparty](#) – Külső könyvtárak.
- | | | — [aes-js](#) – AES titkosításhoz könyvtár
- | | | — [index.js](#)
- | | | — [bootstrap-4.4.1-dist](#) – A menühöz függvénykönyvtár
- | | | — [css](#)
- | | | | — [bootstrap.css](#)
- | | | | — [bootstrap.css.map](#)

3.4 Megvalósítás

					└─	bootstrap-grid.css
					└─	bootstrap-grid.css.map
					└─	bootstrap-grid.min.css
					└─	bootstrap-grid.min.css.map
					└─	bootstrap.min.css
					└─	bootstrap.min.css.map
					└─	bootstrap-reboot.css
					└─	bootstrap-reboot.css.map
					└─	bootstrap-reboot.min.css
					└─	bootstrap-reboot.min.css.map
				└─		js
				└─		bootstrap.bundle.js
				└─		bootstrap.bundle.js.map
				└─		bootstrap.bundle.min.js
				└─		bootstrap.bundle.min.js.map
				└─		bootstrap.js
				└─		bootstrap.js.map
				└─		bootstrap.min.js
				└─		bootstrap.min.js.map
			└─			js-sha256 – A sha256-hoz függvénykönyvtár
			└─			sha256.js
			└─			sha256.min.js
			└─			simple-jsonrpc-js – Az RPC-hez függvénykönyvtár
			└─			simple-jsonrpc-js.js
			└─			simple-jsonrpc-js.min.js
			└─			bootstrap.min.js
			└─			bootstrap.min.js.map
			└─			jquery-3.4.1.slim.min.js – JQuery a bootstrap-hoz
			└─			popper.min.js – Popper a bootstrap-hoz
			└─			popper.min.js.map – Popper a bootstrap-hoz
		└─				components – A kliens komponensei

3.4 Megvalósítás

			— controller – Az oldalak kontrollerjei
			— ChgPasswordControllerService.js – Jelszó módosítás kontrollere
			— ControllerServiceBase.js – A kontrollerek őse, pár alap szolgáltatás van benne
			— CreateUserControllerService.js – Új felhasználó létrehozásának a kontrollere
			— LoginControllerService.js – A bejelentkezés kontrollere
			— MainControllerService.js – A főoldal kontrollere
			— PhoneBookFileControllerService.js – A telefonkönyvfájl kontrollere
(a			SecretFileController-ből származik
			— SampleControllerService.js – Egy példa kontroller
			— SecretFileControllerService.js – A titkos fájlok kontrollere
			— TxtFileControllerService.js – A txt fájlok kontrollere (a SecretFileController-ből származik)
			— file – A fájlkezeléssel foglalkozó osztályok mappája
			— DirManagerService.js – A mappakezeléssel foglalkozik
			— FileFactory.js – Fájl osztály létrehozása (.phb és .txt-is)
			— PhoneBookFile.js – Telefonkönyv fájl osztály (ami SecretFile)
			— SecretFile.js – Titkos fájl osztály
			— SecretJson.js – Titkos Json osztály
			— TxtFile.js – Txt fájl osztály, ami SecretFile
			— pageloader – Az oldalak letöltésével, és betöltésével foglalkozik
			— HtmlDownloader.js – A html oldalak letöltése
			— PageLoaderService.js – Az oldalak betöltése
			— UserManager – A felhasználókkal foglalkozik
			— UserManagerService.js – A felhasználók osztálya
			— UserManagerServiceMock.js – A felhasználók osztályának kimockolt verziója
			— interfaces – Interfészek
			— encrypt - Titkosítók

3.4 Megvalósítás

				—	Iencryptor.js	– Kétirányú titkosító interface
				—	Ihash.js	– Egy irányú titkosító interface
				—	file	– Fájl interfészek
				—	IsecretFile.js	– Titkos fájl interfész
				—	RPC	– RPC interfészek
				—	IRPCClient.js	– RPC kliens interfész
				—	UserManager	– User manager interfészek
				—	IuserManagerService.js	– User manager interfész
				—	lib	– Könyvtárak
				—	encrypt	– Titkosító könyvtárak
				—	AESEncryptor.js	– Kétirányú titkosító könyvtár AES-t használva
				—	SHA256Salty.js	– Egyirányú titkosító könyvtár SHA256-ot használva
					sózva	
				—	ErrorObject	– Hiba objektum könyvtár
				—	ErrorObject.js	– Hiba objektum könyvtár
				—	RPCWrapper	– RPC wrapper könyvtár
				—	RpcClients.js	– Az elérhető RPC kliensek tárolója
					(window.theRpcClients)	
				—	RPCWrapperService.js	– Az összes elérhető RPC függvény
				—	SimpleJsonRpc	– A SimpleJsonRpc osztályok az adatátvitelhez
				—	SimpleJsonRpcPOSTClientService.js	– RPC adatátvitel HTTP POST
					protokollal	
				—	SimpleJsonRpcWebSocketClientService.js	– RPC adatátvitel WS
					protokollal	
				—	test	– tesztek
				—	webtest	– webtesztek
				—	lib	– könyvtárak a webteszthez
				—	createFile.js	– robotkattintgatással fájl létrehozása
				—	login.js	– robotkattintgatással bejelentkezés
				—	phonebook.js	– robotkattintgatással telefonkönyv manipulációk
				—	tools.js	– eszközök a web-teszthez

3.4 Megvalósítás

- | | | — [dataMedia.js](#) – Telefonkönyv adatok az ELTE oldaláról
- | | | — [getDataFromMedia.js](#) – adatok letöltéséhez segéd szkript az ELET oldaláról
- | | | — [testPhoneBookAddContacts.js](#) – Egy teszt, ami létrehoz egy telefonkönyvet, és feltölti adatokkal
- | | — [htmlFileDict.js](#) – A htmlFájlok JSON-ja
- | | — [main.js](#) – A main JavaScript fájl
- | — [style](#) – Az oldal stílusa
- | | — [bootstrap.min.css](#) – Bootstraphoz stílus lap
- | | — [bootstrap.min.css.map](#) – Bootstraphoz stílus lap
- | | — [style.css](#) – saját stílus lap
- | — [favicon.ico](#) - ikon
- | — [index.html](#) - főoldal
- | — [testPhoneBook.html](#) – A telefonkönyv teszt indítása

3.4 Megvalósítás

3.4.5 Fejlesztési lehetőségek

A projekt még nagyon a kezdeti időszakában van, így rengeteg fejlesztési lehetőség van.

1. A kliens oldalt megvalósítani Android és PC alkalmazásként is.
2. Több típusú fájl, ne csak txt vagy telefonkönyv fájl, hanem
 1. Fájl típusú, amibe bármilyen fájlt le lehet titkosítani
 2. Naptár titkosítás.
3. Fájl importálás/exportálás.
4. Telefonkönyv importálás (főleg a majd készülő androidos verzióban)
5. UI-ról választható titkosítási forma, hashelési forma.
6. SSL helyett quantum safe kommunikáció
7. Szerver windows support.

3.5 Tesztelési terv

A txt fájlok teszteléséhez Arany János összes költeményeit használom:

<https://mek.oszk.hu/00500/00597/html/index.htm>

A telefonkönyvek teszteléséhez az ELTE honlapján elérhető telefonszámokat használom:

1. ELTE IK Média- és Oktatásinformatikai Tanszék > A Tanszékről > Oktatók és munkatársak

<https://mot.inf.elte.hu/munkatarsak>

3.5.1 Előkészületek

Létrehozok egy üres adatbázist a projekt mellé, és megcsinálom a szükséges config fájlt:

```
{  
  "host": "localhost",  
  "logFile": "/var/tmp/rpd_test_[[now]].log",  
  "open_port": 11080,  
  "secure_port": 11443,  
  "enable_create_user": true,  
  "logLevel": "DEBUG",
```

3.5 Tesztelési terv

```
"show_encrypted_data": true,  
"crt_file": "[[dir_project]]/../../key/rpd.crt",  
"key_file": "[[dir_project]]/../../key/rpd.key",  
"web_root": "[[dir_project]]/src/web",  
"data_dir": "[[dir_project]]/../../test_data"  
}
```

3.5.2 Eset 1: A szerver elindítása (black box)

`./run_server --configFile config.test.json`

Elvárt eredmény

1. Loggolja a konfigurált beállításokat:

```
[2020-04-08 12:25:01,276][INFO] Loglevel: INFO  
[2020-04-08 12:25:01,440][INFO] Runner command: ./run_server.py --configFile  
config.test.json  
[2020-04-08 12:25:01,440][INFO] Config:  
[2020-04-08 12:25:01,440][INFO] configFile: config.test.json  
[2020-04-08 12:25:01,440][INFO] debug: False  
[2020-04-08 12:25:01,440][INFO] logLevel: INFO  
[2020-04-08 12:25:01,440][INFO] logFile: /var/tmp/rpd_test_20200408_122501.log  
[2020-04-08 12:25:01,440][INFO] logFormat: [% (asctime)s][%(levelname)s] %  
(message)s  
[2020-04-08 12:25:01,440][INFO] show_rpc_message: False  
[2020-04-08 12:25:01,440][INFO] open_port: 11080  
[2020-04-08 12:25:01,440][INFO] debug_open_port: 8081  
[2020-04-08 12:25:01,440][INFO] secure_port: 11443  
[2020-04-08 12:25:01,441][INFO] host: localhost  
[2020-04-08 12:25:01,441][INFO] crt_file:  
/home/somla/working/rpd/master/../../key/rpd.crt  
[2020-04-08 12:25:01,441][INFO] key_file:  
/home/somla/working/rpd/master/../../key/rpd.key  
[2020-04-08 12:25:01,441][INFO] web_root: /home/somla/working/rpd/master/src/web
```

3.5 Tesztelési terv

```
[2020-04-08 12:25:01,441][INFO] data_dir:
/home/somla/working/rpd/master/../test_data
[2020-04-08 12:25:01,441][INFO] test_dir: /var/tmp/real_private_data
[2020-04-08 12:25:01,441][INFO] salt: My own Salt
[2020-04-08 12:25:01,441][INFO] server_salt: Server salt
[2020-04-08 12:25:01,441][INFO] enable_create_user: True
[2020-04-08 12:25:01,441][INFO] show_encrypted_data: True
[2020-04-08 12:25:01,441][INFO] defaultRpcClient:
SimpleJsonRpcWebSocketClientService
[2020-04-08 12:25:01,441][INFO] hideMessageTime: 4000
```

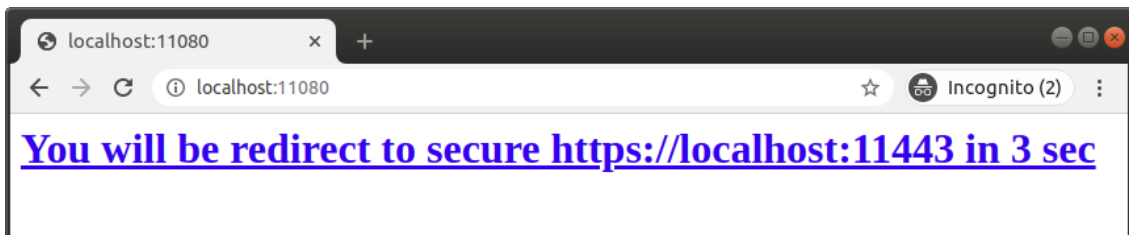
2. loggolja a szerver elérhetőségeit:

```
[2020-04-08 12:25:01,443][INFO] HTTPS Server starting... https://localhost:11443/
[2020-04-08 12:25:01,444][INFO] HTTP redirect Server starting...
http://localhost:11080/
```

3. írja ki őket a /var/tmp/rpd_test_*.log (a * helyére az aktuális dátumot várom)

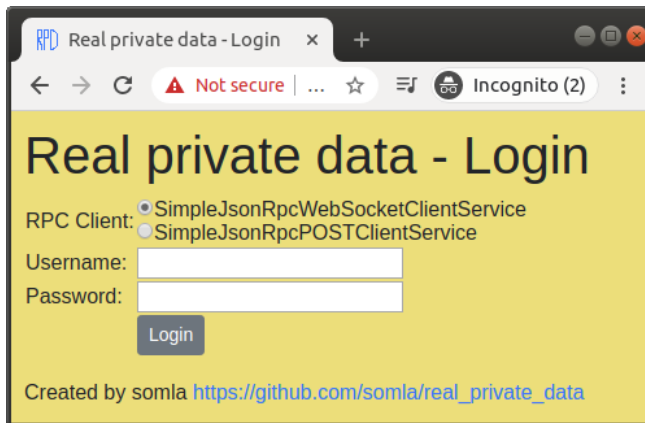
A /var/tmp/rpd_test_20200408_122501.log fájl tényleg létrejött, és tényleg ugyanaz van benne, mint a képernyőn.

4. A <http://localhost:11080/> -re kattintva jussunk el az átirányító oldalra, és az irányítson át minket a titkosított oldalra 3 másodperc múlva



26. ábra: Átirányító oldal

3.5 Tesztelési terv



27. ábra: Bejelentkező oldal

3.5.3 Eset 2: Felhasználók létrehozása konzolból (black box)

2 felhasználót fogok létrehozni

Felhasználónév: test_user1 **password:** password1

Felhasználónév: test_user2 **password:** password2

```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: test_user1
Enter your password:
Enter your password again:
Registration was successfully
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: test_user2
Enter your password:
Enter your password again:
Registration was successfully
~/working/rpd/master/src/python:master$
```

28. ábra: Felhasználó létrehozása CLI-ből

Elvárt eredmény

1. Hozzon létre két felhasználói mappát:
2. Be tudjak lépni a felhasználókkal, ezt lásd lejjebb a bejelentkezés tesztelésénél.

```
File Edit View Search Terminal Help
~/working/rpd/test_data$ ls
a6685c94348208f0316c8ba67b0df0897a7f820c286a126649c81bf42aa13fd2
d4efaef0a0d894920ccc97ada5a54f04555a1621d4c050e7af8348b598daeee7
~/working/rpd/test_data$
```

29. ábra: Titkos felhasználói mappák

3.5 Tesztelési terv

3.5.4 Eset 3: Üresen hagyott mezők felhasználó létrehozása közben

Vagy a felhasználónevet, vagy a jelszó mezőt, vagy mindkettőt hagyjuk üresen

Elvárt eredmény

```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username:
Enter your password:
Enter your password again:
Username and/or password is empty
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username:
Enter your password:
Enter your password again:
Username and/or password is empty
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: user
Enter your password:
Enter your password again:
Username and/or password is empty
~/working/rpd/master/src/python:master$
```

30. ábra: Hibás felhasználó létrehozás (üres felhasználónév vagy jelszó, vagy jelszó mégegyszer mező)

Username and/or password is empty üzenet, a data dir változatlan hagyása

3.5.5 Eset 4: Jelszó és jelszó mégegyszer nem egyezik (CLI)

```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: Gibbs Jakab
Enter your password:
Enter your password again:
password and password again is not equal
~/working/rpd/master/src/python:master$
```

31. ábra: A jelszó és a jelszó mégegyszer mező nem egyezik

Elvárt eredmény

Hibaüzenet, test_data dir ne változzon

3.5.6 Eset 5: Létező felhasználó hozzáadása azonos jelszóval

Meg kell jegyeztem, hogy itt a felhasználónév és a jelszó páros azonosít egy felhasználót, így például **User1/password1** és **User1/password2** nem ugyanaz a felhasználó.

Gondolkodtam ennek javításán, de nem igazán lehetséges úgy, hogy ne adjon többlet információt a szerver üzemeltetőjének a felhasználóról.

test_user1/password1

3.5 Tesztelési terv

```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: test_user1
Enter your password:
Enter your password again:
Error:User has been already registrated
~/working/rpd/master/src/python:master$
```

32. ábra: Létező felhasználó hozzáadása mégegyszer

Elvárt eredmény

Hibaüzenet, test_data dir ne változzon

3.5.7 Eset 6: Létező felhasználó hozzáadása más jelszóval (cli)

test_user1/password2 létrehozása

```
File Edit View Search Terminal Help
~/working/rpd/master/src/python:master$ ./create_user.py --configFile config.test.json
Enter your username: test_user1
Enter your password:
Enter your password again:
Registration was successfully
~/working/rpd/master/src/python:master$
```

33. ábra: Létező felhasználó hozzáadása más jelszóval

Elvárt eredmény

1. Hozzon létre egy új felhasználói mappát
2. Be tudjak lépni az új felhasználóval, ezt lásd lejjebb a bejelentkezés tesztelésénél.

```
File Edit View Search Terminal Help
~/working/rpd/test_data$ ls
703b4893807033a93c5c2782ea515205c2fccd1ee8cc8e7958ece471a1dbad2c
a6685c94348208f0316c8ba67b0df0897a7f820c286a126649c81bf42aa13fd2
d4efaef0a0d894920ccc97ada5a54f04555a1621d4c050e7af8348b598daeee7
~/working/rpd/test_data$
```

34. ábra: Új mappa létrejött a felhasználónak

3.5 Tesztelési terv

3.5.8 Eset 7: Belépés hibás jelszóval (GUI)

Bejelentkezés a következő felhasználókkal

Felhasználónév	Jelszó
I am not exist	I am not exist
test_user1	almafa
test_user2	dinnye

(SimpleJsonRpcWebSocketClientService és SimpleJsonRpcPOSTClientService segítségével is)

Elvárt eredmény

Hibaüzenet

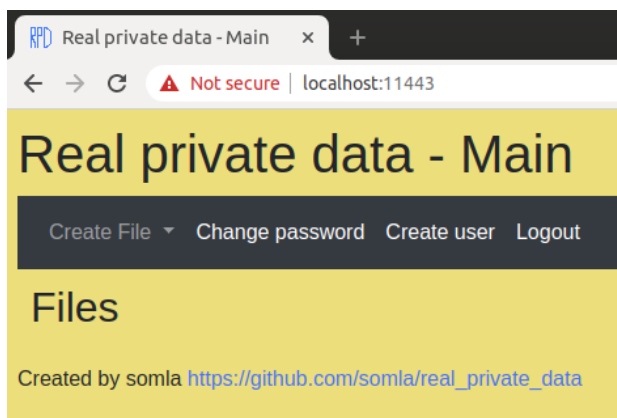
3.5.9 Eset 8: Bejelentkezés valós felhasználókkal (GUI)

Bejelentkezés a következő felhasználókkal (SimpleJsonRpcWebSocketClientService és SimpleJsonRpcPOSTClientService segítségével is)

Felhasználónév	Jelszó
test_user1	password1
test_user1	password2
test_user2	password2

Elvárt eredmény

Bejelentkezés az oldalra, és a main oldalra irányítás.



35. ábra: Főoldal (még nincs egy fájl sem)

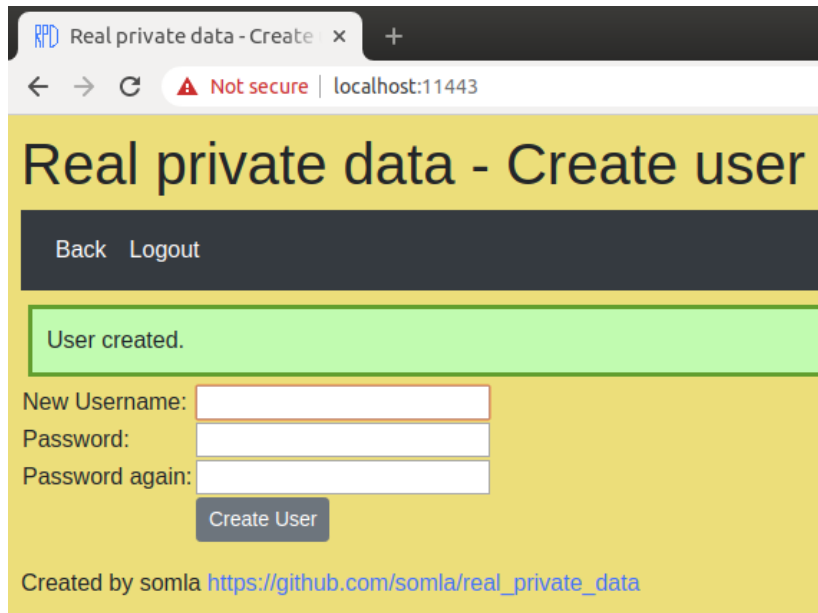
3.5 Tesztelési terv

3.5.10 Eset 9: Felhasználó létrehozása (GUI)

test_user3/password3 létrehozása

Elvárt eredmény

1. sikeres létrehozás
2. sikeres bejelentkezés az új felhasználóval (lásd 8. eset: Bejelentkezés valós felhasználókkal (GUI)).

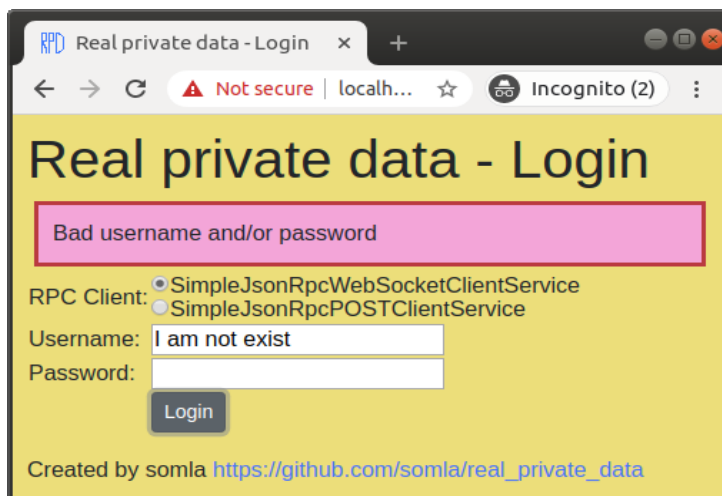


36. ábra: Új felhasználó létrehozva

3.5.11 Eset10: Bejelentkezés nem létező felhasználóval

Bejelentkezés „I am not exist” felhasználóval

Elvárt eredmény:



37. Ábra: Nem létező felhasználó

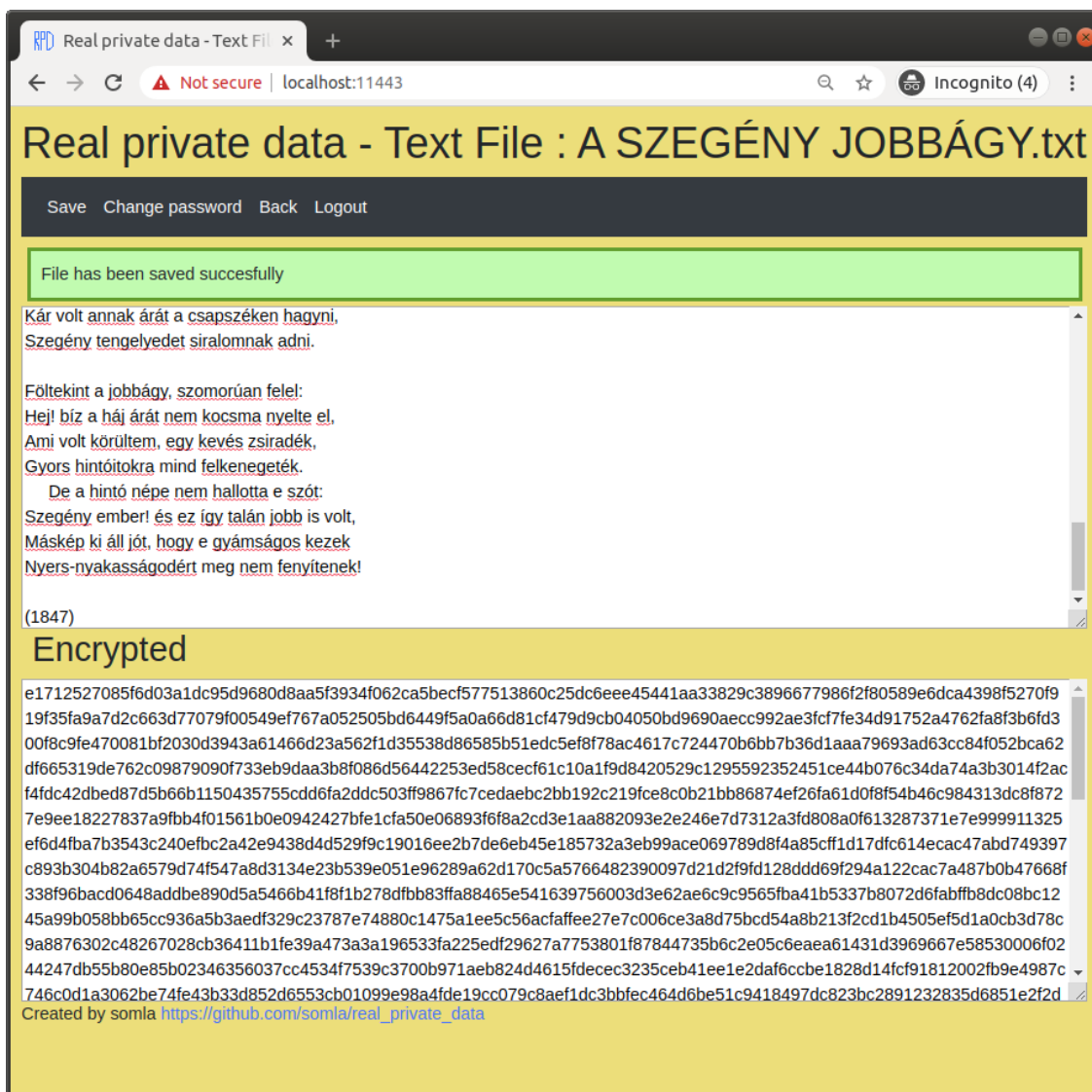
3.5 Tesztelési terv

3.5.12 Eset11: Txt Fájl létrehozása

Hozzunk létre pár txt fájlt.

Elvárt eredmény

A fájlok létrejönnek, és meg is tudjuk őket nyitni, lásd lejjebb



38. ábra: Új text fájl létrehozva

3.5 Tesztelési terv

3.5.13 Eset 12: Telefonkönyv fájl létrehozása

Ehhez csináltam egy teszt robotot, ami létrehoz egy Telefonkönyv fájlt, és feltölti adatokkal.

<https://localhost:11443/testPhoneBook.html>

waitTime: Megmondja, hogy mennyit várjon a teszt két művelet között

username: Melyik felhasználóval lépjen be

password: Mi a felhasználó jelszava

fileName: Mi legyen a létrehozandó fájl neve

filePassword: Mi legyen a jelszava

Real private data - Login

RPC Client: ☒ SimpleJsonRpcWebSocketClientService ☐ SimpleJsonRpcPOSTClientService

Username:

Password:

Login

Test

waitTime:

username:

password:

fileName:

filePassword:

Start Test

Created by somla https://github.com/somla/real_private_data

39. ábra: Telefonkönyv létrehozása robottal

Elvárt eredmény

A fájl létrehozása, és a telefonkönyv adatok tárolása.

3.5.14 Eset 13: Ellenőrizzük, hogy a szerveren tárolt adatok valóban titkosak!

1. Kिलistázzuk a mappákat tee paranccsal
2. megnézzük a fájlokat cat paranccsal
3. megnézzük a fájlokat hexdump paranccsal

```
File Edit View Search Terminal Help
~/working/rpd/test_data$ tree
.
├── 703b4893807033a93c5c2782ea515205c2fccd1ee8cc8e7958ece471a1db
ad2c
├── a6685c94348208f0316c8ba67b0df0897a7f820c286a126649c81bf42aa1
3fd2
│   ├── 003fc8ec29bc1a8c50f0c4408a674344176e37208b13fa017e0e32e6
46d9ca6b8b54d4258da4db6b9aa9e611ee252faa641e684129e07424e22807ea
3ea471b50bad5c2758bcc37bfc5eced07028fb66d7d085
│   ├── 043fceb26bd15d153a493458d344618413b6b27df1ef901290f3de1
408ac83fd80e83248af78a6d9ba5b21fef2579ab62166b142fe12421b2295de2
39ac74b50bad42be94b320a0fd4a2d1b7129f88b22efa86af09d91
│   ├── 066acdba28ea1ad053f2c4168c624b4f126a6a77d61af855225736e5
408a9d3fdd55d427d8f08a6acefdb512b6297ea9311b6e172be12425b77257e0
3aae21e40bdf50aa88b5aa64fb5c577f4a13ce
│   ├── 076ccdb42ce846df00f7ce11dc66404c133b3f25df13aa53230f36b9
168d9f32db0584278cf08a3cccacb013e4277bae641e60142ce02874e27e5ce7
3cf871e50fca48af9ea6aa09e7524392b33fec8b22e6a80bc9c5a0c49a4cf1ee
85b4c6640bfc21732aebcc512bf5
│   ├── 5136c9ef2de715d057a7c7108e3d444c433e6c71d848ff017c0567b6
1b8acf6bd501852388a38b6dcef9e740e3202efe30186f142ce87725e47906e1
6fab72e70bad47a583a620ba935f4892bf25e307e886853cf0
│   ├── 516fc9bd27ed1a8a50a094438d30441a416c3b22d818ae012d0f30b7
44dfcb6cdf54d3738df0da6ec9f8e113ef2378ff301f681578e82220e92a57b6
3aa824e81c4e90a890a7b909e3565994ae2df306e6e3df30fc91
│   └── 5668cdbf7ae6428851f293178132401d156e6927dd18f8042b5733b3
1b889f6c8b03d9278ea48f38ceacb611b5257efd644d61117eb32521b02d52b3
61a476b70fc145a1f1bda809fed0a435570a976860018264cb8e91e8a24cd9d6
a719364520c7838e01c5834c73d59bdd29d96144ff8f3385
├── cd1699c8134c04ca727a99e2652250e7fd37bbf3ce14e9c41a539fe67b3bf
e17a
└── d4efaef0a0d894920ccc97ada5a54f04555a1621d4c050e7af8348b598da
eee7

4 directories, 7 files
~/working/rpd/test_data$
```

40. ábra: Titkos felhasználónevek, és fájlnevek

Itt látható, hogy mind a mappák (felhasználók nevei), és a benne lévő fájlok nevei is titkosítottak.

3.5 Tesztelési terv

```
File Edit View Search Terminal Help
286a126649c81bf42aa13fd2$ cat 003fc8ec29bc1a8c50f0c4408a67434417
6e37208b13fa017e0e32e646d9ca6b8b54d4258da4db6b9aa9e611ee252faa64
1e684129e07424e22807ea3ea471b50bad5c2758bcc37bfc5eced07028fb66d7
d085
3~*0=wTfRSQDZ:~7ued}  N(
j@dew8x#R}M,
o)1h%`j Bb~VeeR1CxmGr{>Y{R$ 7=%
'200k|S%.PF, '=22[46iln&S!>~W
uHE7.UQHnq^|30F&UXS.HW

BV::Pod~q>
8w2L  m27  T*  >am`R}M]  5  p  N
<2W
weede{f  wwW:  Z  t3  *  N|w/cop7B"
Kopçg  1s8K[Pee  .ebb  .we
am>en3+t@²)#!2_%"-eg|ks|py|3MHT4?d[4}
{&Л  p  ;y$/  L  )Y  zX
d  Hoo  й+oG
_  wy  3  J  C
k  /d^*b/  wf*Δ!"  p
&
D  b  U_ox1J?  X  S  /  uk  P  (qG  rx  C  C  A
1U~  #  'N  V  !%  '  {  I  Z  `yZ  M  9YN
T  Vg  {  ^  I  ^  [  iYh  `  '  5  3  T  Rg  ~  i  K  -  q  F  9  &  i
|  =>  6  :  K  6  !  D  )  q  M  >  .  '  Z  ;  E  Z  J  0
(  ^  M  v  l  v
(  [  ]  AY+Lb  |  ,  [  ~  ,  Ć  Z  d  >  N
*  w  -  \  15  )
9
F  R  <  2  0  0  M  0  )  5  ?  *  [  D  s  W  <  .  t  l  ;  ,
9  Č  F  [  [  !  o  =  *  n  \  t  !  k  R  2  2  ?  Q  -  i  >
G  ?  1  P  j  ~  P  E  ^  Θ  0  9  .  !  H  ~  C  *  )  i
~/working/rpd/test_data/a6685c94348208f0316c8ba67b0df0897a7f820c
286a126649c81bf42aa13fd2$
```

41. ábra: Titkos fájl tartalma cat paranccsal megjelenítve

Itt látható, hogy a fájl cat-tal megjelenítve is értelmetlen

3.5 Tesztelési terv

3.5.15 A teszt közben létrejött felhasználók és fájlok

3.5.15.1 Felhasználók

Felhasználónév	Jelszó
test_user1	password1
test_user1	password2
test_user2	password2
test_user3	password3

3.5.15.2 Fájlok

Felhasználó	Fájl	Jelszó
test_user1/password1	A MÉH ROMÁNCA.txt	Petofi
test_user1/password1	A SZEGÉNY JOBBÁGY.txt	Petofi
test_user1/password1	A VARRÓ LEÁNYOK.txt	Petofi
test_user1/password1	ARANYAIMHOZ.txt	Petofi
test_user1/password1	EGYKORI TANÍTVÁNYOM EMLÉKKÖNYVÉBE.txt	Petofi
test_user1/password1	VÁLASZ PETŐFINEK.txt	Petofi
test_user1/password1	ELTE IK Média- és Oktatásinformatikai Tanszék.phb	Media

4 Forrásjegyzet

- [1] tornado hivatalos oldala: <https://www.tornadoweb.org/en/stable/> [Hozzáférés dátuma: 28 04 2020]
- [2] tornado GitHub oldala: <https://github.com/tornadoweb/tornado/> [Hozzáférés dátuma: 28 04 2020]
- [3] tornado PyPi oldala: <https://pypi.org/project/tornado/> [Hozzáférés dátuma: 28 04 2020]
- [4] jsonrpcserver GitHub oldala: <https://github.com/bcb/jsonrpcserver> [Hozzáférés dátuma: 28 04 2020]
- [5] jsonrpcserver PyPi oldala: <https://pypi.org/project/jsonrpcserver/> [Hozzáférés dátuma: 28 04 2020]
- [6] pandas hivatalos oldala: <https://pandas.pydata.org/> [Hozzáférés dátuma: 28 04 2020]
- [7] pandas GitHub oldala: <https://github.com/pandas-dev/pandas> [Hozzáférés dátuma: 28 04 2020]
- [8] pandas PyPi oldala: <https://pypi.org/project/pandas/> [Hozzáférés dátuma: 28 04 2020]
- [9] Pyexcel-ods GitHub oldala: <https://github.com/pyexcel/pyexcel-ods> [Hozzáférés dátuma: 28 04 2020]
- [10] Pyexcel-ods PyPi oldala: <https://pypi.org/project/pyexcel-ods/> [Hozzáférés dátuma: 28 04 2020]
- [11] aes-js NPM oldala: <https://www.npmjs.com/package/aes-js> [Hozzáférés dátuma: 28 04 2020]
- [12] Aes-js hivatalos oldala: <https://cdn.rawgit.com/ricmoo/aes-js> [Hozzáférés dátuma: 28 04 2020]
- [13] bootstrap hivatalos oldala: <https://getbootstrap.com/> [Hozzáférés dátuma: 28 04 2020]
- [14] jquery hivatalos oldala: <https://jquery.com/> [Hozzáférés dátuma: 28 04 2020]
- [15] Js-sha256 NPM oldala: <https://www.npmjs.com/package/js-sha256> [Hozzáférés dátuma: 28 04 2020]

4 Forrásjegyzet

[16] popper.js hivatalos oldala: <https://popper.js.org/> [Hozzáférés dátuma: 28 04 2020]

[17] Simple-jsonrpc-js GitHub oldala: <https://github.com/jershell/simple-jsonrpc-js>
[Hozzáférés dátuma: 28 04 2020]

[18] Google Chrome hivatalos oldala: <https://www.google.com/chrome/> [Hozzáférés dátuma: 28 04 2020]

[19] Mozilla Firefox hivatalos oldala: <https://www.mozilla.org/en-US/firefox/>
[Hozzáférés dátuma: 28 04 2020]