

Document Organiser

Dokumentáció

Angular alapú szoftverfejlesztés

Önálló laboratórium 1

Tartalom

| | |
|--|----|
| Tartalom | 2 |
| A rendszer célja, funkciói és környezete | 3 |
| Rövid leírás | 3 |
| Rendszer funkciói | 3 |
| Megvalósítás | 4 |
| Backend | 4 |
| User entitás | 4 |
| Doctype entitás | 4 |
| Userdoc entitás | 5 |
| Levelezés | 6 |
| Frontend | 7 |
| Grafikus felhasználói felület | 7 |
| Üzleti logikai réteg | 10 |
| Telepítési leírás | 11 |
| Program készítése során felhasznált eszközök | 12 |
| Összefoglalás | 13 |
| Továbbfejlesztési lehetőségek | 14 |
| Hivatkozások | 15 |

A rendszer célja, funkciói és környezete

Rövid leírás

A cél egy egyszerű dokumentumkezelő web alkalmazás megvalósítása, amelybe a felhasználók felvihetik a dokumentumaikat. Az alkalmazás fő célja az, hogy a felhasználók egy helyen elérjék minden dokumentumukat, valamint hozzáférjenek ezek régebbi verzióihoz. Ezen felül az alkalmazás e-mailben értesíti a felhasználókat arról, ha hamarosan lejárna valamely dokumentumuk, így biztosítva, hogy időben megújítsák azt.

Rendszer funkciói

A következő funkciókat várjuk el a rendszertől a végfelhasználó szempontjából:

- Felhasználó regisztrálása, bejelentkezés (authenticáció)
- Felhasználó dokumentumai közé újak felvétele
- A meglévő dokumentum adatainak módosítása
- File-ok felvétele (hozzáadása) a dokumentumokhoz
- A meglévő dokumentumok törlése az adatbázisból
- Felhasználói adatok módosítása
- Email értesítés küldése hamarosan lejáró dokumentumokról

Ezen felül a Superadmin a következő funkcionálisokat éri el:

- Dokumentumtípusok felvétele
- Dokumentumtípusok szerkesztése
- Dokumentumtípusok törlése
- Hozzáférés más felhasználók dokumentumaihoz

Megvalósítás

Az alkalmazást itt két részre különböztetve, backend – a szerver, amely az adatbázis elérését végzi, és frontend – ami a felhasználókkal való kommunikálásért felelős, alpontokban fogom taglalni, melyek az alkalmazás keretein belül http kérés-válasz üzenetekben kommunikálnak egymással.

Az alkalmazás frontendje jelen konfiguráció szerint a **4040**-es porton érhető el, míg a backend a **4041**-es porton található meg.

Backend

A projekt adatainak tárolásához a mongoDB-t választottam, ugyanis a dinamikus mennyiségű részlet miatt egy document-database-t tartottam a legpraktikusabbnak.

Az adatmodellek definíciói megtalálhatóak a *./Backend/models* mappa fájljában. A következő alfejezetekben mindegyik modell külön részletesen ismertetve lesz.

User

Célja: Egy felhasználói fiókot reprezentál az adatbázisban

| Entitás mezőnév | mongoose megkötés | MongoDB adattípus |
|-----------------|-------------------|-------------------|
| id | String | ObjectId |
| email | String | String |
| password | String | String |
| name | String | String |
| role | String enum | String |

- **id** mező egyedi azonosító, elsődleges kulcs, mely segítségével azonosítható az entitás.
- **email** szintén egyedi azonosító, minden fiókhoz kötetendő egy email cím.
- **password** a felhasználó által beállított jelszó, mely titkosítva van eltárolva
- **name** a felhasználó teljes neve
- **role** mező megadja a felhasználó hozzáférési szintjét (enum lehetséges értékei: *'User'*, *'Administrator'*, *'Superadmin'*)

Doctype

Célja: Egy dokumentum típust, és az ahhoz tartozó extra mezőket hivatott eltárolni

Egy dokumentumnak kötelező meghatározni a típusát, és ez alapján kapja meg a részletes adataihoz tartozó mezőket (pl.: Személyigazolványhoz tartozik szám, születési dátum, nem, stb.).

Ez alapján a Doctype-nak el kell tárolnia a típus nevét, ezeket a mezőket és a hozzájuk tartozó nevet valamint típust. A közös mezőkhöz tartozó adatokat felesleges tárolni, ugyanis mindegyik típusnál egyformák.

| Entitás mezőnév | mongoose megkötés | MongoDB adattípus |
|-----------------|-------------------|-------------------|
| id | String | ObjectId |
| name | String | String |
| details | Array of Objects | Array |

- **id** egyedi azonosító, elsődleges kulcs, segítségével azonosítható az entitás
- **name** a felhasználó által megadott típus neve, tetszés szerint bárminek lehet hívni
- **details** a típus részletes (nem közös) mezőinek a specifikálása, egy array-beli objektum két kulcsból áll amelyek a key és keyType, mindkét mező típusa String, és az utóbbi egy következőkből álló enum: 'Number', 'String', 'Date'.

Userdoc

Célja: A felhasználó által felvett dokumentum minden adatát tárolni.

A userdoc one-to-many kapcsolatban áll a felhasználókkal és a típusokkal, azaz egy dokumentumnak egyetlen gazdája és típusa lehet, de egy felhasználóhoz és típushoz tartozhat akármennyi dokumentum.

Emiatt fel kell venni külső kulcsként azt a felhasználói id-t, akihez tartozik, valamint a típus id-t, és ezeket el szükséges tárolni. Ehhez szükséges felvenni a sémában egy User típusú owner, és egy Doctype típusú doctype mezőt.

| Entitás mezőnév | mongoose megkötés | MongoDB adattípus |
|-----------------|-------------------|-------------------|
| id | String | ObjectId |
| name | String | String |
| doctype | ObjectId | ObjectId |
| owner | ObjectId | ObjectId |
| details | Array of Objects | Array |
| expires_at | Date | Date |

| | | |
|-------------|----------------------------------|---------|
| currentfile | {date: Date, location: String} | Object |
| oldfiles | {date: Date, location: String}[] | Array |
| upgrade | Boolean | Boolean |

- **id** egyedi azonosító, elsődleges kulcs, segítségével egyedien azonosítható az adatbázisban
- **name** a dokumentum tetszőlegesen állítható neve
- **doctype** a hozzákapcsolt doctype id-je
- **owner** a hozzákapcsolt felhasználó id-je
- **details** hasonlóan a doctypehoz a típus részletes (nem közös) mezőinek a specifikálása, ezúttal a hozzájuk tartozó értékkel együtt, egy array-beli objektum három kulcsból áll amelyek a key, keyType és value, mindhárom mező típusa String, és a keyType egy következőkből álló enum: 'Number', 'String', 'Date'.
- **expires_at** a dokumentum lejárat dátuma
- **currentfile** a dokumentumhoz tartozó legfrissebb file feltöltési ideje és elérési címe
- **oldfiles** a dokumentumhoz tartozó régebbi file-ok feltöltési ideje és elérési címe
- **upgrade** egy igaz-hamis érték, mely jelzi, hogy a dokumentum utolsó frissítése óta történt-e változás a dokumentumhoz tartozó doctype részleteiben, ezáltal frissíteni szükséges-e a dokumentumot.

Levelezés

Az alkalmazás levelezését a nodemailer valósítja meg, annak ütemezését pedig a cron scheduler végzi.

Minden hét elején lefut egy adatbázis lekérdezés, amely megállapítja, mely felhasználóknak van a következő 4 hétben lejáró dokumentuma. Ezeknek a felhasználóknak küld egy e-mailt, melyben felsorolja melyek a hamarosan lejáró dokumentumaik, és ezek mikor járnak le, valamint biztosít egy linket a web alkalmazáshoz a gyors elérés érdekében.

Frontend

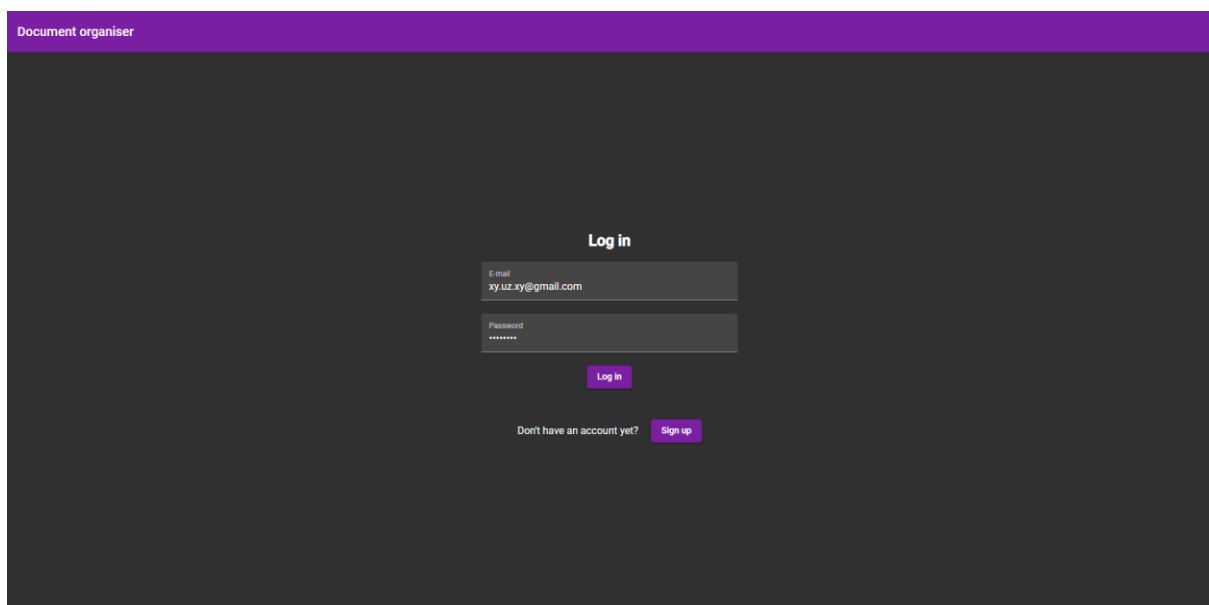
A webalkalmazás jelen körülmények között a localhost 4040-es portján érhető el, ezt a címet production-be való kihelyezés esetén frissíteni szükséges a backend CORS policyjában, valamint a frontend environment fájljában is szükséges a backend majd production-beli címének felvétele.

Grafikus felhasználói felület

Célja: A felhasználók számára egyszerű, könnyen átlátható felületet nyújtani, az összes funkciót elérhetővé tenni.

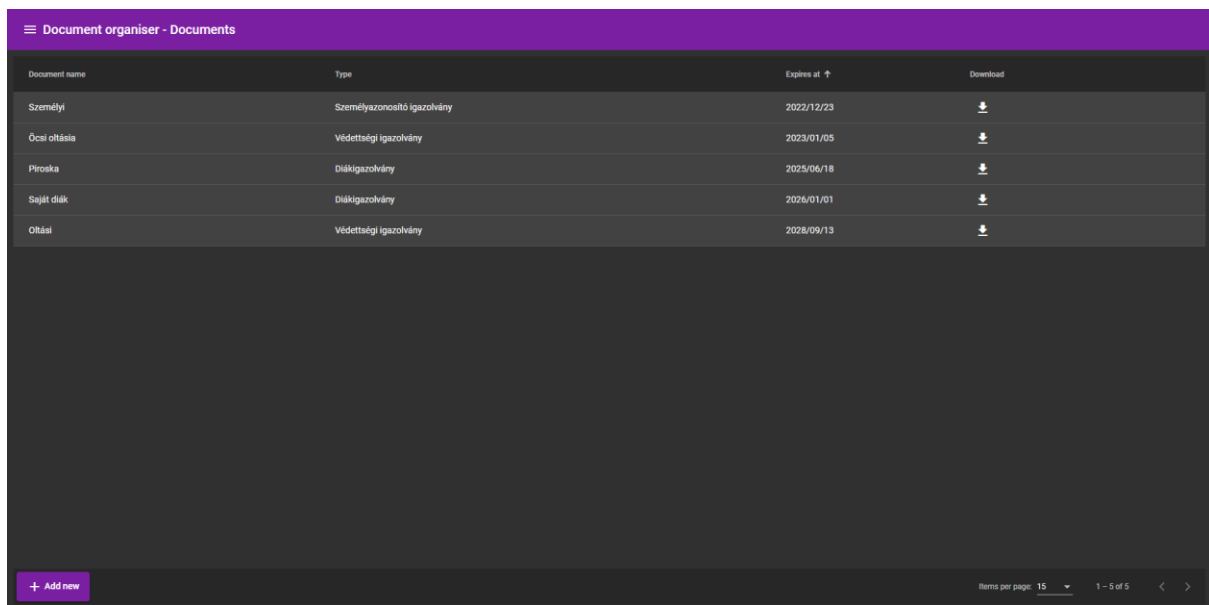
A document organiser felhasználói felülete egy Angular keretrendszerre épült weboldal, amely az Angular Material könyvtár komponenseiből van felépítve, így a weboldal kinézete és működése az átlagfelhasználó által már valószínűleg ismert és megszokott Google material design-t alkalmazó elemekből áll.

A felhasználót a frontend címére navigálva egy bejelentkező és regisztráló felület fogadja, amely segítségével autentikálni tudja magát, sikeres bejelentkezés után pedig átirányítódik a Documents felületre. Az alkalmazás többi képernyője nem elérhető bejelentkezés nélkül, autentikáció hiányában a felhasználó átirányításra kerül a bejelentkező felületre.



1.ábra: A bejelentkező felület.

A sikeres bejelentkezést követően a felhasználó az alkalmazás főoldalára érkezik, egy listára, ahol láthatja a felvett dokumentumait felsorolás formában.



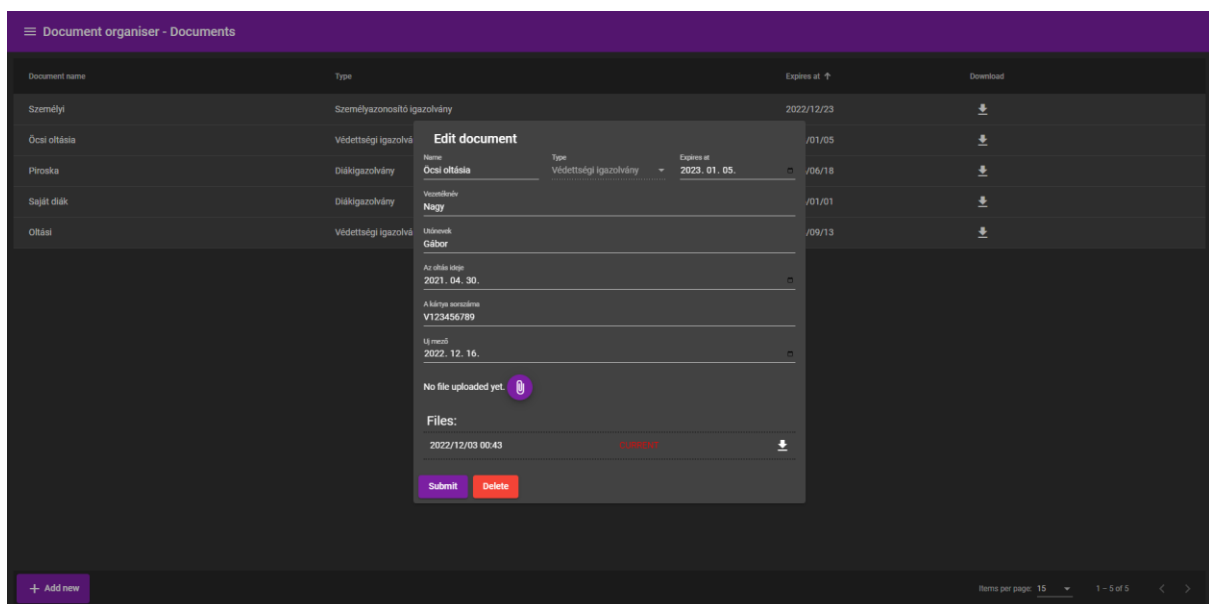
The screenshot shows the 'Document organiser - Documents' interface. It features a table with the following columns: Document name, Type, Expires at, and Download. The table contains five rows of document information. Below the table, there is a '+ Add new' button and a pagination bar showing 'Items per page: 15' and '1 - 5 of 5'.

| Document name | Type | Expires at | Download |
|---------------|-----------------------------|------------|--------------------------|
| Személyi | Személyazonosító igazolvány | 2022/12/23 | Download |
| Ócsei oltásia | Védettségi igazolvány | 2023/01/05 | Download |
| Piroska | Diákigazolvány | 2025/06/18 | Download |
| Saját diák | Diákigazolvány | 2026/01/01 | Download |
| Oltási | Védettségi igazolvány | 2028/09/13 | Download |

2.ábra: A documents felület felhasználói nézete.

Az oldal alján elhelyezkedő gombra kattintva a felhasználó felvehet új dokumentumokat a listába, valamint amennyiben a lista egy elemére kattint, megjelenik az adott dokumentum információja, amelyet szerkeszthet, vagy a felugró felület Delete gombja hatására eltávolíthat a listából. A szerkesztés és új felvétele funkciók ugyanazt a komponenst használják.

A hozzáadó felület alján található kapocs gomb segítségével lehetőség van file felvételére a dokumentumokhoz, e file-ok pedig ameddig él a dokumentum naplózva eltárolásra kerülnek, hogy a felhasználó később is meg tudja tekinteni a dokumentuma régebbi verzióját.



The screenshot shows the 'Document organiser - Documents' interface with the 'Edit document' modal open. The modal contains the following fields: Name (Ócsei oltásia), Type (Védettségi igazolvány), Expires at (2023. 01. 05.), Version (V123456789), and a list of files. The 'Files' section shows a file named '2022/12/03 00:43' with a download icon. The modal also includes 'Submit' and 'Delete' buttons.

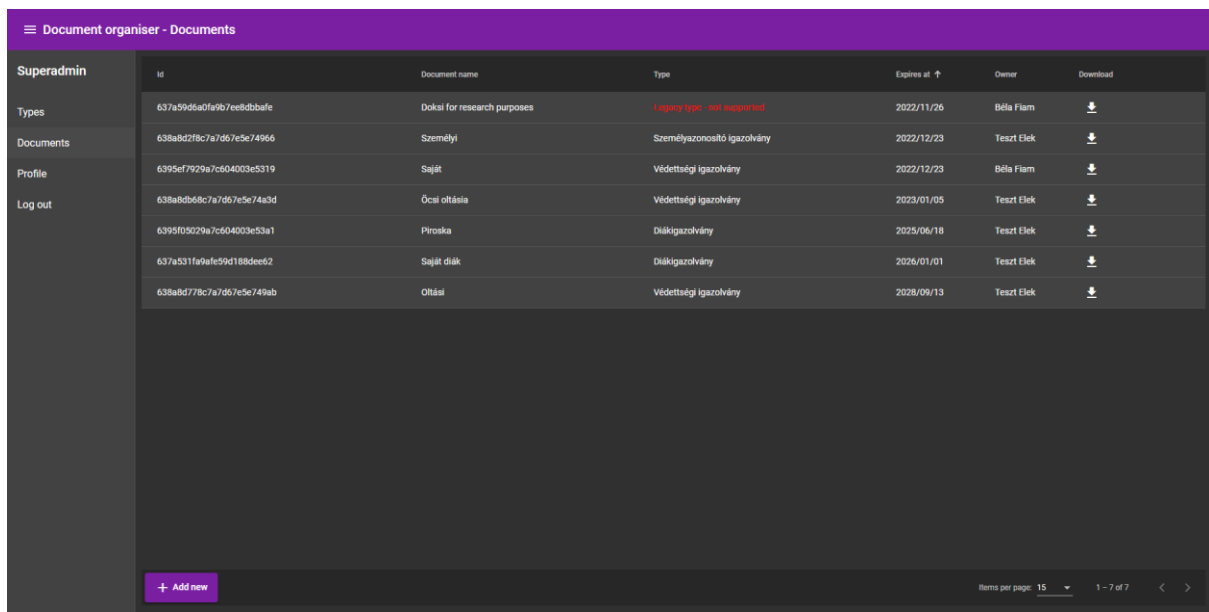
| Document name | Type | Expires at | Download |
|---------------|-----------------------------|------------|--------------------------|
| Személyi | Személyazonosító igazolvány | 2022/12/23 | Download |
| Ócsei oltásia | Védettségi igazolvány | 2023/01/05 | Download |
| Piroska | Diákigazolvány | 2025/06/18 | Download |
| Saját diák | Diákigazolvány | 2026/01/01 | Download |
| Oltási | Védettségi igazolvány | 2028/09/13 | Download |

3.ábra: A document frissítő és hozzáadó felület.

Azon dokumentumok, amelyeknek a típusát eltávolították a rendszerből a felhasználók számára már csak megtekinthetők és törölhetők lesznek, míg azon dokumentumok melyek

típusán változtattak az adminisztrátorok egészen addig csak megtekinthetőek vagy törölhetőek, amíg a felhasználó bele nem egyezik a dokumentuma verziójának és adatainak frissítésébe az ilyenkor megjelenő Update gomb használatával.

Adminisztrátori jogosultsággal a felület minden felhasználó dokumentumait kilistázza, és ezen felül többletinformációt is nyújt róluk az adminisztrátor számára.



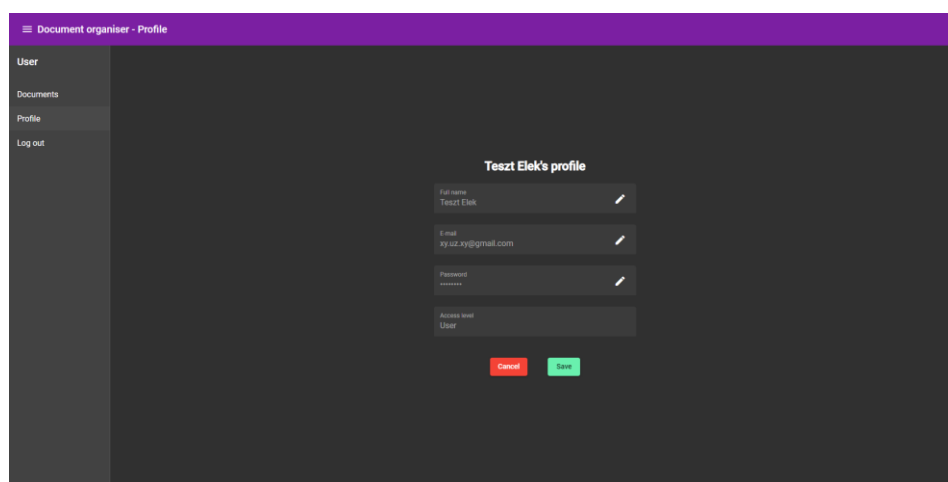
| Id | Document name | Type | Expires at | Owner | Download |
|--------------------------|-----------------------------|-----------------------------|------------|------------|----------|
| 637a59d6a0fa9b7e8dbbaf6 | Dokai for research purposes | Legacy type - not supported | 2022/11/26 | Béla Fiam | Download |
| 638a8d2f8c7a7d67e5e74966 | Személyi | Személyazonosító igazolvány | 2022/12/23 | Teszt Elek | Download |
| 6395ef7929a7c604003e5319 | Saját | Védettségi igazolvány | 2022/12/23 | Béla Fiam | Download |
| 638a8db68c7a7d67e5e74a3d | Ócsi ottási | Védettségi igazolvány | 2023/01/05 | Teszt Elek | Download |
| 6395f05029a7c604003e53a1 | Piroka | Diákigazolvány | 2025/06/18 | Teszt Elek | Download |
| 637a531fa9a7e59d188de62 | Saját diák | Diákigazolvány | 2026/01/01 | Teszt Elek | Download |
| 638a8d778c7a7d67e5e749ab | Ottási | Védettségi igazolvány | 2028/09/13 | Teszt Elek | Download |

4.ábra: A documents felület adminisztrátori jogosultsággal.

A bal felső sarokban elérhető hamburger menüre kattintva elérhető az alkalmazás navigációs sávja. Ennek segítségével elérhető az alkalmazás többi felülete.

A hamburger menüt használva a felhasználó el tud navigálni a Profile-ra, a Logout menüpontot kiválasztva kijelentkezni az alkalmazásból, vagy ha superadmin, akkor a Types menüpontot választva a típusokat listázó felületre léphet át.

A Profile felületen a felhasználó megtekintheti, és megváltoztathatja az adatait.



Document organiser - Profile

User

Documents

Profile

Log out

Teszt Elek's profile

Full name
Teszt Elek

Email
xyz.xy@gmail.com

Password

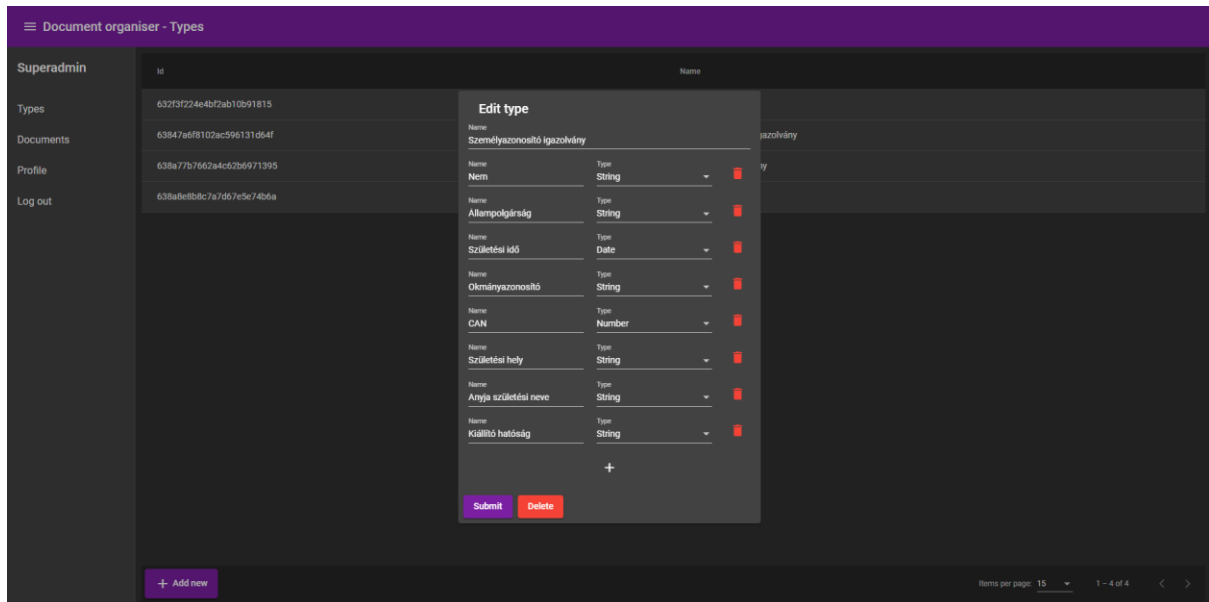
Access level
User

Cancel Save

5.ábra: A profile felület.

A types felület nagyjában hasonlít a documents felülethez, csak ezen a doctype-ok kerülnek megjelenítésre.

Az oldal alján található add gomb hatására megnyílik a doctype szerkesztő felület, amelynek hatására megnyílik a típus szerkesztő felület, amelyen az adminisztrátornak az adott típushoz lehetősége van felvenni, módosítani és törölni részlet-mezőket. Ezt a szerkesztő felületet a lista egy elemére kattintva is meg lehet nyitni, amely esetben a kiválasztott doctype adatai jelennek meg rajta, és lehetősége van az adminisztrátornak frissíteni, vagy éppenséggel törölni a típust.



6.ábra: A doctype frissítő és hozzáadó felület.

Üzleti logikai réteg

Célja: Az API-tól kapott adatok alapján kiszorgálni a grafikus felhasználói felületet.

A kliensalkalmazáson belül minden komponenshez tartozik egy osztály amiben az API-tól lekért adatokat az alkalmazás a grafikus felületének szükséges formátumra hozza. Ugyanúgy a felhasználó által bevitt adatok API által elvárt formátumra alakításáért is ez az osztály felelős.

Az átalakításra a html multipart/form üzenet limitáció miatt van szükség, ugyanis ezen csak blob vagy string formátumban utaztatható adat, így a JSON formátumok, mint például tömbök átalakításra kerülnek.

Az alkalmazás az Angular struktúra szervízeit és az rxjs könyvtár Observable struktúráját használva kommunikál az API-val.

Telepítési leírás

A webalkalmazásnak szüksége van telepített MongoDB-re a futtató számítógépen, valamint azon belül egy adatbázis példányra, melyek adatait fel szükséges tüntetni a *2022-document-organiser/Backend/config/db.js* fájl *mongoose.connect* sorában a következő formátumban:

```
mongoose.connect('mongodb://localhost/docOrg');
```

A *2022-document-organiser/Backend* mappán belül ki szükséges adni a következő parancsot, mely a definiált node modules-okat installálja nekünk:

```
npm install
```

A futó folyamat végeztével már futtatható is a backend a következő paranccsal:

```
node index.js
```

A frontend elindításához szükséges a *2022-document-organiser/document-organiser* mappán belül is kiadni a következő parancsot:

```
npm install
```

Majd ezek után a webalkalmazás elindításához elegendő a következő parancsot kiadni:

```
npm run serve
```

Program készítése során felhasznált eszközök

- Visual Studio Code
 - Felhasználás: fejlesztőkörnyezet
- MongoDB [\[1\]](#)
 - Felhasználás: adatbázis-kezelő rendszer
- Robo 3T [\[2\]](#)
 - Felhasználás: MongoDB adatbázis menedzselő szoftver
- ExpressJS [\[3\]](#)
 - Felhasználás: Szerver oldali Node.js framework
- GitHub
 - Felhasználás: Verziókezelés, kollaboráció
- Angular [\[4\]](#)
 - Felhasználás: Kliens oldali web-platform
- Angular Material [\[5\]](#)
 - Felhasználás: Material design komponensek Angular platformhoz

Összefoglalás

A félév során implementáltam és dokumentáltam a Document Organiser-nek elnevezett dokumentumkezelő webalkalmazást. Az így elkészült program segítségével a felhasználók tárolni tudják saját dokumentumaikat és irataikat, valamint értesítést kaphatnak arról, mikor ezek lejárta közeleg.

Adminisztrátorok pedig szerkeszthetik a felhasználók számára elérhető dokumentumtípusokat, valamint minden felhasználó dokumentumaihoz is limitált hozzáférést kapnak, ezen felül a szuperadminok kezelhetnek a rendszeren belül mindent.

Az elkészült alkalmazás MongoDB adatbázisra épül, mely az adatokat helyi példányban tárolja el. Az ebben eltárolt adatokat a mongoose könyvtár segítségével éri el a szerver, mely maga pedig NestJS-re épül.

A frontendet Angular segítségével építettem fel, azon belül is Angular Material elemeket alkalmazva, mely eredményeképp egy jól működő, flexibilis, modern és könnyen átlátható alkalmazást sikerült elkészíteni.

Továbbfejlesztési lehetőségek

Az alkalmazás bár használható, koránt sem tökéletes, a grafikus felülete modernizálásra szorul, de e kívüli, funkcionalitásbeli továbbfejlesztési lehetőségek többek közt a következők:

- Egy alkalmazáson belüli értesítés-szolgáltatás, amely pontosabb információval szolgál esetleges hibákról vagy háttérbeli történésekről
- Egy felugró-ablak szolgáltatás, amely megerősítés kéréseket, vagy értesítéseket kommunikál a felhasználó felé.
- Mobil nézet támogatása
- Lokalizáció nyelvválasztóval, például i18n segítségével
- Adminisztrátori funkciók bővítése (pl. dokumentum létrehozása egy felhasználó nevében)
- Regisztrációs lehetőségek és funkcionalitás bővítése (pl. Google/Facebook authentication, jelszó emlékeztető, email megerősítés)
- Fizetős funkció (pl. GoPay-el)

Hivatkozások

- [1] MongoDB hivatalos honlap
<https://www.mongodb.com/>
- [2] Robo 3T hivatalos honlap
<https://robomongo.org/>
- [3] ExpressJS hivatalos honlap
<https://expressjs.com/>
- [4] Angular hivatalos honlap
<https://angular.io/>
- [5] Angular Material hivatalos honlap
<https://material.angular.io/>