DOKUMENTÁCIÓ

„LIBMAN” ALKALMAZÁS

(Könyvtárkezelő rendszer)

VIZSGAMUNKA

BUDAPESTI ZSIDÓ HITKÖZSÉG KÜLKERESKEDELMI TECHNIKUM

SZOFT II / 1 / E

Készítette: Szőllősi-Maruzs Eszter

Resperger Patrik

Grünhut Gábor

v 2.1.

2024-04-18

Tartalom

I. A feladat leírása és a megoldástól elvárt követelmények meghatározása

1. A téma ismertetése, témaválasztás indoklása

A projekt célja egy könyvtárkezelő rendszer létrehozása, amely támogatja a könyvek kezelését egy webes interfészen keresztül, beleértve a keresést, hozzáadást, frissítést és törlést is (CRUD műveletek).

Az elkészült alkalmazás elsősorban intézményi (köznevelési, kulturális) körülmények közt hasznosítható, de nem zárható ki magáncélú felhasználása sem (az adatbázis kialakításának a lehetősége okán, magánkönyvtárak esetében).

Az elsődleges cél az volt, hogy az oktatási intézményünk, a BZSH Külkereskedelmi Technikum részére készítsünk egy hasznosítható alkalmazást.

2. Elvárt követelmények meghatározása

A könyvtárkezelő rendszernek tudnia kell kezelni a „Ki?”, „Mit?” és „Mikor?” kérdéskört.

- „Ki?”

  Milyen felhasználói kör és milyen jogosultságokkal tudja használni az alkalmazást? Az **adminisztrátor**nak szükséges a CRUD műveleteket elvégeznie, valamint a felhasználóhoz rendelnie a „kölcsönzött” könyveket. A **felhasználó** az adatbázisban szereplő tételek (könyvek) megtekintésére jogosult.

- „Mit?”

  Könyvek adatainak kezelésére és kölcsönzésre szolgál az alkalmazás. A könyvekhez tartozó adattípusok meghatározása a cél, mint pl.: cím, szerző, kiadás éve, kiadó megnevezése, ISBN, rendelkezésre álló mennyiség, kategória (angol nyelven: title, author, year, publisher, ISBN, quantity, category).

- „Mikor?”

  A felhasználó általi könyvkölcsönzés meghatározott dátummal történik, és az adminisztrátornak (vagy egy előre beállítható automatizmusnak) meg kell határoznia azt is, hogy a kölcsönzés időtartama meddig tart, vagyis mikor jár le a kölcsönzött könyv visszaadásának a határideje.

Az alkalmazás a fenti kérdéskörök mentén került kialakításra.

II. Fejlesztői dokumentáció

1. A futási környezet leírása

A „Libman” egy webes alkalmazás, és mint ilyen, a legtöbb, széles körben elterjedt web-böngészővel együttműködik. Specifikus, minimális hardware igényről nincsen tudomásunk azzal kapcsolatban.

2. fejlesztői környezet leírása

Az alábbiakban

- Atlas MongoDB adatbázis szerver
- Visual Studio Code
- Vite/React
- az ESLint ellenőrzést végez a projekt teljes kódján.

Az alábbi függőségeket használja a projekt (a verziószámok az adott függőség verzióját jelölik):

- *@tanstack/react-query:* React alkalmazásban használható adatlekérdezési könyvtár.
- *@tanstack/react-query-devtools:* a React Query fejlesztőeszközei.
- *axios:* HTTP kliens a kliensoldali hálózati kérésekhez.
- *dayjs:* dátum és idő kezelő könyvtár.
- *react:* React könyvtár.
- *react-dom:* a React DOM manipulációs könyvtár, amely lehetővé teszi a React elemek renderelését a DOM-ba.
- *react-icons:* ikonokat tartalmazó React komponensek.
- *react-router-dom:* React alkalmazásokhoz használt router.
- *react-toastify:* értesítések megjelenítését teszi lehetővé React alkalmazásokban.
- *styled-components:* a CSS-in-JS stílusok létrehozását és kezelését lehetővé tevő könyvtár.

3. Frontend

Az alkalmazásban az alábbi „Frontend” specifikus forrásfájlok és konfigurációs állományok találhatóak (könyvtárszerkezet alapján):

- client/: A könyvtár tartalmazza a kliens oldali alkalmazás forrásfájljait és konfigurációs állományait.

  .eslintrc.cjs: ESlint konfigurációs fájl.

  .gitignore: A Git által figyelmen kívül hagyandó fájlok és könyvtárak listája.

  index.html: A kliens oldali alkalmazás kezdőoldala.

  postcss.config.js: PostCSS konfigurációs fájl.

  README.md: A projekt dokumentációja.

  tailwind.config.js: Tailwind CSS konfigurációs fájl.

  vite.config.js: Vite (a kliens oldali fejlesztési szerver) konfigurációs fájl.

  src/: A forráskód helye.

    App.jsx: Az alkalmazás fő komponense.

    index.css: A globális stílusokat tartalmazó CSS fájl.

    main.jsx: Az alkalmazás belépési pontja.

      assets/: Képek és egyéb statikus erőforrások.

      components/: Az alkalmazás komponensei.

      pages/: Az alkalmazás oldalai.

      utils/: Segédfüggvények és eszközök.

A components/ könyvtárban található komponensek részletesebb leírása a 4.1. pontban található.

A következő technológiák kerültek felhasználásra a webes oldali rész kialakításakor:

- *React (JSX)*

  A felhasználói felületek fejlesztésére, valamint az elemek JavaScript kódban történő leírásának a lehetősége miatt választottuk ezt a technológiát. A HTML-szerű struktúra, az egyszerűbben karbantartható és más fejlesztők által is könnyebben átlátható kód miatt. Nem utolsó sorban a React segíti az UI komponensek újrafelhasználhatóságát, amivel mi is éltünk az alkalmazás elkészítése során.

- *Tailwind CSS*

A "utility-first" típusú CSS keretrendszer kiválasztásakor szintén a gyorsaság és az ahhoz mért hatékonyság volt az elsődleges szempont számunkra. A kisméretű, újrahasznosítható osztályok segítségével az alkalmazás stíluselemeit könnyebben tudtuk kezelni, egyben rendkívül homogén kinézetet tudtunk biztosítani.

- *Axios*

  A backend API-hoz való kommunikáció szempontjából, a kérések küldésének és fogadásának egyszerűsítése miatt került alkalmazásra a technológia.

A három, fenti technológia együttes alkalmazásával egy rugalmas, megjelenésében gyorsan és egyszerűen testreszabható alkalmazás kialakítása történt meg.

3.1. Komponensek

Az alkalmazásban a következő Frontend komponensek - felhasználói felület elemek - kerültek kialakításra (az adott alkalmazás neve alatti rövid blokkban kerül ismertetésre a komponens által végrehajtandó feladat, az *osztályváltozók és metódusok*):

a) EditBook.jsx

A komponens feladata: Egy meglévő könyv adatainak frissítése.
Osztályváltozók: Form adatok tárolása és kezelése.
Metódusok: javascript

```
componentDidMount() {
  this.fetchBookDetails();
}

fetchBookDetails = async () => {
  const { bookId } = this.props.match.params;
  try {
    const response = await axios.get(`/api/books/${bookId}`);
    this.setState({ ...response.data });
  } catch (error) {
    console.error('Error fetching book details:', error);
  }
}

handleSubmit = async (event) => {
  event.preventDefault();
  const { bookId } = this.props.match.params;
  try {
    await axios.patch(`/api/books/${bookId}`, this.state);
    alert('Book updated successfully!');
    this.props.history.push('/books');
  } catch (error) {
    console.error('Failed to update book:', error);
    alert('Failed to update book!');
  }
```

```
}
```

***b)* BookList.jsx**

A komponens feladata: Az összes könyv listájának megjelenítése.
Osztályváltozók: Könyvek listája (state-ben tárolva).
Metódusok: javascript

```
componentDidMount() {
  this.fetchBooks();
}

fetchBooks = async () => {
  try {
    const response = await axios.get('/api/books');
    this.setState({ books: response.data });
  } catch (error) {
    console.error('Error fetching books:', error);
  }
}

renderBooks = () => {
  return this.state.books.map(book => (
    <div key={book._id}>
      <h3>{book.title}</h3>
      <p>Author: {book.author}</p>
    </div>
  ));
}
```

***c)* AddBook.jsx**

A komponens feladata: Új könyv hozzáadása az adatbázisba.
Osztályváltozók: Form adatok tárolása és kezelése.
Metódusok: javascript

```
handleSubmit = async (event) => {
  event.preventDefault();
  try {
    await axios.post('/api/books', this.state);
    alert('Book added successfully!');
    this.setState({ title: '', author: '', year: '', publisher: '', ISBN: '', quantity: '', category: '' });
    this.props.history.push('/books');
  } catch (error) {
    console.error('Failed to add book:', error);
    alert('Failed to add book!');
  }
}
```

***d)* BookDetail.jsx**

A komponens feladata: Egy kiválasztott könyv részletes adatainak megjelenítése.
Osztályváltozók: Kiválasztott könyv adatai.
Metódusok: javascript

```
componentDidMount() {
  this.fetchBookDetail();
}

fetchBookDetail = async () => {
  const { bookId } = this.props.match.params;
  try {
    const response = await axios.get(`/api/books/${bookId}`);
    this.setState({ book: response.data });
  } catch (error) {
    console.error('Error fetching book detail:', error);
  }
}
```

e) DeleteBook.jsx

A komponens feladata: Meglévő könyv törlése az adatbázisból.

Metódusok: javascript

```
handleDelete = async () => {
    const { bookId } = this.props.match.params;
    try {
        await axios.delete(`/api/books/${bookId}`);
        alert('Book deleted successfully!');
        this.props.history.push('/books');
    } catch (error) {
        console.error('Error deleting book:', error);
        alert('Error deleting book!');
    }
}
```

További komponensek, melyek az alkalmazás működéséhez szükségesek:

- *BorrowedBook.jsx:* megjeleníti a felhasználók kölcsönzött könyveit és az ezekkel kapcsolatos információkat.
- *Dropdown.jsx:* legördülő lista komponens.
- *Navbar.jsx:* navigációs sáv komponens, melyet az alkalmazásban a fő navigációhoz használunk.
- *Sidebar.jsx:* az oldalsáv komponens, amelyet az alkalmazás a navigációhoz vagy az alkalmazás beállításaihoz használ.
- *User.jsx:* felhasználói adatokat megjelenítő komponens.
- *UsersContainer.jsx:* konténer komponens, amely megjeleníti az összes felhasználót az alkalmazásban.

3.2. fejlesztési lehetőségek

Keresési komponens implementálása az alkalmazásba, hogy az összes, adatbázisban lévő könyvből a felhasználó és az admin kereséseket tudjon végrehajtani.

Beállítási lehetőségek, amelyek során a felhasználó jobban testre tudja szabni az alkalmazást.

Statisztikák: az adatbázisban lévő könyvekre és a felhasználókra vonatkozóan.

Nyelvi beállítások.

A későbbiekben szükségessé válhat mobilalkalmazás készítése a könyvtári rendszerhez.

5. Backend

Az alkalmazásban az alábbi „Backend” specifikus kontrollerek és köztes szoftverek (middleware) találhatóak:

- controllers/: a könyvár a szerver oldali vezérlőket (controller) tartalmazza, amelyek az üzleti logikát végzik.

  authController.js: Felhasználó-kezelési funkciókat tartalmazó vezérlő.

  bookController.js: Könyvkezelési funkciókat tartalmazó vezérlő.

  userController.js: Felhasználó-kezelési funkciókat tartalmazó vezérlő.

- middleware/: Tartalmazza a middleware-eket, amelyek a bejövő kéréseket dolgozzák fel a szerveren.

  authMiddleware.js: Az autentikációt kezelő middleware.

  errorHandlerMiddleware.js: Hibakezelő middleware.

  validationMiddleware.js: Érvényesség-ellenőrző middleware.

A következő technológiák kerültek felhasználásra a szerver oldali rész kialakításakor:

- Node.js, nyílt forráskódú, JavaScript alapú futtatókörnyezet, amely lehetővé teszi a JavaScript programok futtatását a szerveroldalon. Az alkalmazásban az aszinkron és eseményvezérelt tulajdonsága került elsősorban felhasználásra, ami lehetővé tette a hatékony és skálázható szerveralkalmazást.

- Express.js webes alkalmazási keretrendszerre a minimalizmusa miatt esett a választás. A Node.js-hez kapcsolódva lehetővé tette a könnyű kezelést és az alkalmazás gyors és hatékony építését (pl.: könnyen használható útvonal kezelés miatt).

- MongoDB-re amiatt esett a választásunk, mivel NoSQL adatbázisrendszert szerettünk volna, amely dokumentum-orientált tárolást biztosít (nem utolsó sorban a Node.js alapú alkalmazások háttéradatbázisaként az egyik leggyakrabban használt). A JavaScripttel való jól integrálhatósága szintén szempont volt.

- Mongoose ODM (Object-Document Mapper). A modellek és sémák definiálásához és az adatok strukturált kezeléséhez a MongoDB adatbázissal való kiváló kommunikációja alapján használtuk.

5.1. Adatbázis

Az adatbázisban MongoDB táblák szerepelnek. Mivel a MongoDB egy dokumentumalapú NoSQL adatbázisrendszer, így a hagyományos relációs adatbázisok esetében meglévő adatszerkezet leképezése nehézségekbe ütközik.

A jelen projekt esetében a MongoDB-re a nagy adatmennyiségek tárolására és kezelésére való képessége miatt esett a választásunk, és azért is mivel hatékonyan skálázódik fel és le.

5.2. API végpontok (Könyvek kezelése)

Az alkalmazásban az alábbi API végpontok nevesíthetőek:

Ssz.	*API végpontok megnevezése*	*Mire szolgál*
1.	GET /api/books	Összes könyv lekérdezése.
2.	POST /api/books	Új könyv hozzáadása.
3.	GET /api/books/{id}	Könyv lekérdezése azonosító alapján.
4.	PATCH /api/books/{id}	Könyv frissítése.
5.	DELETE /api/books/{id}	Könyv törlése.

1. ***GET /api/books:*** Ez a végpont az összes könyv lekérdezésére szolgál. A válasz egy JSON formátumú tömb lesz, amely minden könyvet reprezentál, például:

```
[
 {
  "id": "97832345667890",
  "title": "The Universe Explained",
  "author": "Carl Sagan",
  "year": "2009",
  "publisher": "CosmoBooks",
  ...
 },
 {
  "id": "9780123456789",
```

"title": "The Joy of Calculus 20",
"author": "Alan Turing",
"year": "2014",
"publisher": "MathPress",
...
},
...
]

2. ***POST /api/books:*** Ez a végpont egy új könyv hozzáadására szolgál. A kliensnek JSON formátumú adatot kell elküldenie a következőképpen:

{
"title": "The Universe Explained",
"author": "Carl Sagan",
"year": "2009",
"publisher": "CosmoBooks",
...
}

3. ***GET /api/books/{id}:*** Ez a végpont egy adott könyv lekérdezésére szolgál az azonosítója alapján. Az ’id’ a könyv azonosítója. A válasz egy JSON objektum lesz, amely egyetlen könyvet reprezentál.

{
"id": "97832345667890",
"title": "The Universe Explained",
"author": "Carl Sagan",
"year": "2009",
"publisher": "CosmoBooks",
...
}

4. ***PATCH /api/books/{id}:*** Ez a végpont egy adott könyv frissítésére szolgál az azonosítója alapján. A kliens JSON formátumú adatot küld, amely tartalmazza a módosítani kívánt tulajdonságokat. Például:

   {

   "title": "The Universe **Un**explained",

   "author": "Carl Sagan",

   ...

   }

5. ***DELETE /api/books/{id}:*** Ez a végpont egy adott könyv törlésére szolgál az azonosítója alapján. A DELETE kérés paraméterként tartalmazza az eltávolítandó könyv azonosítóját. A válasz egy megerősítő üzenet.

III. Felhasználói dokumentáció

Kedves Felhasználó!

A „Libman” webes alkalmazás egy könyvtári alapfunkciókat ellátó segédalkalmazás. Az alapfunkciók alatt értjük a következőket:

- *felhasználó kezelése (azonosítása)*

  Felhasználó, azaz könyvtári „kölcsönző” személy képes bejelentkezni és egy azonosító számmal (ID number, vagy ID) képes egyedi módon a bejelentkezést követően magát azonosítani. Ez az azonosítás az első bejelentkezést – avagy adminisztrátor általi regisztrációt – követően nem vész el, tehát nem szükséges minden későbbi bejelentkezés alkalmával újból regisztráltatnia a felhasználónak magát. A munkafoyamat végével a felhasználónak lehetősége van kijelentkezni.

- *Könyv címeinek (adatainak) tárolása*

  Az alkalmazás tárolja – mint egy fizikai könyvtárban a katalógus – a kikölcsönözhető könyvek címeit (és még számos adatot az adott könyvvel kapcsolatban, ezeket majd a könyvekről szóló menüpont alatt részletezzük).

- *Könyvek felhasználóhoz rendelése*

  Mint egy fizikai könyvtárban, a webes alkalmazásban is lehetőség van a „kikölcsönzött” könyveknek a „kölcsönvevőhöz”, azaz a felhasználóhoz történő hozzárendeléséhez. Ezzel a funkcióval nyomonkövethető, hogy ki, mikor és melyik könyvet kölcsönözte ki.

- *Könyvek keresése a könyvtárban (adatbázisban)*

  A felhasználó a könyv címére (adataira) történő kereséssel találatot érhet el az adatbázisban. Magyarán, ha tudod, hogy mit keresel, akkor nagy eséllyel, könnyen kideríthetted, hogy megvan-e a könyvtárban?

A következő oldalakon az alkalmazás funkcióit nézzük át részletesebben.

1. Felhasználó be- és kijelentkezése a webes alkalmazásba

A webes alkalmazásba az ún. bejelentkezési, vagy „Log in” felületen tudsz bejelentkezni, hogy aztán a saját felhasználói neved alatt tudd a különböző szolgáltatásokat, további funkciókat igénybe venni.

A bejelentkezést követően a képernyő jobb felső sarkában a következő „felhasználói” legördülő menüt fogod látni. Helyesebben a kerek, színes körben a te neved kezdőbetűinek a monogramja lesz látható, majd utána neked, mint felhasználónak a teljes neve (vezetéknév és keresztnév).

A könyvtárban történő keresést, „kölcsönzést”, vagy csak a hozzád rendelt, azaz kivett könyvek ellenőrzését követően célszerű a „felhasználói” legördülő menü segítségével kilépni.

A webalkalmazásból történő kilépéssel meg tudod akadályozni, hogy illetéktelenek a te felhasználói profiloddal, azonosítóddal bármilyen lehetséges módon visszaéljenek.

A kilépéshez a „felhasználói” legördülő menüben a „Log out”, azaz „Kilépés” gombra kell klikkelned.

(A „Log out” funkciót aktiváló gombról készült ábrát lásd alább.)

2. Felhasználó „könyvkölcsönzése” a webes alkalmazásban

Az alábbi a képernyőképen láthatod a ***felhasználókhoz*** *(Users)* ***rendelt kölcsönzött könyvekre vonatkozó adatok****at*.

A webalkalmazásban a „Libman” logó alatt elhelyezkedő „Books” („Könyvek”) felíratú nyomógomb alatt lévő „Users” gombra kattintással tudod ezt a felületet előhozni.

A világoszöld keret most egy felhasználót jelöl ki annak érdekében, hogy jobban megvizsgálhassuk, mit is mutat meg nekünk. A látható, megjelenített adatok a következők:

- a felhasználó neve

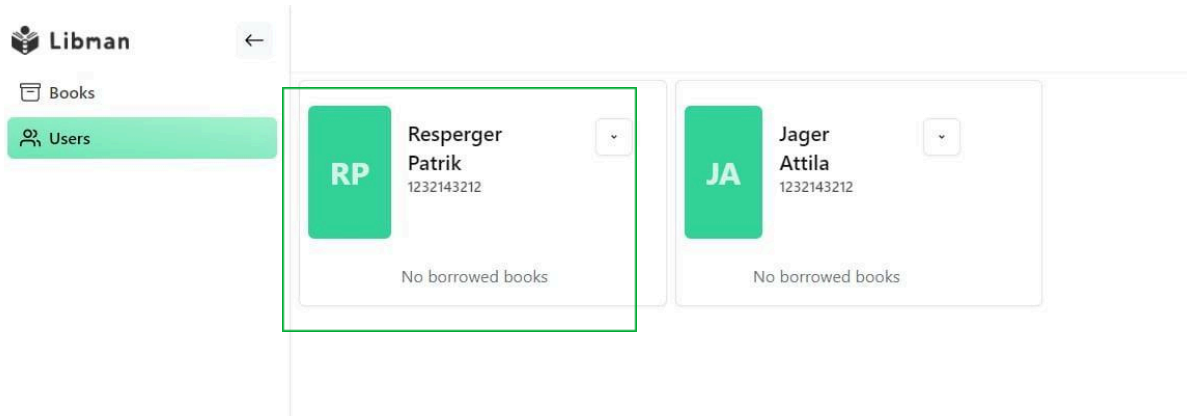
  (Vezetéknév és Keresztnév egymás alatt, külön sorban)

- a felhasználó azonosító száma (ID number, vagy ID)
- nagy zöld, nyomógomb szerű felületen a felhasználó nevének kezdőbetűiből származtatott monogram.

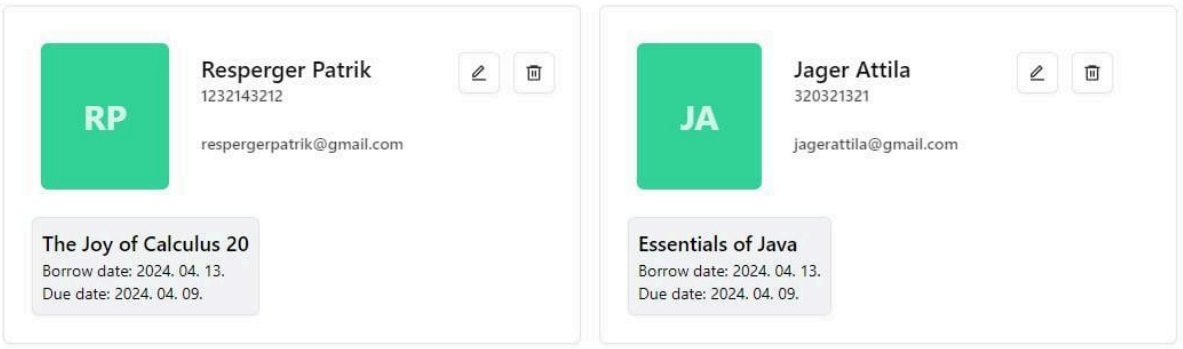
  (Pl.: Resperger Patrik esetében „RP” lesz)

- alatta a státusz, hogy rendelkezik-e kölcsönzött könyvvel a felhasználó, tehát a nevéhez van-e könyvtári könyv rendelve.

  (az alábbi esetben a „No borrowed book” státusz annyit jelent angol nyelven, hogy „Nincs kölcsönzött könyv”)



A következő képernyőképen viszont az látható, ha már rendelkezik „kölcsönzött” könyvvel a felhasználó, tehát már rendeltek hozzá olyan – az adatbázisban megtalálható – könyvet, amit egyébként fizikai értelemben valóban kikölcsönzött.



A könyv címe alatt (Pl.: Resperger Patrik esetében a „The Joy of Calculus 20” című könyv került kikölcsönzésre) látható a „kölcsönzés dátuma” (ami angol nyelven „Borrow date”) év-hónap-nap formátumban, majd az alatt a „kölcsönzés lejáratának a dátuma” (ami angol nyelven „Due date) szintén év-hónap-nap formátumban.

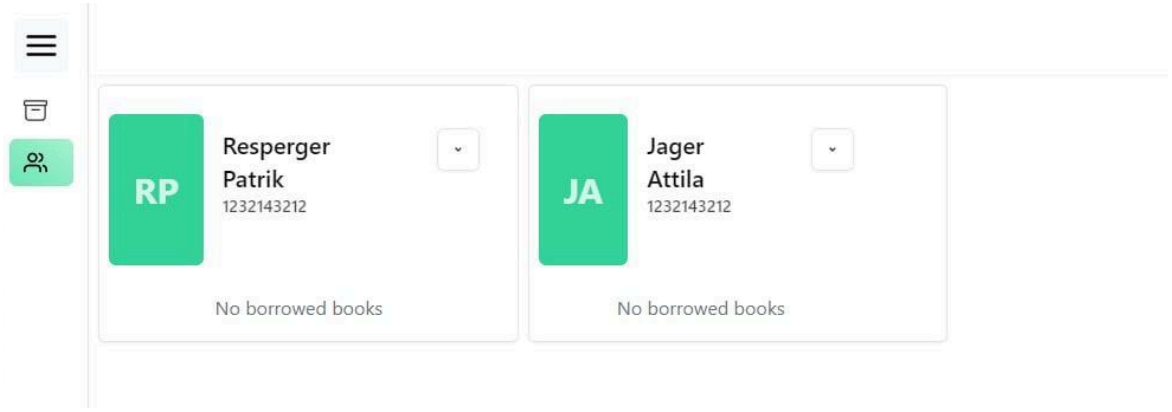
Figyeljük meg, hogy a felhasználó neve alatt nem kizárólag az automatikusan generált azonosító szám jelenik meg, hanem a regisztrácó során megadott – az adminisztrátor által felvitt – e-mail cím is.

3. A képernyő jobb oldalán elhelyezkedő menüpontok minimalizálása

Fontos funkció, emiatt külön említést érdemel. Hasznossága a könyvek keresése során mutatkozik meg, amikor nagyobb képernyőfelületen szeretnénk a találati eredményeket megtekinteni.

Amint az alábbi képen is látható a bal oldali menüsor minimalizálásával lett megoldva.

A



funkciók természetesen nem tűnnek el, így a felhasználókra vonatkozó menüpont, vagy a könyvekre vonatkozó menüpont megmarad és működik, szintén a főmenü.

4. Könyvek menüpontok

A felhasználói regisztráción túl a könyvek menüpontban („Books”) szintén szignifikáns különbség van a felhasználó és az adminisztrátor közt. Felhasználóként te meg tudod tekinteni az adatbázisban lévő könyvek címeit (adatait).

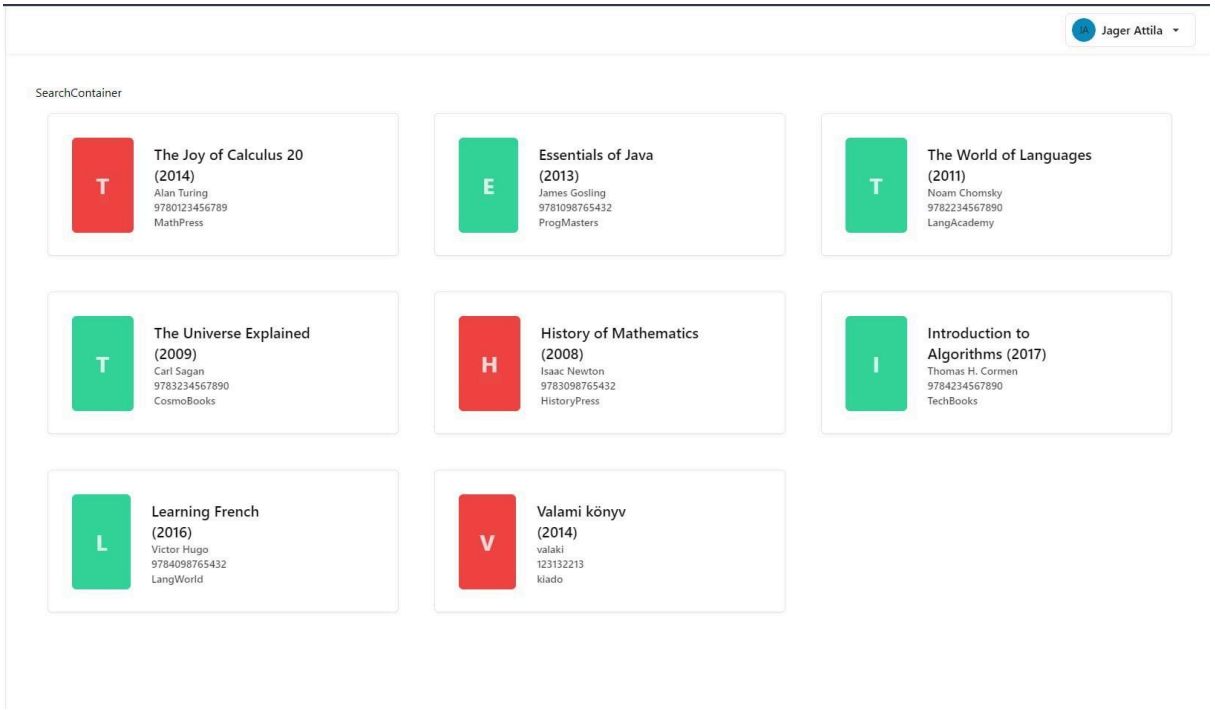
- ***felhasználó általi megtekintés***

  Amint az az alábbi képen is látszik felhasználóként a „Search Containerben”, vagyis a keresési keretben lehetséges a könyvtárban (az adatbázisban) meglévő könyveket kilistázni.

- ***Könyvek adatai***

  A könyv legjellemzőbb adata a címe, de az adatbázis egyéb adatokat (jellemzőket) is tárol az egyes könyvekről, úgy mint kiadás éve, író (szerkesztő, közreműködő személy) neve, könyv azonosítószáma (Book ID), kiadó megnevezése.

Ha az adott könyv az adatbázisban szerepel és rendelkezésre is áll – vagyis az összes példánya nem lett hozzárendelve másik felhasználóhoz – akkor az adminisztrátor hozzánk, mint felhasználóhoz tudja rendelni.



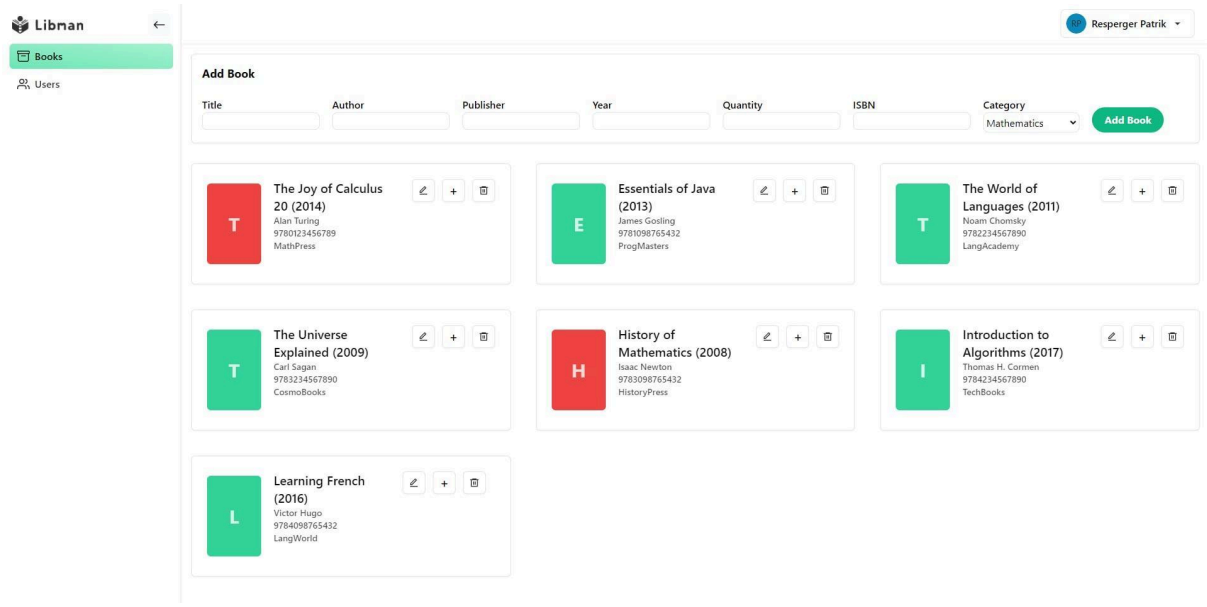
5. Érdekesség – Az adminisztrátor könyvek felülete

Csak érdekességként – és a könyv hozzárendelés funkció miatt – mutatjuk itt be tájékoztató jelleggel, az adminisztrátor cselekvési lehetőségeit az adatbázisban található könyvekkel kapcsolatban.

Az alábbi árbán jól látható, hogy az adminisztrátor képes az adatbázisban a következő műveletek elvégzésére:

- új könyv felvitele
- meglévő könyv adatainak módosítása
- könyv törlése

És természetesen az adminisztrátor fogja a felhasználóhoz hozzárendelni a kiválasztott könyvet is.



IV. A készítő adatai.

A projekt készítésében az alábbi tanulók vettek részt:

Szőllősi-Maruzs Eszter

Resperger Patrik

Grünhut Gábor