Külkereskedelmi Technikum

**Byte Brigade**

**Dokumentáció**

**BOOKHUB**

**online használtkönyv-csere platform**



Készítették:

Balázs Réka, Gerecs Diána, Csajbók-Reményi László

Budapest

2024

**Tartalomjegyzék**

1.Bevezetés1

1.1 Témaválasztás indoklása2

1.2 AI által generált képek használata3

1.3 Hasonló weboldal3

1.4 A project jelenlegi állapota és a jövője3

2. Fejlesztői dokumentáció6

2.1 Fejlesztői környezet6

2.2 Fejlesztéshez használt operációs rendszerek, böngészők6

2.3 Fejlesztéshez használt programok, programozási nyelvek és eszközök7

2.4 Adatbázis10

2.5 Változók és mezők elnevezése12

2.6 Frontend kialakítása Angulárral11

2.7 Osztályok elnevezése a Bulma keretrendszeréhez igazodva13

2.8 Backend megvalósítása SpringBoot alkalmazással14

2.9 Fejlesztési lehetőségek20

3 Tesztelés21

4 Felhasználói dokumentáció23

4.1 Böngészés a honlapon22

4.2 Regisztráció a honlapon23

4.3 Elérhető funkciók a bejelentkezés után23

4.4 Könyv felajánlása24

4.5 Könyv lefoglalása24

4.6 Hibajelzések jelentése24

5. Összefoglalás25

**1 Bevezetés**

**1.1 Témaválasztás**

*„A lélek hőmérője: a könyv.  Mondd, meg mit olvasol és megmondom: mi vagy. Megmondom mi az akaratod, célod, vágyad, álmod.” (Juhász Gyula)*

A Byte Brigade által megálmodott és megalkotott Bookhub vállalat weblapja alkalmas arra, hogy azokat az embereket kösse össze, akik szeretnek olvasni, és szívesen adják tovább azt, amit már elolvastak, vagy keresgélnek azok között a könyvek között, amik elérhetőek. Ezáltal ismeretségek születhetnek azonos érdeklődésű emberek között.

Mindeközben felhasználóink felelősséggel viseltetnek a környezet felé, tisztában vannak azzal, hogy rajtunk múlik, mit adunk tovább a gyermekeinknek. Fejlesztőként ehhez azzal járulhatunk hozzá, hogy lehetőséget biztosítunk arra, hogy a saját könyveiket szabadon felajánlják, cserélgessék egymás között, mindezt olyan rendezett körülmények között, amely egyszerre védi a felhasználók adatait, és minél inkább csökkenti a lehetőségét annak, hogy valaki visszaéljen a felajánlók jóindulatával.

Szeretnénk elérni, hogy amikor valaki tovább szeretné adni a könyveit bármilyen oknál fogva, akkor ne ütközzön abba az akadályba, hogy nem tudja kinek, és emiatt egy jó állapotú könyv a hulladékban végzi, pedig más szívesen olvasná, csak nincs rá módja, hogy megvegye, esetleg nem is tud róla, hogy másnak a birtokában van, de már nincsen szüksége rá. Mindezeken túl olyan könyveket is megóvhatnánk a kidobástól, amiket már csak használtan lehet megszerezni egy-egy antikváriumban, pedig az oktatás során egyszerűen nélkülözhetetlen.

Nagy mennyiségű könyvek cseréjét az adminoknak lehetne jelezni, az adminisztrációs terhek csökkentése érdekében. Leginkább felsőfokú oktatási intézményekben jellemző, hogy a tantárgy oktatási színvonalát jelentősen csökkentheti egy-egy már el nem érhető könyv, amit a tanulók csak addig használnak, amíg meg nem szerzik az oklevelet.

**1.2 AI által generált képek használata**

A tervezgetés kezdeti fázisában elkezdtünk AI által generált képeket gyártani Hamar kialakult az egyetértés abban, hogy ezek megfelelő alapot fognak adni a vizuális megjelenéshez.

Az első a *Gencraft – AI Art Image and Video Generator* (<https://gencraft.com/>) volt, ezzel ingyenesen egy nap korlátos számú képet lehet legenerálni.

A második az *Image Creator from Microsoft Designer* ([A Microsoft Tervező Képkészítője (bing.com)](https://www.bing.com/images/create?toWww=1&redig=54A8DCDEEAC14C918E6B53B3D6112638)), ezzel korlátlan számú kép generálható egy nap, de az első 15 gyorsabban generálható.

A téma alapját a mackók adták, mégpedig azért, mert a fejlesztés korai szakaszában Gerecs Diána fejlesztő javaslatára bevezettük a rendszeres gumicukor fogyasztást, ami többnyire mackó formájú volt. Így aztán a mackók uralták a témákat a képeknél. Külön-külön generáltuk a képeket Csajbók-Reményik László és Balázs Réka. Az online találkozók alkalmával közösen döntöttük el, hogy melyik felületre melyik kép kerülhet. Tapasztalatok szerint lényegesen több képet kell generálni, mint amennyi végül is jó lesz.

A képek generálása közben sokat megtanultunk arról, hogy hogyan lehet és érdemes kiadni az utasításokat olyan módon a mesterséges intelligenciának, hogy az olyan képeket generáljon, amik közel állnak az elképzeléseinkhez. A végeredmény magáért beszél, azok a külső szemlélők, akik megtekintették a weblapot ezekkel a képekkel egyöntetűen elismerően nyilatkoztak róluk.

**1.3 Hasonló weboldal**

Jelenleg hasonló működő weboldalt nem találtunk, ami Magyarországon működik, ingyenesen elérhető, és használható. Azonban több olyan fórumot is találtunk, ahol igény mutatkozik arra, hogy jó lenne egy ilyen szolgáltatás, ahol a használt könyveket cserélhetik a regisztrált felhasználók.

Több nagyvárosban vannak kihelyezett ingyenes könyvmegállók, és ezeket szívesen használják az emberek. Ide azonban személyesen kell elmenni, és azok számára, akik közelében nincsen ilyen kihelyezve nem jelent megoldást. Arról nem beszélve, hogy sokkal egyszerűbb úgy keresni a tartalmak között, hogy csak akkor veszi fel valaki a kapcsolatot a felajánlóval, ha az ő számára érdekes könyv van a kínálatban. A kihelyezett könyvek esetében azonban ez a kényelmesebb áttekintése a választéknak egyszerűen nem lehetséges.

**1.4 A projekt jelenlegi állapota és jövője**

A projekt jelenlegi állapotában csak egy elképzelt vállalat egy jól felépített szerethető weboldallal, olyan képekkel, amiket mindenki szívesen nézeget, és aminek sokan örülnének, ha megvalósulna. Egyelőre nem kerestünk hozzá támogatót, és nem tervezzük azt, hogy ezt valóban életre keltsük, azonban nem is zárjuk ki teljesen ennek az lehetőségét.

A korlátokat egyértelműen az anyagi forrás megteremtése okozza, illetve az, hogy jelenleg egyikünk sem dolgozik főállásban fejlesztőként, így nehézséget okozna a fenntartása és üzemeltetése annak ellenére, hogy akiknek megmutattuk tetszetősnek találták, és látnának benne fantáziát.

**2. Fejlesztői dokumentáció**

**2.1 Fejlesztői környezet**

A hozzáférési lehetőségek úgy vannak kialakítva, hogy minden admin hozzáfér a forráskódokhoz, aki meghívást kapott a közös Github tárolóhoz. A backend és a frontend illetve a dokumentáció egyértelműen elkülönül a könyvtárszerkezetben. A fejlesztés közben verzió követést alkalmaztunk, az aktuális fejlesztés feltöltése előtt letöltöttük a további változtatásokat, ezen kívül rendszeresen tartottunk online megbeszéléseket zoom-on.

A fejlesztés folytatásához elsőként a Github-ról kell letölteni a program aktuális állapotát. A változtatások feltöltéséhez jogosultság szükséges.

Github link : <https://github.com/BalazsR2022/Byte_Brigade.git>

Az Angulár 16.0.1-es verziót használtuk, ehhez igazodjon a többi letöltött program kompatibilis verziója. A letöltendő programok listája az alábbiakban a használt programok cím alatt található.

A frontend indítása az Visual Studio Code-ban történik, npm start paranccsal. A XAMPP Controll panelen az Apache ,MySQL, Tomcat indítása szükséges. A helyi gépen létre kell hozni egy book nevű adatbázis-t. A backend indítása az IntelliJ IDEA-ben lehetséges, futtatáskor feltölti adatokkal az adatbázist.

**2.2 Fejlesztéshez használt operációs rendszerek:**

|  |  |
| --- | --- |
|  | Windows |
| A képen fekete, kör, fekete-fehér, csillagászat látható  Automatikusan generált leírás | Debian |

**Fejlesztéshez használt böngészők:**

|  |  |
| --- | --- |
|  | Edge |
| A képen kör, Színesség, Grafika, Grafikus tervezés látható  Automatikusan generált leírás | Chrome |
| A képen kör, Színesség, gömb, bolygó látható  Automatikusan generált leírás | Chromium |

**Fejlesztéshez használt programok:**

|  |  |
| --- | --- |
| logo | Visual Studio Code |
| A képen szimbólum, Betűtípus, Grafika, képernyőkép látható  Automatikusan generált leírás | IntelliJ |
| A képen szimbólum látható  Automatikusan generált leírás | Angular |
|  | Bulma keretrendszer |
| A képen szimbólum, Grafika, tervezés látható  Automatikusan generált leírás | XAMPP |
| A képen szöveg, embléma, Betűtípus, tervezés látható  Automatikusan generált leírás | PHPMyAdmin |
| A képen óra, multimédia, tervezés látható  Automatikusan generált leírás | Terminátor |
|  | Chocolatey |
| A képen fekete, sötétség látható  Automatikusan generált leírás | Postman |

**Fejlesztéshez használt programozási nyelvek:**

|  |  |
| --- | --- |
| **A képen fekete, sötétség látható  Automatikusan generált leírás** | **HTML** |
| **A képen tervezés, művészet látható  Automatikusan generált leírás alacsony megbízhatósággal** | **JAVA** |
| **A képen Grafika, Színesség, zöld látható  Automatikusan generált leírás** | **CSS** |
| A képen embléma, Betűtípus, szimbólum, szöveg látható  Automatikusan generált leírás | **PHP** |
|  | **JavaScript** |

**Fejlesztéshez használt eszközök**

DESKTOP-P5Q6LET Processzor AMD Ryzen 5 3600 6-Core Processor 3.59 GHz Memória mérete 16,0 GB , Termékazonosító 00330-80000-00000-AA780 Rendszer típusa 64 bites operációs rendszer, x64-alapú processzor Kiadás Windows 10 Pro Verzió 22H2 Minőség Windows Feature Experience Pack 1000.19054.1000.0

ASUS-VIVOBOOK X515EA-BQ1182 15.6’’ FullHD laptop, Intel Core i3-115G4, 8GB, 256G SSd, Intel

ASUS VIVOBOOK S15 S513 OLED Intel® Core™ i7-1165G7 Processor (12M Cache, up to 4.70 GHz, with IPU) 16GB DDR4 3200MHz RAM (2x8GB) Intel® Iris® Xe Graphics

**2.4 Adatbázis**

A *Bookuser* adattábla, amely tárolja a regisztrál felhasználók adatait. a ki-és bejelentkezéshez szükséges. Elnevezése egyértelműen utal arra, hogy a regisztrált felshasználó egyben könyvtulajdonos, hiszen vagy felajánl könyvet, vagy nagy valószínűséggel éppen foglalni fog egyet és ebből a célból regisztrál.

Az első mező az *id* “AUTO\_INCREMENT” tulajdonsággal rendelkezik, automatikusan jön létre a regisztrációkor. Hossza 11 karakter lehet, ez elegendő a felhasználók egyedi id azonosítójának tárolásához, amely egyértelműen megkülönböztet minden egyes felhasználót úgy, hogy az adatok egyértelműen elkülönüljenek más regisztrált felhasználók adataitól. A tulajdonság *int* váltózóként van tárolva, mely az integer rövidítése, egész számok tárolására szolgál, nincs szükség ennél összetettebb azonosító használatára ennek a honlapnak a használatához.

Az int változóban tárolt adatok kis méretű adatok, kevés helyet igényelnek az adatbázisban. Az egyik legelterjedtebb változó az adatbázisokban, az adatbázis kezelő rendszerek támogatják a használatát. Nem ismétlődhet ez a tulajdonság, ugyanaz az *id* nem lesz kiosztva a felhasználó adatainak törlése után sem másik regisztrált felhasználó részére. Egyértelmű azonosítást tesz lehetővé, használható idegen kulcsként más adattáblával történő összekötéshez.

A *username* tulajdonság 50 karakter hosszú lehet, szabadon választott a felhasználó regisztrációjánál. Ez a hosszúság valószínűleg elég lesz egy átlagos felhasználónév megadásához. A felhasználó a felhasználónév és a jelszó megadásával tud belépni a honlapra, könyveket felajánlani és foglalni.

A password tulajdonság tárolja a regisztrált felhasználó által szabadon választott jelszót, hossza maximum 255 karakter lehet.

A *Books* adattábla a regisztrált felhasználók által felajánlott könyvek adatait tárolja. Az *id* tulajdonság automatikusan jön létre, annak beállítása illetve utólagos módosítása nem lehetséges. Az *owner\_id* tartalma azonos a bookusers tábla *id*-vel, elsődleges kulcsként funkcionál, biztosítja a két adattábla közötti kapcsolatot.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

**2.5 Változók és mezők elnevezése**

A frontendben és a backendben is a tiszta kódnak megfelelő változóneveket használtunk, amelyek egyértelműen utalnak az adott változó céljára és használatára.

Frontend deklarált változónevek:

* user (regisztrált felhasználó)
* name(regisztrált felhasználó neve)
* email(regisztrált felhasználó emailcíme)
* password(regisztrált felhasználó jelszava)
* county(regisztrált felhasználó által jelölt megye)

Backend deklarált változónevek:

* book (felajánlott könyv)
* id (könyv id-ja)
* title (könyv címe)
* author (könyv szerzője)
* publisher (felajánló regisztrált felhasználó)
* category (könyv kategóriája)
* county (regisztrált felhasználó által jelölt megye)
* quality (könyv állapota)

A regisztrációkor megadott adatokat tároljuk változókban, azoknak vizsgálata csak a formai követelményeknek való megfelelés szempontjából szükséges. Azok valódiságát semmilyen szempontból nem vizsgáljuk, tartalmukért a regisztrált felhasználó a felelős.

**2.6 Frontend kialakítása Angulárral**

Az Angular használatában hamar meg tudtunk állapodni. Mellette szóltak a következő érvek:

1. *Keresztplatformos támogatás:* Angular támogatja a keresztplatformos fejlesztést, amely lehetővé teszi az alkalmazások futtatását többféle platformon, beleértve a webböngészőket, mobilalkalmazásokat és asztali alkalmazásokat is.
2. *Kiterjedt funkciókészlet:* Angular számos beépített funkciója és modullja megkönnyíti a fejlesztést, mint például komponensek, modulok, szolgáltatások, HTTP kliens, formokkezelés.
3. *MVC architektúra:* Az Angular keretrendszer MVC (Model-View-Controller) architektúrát alkalmaz, amely lehetővé teszi az alkalmazások komponensekre és modulokra történő logikus felosztását, ami javítja az alkalmazás karbantarthatóságát és skálázhatóságát.
4. *Egyértelmű dokumentáció*: Az Angularnak nagyon jó dokumentációja van, és mivel már azelőtt elkezdtük az ismerkedést a programmal, hogy az iskolában tananyag lett volna, mindeképpen olyan keretrendszert keretünk, amit egyedül is el tudunk kezdeni tanulni.
5. *Tesztelhetőség:* Angular támogatja az egységteszteket és az integrációs teszteket, amelyek segítenek biztosítani az alkalmazások stabilitását és megbízhatóságát. Egyszerűsíti a tesztelést a projektmunkában.
6. *Optimalizált teljesítmény:* Az Angular beépített eszközökkel rendelkezik a teljesítmény optimalizálására, gyors betöltési időt tesz lehetővé. Ez egy honlapnál jelentős előnyt jelent. A felhasználók nem szeretnek sokat várni egy honlap betöltődésére.
7. *Rugalmas adatkötés*: Az Angular kétirányú adatkötést biztosít, amely lehetővé teszi az adatok dinamikus megjelenítését az alkalmazás felületén, és azonnali frissítést biztosít az adatok megváltozása esetén.

|  |  |
| --- | --- |
| ***komponens neve*** | ***feladata*** |
| *about* | bemutatkozás, célok bemutatása, küldetés meghatározása |
| *bulma-card* | dinamikusan generált listát jelenít meg a könyvekről |
| *bulma-footer* | láblécet jelenít meg a honlapon |
| *contact* | kapcsolati adatok megjelenítése |
| *home* | a Bulma Card komponensét használja a felhasználói felületének megjelenítésére |
| *login* | bejelentkezéshez szükséges űrlapot jelenít meg |
| *navbar* | navigációs sávot jelenít meg a honlap tetején. A logóra való kattintás visszaviszi az oldal főoldalára míg a hamburger ikonra való kattintás megnyitja vagy bezárja a mobilmenüt. |
| *redbutton* | a felhasználót visszanavigálja arra az oldalra, ahol nem a jó gombot nyomta meg |
| *service* | bemutatja a honlapon igénybevehető szolgáltatásokat |
| *signup* | feliratkozáshoz szükséges űrlapot jelenít meg |
| *surprise* | lehetőséget biztosít a felhasználónak egy véletlenszerű könyv választására |
| *user* | megjeleníti a regisztrált felhasználó adatait |

**2.7 Osztályok elnevezése a Bulma keretrendszer szerint**

Például az alábbi kódrészletben a container osztály használatával a div elem felveszi a konténer tulajdonságait. és a tartalom középre igazítva jelenik meg igazodva az oldal szélességéhez

A column class segítségével a "has-text-primary-dark" egy osztálynevet jelöl, amelyet a Bulma CSS keretrendszer használ. Ez a stílus beállítja a szöveg színét sötét árnyalatú kékre. A Bulma dokumentációban kereshetető ki a megfelelő színárnyalat, amit látni szeretnénk a megjelenő honlapon. A „columns” class hozzáadásával az alárendelt div elem a Bulma grid rendszerének megfelelően jelenik meg, ami sokkal esztétikusabb megjelenítést tesz lehetővé az eredeti html kódhoz képest egy egyszerű kódmódosítással.

A margin értéket sem szükséges külön beállítani a css kódban, az mt-2 class hozzáadásával margin-top értéket ad az oszlopoknak, így a felső margót növelve beállítja a megfelelő távolságot a honlapon megjelenő elemek között. A column class az oszlopok egymás mellé rendezésében segít.

A card class kártyaszerű megjelenést ad a benne elhelyezett elemeknek. Ez a kártya szerű megjelenés leegyszerűsítette a képek megjelenését és formázását az oldalon, egységes megjelenést tett lehetővé.

Az alábbi szín a következő színkódot jelöli a táblázat szerint: "hsl(171, 100%, 29%)", amely a HSL (Hue, Saturation, Lightness) színmodell alapján határozza meg a szín árnyalatát, telítettségét és világosságát. Ebben az esetben a szín egy kék árnyalata.

A title class használatával a paragraph elem kep kiemelt megjelenést további formázás nélkül is.

<div class="container">

  <div class="columns mt-2">

    <div class="column">

      <div class="card has-background-primary-light">

        <div class="card-content">

          <p class="title has-text-primary-dark has-text-primary-dark">

            "Olvass, cserélj, játssz."

          </p>

**2.8 Backend megvalósítása SpringBoot alkalmazással**

A backend a SpringBoot alkalmazásban lett megvalósítva. A SpringBoot használata nagyban egyszerűsítette a backend megvalósítását beépített mechanizmusok segítségével. Ezeknek a funkciónak az eléréséhez importokat kell az egyes fájlok elején használni.

A program belépési pontja az alábbiakban található *BackendApplication.java* fájl. A *public static void main(String[] args* metódus van benne definiálva, amely a program végrehajtásának kezdetén indul el. Ez a metódus hívja meg a Spring Boot alkalmazás indításához szükséges *SpringApplication.run(BackEndApplication.class, args)* függvényt.

A fájl tartalmazza továbbá az *@SpringBootApplication* annotációt is, ami azt jelzi, hogy ez a fájl a Spring Boot alkalmazás fő konfigurációs osztálya. Az annotáció kombinálja a *@Configuration, @EnableAutoConfiguration, és @ComponentScan* annotációkat, ezáltal automatikusan konfigurálja a Spring alkalmazást és betölti az összes szükséges beállítást. Ez teszi lehetővé, hogy az alkalmazás könnyedén futtatható legyen a Spring Boot alapértelmezett beállításaival. A fájlban található *@Bean* annotáció pedig egy *PasswordEncoder* típusú bean létrehozására szolgál. Ez a bean a jelszavak titkosítására szolgáló *BCryptPasswordEncoder* példányt hozza létre, amelyet később a biztonsági konfigurációban lehet használni.

package com.geolidth.BackEnd;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
  
@SpringBootApplication  
public class BackEndApplication {  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(BackEndApplication.class, args);  
 }  
  
}

Az alábbiakban található a ***BookCotroller.java*** fájl tartalma, amely a könyvekkel kapcsolatos HTTP kérések kezéséért felelős. A fájlban a REST API funkciói valósulnak meg a következő módon:

1. *Könyvek lekérdezése: Az @GetMapping* annotáció segítségével a *getBooks* metódus felelős az összes könyv lekérdezéséért.
2. *Könyv részleteinek lekérdezése: Az @GetMapping("/{id}")* annotációval ellátott *getById* metódus felelős egy adott könyv részleteinek lekérdezéséért az azonosító alapján.
3. *Könyv mentése: Az @PostMapping* annotációval ellátott *save* metódus felelős az új könyv mentéséért. A kliens által küldött új könyv adatait egy *NewBook* objektumban kapja meg, és aztán a *bookService* segítségével elmenti az adatbázisba.
4. *Könyv frissítése: Az @PutMapping("/{bookId}")* annotációval ellátott *updateBook* metódus felelős egy meglévő könyv frissítéséért. A kliens által küldött frissített könyv adatokat egy *UpdateBook* objektumban kapja meg, és aztán a *bookService* segítségével frissíti az adatbázisban.
5. *Könyv törlése: Az @DeleteMapping("/{bookId}")* annotációval ellátott *deleteBook* metódus felelős egy könyv törléséért az azonosító alapján.

package com.geolidth.BackEnd.Controllers;  
  
import com.geolidth.BackEnd.models.dao.BookUser;  
import com.geolidth.BackEnd.models.dto.UpdateBook;  
import com.geolidth.BackEnd.models.dao.Book;  
import com.geolidth.BackEnd.models.dto.NewBook;  
import com.geolidth.BackEnd.services.BookService;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.core.Authentication;  
import org.springframework.web.bind.annotation.\*;  
  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
 private final BookService bookService;  
 public BookController(BookService bookService) {  
 this.bookService = bookService;  
 }  
  
 @GetMapping  
 public ResponseEntity<List<Book>> getBooks(@RequestParam Optional<String> q) {  
 List<Book> books = bookService.getBooks(q);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(books);  
 }  
 @GetMapping("/{id}")  
 public ResponseEntity<Book> getById(@PathVariable int id) {  
 Book book = bookService.getById(id);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(book);  
 }  
  
 @PostMapping  
 public ResponseEntity<Book> save(@RequestBody NewBook newBook, Authentication auth) {  
 BookUser user = (BookUser) auth.getPrincipal();  
 Book savedBook = bookService.save(user.getId(), newBook);  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).body(savedBook);  
  
 }  
 @PutMapping("/{bookId}")  
 public ResponseEntity<Book> updateBook(  
 @PathVariable int bookId,  
 @RequestBody UpdateBook updateBook, Authentication auth) {  
 BookUser user = (BookUser) auth.getPrincipal();  
 Book updatedBook = bookService.updateBook(user.getId(), bookId, updateBook);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(updatedBook);  
 }  
  
  
 @DeleteMapping("/{bookId}")  
 public ResponseEntity<?> deleteBook(@PathVariable Integer bookId, Authentication auth) {  
 BookUser user = (BookUser) auth.getPrincipal();  
 bookService.deleteBook(user.getId(), bookId);  
 return ResponseEntity.*status*(HttpStatus.*NO\_CONTENT*).build();  
 }  
  
  
 }

*LoginController* osztály felelős a bejelentkezési adatok fogadásáért, validálásáért és továbbításáért a *LoginService* felé, majd a válasz visszaadásáért a kliensnek.

package com.geolidth.BackEnd.Controllers;  
  
import com.geolidth.BackEnd.models.dto.Login;  
import com.geolidth.BackEnd.models.dto.Token;  
import com.geolidth.BackEnd.services.LoginService;  
import lombok.AllArgsConstructor;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import static org.springframework.http.HttpStatus.*OK*;  
  
@RestController  
@AllArgsConstructor  
public class LoginController {  
  
 private LoginService loginService;  
  
 @PostMapping("/login")  
 public ResponseEntity<Token> login(@RequestBody Login login) {  
 return ResponseEntity.*status*(*OK*).body(loginService.login(login));  
  
  
 }  
}

A *UserController* felelős azért, hogy kezelje az olyan felhasználókkal kapcsolatos kéréseket, mint például új felhasználó létrehozása vagy egy meglévő felhasználó lekérdezése.

package com.geolidth.BackEnd.Controllers;  
  
import com.geolidth.BackEnd.models.dao.BookUser;  
import com.geolidth.BackEnd.models.dto.NewUser;  
import com.geolidth.BackEnd.services.BookService;  
import com.geolidth.BackEnd.services.UserService;  
import lombok.AllArgsConstructor;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import static org.springframework.http.HttpStatus.*CREATED*;  
import static org.springframework.http.HttpStatus.*OK*;  
  
@RestController  
@RequestMapping("/users")  
@AllArgsConstructor  
public class UserController {  
  
 private UserService userService;  
  
  
 @GetMapping("/{id}")  
 public ResponseEntity<BookUser> getById(@PathVariable Integer id) {  
 return ResponseEntity.*status*(*OK*).body(userService.getById(id));  
 }  
  
 @PostMapping  
 public ResponseEntity<BookUser> signUp(@RequestBody NewUser newUser) {  
 return ResponseEntity.*status*(*CREATED*).body(userService.save(newUser));  
 }  
}

Az exceptions fájlok felelnek a kivételkezelésért. Minden olyan esetben, amikor a program nem olyan adatot kap a felhasználótól, ami a működéséhez szükséges vagy figyelmeztetést, vagy hibaüzenetet kap, amelyből egyértelműen kiderül, hogy mit tud tenni a honlap hibamentes használatának folytatásához.

Ezek a fájlok a következők:

BookNotFoundException-keresett könyv nem található az adatbázisban

ForbiddenActionException-meg nem engedett műveletet próbál végrehajtani a felhasználó

NoSuchBookException- a keresett könyv nem található

NoSuchUserException-ilyen felhasználó nem található

Alábbiakban a *RestControllerExceptionHandler* fájl tartalma található. Ez a fájl felel azért, hogy a felhasználó a megfelelő hibaüzenetet kapja téves adatbevitel esetén:

package com.geolidth.BackEnd.exceptions;  
  
import com.geolidth.BackEnd.models.dto.ErrorMessage;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.MissingRequestHeaderException;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.RestControllerAdvice;  
  
import java.util.NoSuchElementException;  
  
import static org.springframework.http.HttpStatus.\*;  
  
@RestControllerAdvice  
public class RestControllerExceptionHandler {  
 @ExceptionHandler(MissingRequestHeaderException.class)  
 public ResponseEntity<ErrorMessage> handleBadRequestExceptions(MissingRequestHeaderException e) {  
 return ResponseEntity.*status*(*BAD\_REQUEST*).body(new ErrorMessage(e.getMessage()));  
 }  
  
 @ExceptionHandler(WrongUsernameOrPasswordException.class)  
 public ResponseEntity<ErrorMessage> handleUnauthorizedExceptions(WrongUsernameOrPasswordException e) {  
 return ResponseEntity.*status*(*UNAUTHORIZED*).body(new ErrorMessage(e.getMessage()));  
 }  
  
 @ExceptionHandler(ForbiddenActionException.class)  
 public ResponseEntity<ErrorMessage> handleForbiddenActionExceptions(ForbiddenActionException e) {  
 return ResponseEntity.*status*(*FORBIDDEN*).body(new ErrorMessage(e.getMessage()));  
 }  
  
 @ExceptionHandler(NoSuchElementException.class)  
 public ResponseEntity<ErrorMessage> handleNotFoundExceptions(NoSuchElementException e) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(new ErrorMessage(e.getMessage()));  
 }  
}

Néhány fontos részlet a kódban:

1. *@RestControllerAdvice* annotáció: Ez az annotáció jelzi, hogy ez az osztály globális kivételkezelő osztály, amely az összes *@RestController* osztályban keletkező kivételt kezeli.
2. *@ExceptionHandler* annotációk: Ezek az annotációk meghatározzák, hogy milyen típusú kivételeket kezeljen az adott metódus. Amikor a megadott típusú kivétel keletkezik, a megfelelő metódus hívódik meg.
3. *handleBadRequestExceptions(), handleUnauthorizedExceptions(), handleForbiddenActionExceptions(), handleNotFoundExceptions():* Ezek a metódusok a megfelelő típusú kivételeket kezelik, és visszaadnak egy *ResponseEntity* objektumot, amely tartalmazza a megfelelő HTTP státuszkódot és üzenetet.
4. *ErrorMessage* objektum: Ez egy egyéni DTO (adatátviteli objektum), amely egy üzenetet tartalmaz, és visszaküldi azt a kliensnek a kivétel kezelése során.

WrongUsernameOrPasswordException- hibás felhasználónév vagy jelszó

Hiányzó könyvek kivételkezelésével foglalkozó osztályok összehasnolítása:

1. *NoSuchBookException* :

1.1.Kiterjeszti a *NoSuchElementException* osztályt.

1.2.Ez a fajta kivétel olyan helyzetekben kerül használatra, amikor egy adott könyvet nem lehet megtalálni vagy elérni.

1.3A konstruktorban beállítja a kivétel üzenetét egy konstans értékre, ami jelzi, hogy a keresett könyv nem található.

1. *BookNotFoundException* :

2.1Kiterjeszti a *RuntimeException* osztályt.

2.2 Ez a fajta kivétel általában olyan általános helyzetekben kerül használatra, amikor egy könyv nem található meg vagy nem érhető el, de nem szükségszerűen kapcsolódik a *NoSuchElementException*-hoz.

* 1. A konstruktorban paraméterként megadható egy üzenet, amely jelzi a kiváltó okot vagy részleteket a kivételről.

Összességében a *NoSuchBookException* specifikusabb típusú kivételt jelöl, amely azt jelzi, hogy egy adott könyv nem található, míg a *BookNotFoundException* általánosabb kivétel típus, amely akár bármilyen okból keletkezhet, ha egy könyv nem található vagy elérhető.

A Book.java fájl a com.geolidth.BackEnd.models.dao csomagban részt vesz az adatbázis-séma modellezésében, és az alkalmazásban a könyvek tárolásáért és kezeléséért felelős. Néhány fontos dolog, amit érdemes megjegyezni ebben az osztályban:

1. Az *@Entity* annotációval megjelölt osztályokat JPA entitásoknak tekintik, amelyek adatbázistáblákhoz rendelődnek.
2. A *@Table* annotáció segítségével meghatározható, hogy melyik adatbázistábla tartalmazza az entitást. Ebben az esetben a tábla neve "books".
3. A *@Id* annotációval megjelölt mező az adatbázisban a főkulcs (primary key).
4. A *@GeneratedValue* annotáció jelzi, hogy az *id* mező automatikusan generálódik, és az adatbázis generált értéket használja.
5. A *@Column* annotáció segítségével konfigurálhatók az oszlopok a táblában.
6. A *@ManyToOne* annotáció jelzi, hogy ez a könyv entitás egy másik entitáshoz, egy *BookUser* objektumhoz tartozik.
7. A *@JsonIdentityReference* és *@JsonProperty* annotációk segítenek az objektum hivatkozásainak kezelésében, amikor JSON-be kerül konvertálásra.

A BookUser.java osztály felelős az alkalmazásban a felhasználók tárolásáért és kezeléséért felelős, valamint a felhasználókhoz rendelt könyvek kezeléséért. Néhány annotáció, amely az előző fájlban nem volt használva:

1. A *@OneToMany* annotáció jelzi, hogy ez a felhasználó entitás több könyvet is tartalmazhat, és az adatbázisban ez a kapcsolat egy-minden kapcsolatot jelent.
2. Az osztály implementálja a Spring Security *UserDetails* interfészt, ami azt jelenti, hogy ez a felhasználó osztály a Spring Security keretrendszer által kezelt felhasználót reprezentálja.
3. Az *@JsonIgnore* annotáció jelzi, hogy a jelszó mezőt nem kell JSON formátumban reprezentálni, így az nem fog megjelenni az API válaszokban

A dto csomagban található fájlok DTO (Data Transfer Object) osztályokat definiálnak a *com.geolidth.BackEnd.models.dto* csomagban. Ezeknek a dto fájloknak a célja, hogy hibákat tartalmazó üzeneteket reprezentáljon a backend alkalmazásban. Néhány fontos dolog, amit meg kell jegyezni ebben az osztályban:

1. Az *@Data* annotáció a Lombok könyvtár egyik funkcióját jelenti, ami automatikusan generál gettereket, settereket, *toString(), equals() és hashCode()* metódusokat az osztály attribútumaihoz.
2. Az *ErrorMessage* osztály két attribútumot tartalmaz: *status és message.*
3. Az osztályban van egy konstruktor, ami egy *message* paramétert vár. Ez a konstruktor inicializálja az *status* attribútumot az "error" szöveggel, és a *message* attribútumot a paraméterben kapott üzenettel.

**2.9 Fejlesztési lehetőségek**

Megfelelő komponensek hozzáadásával a honlap alkalmassá tehető regisztrált felhasználólk közötti közvetlen kommunikációra is forum felület kialakításával. Ez azonban mindenféleképpen erőforrás igényes, folyamatos felügyeletet igényel. Szükséges szűrőprogramok alkalmazása a kommunkáció stílusának és céljának ellenőrzésére . Ebben az esetben a felhasználói feltételek olyan kialakítása válna szükségessé amellyel a legtöbb visszaélés megakadályozható.

Lehetséges az adminok részére blog kialakítása, ahol személyes élményeket oszthatnának meg a felhasználókkal elősígítve az oldal aktív használatát.

Személyes találkozók megszervezéséhez nyújthat segítséget egy olyan menüpont, ahol az adott évben tervezett személyes találkozások láthatók. Ez megjelenhet továbbá egy

**3 Felhasználói dokumentáció**

**3.1. Honlap használata**

A weboldal használata lehetséges asztali gépről, laptopról, táblagépről, vagy mobiltelefonról. Webböngésző telepítésén kívül más program telepítésére és futtatására nem lesz szükséged, csak internetkapcsolatra.

Az alábbi böngészők bármelyikét elindíthatod:

|  |  |  |  |
| --- | --- | --- | --- |
| A képen kör, Színesség, Grafika, Grafikus tervezés látható  Automatikusan generált leírás |  | A képen rajzfilm látható  Automatikusan generált leírás | A képen kör, Színesség, gömb, bolygó látható  Automatikusan generált leírás |

Dupla kattintással tudod megnyitni a böngészőt és a következő url-t kell beírnod a kereső mezőbe:

A Bookhub weboldala egy különleges lehetőséget biztosít minden könyvszerető embernek, hogy a könyveid ne csak porosodjanak a polcodon, hanem új otthont és új gazdát találjanak. Fedezd fel a könyvcsere kalandját, ahol nem csak olvasol, hanem megiszthatod és cserélheted is a kedvenc történeteidet. csak jelentkezz be, és kezdődjön a kaland!

**3.2 Honlap funkciói**

A képen szimbólum, Grafika, képernyőkép, Téglalap látható

Automatikusan generált leírás Erre az ikonra kattintva minden oldalról a főoldalra jutsz vissza. A főoldalon az oldal tetején megjelenő menüben tudsz választani a honlap szolgáltatásai között.

A könyvek a főoldalon töltődnek be, itt megtekintheted a választékot bejelentkezés nélkül is. Válassz egy könyvet, jelezd a tulajdonosnak, és beszéljétek meg hogyan tudod átvenni. Töltsd fel régi könyveidet, és várd, hogy jelentkezzen érte az új tulajdonos!

Regisztrációhoz a Sign up menüpontban tudod megadni az adataidat. Az adatok valódisága a te felelősséged.

A képen képernyőkép látható

Automatikusan generált leírás

Adatvédelem: az ön adatait(nevét, címét, elérhetőségét) csak akkor ismerheti meg más felhasználó, amikor jelzi önnek, hogy valamelyik könyvét szeretné átvenni, és felveszik a kapcsolatot egymással. Kérjük járjon el figyelmesen, hogy kivel veszi fel a kapcsolatot. A felhasználó felelőssége, hogy valós adatokat adjon meg, azt a weboldal adminjainak nem áll módjában ellenőrizni. Visszaélés esetén kérem jelezze az adminoknak, hogy meg tudjuk tenni a további intézkedéseket.

Ha nem tudsz választani, kérd meg a zsákbamackót, hogy segítsen!

A képen szöveg, képernyőkép, szoftver, Webhely látható

Automatikusan generált leírás

Ha kiváncsi vagy az alapítókra válaszd az about menüpontot.

A kapcsolat menüpontban nyugodtan üzenhetsz az adminisztrátoroknak bármilyen problémával vagy kéréssel. Ne habozz igénybevenni ezt a lehetőséget, ha valamit nem találsz, vagy Vannak ötleteid, hogyn fejleszthetnénk tovább a honlapot. Tudod jelezni nekünk továbbá a tömeges cserére vonatkozó kérésedet is, így nem kell minden egyes könyvet egyesével felrögzítened.

A könyvek cseréjének gyakorlati megvalósítását segítendő a könyveknél jelezve van a megye, ahol átvehető személyesen .

Minden olyan könyvnél tudod az átvételi szándékot, amelyik a honlapon látható. A már lefoglalt könyvek eltűnnek a kínálatból, így nem tudsz olyan könyvet véletlenül lefoglalni, amit más már lefoglalt magának.

**3.3Hibajelzések, elérhetőségek**

Hibás adatok a regisztrációnál: Kérlek ellenőrizd, hogy a név mezőben nem szerepel-e téves karakter, a cím mezőben nem szerepel-e hibás karakter, illetve minden mezőt kitöltöttél-e a regisztrációs felületen?

Hibás adatok a könyvnél: Ellenőrizd újra, hogy a könyv címe nem túl hosszú-e? Ha nem fér ki a rendelkezésre álló mezőben, használj rövidítést.

Amennyiben a honlap hibásan jelenik meg, ellenőrizd, hogy a böngésződ frissítve van, és valóban a legújabb verziót használod.

Az adminok elérhetőségét az about menüpont alatt találod, a honlapon keresztül tudsz nekünk üzenetet küldeni. Ne habozz jelezni bármilyen hibát, amit találsz!

**4.Összefoglalás**