

Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku

Sistem za upravljanje bibliotekama

Projekat iz predmeta Praktikum - Napredne web tehnologije

ČLANOVI TIMA:

1. Dalila Kršlak (2226/18906)
2. Tajra Selimović (2296/18972)
3. Nejra Adilović (2218/19061)

ODGOVORNI NASTAVNIK:

V. prof. dr Anel Tanović

ODGOVORNI ASISTENT:

Irfan Prazina Mr. dipl. ing

Sarajevo, 2025.

SADRŽAJ

1. Opis teme.....	4
2. Funkcionalnosti sistema.....	5
2.1. Upravljanje knjigama (superadmin, admin i bibliotekari).....	5
2.2. Upravljanje bibliotekama (superadmin).....	5
2.3. Pretraga i filtriranje knjiga (korisnik).....	5
2.4. Rezervacija knjige (korisnik).....	5
2.5. Posudba i vraćanje knjiga (bibliotekar).....	6
2.6. Evidencija zakašnjenja i penalizacija (automatski).....	6
2.7. Statistika najposuđivanijih i razmijenjenih knjiga (superadmin, admin, bibliotekar).....	6
2.8. Upravljanje korisnicima i osobljem (superadmin).....	6
2.9. Praćenje nedostupnih knjiga (bibliotekar).....	6
2.10. Razmjena knjiga između biblioteka.....	7
3. Skice korisničkih interfejsa.....	8
4. ER Dijagram.....	14
5. Primjeri sličnih aplikacija.....	15
6. Mikroservisi.....	16
6.1. Servis za upravljanje knjigama, rezervacijama i posudbama.....	16
6.2. Servis za upravljanje bibliotekama i korisnicima.....	17
6.3. Servis za upravljanje transferima knjiga.....	18
7. CRUD operacije za svaki mikroservis.....	19
7.1. Servis za upravljanje knjigama, rezervacijama i posudbama.....	19
7.2. Servis za upravljanje bibliotekama i korisnicima.....	23
7.3. Servis za upravljanje transferima knjiga.....	27
8. Service Discovery i Load Balancing.....	33
9. Sinhrona komunikacija između mikroservisa.....	36
10. Dijagrami toka komunikacije.....	38
10.1. Kreiranje rezervacije i provjera korisnika.....	38
10.2. Posudba knjige uz provjeru dostupnosti i korisničkog statusa.....	39
10.3. Iniciranje transfera primjerka knjige između biblioteka.....	40
11. Upravljanje sigurnošću mikroservisa.....	41
11.1. Sigurnosna rješenja u mikroservisnoj arhitekturi.....	41
11.2. Primjena JWT rješenja u našem sistemu.....	42
11.3. Konfiguracija API Gateway-a.....	44
12. Asinhrona komunikacija i saga obrazac uz RabbitMQ.....	48
13. Dockerizacija sistema i orkestracija putem Docker Compose-a.....	50
14. Rad aplikacije – frontend prikaz.....	52

14.1. Početna stranica i login.....	52
14.2. Korisnik - Izgled stranice.....	53
14.3. Bibliotekar - Izgled stranice.....	56
14.4. Admin - Izgled stranice.....	62
14.5. Superadmin - Izgled stranice.....	63
15. Zaključak.....	68

1. Opis teme

Sistem za upravljanje bibliotekama predstavlja naprednu softversku platformu dizajniranu za centralizovano upravljanje više biblioteka koje funkcionišu unutar iste mreže, kao što je univerzitetski sistem, gradska mreža biblioteka ili organizacija s više odjela. Svaka pojedinačna biblioteka u mreži zadržava određeni nivo autonomije, s vlastitim administratorima i zaposlenicima koji su zaduženi za lokalno upravljanje fondom knjiga, korisnicima i dnevnim operacijama. Iznad svih lokalnih biblioteka nalazi se superadministrator, koji ima potpun uvid u funkcionisanje cjelokupnog sistema i ovlaštenja da interveniše, postavlja politike, odobrava razmjene i nadgleda opšte performanse sistema.

Jedna od ključnih karakteristika ovog sistema jeste mogućnost međusobne komunikacije između biblioteka, što omogućava jednostavnu i brzu razmjenu knjiga i druge bibliotečke građe. Ukoliko jedna biblioteka nema određenu knjigu, korisnik može provjeriti dostupnost te knjige u drugim bibliotekama u mreži i izvršiti rezervaciju ili zahtjev za posudbu, pri čemu se proces može automatizirati kroz sistem.

Za krajnje korisnike sistem nudi intuitivan i jednostavan korisnički interfejs putem kojeg mogu pretraživati dostupnu literaturu prema različitim kriterijima (naslov, autor, žanr, godina izdanja, itd.), pratiti trenutno stanje zaduženih knjiga, rezervisati ili produžiti posudbu, te dobijati obavijesti o isteku roka vraćanja ili dostupnosti rezervisanih naslova. Administrativni dio sistema omogućava vođenje evidencije o svim knjigama, uključujući informacije o stanju (npr. oštećenje, zaduženost, lokacija), te omogućava osoblju da prate radno vrijeme, zaduženja i druge aktivnosti. Sistem automatski generiše izvještaje o posudbama, najčitanijim knjigama, korisničkim statistikama i internim aktivnostima osoblja, što znatno olakšava upravljanje resursima i planiranje budućih nabavki.

Implementacijom ovakvog sistema postiže se značajno unapređenje u efikasnosti poslovanja, smanjuje se potreba za ručnim unosom podataka i administrativnim greškama, te se korisnicima omogućava brža i pouzdana dostupnost knjiga, bez obzira na lokaciju biblioteke. Na taj način, biblioteke postaju savremeni informacijski centri povezani u jedinstvenu digitalnu mrežu koja odgovara na potrebe modernog korisnika.

2. Funkcionalnosti sistema

Sistem obuhvata niz funkcionalnosti dizajniranih za olakšano i intuitivno korištenje. U nastavku je predstavljeno deset najvažnijih.

2.1. Upravljanje knjigama (superadmin, admin i bibliotekari)

Sistem omogućava dodavanje novih knjiga sa svim relevantnim podacima, uključujući naslov, autora, žanr, ISBN i dostupnost. Administratori i bibliotekari mogu uređivati informacije o knjigama, ažurirati status dostupnosti te brisati knjige koje više nisu u upotrebi. Knjige su organizovane tako da korisnici lahko mogu pretraživati i pregledavati njihove osnovne podatke.

2.2. Upravljanje bibliotekama (superadmin)

Superadmin može dodavati i uređivati biblioteke unutar sistema, unoseći podatke poput naziva, adrese i kontakt informacija. Svaka biblioteka može imati svog administratora i bibliotekare koji upravljaju njenim radom. Ova funkcionalnost omogućava centralizovanu kontrolu nad svim bibliotekama u sistemu.

2.3. Pretraga i filtriranje knjiga (korisnik)

Korisnici mogu pretraživati knjige na osnovu različitih kriterija, uključujući naslov, autora, žanr ili dostupnost u određenoj biblioteci. Sistem omogućava pregled osnovnih informacija o knjizi, uključujući broj dostupnih primjeraka. Osim toga, korisnici mogu filtrirati rezultate kako bi lakše pronašli knjige koje ih zanimaju.

2.4. Rezervacija knjige (korisnik)

Ako je knjiga dostupna u biblioteci, korisnik može izvršiti online rezervaciju i preuzeti je u dogovorenom roku. U slučaju da knjiga trenutno nije dostupna, sistem omogućava stavljanje korisnika na listu čekanja, gdje će biti obaviješten čim se knjiga vrati. Rezervacije se bilježe u sistemu, a bibliotekari ih mogu pratiti i obrađivati.

2.5. Posudba i vraćanje knjiga (bibliotekar)

Bibliotekari registruju posudbu knjiga, unoseći podatke o korisniku, datumu posudbe i krajnjem roku vraćanja. Kada korisnik vrati knjigu, sistem automatski ažurira njenu dostupnost u biblioteci. Ova funkcionalnost omogućava precizno praćenje statusa knjiga i sprječava gubitak knjiga.

2.6. Pregled rezervacija i posudbi (korisnik)

Sistem omogućava korisnicima da u svakom trenutku pregledaju sve svoje aktivne rezervacije i posudbe, zajedno s informacijama o datumu isteka. Na taj način korisnici mogu samostalno pratiti rokove i na vrijeme planirati vraćanje knjiga, čime se unapređuje korisničko iskustvo i smanjuje potreba za dodatnim intervencijama biblioteke.

2.7. Statistika najposuđivanijih i razmijenjenih knjiga (superadmin, admin, bibliotekar)

Sistem generiše izvještaje o najposuđivanijim knjigama u svakoj biblioteci ili na nivou cijelog sistema. Statistika uključuje broj posudbi za određeni vremenski period, što omogućava bibliotekarima da prate interes korisnika i optimizuju fond knjiga. Također, funkcionalnost uključuje praćenje transfera knjiga između biblioteka, bilježeći podatke o zahtjevima za razmjenu, datumima slanja i primanja te odgovornim osobama.

2.8. Upravljanje korisnicima i osobljem (superadmin)

Superadmin ima mogućnost kreiranja, uređivanja i brisanja korisničkih naloga za bibliotekare i administratore biblioteka. Svaki korisnik se povezuje sa svojom bibliotekom, a sistem omogućava dodjelu različitih nivoa pristupa. Ova funkcionalnost omogućava efikasnu organizaciju osoblja i kontrolu nad administrativnim pravima unutar sistema.

2.9. Praćenje nedostupnih knjiga (bibliotekar)

Kada korisnik traži knjigu koja nije dostupna u njegovoj biblioteci (npr. sve kopije su posudene), sistem je automatski označava kao nedostupnu. Ova oznaka omogućava bibliotekarima da prate potražnju za određenim naslovima i, ako je potrebno, pokrenu zahtjev za transfer iz druge

biblioteke. Na taj način se osigurava bolja dostupnost knjiga i efikasnija razmjena između biblioteka.

2.10. Razmjena knjiga između biblioteka

Biblioteke mogu međusobno komunicirati i slati zahtjeve za razmjenu knjiga kada određeni naslov nije dostupan u jednoj biblioteci, ali jeste u drugoj. Sistem omogućava praćenje transfera knjiga, od podnošenja zahtjeva do potvrde prijema. Ova funkcionalnost poboljšava dostupnost knjiga korisnicima i optimizuje njihovu raspodjelu između biblioteka.

3. Skice korisničkih interfejsa

The screenshot shows the BookWorm Admin application interface. At the top, there is a header bar with the title "BookWorm Admin" and "Admin". On the right side of the header, it shows the time "12:29 PM" and the date "March 15th, 2025". There is also a gear icon for settings.

The main content area is titled "Book Management". It features a search bar with the placeholder "Search by ID or Title" and a button labeled "Add Book".

The central part of the screen is a table with the following columns: ISBN, Title, Author, Genre, Availability, and Action. The table contains ten rows of data, all of which represent the book "The Fortress" by Meša Selimović. The "Availability" column alternates between "Available" and "Borrowed". The "Action" column contains three icons: a pencil, a trash can, and a magnifying glass.

ISBN	Title	Author	Genre	Availability	Action
1	The Fortress	Meša Selimović	Novel	Available	
1	Silence	Meša Selimović	Novel	Borrowed	
1	The Fortress	Meša Selimović	Novel	Available	
1	The Fortress	Meša Selimović	Novel	Available	
1	Circle	Meša Selimović	Novel	Borrowed	
1	The Fortress	Meša Selimović	Novel	Available	
1	The Fortress	Meša Selimović	Novel	Available	
1	The Fortress	Meša Selimović	Novel	Available	
1	The Fortress	Meša Selimović	Novel	Borrowed	
1	The Fortress	Meša Selimović	Novel	Available	

On the left side of the interface, there is a vertical sidebar with the following navigation options:

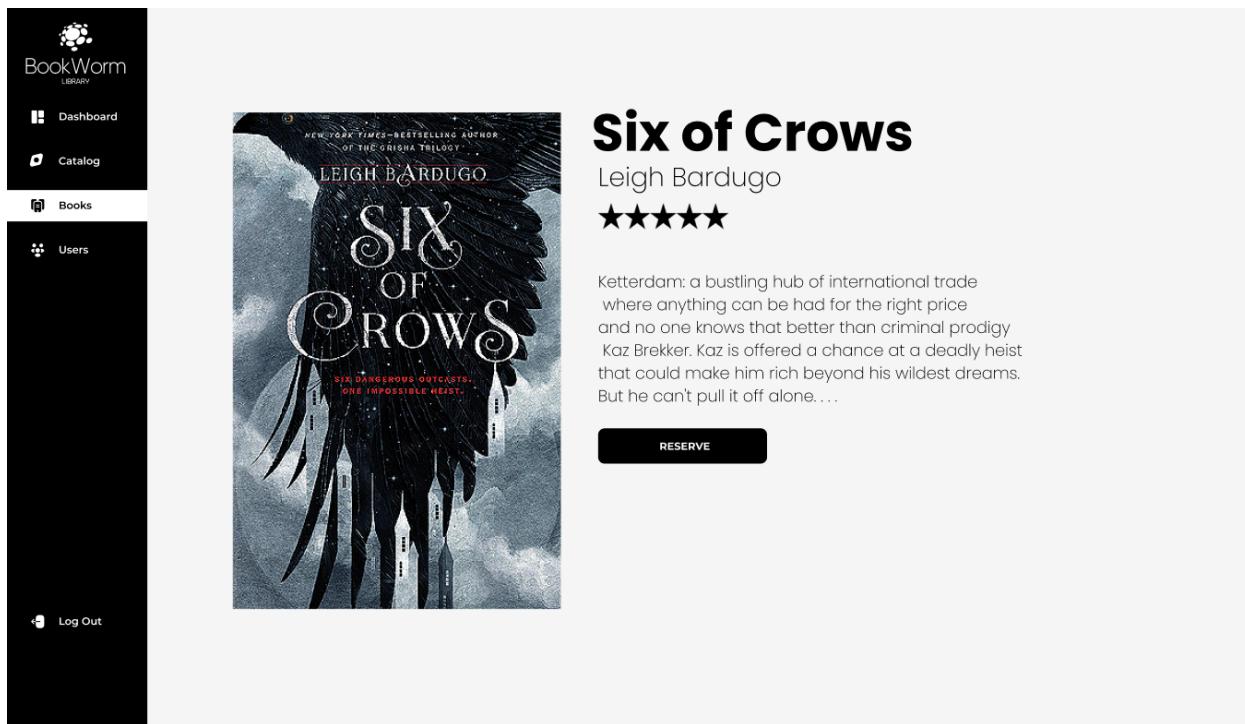
- Dashboard
- Catalog
- Books
- Users
- Branches
- Log Out

Slika 3.1. Upravljanje knjigama

ID	Name	Contact	Location	Action
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	
1	BookWorm Matara	0412410984	Matara	

Slika 3.2. Upravljanje bibliotekama

Slika 3.3. Pretraga i filtriranje knjiga



Slika 3.4. Rezervacija knjiga

Borrowed Books		Overdue Borrowers			
ID	User ID	Amount	Due Date	Date & Time	Action
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	
001	001	002 Books	13 - 03 - 2024	25-02-2024 10:39:43	

Slika 3.5. Posudba i vraćanje knjiga

The screenshot shows the BookWorm Admin dashboard. On the left is a sidebar with a dark theme containing the BookWorm logo, a navigation menu with 'Dashboard', 'Catalog', 'Books', 'Users', 'Branches', and 'Log Out', and a user profile for 'BookWorm Admin Admin'. The main content area has tabs for 'Borrowed Books' (selected) and 'Overdue Borrowers'. A search bar at the top right says 'Search by ID'. Below is a table with columns: ID, User ID, Title, Due Date, Date & Time, and Action. The table lists six entries for user ID 1, all due on 13-03-2024 at 10:39:43.

ID	User ID	Title	Due Date	Date & Time	Action
001	1	Six of Crows	13 - 03 - 2024	25-02-2024 10:39:43	
001	1	It Ends with Us	13 - 03 - 2024	25-02-2024 10:39:43	
001	1	Harry Potter Part 2	13 - 03 - 2024	25-02-2024 10:39:43	
001	1	Hunger Games	13 - 03 - 2024	25-02-2024 10:39:43	
001	1	Little Women	13 - 03 - 2024	25-02-2024 10:39:43	
001	1	Pirde and Prejudice	13 - 03 - 2024	25-02-2024 10:39:43	

Slika 3.6. Evidencija zakašnjenja i penalizacija

The screenshot shows the BookWorm Librarian dashboard. The sidebar is identical to the Admin version. The main area features a pie chart comparing 'Total Borrowed Books' (dark grey) and 'Total Returned Books' (light grey). Below the chart are summary statistics: 10 Branch count, 150 Users, and 1500 Books. To the right is a 'Book Transfers' section with a 'Request Transfer' button and a list of four transfer requests involving branches Matara, Malta, and Alipatina. At the bottom are two cards: 'Most Borrowed Books' (listing Six of Crows, It Ends with Us, and Normal people) and 'Books Statistic' (listing Steve Jobs, Zlatan Ibrahimović, Pride and Prejudice, Lies, Silence, and Prayer).

Slika 3.7. Statistika najposuđivanijih i razmijenjenih knjiga

ID	Name	Email	Role	Library	Action
1	Mehmed Selimović	mselimovic@gmail.com	User	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	Librarian	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	Librarian	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	Librarian	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	Librarian	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	User	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	User	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	User	BookWorm Matara	
1	Mehmed Selimović	mselimovic@gmail.com	User	BookWorm Matara	

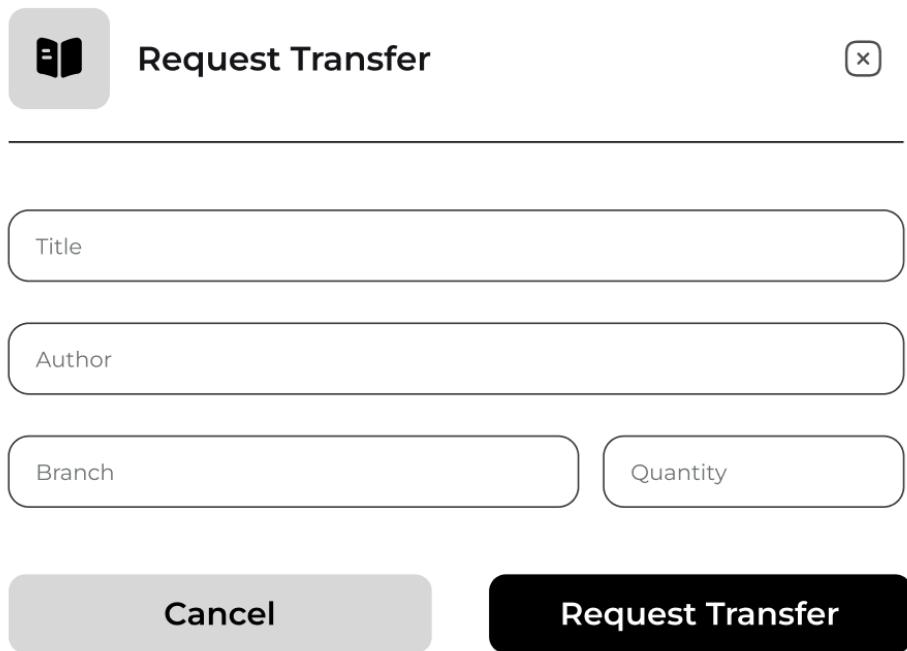
Slika 3.8. Upravljanje korisnicima i osobljem

Book Details

Book ID: 1	Borrowed Copies: 15
Title: Six of Crows	Available Copies: 0
Author: Leigh Bardugo	
Genre: Fantasy	

Request Transfer **Close**

Slika 3.9. Praćenje nedostupnih knjiga



The image shows a mobile application interface titled "Request Transfer". At the top left is a book icon. To the right of the title is a close button (an "X"). Below the title is a horizontal line. There are four input fields: "Title", "Author", "Branch", and "Quantity", each enclosed in a rounded rectangle. At the bottom are two buttons: "Cancel" (gray background) and "Request Transfer" (black background).

Request Transfer

Title

Author

Branch

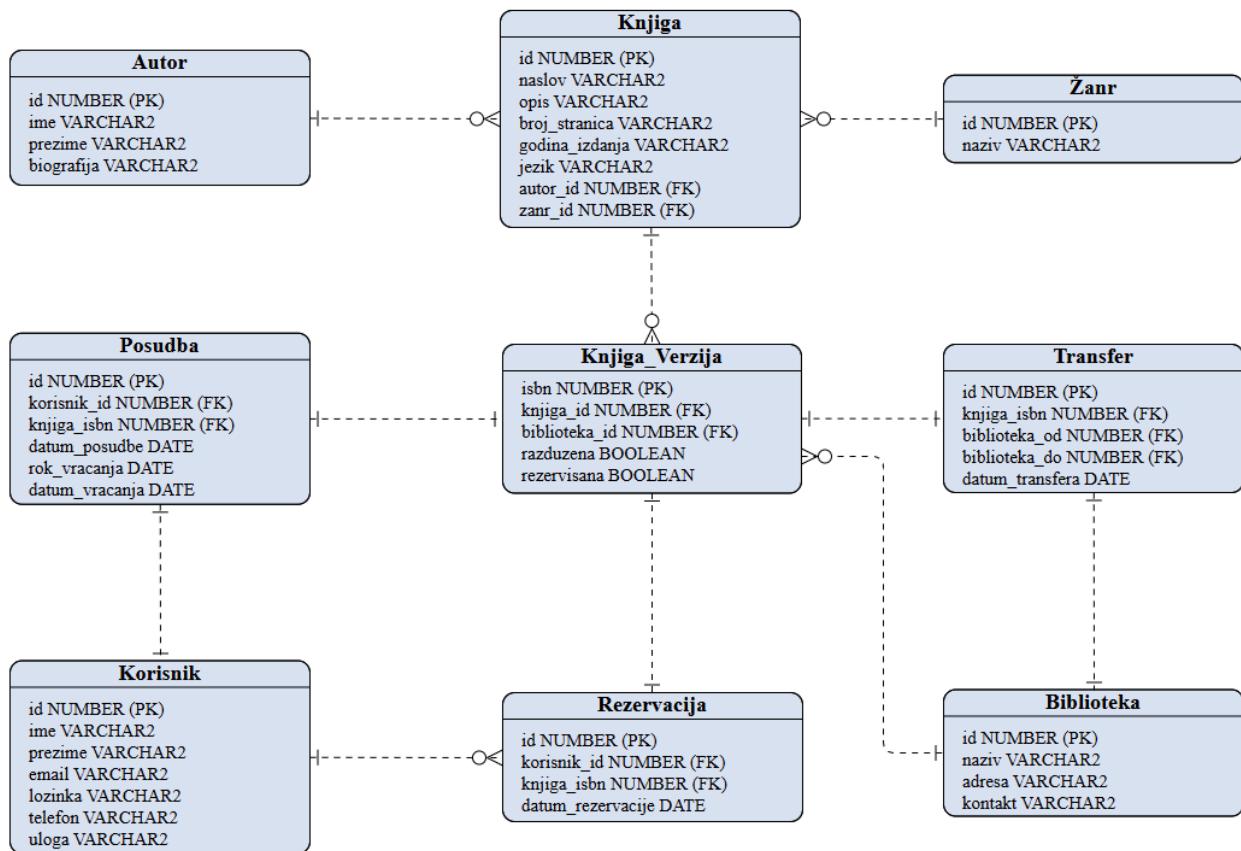
Quantity

Cancel Request Transfer

Slika 3.10. Razmjena knjiga između biblioteka

4. ER Dijagram

Na slici ispod možemo vidjeti kako izgleda naš Entity Relationship Diagram (ERD). Ono što je interesantno jeste da imamo dvije klase za knjigu koje se zovu *Knjiga* i *Knjiga_Verzija*. *Knjiga* u bazi podataka će predstavljati jednu vrstu knjige, kao naprimjer knjiga *Derviš i smrt*, dok *Knjiga_Verzija* će predstavljati određenu verziju knjige sa svojim identifikacijskim ključem ISBN. Ovo znači da ćemo u bazi podataka čuvati u tabeli *Knjiga* određenu knjigu, dok *Knjiga_Verzija* će predstavljati verziju te knjige odnosno tačno određenu knjigu.



Slika 4.1. ER dijagram

5. Primjeri sličnih aplikacija

Na tržištu već postoji nekoliko aplikacija i sistema koji omogućavaju upravljanje bibliotekama, olakšavaju posudbu i razmjenu knjiga te unapređuju korisničko iskustvo kroz digitalizovane servise. U nastavku su navedeni neki od najpoznatijih primjera takvih aplikacija i platformi.

1. Biblioteka Sarajeva: <https://www.bgs.ba/>
2. Knjižnice grada Zagreba: <https://www.kgz.hr/hr>
3. Library of Congress: <https://www.loc.gov/>

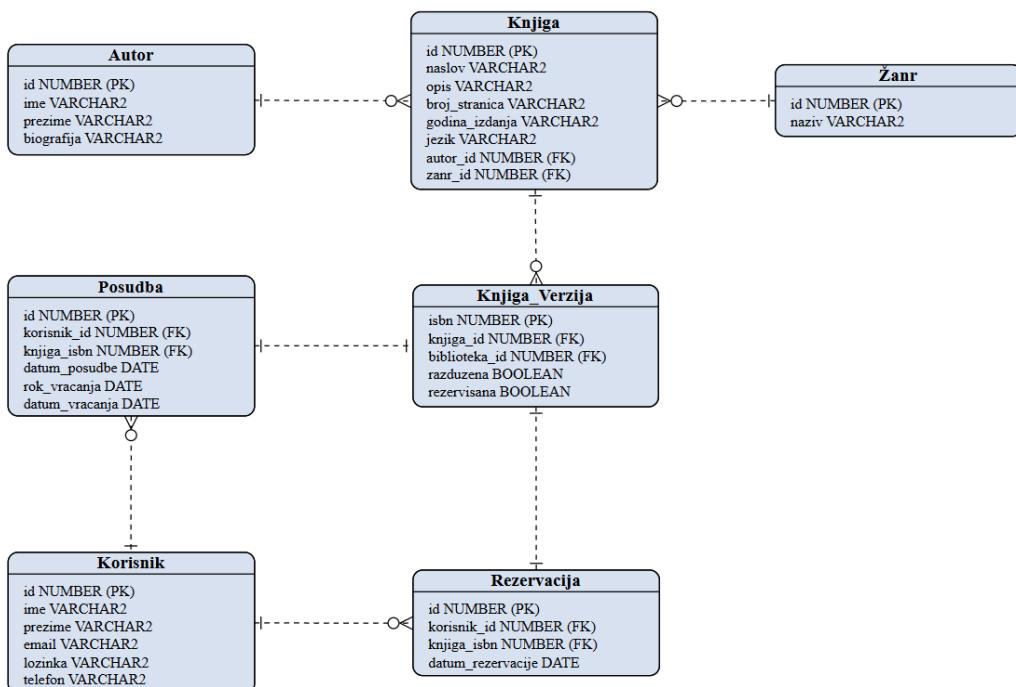
6. Mikroservisi

Sistem je podijeljen na tri glavna mikroservisa, pri čemu svaki mikroservis obavlja specifične funkcionalnosti unutar sistema za upravljanje bibliotekama.

6.1. Servis za upravljanje knjigama, rezervacijama i posudbama

Ovaj mikroservis obuhvata sve funkcionalnosti vezane za upravljanje knjigama i procesima posudbe. Glavne funkcionalnosti uključuju:

- Upravljanje knjigama (dodavanje, ažuriranje, brisanje i pregled knjiga)
- Praćenje nedostupnosti knjiga i obaveštavanje korisnika
- Pretraga i filtriranje knjiga prema različitim kriterijima
- Prikupljanje statistike o najposuđivanim knjigama
- Rezervacija knjiga i vođenje liste čekanja
- Posudba i vraćanje knjiga uz praćenje rokova
- Evidencija zakašnjenja i penalizacija korisnika

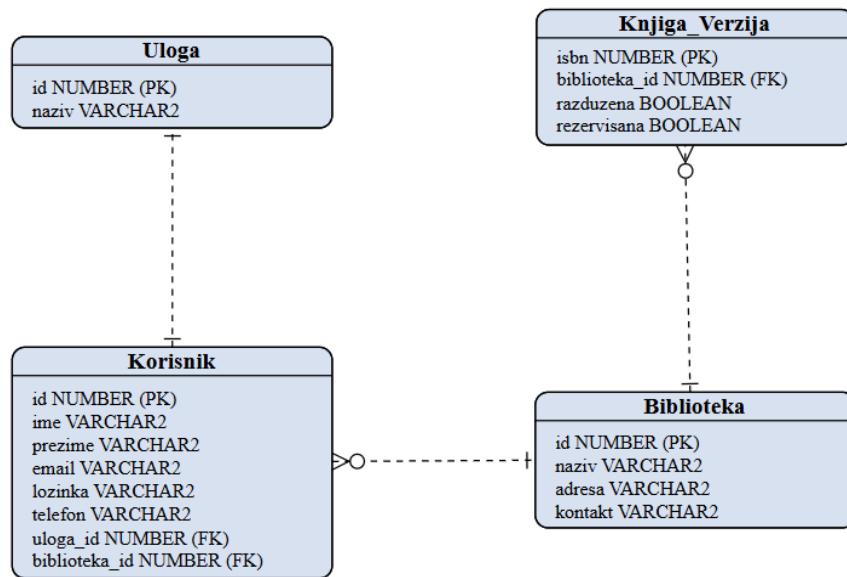


Slika 6.1. ERD servisa za upravljanje knjigama, rezervacijama i posudbama

6.2. Servis za upravljanje bibliotekama i korisnicima

Ovaj mikroservis se bavi administrativnim funkcijama sistema, omogućavajući upravljanje bibliotekama, korisnicima i osobljem. Njegove glavne funkcionalnosti su:

- Upravljanje bibliotekama (kreiranje, ažuriranje, brisanje i pregled biblioteka)
- Upravljanje korisnicima i osobljem (registracija, autorizacija i evidencija aktivnosti)
- Upravljanje pristupnim pravima i povezivanje korisnika s bibliotekama

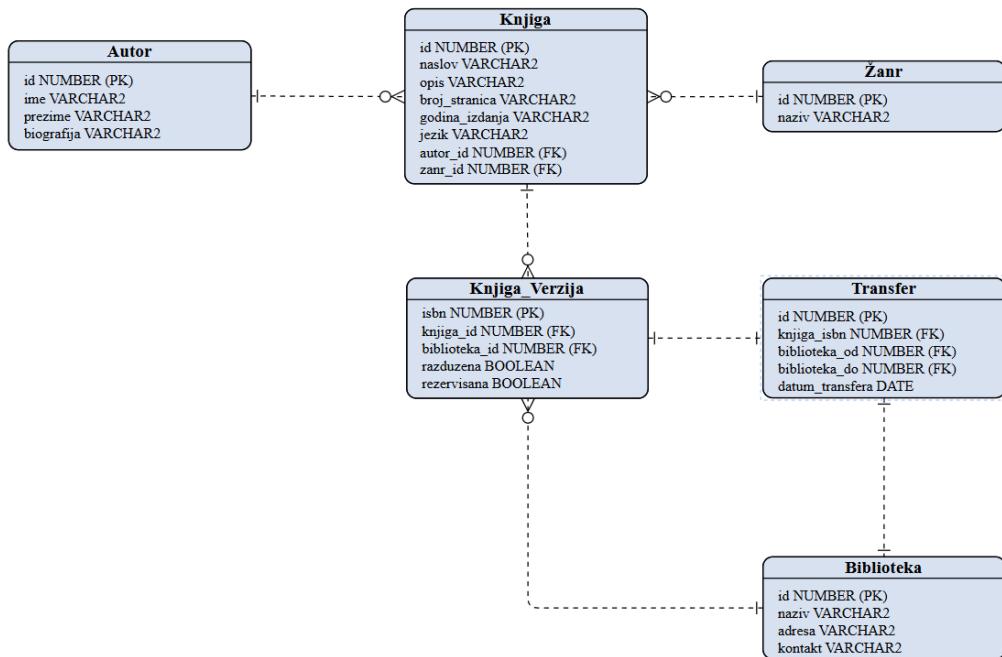


Slika 6.2. ERD servisa za upravljanje bibliotekama i korisnicima

6.3. Servis za upravljanje transferima knjiga

Kako bi se omogućila razmjena knjiga između različitih biblioteka unutar sistema, razvijen je posebni mikroservis za upravljanje transferima knjiga. Ovaj mikroservis pruža sljedeće funkcionalnosti:

- Upravljanje zahtjevima za razmjenu knjiga između biblioteka
- Praćenje statusa razmjene (odobreni, u toku, završeni transferi)
- Evidencija odgovorne osobe i datuma razmjene



Slika 6.3. ERD servisa za transfer knjigama

7. CRUD operacije za svaki mikroservis

U nastavku možemo vidjeti rezultate svih CRUD operacija pozvanih nad klasama *Žanr*, *Korisnik* i *Autor* respektivno za sva tri mikroservisa.

7.1. Servis za upravljanje knjigama, rezervacijama i posudbama

GET - Ispis svih žanrova (uspjeh)

Code	Details
200	<p>Response body</p> <pre>[{ "name": "Roman" }, { "name": "Roman" }, { "name": "Roman" }, { "name": "Roman" }, { "name": "Drama" }, { "name": "Poezija" }]</pre>

GET - Ispis određenog žanra pomoću IDa (uspjeh)

Name	Description
id * required	ID of the genre <input type="text" value="integer(\$int64) (path) 5"/>

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8083/genre/5' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8083/genre/5
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "name": "Drama" }</pre>

GET - Ispis određenog žanra pomoću IDa (neuspjeh)

The screenshot shows a REST API documentation interface. At the top, there is a form for a GET request:

Name	Description
id * required	ID of the genre
integer(\$int64) (path)	100

Below the form are two buttons: "Execute" and "Clear".

Under the "Responses" section, there is a "Curl" code block:

```
curl -X 'GET' \
'http://localhost:8083/genre/100' \
-H 'accept: */*'
```

Then, the "Request URL" is shown:

```
http://localhost:8083/genre/100
```

Under "Server response", there is a table:

Code	Details
400 Undocumented	Error: response status is 400

Under "Response body", the JSON output is:

```
{  
    "error": "validation",  
    "message": "Genre with given ID does not exist!"  
}
```

POST - Kreiranje novog žanra (uspjeh)

The screenshot shows a REST API documentation interface. At the top, there is a "Curl" code block:

```
curl -X 'POST' \
'http://localhost:8083/genre' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{  
    "name": "Drama"  
}'
```

Below it, the "Request URL" is shown:

```
http://localhost:8083/genre
```

Under "Server response", there is a table:

Code	Details
200	

Under "Response body", the JSON output is:

```
{  
    "name": "Drama"  
}
```

Under "Response headers", the output is:

```
connection: keep-alive  
content-type: application/json  
date: Mon, 26 May 2025 18:44:47 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked
```

PUT - Izmjena postojećeg žanra pomoću IDa (uspjeh)

PUT /genre/{id} Update genre

Parameters

Name	Description
id * required integer(\$int64) (path)	2

Request body required

```
{  
    "name": "Fikcija"  
}
```

Code	Description
200	OK Media type /* Controls Accept header. Example Value Schema

```
{  
    "name": "string"  
}
```

PUT - Izmjena postojećeg žanra pomoću IDa (neuspjeh)

Server response

Code	Details
404 <i>Undocumented</i>	Error: response status is 404

Response headers

```
connection: keep-alive  
content-length: 0  
date: Mon, 26 May 2025 18:59:40 GMT  
keep-alive: timeout=60
```

DELETE - Brisanje postojećeg žanra (uspjeh)

DELETE /genre/{id} Delete genre

Parameters

Name	Description
id * required integer(\$int64) (path)	7

Code Details

Code	Details
200	Response body Genre with ID 7 was deleted successfully.

DELETE - Brisanje postojećeg žanra (neuspjeh)

DELETE /genre/{id} Delete genre

Parameters

Name	Description
id * required integer(\$int64) (path)	100

Code Details

Code	Details
400 <i>Undocumented</i>	Error: response status is 400 Response body { "error": "validation", "message": "Genre with given ID does not exist!" }

7.2. Servis za upravljanje bibliotekama i korisnicima

GET - Ispis svih korisnika (uspjeh)

```
[  
  {  
    "firstName": "Marko",  
    "lastName": "Marić",  
    "email": "mmaric1@gmail.com",  
    "phone": "062361598",  
    "roleId": 1,  
    "libraryId": 1  
  },  
  {  
    "firstName": "Marko",  
    "lastName": "Marić",  
    "email": "mmaric1@gmail.com",  
    "phone": "062361598",  
    "roleId": 2,  
    "libraryId": 2  
  },  
  {  
    "firstName": "Marko",  
    "lastName": "Marić",  
    "email": "mmaric1@gmail.com",  
    "phone": "062361598",  
    "roleId": 3,  
    "libraryId": 3  
  },  
]
```

GET - Ispis određenog korisnika putem IDa (uspjeh)

```
{  
  "firstName": "Marko",  
  "lastName": "Marić",  
  "email": "mmaric1@gmail.com",  
  "phone": "062361598",  
  "roleId": 1,  
  "libraryId": 1  
}
```

GET - Ispis određenog korisnika putem IDa (neuspjeh)

```
{  
  "error": "validation",  
  "message": "User with given ID does not exist!"  
}
```

POST - Kreiranje korisnika (uspjeh)

```
{  
    "email": "nejra.adilovic@example.com",  
    "firstName": "Nejra",  
    "lastName": "Adilovic",  
    "libraryId": 1,  
    "password": "sigurna_lozinka123",  
    "phone": "0987654321",  
    "roleId": 2  
}
```

```
{  
    "firstName": "Nejra",  
    "lastName": "Adilovic",  
    "email": "nejra.adilovic@example.com",  
    "phone": "0987654321",  
    "roleId": 2,  
    "libraryId": 1  
}
```

POST - Kreiranje korisnika (neuspjeh)

The screenshot shows a POST request to `http://localhost:8084/user`. The request body is identical to the successful one above, containing valid JSON. However, the response is a `400 Bad Request` with the message `"error": "validation", "message": "Invalid email format!"`, indicating that the validation step failed due to an invalid email address.

```
1 {  
2     "email": "nejraexample.com",  
3     "firstName": "Nejra",  
4     "lastName": "Adilovic",  
5     "libraryId": 1,  
6     "password": "sigurna_lozinka123",  
7     "phone": "0987654321",  
8     "roleId": 2  
9 }
```

```
1 {  
2     "error": "validation",  
3     "message": "Invalid email format!"  
4 }
```

PUT - Ažuriranje korisnika (uspjeh)

```
{  
    "firstName": "Darko",  
    "lastName": "Marić",  
    "email": "mmaric1@gmail.com",  
    "phone": "062361598",  
    "roleId": 1,  
    "libraryId": 1  
}
```

```
{  
    "firstName": "Darko",  
    "lastName": "Marić",  
    "email": "mmaric1@gmail.com",  
    "phone": "062361598",  
    "roleId": 1,  
    "libraryId": 1  
}
```

DELETE - Brisanje korisnika

```
User with ID 1 was deleted successfully.
```

GET - Pronađi biblioteku prema ID-u (uspjeh)

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:8084/library/1
- Method:** GET
- Headers:** (8)
- Body:** (empty)
- Response Status:** 200 OK
- Response Time:** 45 ms
- Response Size:** 226 B
- Content:** A JSON object representing a library:

```
1 {  
2     "name": "Centar",  
3     "address": "Alipašina",  
4     "contact": "061234567"  
5 }
```

GET - Pronađi biblioteku prema ID-u (neuspjeh)

The screenshot shows the Postman interface for a failed API call. The URL is `http://localhost:8084/library/100`. The response status is **400 Bad Request** with a duration of 32 ms and a body size of 220 B. The response body is a JSON object:

```
1 {  
2   "error": "validation",  
3   "message": "Library with the given ID does not exist!"  
4 }
```

7.3. Servis za upravljanje transferima knjiga

GET - Ispis autora ako je zadat ID (uspjeh)

author Author API

GET /author/{id} Find author by ID

Parameters

Name	Description
id * required	ID of the author
integer(\$int64) (path)	1

Code Details

200 Response body

```
{  
    "firstName": "Ivo",  
    "lastName": "Andrić",  
    "biography": "Dobitnik Nobelove nagrade za književnost."  
}
```

Response headers

```
connection: keep-alive  
content-type: application/json  
date: Mon, 31 Mar 2025 21:43:13 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked
```

GET - Ispis autora ako je zadat ID (neuspjeh)

author Author API

GET /author/{id} Find author by ID

Parameters

Name	Description
id * required	ID of the author
integer(\$int64) (path)	7

Code	Details
400 <i>Undocumented</i>	Error: response status is 400 Response body <pre>{ "error": "validation", "message": "Author with given ID does not exist!" }</pre> Response headers <pre>connection: close content-type: application/json date: Mon, 31 Mar 2025 21:47:46 GMT transfer-encoding: chunked</pre>

PUT - Uređivanje autora ako je zadan ID i tijelo zahtjeva (uspjeh)

PUT /author/{id} Update author

Parameters

Name	Description
id <small>* required</small> integer(\$int64) (path)	2

Request body required

```
{
    "firstName": "Meša",
    "lastName": "Selimović",
    "biography": "Pisac"
}
```

Code	Details
200	Response body <pre>{ "firstName": "Meša", "lastName": "Selimović", "biography": "Pisac" }</pre> Response headers <pre>connection: keep-alive content-type: application/json date: Mon, 31 Mar 2025 21:49:39 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

PUT - Uređivanje autora ako je zadan ID i tijelo zahtjeva (neuspjeh)

PUT /author/{id} Update author

Parameters

Name	Description
id * required integer(\$int64) (path)	7

Request body required

```
{
  "firstName": "Meša",
  "lastName": "Selimović",
  "biography": "Pisac"
}
```

Code Details

Code	Details
400 <i>Undocumented</i>	Error: response status is 400 Response body <pre>{ "error": "validation", "message": "Author with given ID does not exist!" }</pre> Response headers <pre>connection: close content-type: application/json date: Mon,31 Mar 2025 21:53:25 GMT transfer-encoding: chunked</pre>

GET - Ispis svih autora

Code	Details
200	Response body <pre>[{ "firstName": "Ivo", "lastName": "Andrić", "biography": "Dobitnik Nobelove nagrade za književnost." }, { "firstName": "Ivo", "lastName": "Andrić", "biography": "Dobitnik Nobelove nagrade za književnost." }, { "firstName": "Meša", "lastName": "Selimović", "biography": "Pisac" }]</pre> Response headers <pre>connection: keep-alive content-type: application/json date: Mon,31 Mar 2025 21:55:15 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

POST - Kreiranje autora (uspjeh)

POST /author Create author

Parameters

No parameters

Request body required

```
{  
    "firstName": "Marko",  
    "lastName": "Markovic",  
    "biography": "Pisac"  
}
```

Code	Details
200	<p>Response body</p> <pre>{ "firstName": "Marko", "lastName": "Markovic", "biography": "Pisac" }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Mon,31 Mar 2025 21:57:15 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

POST - Kreiranje autora (isto ime)

POST /author Create author

Parameters

No parameters

Request body required

```
{  
    "firstName": "Marko",  
    "lastName": "Markovic",  
    "biography": "Pisac"  
}
```

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "error": "validation", "message": "An author with the same name already exists!" }</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Mon,31 Mar 2025 21:59:48 GMT transfer-encoding: chunked</pre>

DELETE - Brisanje autora sa datim IDem (uspjeh)

DELETE	/author/{id} Delete author				
Parameters					
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>id * required integer(\$int64) (path)</td> <td>4</td> </tr> </tbody> </table>		Name	Description	id * required integer(\$int64) (path)	4
Name	Description				
id * required integer(\$int64) (path)	4				

Code	Details
200	<p>Response body</p> <pre>Author with ID 4 was deleted successfully.</pre> <p>Response headers</p> <pre>connection: keep-alive content-length: 42 content-type: text/plain;charset=UTF-8 date: Mon,31 Mar 2025 22:03:16 GMT keep-alive: timeout=60</pre>

DELETE - Brisanje autora sa datim IDem (neuspjeh)

DELETE	/author/{id} Delete author				
Parameters					
<table border="1"> <thead> <tr> <th>Name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>id * required integer(\$int64) (path)</td> <td>7</td> </tr> </tbody> </table>		Name	Description	id * required integer(\$int64) (path)	7
Name	Description				
id * required integer(\$int64) (path)	7				

Code	Details
400 <i>Undocumented</i>	<p>Error: response status is 400</p> <p>Response body</p> <pre>{ "error": "validation", "message": "Author with given ID does not exist!"}</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Mon, 31 Mar 2025 22:01:34 GMT transfer-encoding: chunked</pre>

8. Service Discovery i Load Balancing

U okviru ovog projekta implementiran je Service Discovery mehanizam koristeći Eureka server, kako bi se omogućilo automatsko otkrivanje i balansiranje opterećenja između mikroservisa u sistemu za upravljanje bibliotekama.

Na slici ispod prikazan je Eureka Dashboard, gdje se jasno vidi da su svi registrovani mikroservisi uspješno prepoznati i aktivni. Kao što smo već ranije naveli, u ovom sistemu postoje tri mikroservisa, a to su book-service, library-service i transfer-service. Možemo vidjeti kako je Eureka uspješno implementirana.

The screenshot shows the Spring Eureka Dashboard interface. At the top, it displays "spring Eureka" and "HOME LAST 1000 SINCE STARTUP".

System Status

Environment	test	Current time	2025-05-28T16:15:50 +0000
Data center	default	Uptime	00:02
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
BOOK-SERVICE	n/a (1)	(1)	UP (1) - a63b74b55401:book-service:8082
LIBRARY-SERVICE	n/a (1)	(1)	UP (1) - dc3aef5a52dd:library-service:8084
TRANSFER-SERVICE	n/a (1)	(1)	UP (1) - g78c7ee0475a:transfer-service:8083

General Info

Name	Value
total-avail-memory	100mb

Nakon ovoga, odlučili smo se da load balancing isprobamo na book servisu, gdje smo za te potrebe kreirali dvije instance book-service servisa. Za ostala dva servisa, broj instanci je ostao isti odnosno imaju po jednu instancu. Nakon pokretanja Eureka Dashboarda možemo vidjeti kako su obje instance book-service servisa uspješno aktivne.

The screenshot shows the Spring Eureka dashboard. At the top, it displays "spring Eureka" and "HOME LAST 1000 SINCE STARTUP". Below this, the "System Status" section shows environment details like "Environment: test", "Data center: default", and various uptime metrics. The "DS Replicas" section lists instances registered with Eureka across three application categories: BOOK-SERVICE, LIBRARY-SERVICE, and TRANSFER-SERVICE. The "General Info" section shows memory usage with "Name: total-avail-memorv" and "Value: 168mb".

Application	AMIs	Availability Zones	Status
BOOK-SERVICE	n/a (2)	(2)	UP (2) - book-service-2 , book-service-1
LIBRARY-SERVICE	n/a (1)	(1)	UP (1) - ee7862a7f34e.library-service:8084
TRANSFER-SERVICE	n/a (1)	(1)	UP (1) - c7c94d0b59e1.transfer-service:8083

Za potrebe testiranja napisali smo skriptu koja šalje 100 zahtjeva prema book-service servisu, koristeći Ribbon kao klijentski load balancer u kombinaciji s Eurekom. Cilj je bio izmjeriti koliko je zahtjeva obradila svaka instanca i ukupno vrijeme potrebno za obradu.

```

MINGW64:/c/Users/Zira3/Desktop/Dalila/ETF/Master/2. semestar/NWT/library...
Zira3@tag-PF5D8L04 MINGW64 ~/Desktop/Dalila/ETF/Master/2. semestar/NWT/library...
$ bash ./script.sh http://localhost:8082/book/self-test
[?] Sending 100 requests...
-----
[?] Test completed in 12536 ms
[?] Results:
book-service-1 handled: 50 requests
book-service-2 handled: 50 requests
Total requests: 100
Total time: 12536 ms

```

Rezultati su prikazani na prethodnoj slici. Možemo vidjeti kako su obje instance book-service servisa primile po 50 zahtjeva, a ukupno vrijeme izvršenja je 12536 ms što je izuzetno dobro vrijeme. Ovi rezultati potvrđuju da je load balancing pravilno konfigurisan, jer su zahtjevi ravnomjerno raspoređeni između dvije instance. Također, omogućeno je efikasnije korištenje resursa i smanjenje vremena odziva sistema.

Uspješnom integracijom Eureke kao servisa za otkrivanje i Ribbon-a za balansiranje opterećenja, postignuta je visoka dostupnost i skalabilnost book-service komponente. Testiranjem je potvrđeno da sistem efikasno koristi sve dostupne instance servisa, ravnomjerno raspoređujući opterećenje i time poboljšavajući performanse i otpornost sistema na opterećenja.

9. Sinhrona komunikacija između mikroservisa

U okviru razvijenog sistema, određene funkcionalnosti zahtijevaju komunikaciju između više mikroservisa u realnom vremenu, tj. putem sinhronih API poziva. Iako mikroservisna arhitektura podstiče korištenje asinhronne komunikacije radi smanjenja međuzavisnosti i povećanja skalabilnosti, postoje situacije u kojima je sinhrona komunikacija neophodna. To su, prije svega, situacije u kojima rezultat komunikacije mora biti odmah dostupan kako bi se nastavilo dalje izvršavanje funkcionalnosti. Takve funkcionalnosti uključuju trenutne provjere dostupnosti podataka, validaciju u međukoracima poslovne logike, te potvrdu statusa resursa.

U ovom sistemu implementirano je više sinhronih API poziva koji povezuju mikroservise book-service, library-service i transfer-service. Naprimjer, kada se korisniku prikazuju rezervacije ili posudbe knjiga, book-service u sebi sadrži samo identifikator korisnika (*userId*). Da bi se prikazali svi ostali relevantni podaci o korisniku (ime, prezime, kontakt i slično), book-service se putem sinhronog poziva povezuje s library-service, koji posjeduje detaljne informacije o korisnicima. Ova vrsta komunikacije implementirana je kako za rezervacije, tako i za posudbe, te za detaljan prikaz pojedinačnih rezervacija i posudbi, čime je omogućeno prikazivanje kompletnih informacija u korisničkom interfejsu.

Pored toga, jedan od važnijih segmenata sistema odnosi se na provjeru dostupnosti knjige. Kada korisnik odabere određenu knjigu, potrebno je prikazati sve njene dostupne verzije (*BookVersion*) i stanje tih verzija po bibliotekama. Da bi se to postiglo, book-service prvo komunicira s library-service kako bi dobavio informacije o tome u kojoj se biblioteci nalaze pojedine verzije knjige. Nakon toga, book-service dodatno komunicira sa transfer-service kako bi se provjerilo da li je neka od verzija trenutno u procesu transfera između biblioteka.

Kao rezultat ovih sinhronih poziva, sistem prikuplja podatke koji omogućuju da se za svaku biblioteku prikaže broj primjeraka određene verzije knjige, podijeljen po statusima: dostupni (*availableCount*), rezervisani (*reservedCount*), posuđeni (*checkedOutCount*) i u transferu (*transferCount*). Takođe se navodi i ukupan broj primjeraka (*totalCount*) koji se nalazi u svakoj biblioteci. Na slici ispod prikazan je primjer takvog odgovora koji sadrži ove informacije strukturirane po bibliotekama:

```
1  [
2  {
3      "libraryId": 1,
4      "libraryName": "Centar",
5      "availableCount": 0,
6      "reservedCount": 1,
7      "checkedOutCount": 1,
8      "transferCount": 1,
9      "totalCount": 3
10 },
11 {
12     "libraryId": 2,
13     "libraryName": "Grbavica",
14     "availableCount": 1,
15     "reservedCount": 1,
16     "checkedOutCount": 1,
17     "transferCount": 0,
18     "totalCount": 3
19 },
20 {
21     "libraryId": 3,
22     "libraryName": "Dobrinja",
23     "availableCount": 1,
24     "reservedCount": 1,
25     "checkedOutCount": 1,
26     "transferCount": 0,
27     "totalCount": 3
28 }
29 ]
```

Ovakav pristup omogućava korisnicima da u realnom vremenu dobiju tačne i ažurne informacije o dostupnosti knjiga u različitim bibliotekama, što je ključno za tačnost i korisničko iskustvo sistema. Također, omogućava pouzdano upravljanje zalihamama knjiga i efikasno planiranje eventualnih transfera između biblioteka.

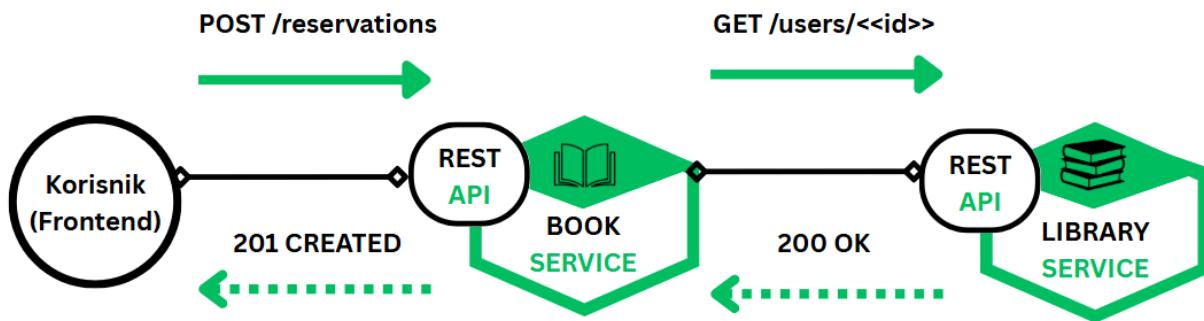
U budućoj fazi razvoja sistema, dio ovih funkcionalnosti će se optimizirati korištenjem asinhronne komunikacije putem poruka i događaja, čime će se dodatno poboljšati skalabilnost i otpornost sistema na greške.

10. Dijagrami toka komunikacije

U nastavku će biti prikazana tri dijagrama toka netrivialne komunikacije. Pod netrivialnom komunikacijom se podrazumijeva ona čiji rezultat zavisi od stanja baza i/ili memorije dva ili više mikroservisa.

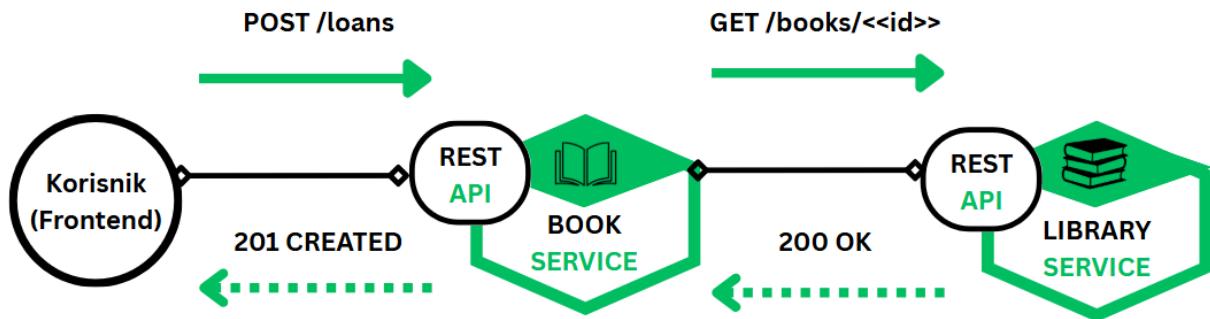
10.1. Kreiranje rezervacije i provjera korisnika

Kada korisnik želi rezervisati knjigu putem korisničkog interfejsa, book-service mora provjeriti identitet korisnika i njegove aktivne rezervacije kako bi omogućio novu. Na dijagramu toka koji se nalazi ispod možemo vidjeti sljedeću komunikaciju. Korisnik (Frontend) šalje zahtjev POST /reservations ka book-service. book-service sinhrono šalje zahtjev GET /users/{id} ka library-service da provjeri da li korisnik postoji i da li ima pravo na novu rezervaciju. Ako library-service vrati status 200 OK, book-service provjerava interni broj trenutnih rezervacija za korisnika. Ako uslovi zadovoljeni, kreira se nova rezervacija u bazi podataka i vraća status 201 CREATED.



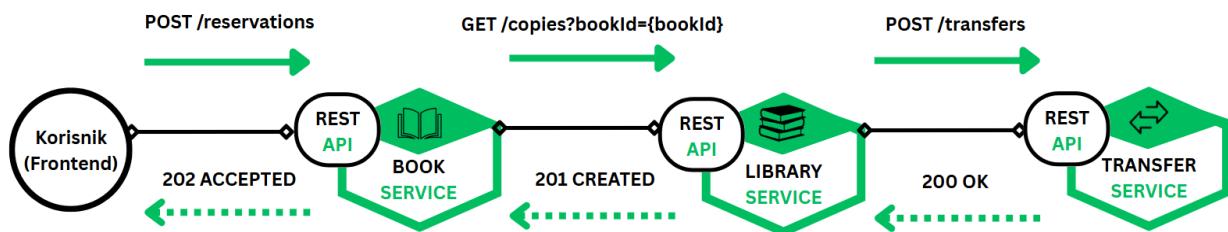
10.2. Posudba knjige uz provjeru dostupnosti i korisničkog statusa

Sljedeći dijagram toka je vezan za posudbu knjige. Korisnik želi posuditi knjigu. Međutim, prije kreiranja posudbe, sistem mora provjeriti da li korisnik ispunjava uslove (nema dugovanja, ima pravo na posudbu) i da li je knjiga dostupna. Frontend šalje POST /loans ka book-service. book-service poziva library-service da provjeri korisnički status (aktivnost, dugovanja). Ako je korisnik validan, book-service provjerava dostupnost primjerka u svojoj bazi. Ako je sve ispravno, posudba se kreira i vraća se 201 CREATED. Ovaj dijagram toka možemo pogledati ispod.



10.3. Iniciranje transfera primjerka knjige između biblioteka

Kada korisnik pokuša rezervisati knjigu u biblioteci u kojoj trenutno nije dostupna, book-service pokreće zahtjev za fizički transfer dostupnog primjerka iz druge biblioteke. Da bi to uradio, potrebno je kontaktirati i library-service i transfer-service. Korisnik (Frontend) šalje POST /reservations ka book-service, navodeći željenu biblioteku. book-service upituje library-service putem GET /copies?bookId={bookId} da pronađe dostupne primjerke te knjige u drugim bibliotekama. Ako pronađe dostupan primjerak u drugoj biblioteci (npr. Biblioteka A), a korisnik je tražio u Biblioteci B, book-service šalje zahtjev POST /transfers ka transfer-service. transfer-service kreira novi zahtjev za transfer između Biblioteke A i Biblioteke B i upisuje ga u svoju bazu. Kada je transfer uspješno kreiran, transfer-service vraća status 201 CREATED. book-service tada upisuje rezervaciju u bazi i vraća korisniku status 202 ACCEPTED, signalizirajući da je rezervacija prihvaćena i da je primjerak u dolasku.



11. Upravljanje sigurnošću mikroservisa

U nastavku ćemo prvo proći kroz neka od mogućih sigurnosnih rješenja u mikroservisnoj arhitekturi, a nakon toga ćemo prikazati uspješno implementirano rješenje.

11.1. Sigurnosna rješenja u mikroservisnoj arhitekturi

Sigurnost u sistemima zasnovanim na mikroservisnoj arhitekturi predstavlja jedan od najvažnijih izazova, budući da se aplikacija više ne sastoji od jedinstvene monolitne strukture, već od niza manjih i samostalnih komponenti koje međusobno komuniciraju putem mreže. Svaki mikroservis potencijalno predstavlja tačku napada, pa je stoga neophodno razmotriti odgovarajuće sigurnosne mehanizme. U okviru istraživanja, analizirana su tri dominantna pristupa za upravljanje sigurnošću u distribuiranim sistemima.

API ključ (API Key) predstavlja najjednostavniji mehanizam zaštite pristupa API-ju, gdje se svakom klijentu dodjeljuje jedinstveni ključ koji se zatim šalje uz svaki zahtjev kao oblik identifikacije. Ključ može biti smješten u URL-u, zaglavljima ili tijelu zahtjeva. Iako je njegova implementacija brza i ne zahtjeva dodatne biblioteke ni složenu infrastrukturu, sigurnosne prednosti su minimalne. API ključevi ne sadrže informacije o korisniku, nisu vremenski ograničeni i teško ih je opozvati bez ručne intervencije. Uz to, ako ključ bude kompromitovan (npr. putem presretanja ili curenja s klijentske strane), napadač može imati dugotrajan pristup sistemu. Zbog toga se API ključevi rijetko koriste u kompleksnijim sistemima koji zahtijevaju preciznu kontrolu pristupa.

OAuth 2.0 je standardni protokol za autorizaciju koji omogućava aplikacijama da korisnicima omoguće pristup ograničenim resursima bez izlaganja njihovih vjerodajnica. Ključna komponenta ovog sistema je Authorization Server, koji izdaje i upravlja pristupnim tokenima nakon što korisnik prođe autentifikaciju. Prednost ovog pristupa je u fleksibilnosti – omogućava grananje permisija, integraciju sa socijalnim loginima i detaljnu kontrolu pristupa. Međutim, njegova složenost predstavlja izazov za manje projekte. Potrebno je implementirati dodatne mehanizme za izdavanje i validaciju tokena, kao i održavanje samog authorization servera, što

može zahtijevati dodatne resurse i pažljivu konfiguraciju. Idealno je rješenje za velike sisteme s višestrukim izvorima identiteta i višerazinskim pristupom.

JWT (JSON Web Token) predstavlja moderno i jednostavno rješenje za autentifikaciju i autorizaciju korisnika u distribuiranim sistemima. JWT je samodostatan token koji u sebi nosi informacije o korisniku – njegov ID, ulogu, dozvole i vrijeme važenja. Nakon što korisnik uspješno izvrši login, server generiše token koji klijent zatim koristi u svim narednim zahtjevima. Prednost JWT-a je u tome što ne zahtijeva upite prema bazi za svaku validaciju – sve informacije se nalaze u samom tokenu i validiraju se pomoću digitalnog potpisa. Osim što olakšava skalabilnost i brzinu sistema, JWT se lako koristi u okruženjima s više servisa jer svaki servis može lokalno validirati token. Ovo rješenje je idealno za mikroservisnu arhitekturu i upravo zbog toga je odabранo u našem projektu.

11.2. Primjena JWT rješenja u našem sistemu

Za implementaciju sigurnosti u okviru mikroservisne arhitekture našeg sistema, odabранo je token-bazirano rješenje koje koristi JWT (JSON Web Token) tehnologiju. JWT omogućava sigurno prenošenje informacija između klijenta i servera bez potrebe za čuvanjem sesija na serveru, čime se sistem čini efikasnijim i skalabilnjim. Implementacija uključuje dodatne komponente kao što su JwtAuthFilter, UserDetailsService, podrška za role-based autorizaciju, kao i refresh token mehanizam koji omogućava produženje sesije bez ponovnog logovanja korisnika.

Autentifikacija se odvija unutar jednog mikroservisa, konkretno u *library-service* servisu, koji obrađuje korisničke zahtjeve za prijavu. Po uspješnom loginu, korisnik dobija dva tokena: access token (koji traje 15 minuta) i refresh token (koji traje 7 dana). Access token se koristi za sve zaštićene zahtjeve, dok se refresh token koristi za generisanje novog access tokena kada prethodni istekne. Ovaj pristup omogućava dobro balansiranu sigurnost i ugodno korisničko iskustvo, jer korisnici ne moraju često ponovo unositi svoje podatke.

U ovom sistemu, API Gateway ne vrši autentifikaciju ni autorizaciju korisnika, već samo prosljeđuje zahtjeve prema odgovarajućim servisima koristeći Eureka service discovery. Ovakva odluka omogućava veću autonomiju svakog mikroservisa, jer svaki servis lokalno obraduje sigurnosne aspekte i donosi odluke o validaciji tokena i dozvoljenim akcijama.

Autorizacija korisnika bazirana je na rolama, koje se definišu pri kreiranju korisnika i čuvaju kao claim unutar JWT tokena (npr. "role": "ADMIN"). Na osnovu tih rola, servisi koriste anotacije kao što je `@PreAuthorize("hasRole('ADMIN')")` kako bi omogućili ili zabranili pristup određenim funkcijama. Spring Security i UserDetailsService osiguravaju dinamičko upravljanje ovim dozvolama.

U arhitektonskom smislu, sistem koristi decentralizovanu autorizaciju, gdje svaki mikroservis samostalno validira token bez dodatnih poziva prema centralnom autorizacionom serveru. Ovaj pristup smanjuje latenciju i eliminiše potrebu za dodatnom infrastrukturom, što je u skladu s jednostavnom i efikasnom arhitekturom projekta. Iako decentralizacija donosi određene izazove, kao što je dupliciranje logike provjere tokena, ona omogućava bolju otpornost sistema i pojednostavljuje razvoj.

Mikroservisi međusobno ne zahtijevaju autorizaciju u trenutnoj fazi razvoja, jer svi vanjski zahtjevi prema sistemu prolaze isključivo kroz API Gateway. Ovim pristupom se onemogućava direktni pristup pojedinačnim mikroservisima izvan sistema, čime se smanjuje sigurnosni rizik izloženosti. Interna komunikacija između servisa odvija se unutar sigurnog okruženja i ne zahtijeva dodatne sigurnosne provjere putem tokena.

Budući da JWT tokeni ne zavise od serverskog stanja (stateless), logout korisnika ne rezultuje trenutnom deaktivacijom tokena. Token ostaje validan do isteka, a logout se realizuje na klijentskoj strani brisanjem tokena iz lokalnog spremišta. Iako je moguće implementirati dodatni mehanizam za invalidaciju tokena, kao što je blacklist preko Redis-a, u trenutnoj fazi razvoja to nije bilo neophodno jer koristimo kratkotrajne access tokene.

Za dalju sigurnost, sistem koristi osvježavanje tokena putem refresh tokena, koji omogućava korisniku da dobije novi access token bez ponovne prijave, sve dok je refresh token važeći. Ovo rješenje pruža bolji balans između sigurnosti i praktičnosti, jer omogućava automatsko produženje sesije bez narušavanja zaštite podataka.

Na kraju, sistem je prilagođen i za pristup s mobilnih uređaja. Zahvaljujući pravilno konfigurisanom CORS-u, svi klijenti (uključujući web i mobilne aplikacije) mogu nesmetano slati zahtjeve prema API-ju. Tokeni se mogu sigurno pohraniti u lokalni storage uređaja i koristiti za autentifikaciju svakog zahtjeva, čime se osigurava univerzalna i sigurna dostupnost sistema na svim platformama.

11.3. Konfiguracija API Gateway-a

U sklopu mikroservisne arhitekture razvijenog sistema za upravljanje bibliotekama, implementiran je API Gateway koristeći Spring Cloud Gateway. Gateway služi kao ulazna tačka za sve zahtjeve upućene ka mikroservisima i omogućava centralizovano upravljanje rutiranjem, sigurnošću i filtriranjem zahtjeva.

The screenshot shows the Spring Eureka dashboard interface. At the top, there's a header with the Spring logo and the word "Eureka". On the right side of the header, it says "HOME LAST 1000 SINCE STARTUP". Below the header, there are three main sections: "System Status", "DS Replicas", and "Instances currently registered with Eureka".

System Status: This section displays various system metrics. It includes fields for Environment (test), Data center (default), Current time (2025-05-29T15:06:23 +0000), Uptime (00:01), Lease expiration enabled (false), Renews threshold (8), and Renews (last min) (0).

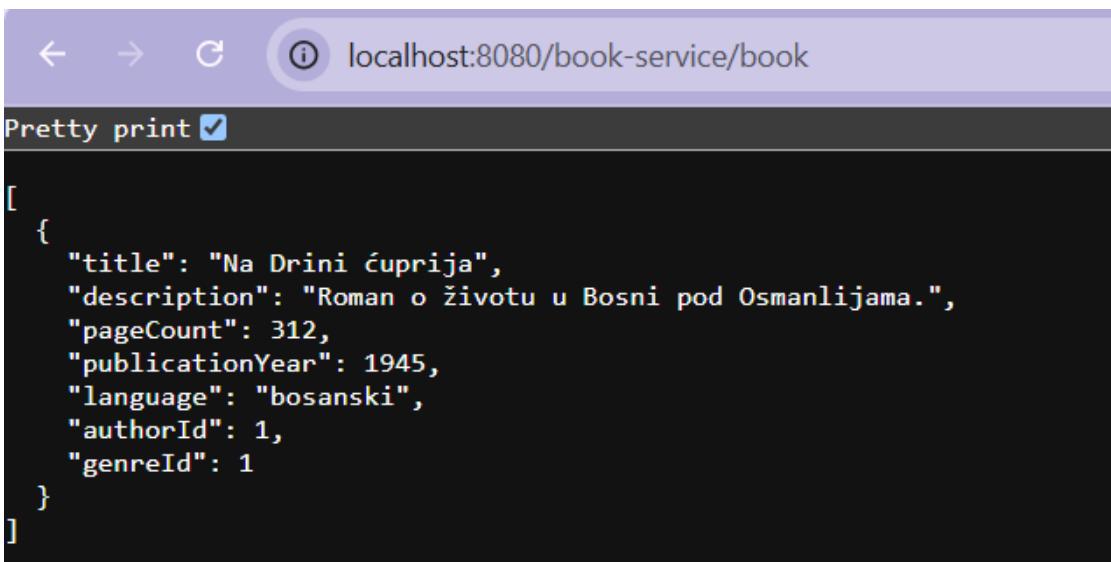
DS Replicas: This section shows a table for "localhost". It has one row with the value "localhost".

Instances currently registered with Eureka: This section displays a table of registered instances across four columns: Application, AMIs, Availability Zones, and Status. The registered instances are:

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 10c70ca63b8b:api-gateway:8080
BOOK-SERVICE	n/a (1)	(1)	UP (1) - 2578048236ea:book-service:8082
LIBRARY-SERVICE	n/a (1)	(1)	UP (1) - 201024ddde309:library-service:8084
TRANSFER-SERVICE	n/a (1)	(1)	UP (1) - 0c07b07522ef:transfer-service:8083

General Info: This section has a table with two columns: Name and Value. It contains one row with the value "localhost".

Na slici iznad prikazan je Eureka dashboard, gdje su prikazane sve trenutno aktivne instance registrovane u sistemu. U ovom slučaju vidimo četiri servisa: book-service, library-service, transfer-service i api-gateway. Prisutnost api-gateway instance potvrđuje da je gateway uspješno pokrenut i registrovan kao servis u okviru Eureka Service Discovery-a, što omogućava dinamičko rutiranje prema drugim servisima bez potrebe za statičkom konfiguracijom njihovih adresa. Na sljedećoj slici prikazan je poziv prema jednoj od ruta sistema putem gateway-a – konkretno prema *book-service/book* ruti, ali kroz API Gateway. Ruta je ispravno proslijedena i vraćen je validan odgovor, što potvrđuje da rutiranje funkcioniše ispravno. Tako, umjesto da se direktno pristupa *book-service* na njegovom portu, poziv se obavlja putem *localhost*.



A screenshot of a web browser window. The address bar shows "localhost:8080/book-service/book". Below the address bar, there is a "Pretty print" checkbox which is checked. The main content area displays a JSON array with one element:

```
[  
  {  
    "title": "Na Drini ćuprija",  
    "description": "Roman o životu u Bosni pod Osmanlijama.",  
    "pageCount": 312,  
    "publicationYear": 1945,  
    "language": "bosanski",  
    "authorId": 1,  
    "genreId": 1  
}]
```

U nastavku imamo prikaz realizacije autorizacije i autentifikacije putem tokena. Na prvoj slici možemo vidjeti kako preko rute login kada se prijavimo u naš sistem izgeneriše se token koji će važiti sljedećih 15 minuta. Osim toga, vidimo da je statusni kod 200 OK što je pozitivno.

Nakon toga, kako bismo provjerili da je sve uredu, prvo smo pozvali rutu GET kojom želimo da provjerimo da li je korisnik koji se prijavljuje autorizovan i možemo vidjeti statusni kod 200 OK i ispis ‘*You are authenticated.*’ . Ovo nam sugerire da je implementacija JWT-a uspješna. Nakon toga, pokušavamo pristupiti ruti admin i njegovom logovanju koristeći izgenerisani token. Također, možemo primijetiti da je uspješan login obzirom da je statusni kod 200 OK i da se ispisuje poruka ‘*Welcome Admin!*’.

POST http://localhost:8084/auth/login

Body (raw JSON)

```

1 {
2   "email": "mmaric1@gmail.com",
3   "password": "lozinka"
4 }

```

200 OK 433 ms 607 B Save Response

Body Cookies Headers (14) Test Results

{ } JSON Preview Visualize

```

1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.
eyJzdWJlOiJtbWFyaWxQdTYlsLmNvbSIsInJvbGUiOiJST0xFX1TRVIiLCJpYXQiOjE3NDg1NzI1ODYsImV4cCI6MTc0ODU3MzQ4Nn0.
xuszbalkjUrzRY90kiNHkzI06ay0ol--M-JnP605Q"
3 }

```

GET http://localhost:8080/library-service/user/secured

Authorization (Bearer Token)

Token

200 OK 388 ms 397 B Save Response

Body Cookies (1) Headers (12) Test Results

Raw Preview Visualize

You are authenticated!

HTTP NWT / admin

GET http://localhost:8080/library-service/user/admin

Authorization (Bearer Token)

Token

200 OK 88 ms 389 B Save Response

Body Cookies (1) Headers (12) Test Results

Raw Preview Visualize

Welcome Admin!

Na prethodnoj slici smo se prijavili kao admin i koristili njegov token tako da je autorizacija bila uspješna. Na slici ispod smo se pokušali prijaviti kao user, odnosno obični korisnik sa tokenom od admina. Kao što možemo vidjeti dobijamo krešku i statusni kod je 500 Server Error, što znači da je sve uspješno implementirano.

The screenshot shows a POSTMAN interface with the following details:

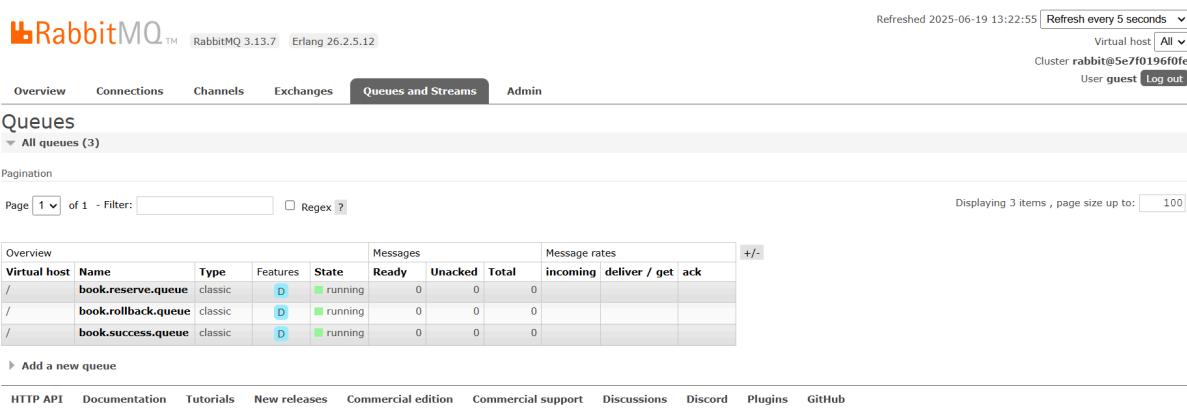
- Method: GET
- URL: <http://localhost:8080/library-service/user/admin>
- Authorization tab is selected, showing "Bearer Token" as the type.
- Body tab shows a JSON response:

```
1 {  
2   "error": "internal",  
3   "message": "An unexpected error occurred"  
4 }
```
- Status bar indicates: 500 Internal Server Error, 26 ms, 544 B.

Spring Cloud Gateway je uspješno konfigurisana komponenta sistema koja omogućava jedinstvenu i sigurnu komunikaciju između klijenta i mikroservisa. Zahvaljujući integraciji s Eurekom, omogućeno je dinamičko rutiranje bez potrebe za ručnim unosom adresa servisa. Također, implementacijom autentifikacije putem tokena osigurana je kontrola pristupa osjetljivim resursima, što je ključni segment sigurnosne arhitekture sistema.

12. Asinhrona komunikacija i saga obrazac uz RabbitMQ

Kako bi se omogućila robusnija i otpornija obrada podataka između mikroservisa, u sklopu sistema implementirana je asinhrona komunikacija koristeći RabbitMQ kao posrednika za razmjenu poruka. Ova funkcionalnost je realizovana prema principima event-based pristupa i Saga choreographije, gdje svaki mikroservis samostalno reaguje na događaje koje primi, bez centralnog orkestratora. Cilj ove implementacije bio je da se postigne koordinacija između dvije lokalne transakcije koje se izvršavaju u različitim mikroservisima, tako da se obezbijedi konzistentnost sistema čak i u slučaju greške.



The screenshot shows the RabbitMQ Management Console interface. At the top, there are navigation links: Overview, Connections, Channels, Exchanges, Queues and Streams (which is the active tab), and Admin. On the right side, there are status indicators: Refreshed 2025-06-19 13:22:55, Refresh every 5 seconds, Virtual host All, Cluster rabbit@5e7f0196f0fe, User guest, and Log out. Below the tabs, it says Queues and All queues (3). There is a pagination section with Page 1 of 1, Filter, Regex, and a note that 3 items are displayed with a page size of 100. A table follows, showing the overview of three queues:

Virtual host	Name	Type	Features	State	Messages			Message rates		
					Ready	Unacked	Total	Incoming	deliver / get	ack
/	book.reserve.queue	classic	D	running	0	0	0			
/	book.rollback.queue	classic	D	running	0	0	0			
/	book.success.queue	classic	D	running	0	0	0			

At the bottom, there is a link to Add a new queue and a footer with links to HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Plugins, and GitHub.

U konkretnom slučaju, rezervacija knjige se ne sastoji samo od jedne baze – već je neophodno ažurirati stanje u mikroservisu zaduženom za knjige (book-service) i u mikroservisu zaduženom za biblioteku (library-service). Da bi se to postiglo, korištene su tri queue-a: book.reserve.queue za osnovnu rezervaciju, book.rollback.queue za poništavanje (inverznu akciju) u slučaju greške, i book.success.queue za potvrdu da su sve transakcije uspješno završene. Svaki mikroservis reaguje na određenu poruku iz queue-a, izvršava lokalnu transakciju, i ako dođe do greške, šalje rollback signal kako bi se prethodno izvršene operacije poništile.

Na slici iznad prikazan je RabbitMQ panel sa pregledom aktivnih queue-ova koji su korišteni za ovu implementaciju. Sve tri queue-e su trenutno u stanju running, što pokazuje da je infrastruktura za asinhronu komunikaciju uspješno konfigurisana i funkcioniše u realnom vremenu. Ova arhitektura omogućava sistemu da postigne veću pouzdanost i otpornost na greške, što je posebno važno u distribuiranim okruženjima s više međuzavisnih servisa.

13. Dockerizacija sistema i orkestracija putem Docker Compose-a

U cilju olakšavanja razvoja, testiranja i pokretanja kompletнog sistema, izvršena je potpuna dockerizacija svih mikroservisa, baza podataka i dodatnih komponenti kao što su RabbitMQ i Eureka server. Svaki mikroservis ima vlastiti Dockerfile, pažljivo konfigurisan da koristi laganu JDK sliku i minimize upotrebu memorije. Posebna pažnja je posvećena optimizaciji kontejnera, gdje su korištene slim varijante baza (npr. postgres:16) i Alpine image-i gdje je bilo moguće, čime se dodatno smanjilo opterećenje sistema.

The screenshot shows the Docker Compose interface for managing a multi-container application. At the top, there are summary statistics: Container CPU usage (11.53% / 2200%), Container memory usage (3.69GB / 15GB), and a link to 'Show charts'. Below this is a search bar and a filter option 'Only show running containers'. The main table lists ten containers under the 'library-management-system' service, each with its name, container ID, image, port(s), CPU usage, last start time, and actions (stop, restart, logs). The table includes columns for Name, Container ID, Image, Port(s), CPU (%), Last started, and Actions. The 'Actions' column contains icons for stopping, restarting, and viewing logs. The bottom of the interface shows system status (RAM 6.50 GB, CPU 2.19%, Disk: 26.44 GB used / limit 1006.85 GB), a terminal link, and a 'New version available' notification.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
rabbitmq	5e7f0196f0fe	rabbitmq:3-management	15672:15672	0.3%	31 minutes ago	[stop] [restart] [logs]
postgres-db	8165e746b489	postgres:16	5432:5432	0%	31 minutes ago	[stop] [restart] [logs]
config-server	7e254aad1402	library-management-system-config-server	8888:8888	3.29%	31 minutes ago	[stop] [restart] [logs]
library-service	61bba1b0c3a8	library-management-system-library-service	8084:8084	0.23%	31 minutes ago	[stop] [restart] [logs]
eureka-server	b9310075bb75	library-management-system-eureka-server	8761:8761	4.55%	31 minutes ago	[stop] [restart] [logs]
system-events	70f30bfbb24c	library-management-system-system-events	9090:9090	0%	31 minutes ago	[stop] [restart] [logs]
api-gateway	f11de01ca23	library-management-system-api-gateway	8080:8080	0.2%	31 minutes ago	[stop] [restart] [logs]
book-service	3ded0ac39982	library-management-system-book-service	8082:8082	0.21%	31 minutes ago	[stop] [restart] [logs]
transfer-service	2454a066ed29	library-management-system-transfer-service	8083:8083	0.22%	31 minutes ago	[stop] [restart] [logs]

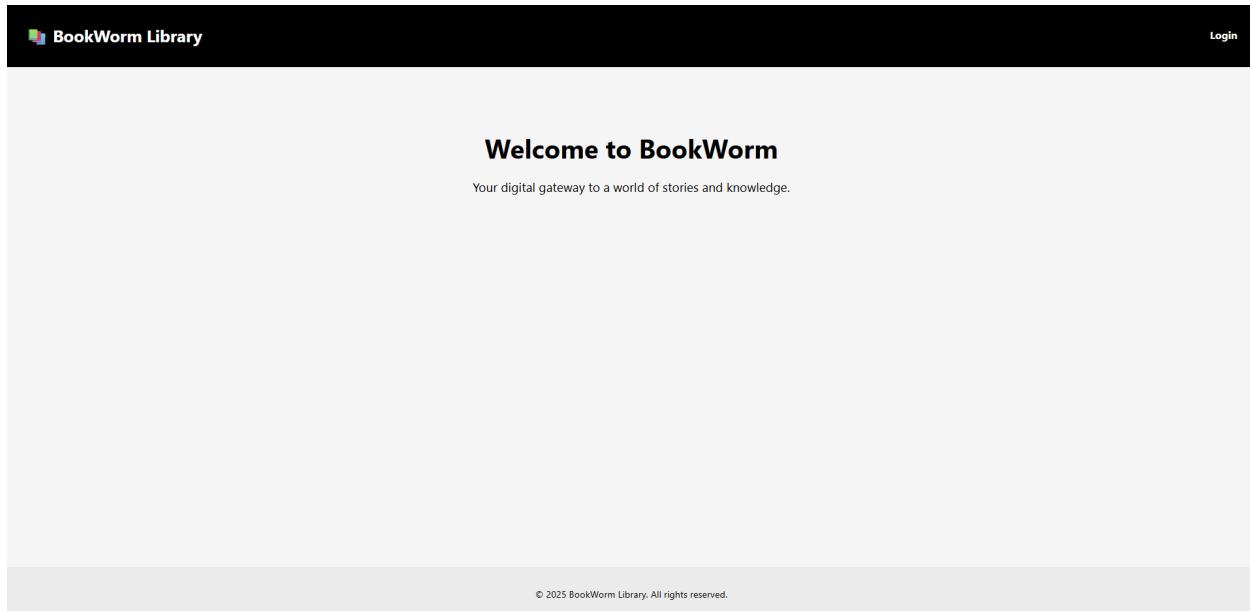
Za potrebe orkestracije i automatizovanog pokretanja svih servisa, kreiran je docker-compose.yml fajl, koji definiše redoslijed pokretanja, mrežne zavisnosti i portove preko kojih komponente komuniciraju. Na slici iznad prikazan je trenutni prikaz aktivnih kontejnera pokrenutih putem Docker Compose-a. Vidljivo je da svi servisi — uključujući book-service, library-service, transfer-service, eureka-server, config-server, api-gateway, system-events, postgres-db, i rabbitmq — funkcionišu stabilno i sinkronizirano unutar istog okruženja.

Ovaj pristup omogućava razvojnom timu da lokalno pokrene kompletan sistem jednim komandnim pozivom, bez potrebe za ručnom konfiguracijom pojedinačnih komponenti. Također, u potpunosti je omogućena interna komunikacija između servisa, zahvaljujući Docker mreži koja povezuje sve kontejnere. Ova infrastruktura olakšava i budući deployment na cloud platformama, jer se svi dijelovi sistema nalaze unutar standardizovanih i lako prenosivih kontejnera.

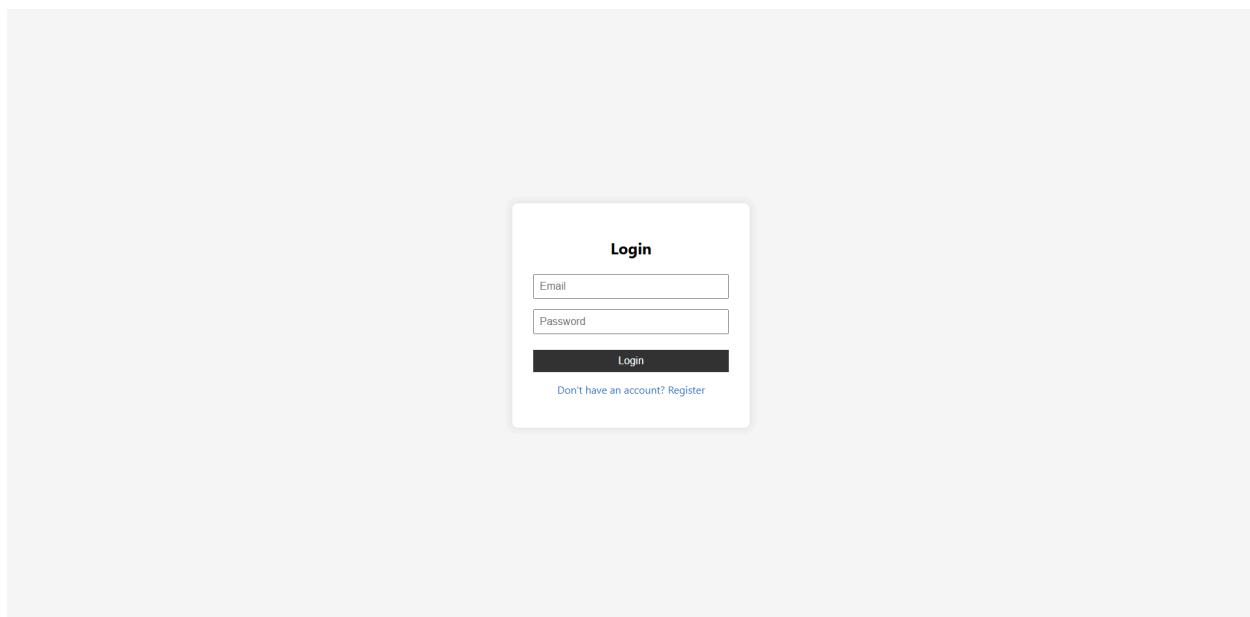
14. Rad aplikacije – frontend prikaz

14.1. Početna stranica i login

Na sljedećoj slici možemo vidjeti prikaz početne stranice.



Nakon što kliknemo na login otvorit će nam se sljedeća stranica.



14.2. Korisnik - Izgled stranice

Kada se prijavimo kao USER možemo vidjeti spisak svih knjiga u biblioteci u kojoj smo prijavljeni.

Book Title	Author	Genre	Availability
Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	Unavailable
The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	Available
Animal Farm	George Orwell	Dystopian	Available
Pride and Prejudice	Jane Austen	Romance	Available
The Shining	Stephen King	Horror	Available

Ukoliko filtriramo po određenom žanru imamo sljedeći ispis.

Book Title	Author	Genre	Availability
Animal Farm	George Orwell	Dystopian	Available

Ukoliko pretražujemo knjige kojih nema u biblioteci.

Explore Books

ann Dystopian

No books found.

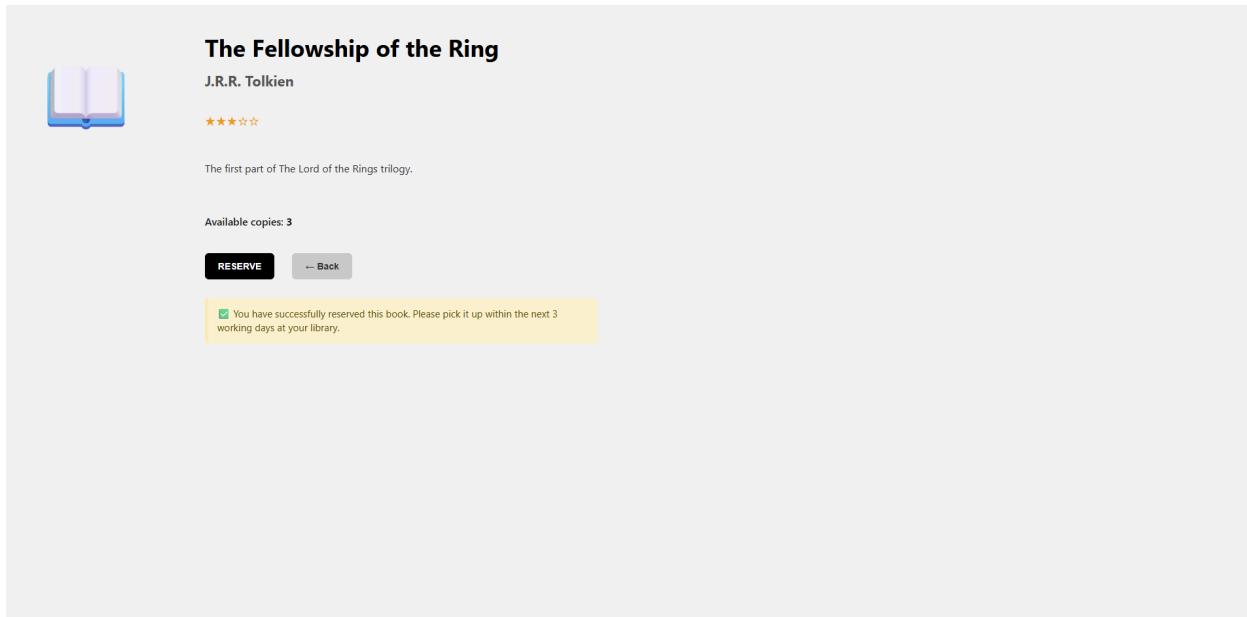
Log Out

Prikaz detalja određene knjige na koju kliknemo je na sljedećoj stranici.

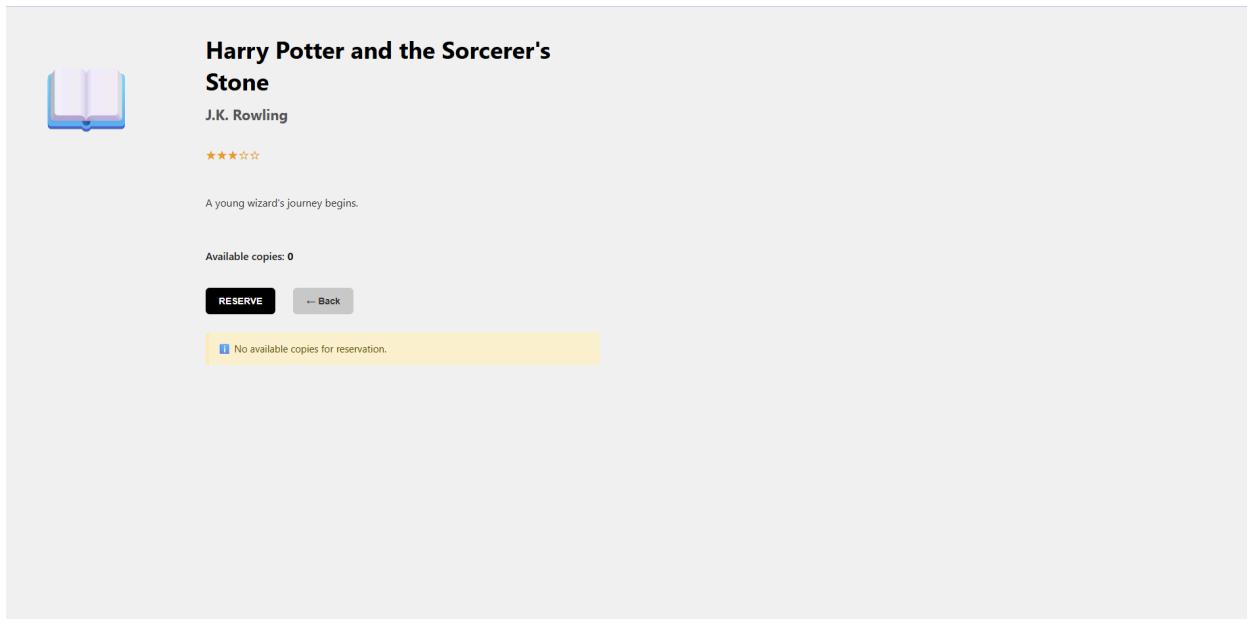
 **The Fellowship of the Ring**
J.R.R. Tolkien

The first part of The Lord of the Rings trilogy.
Available copies: 4
RESERVE ← Back

Nakon uspješne rezervacije knjige dobijemo sljedeću poruku.

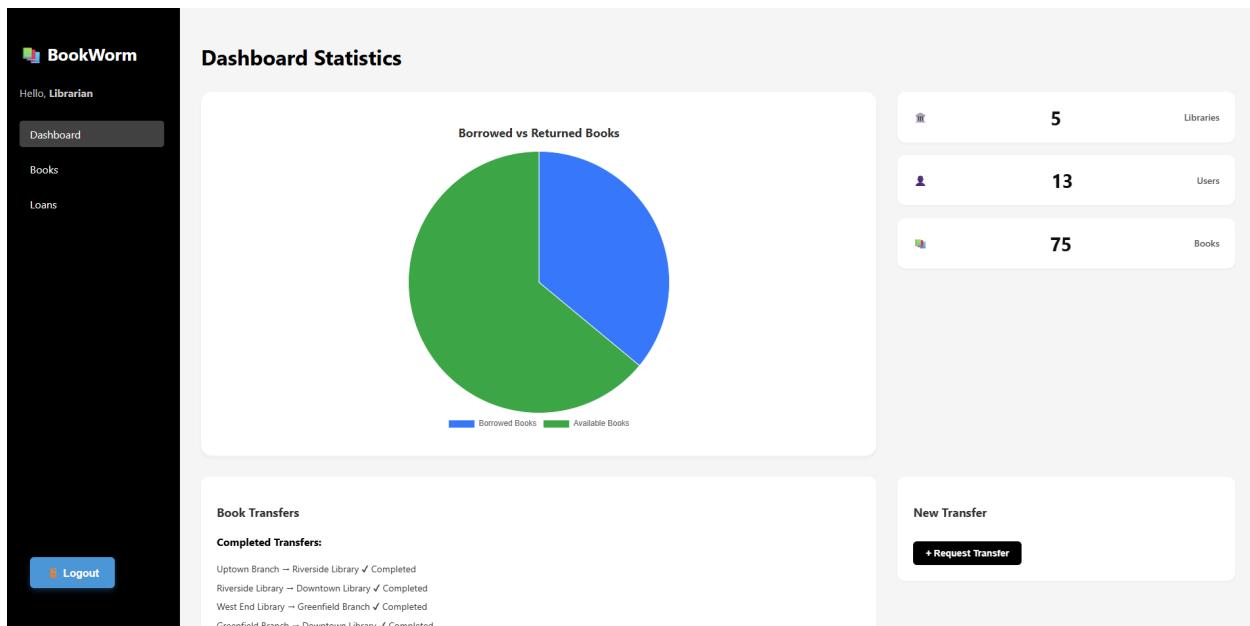


Ukoliko pokušavamo da rezervišemo knjigu koje nema u biblioteci vidjet ćemo obavještenje.

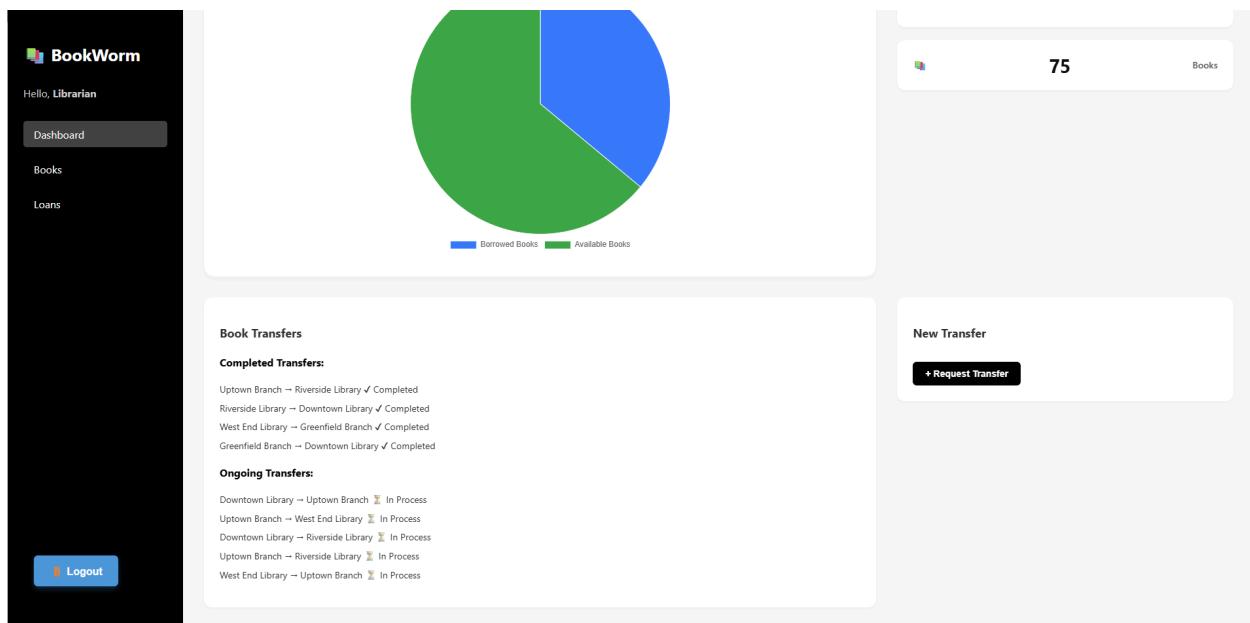


14.3. Bibliotekar - Izgled stranice

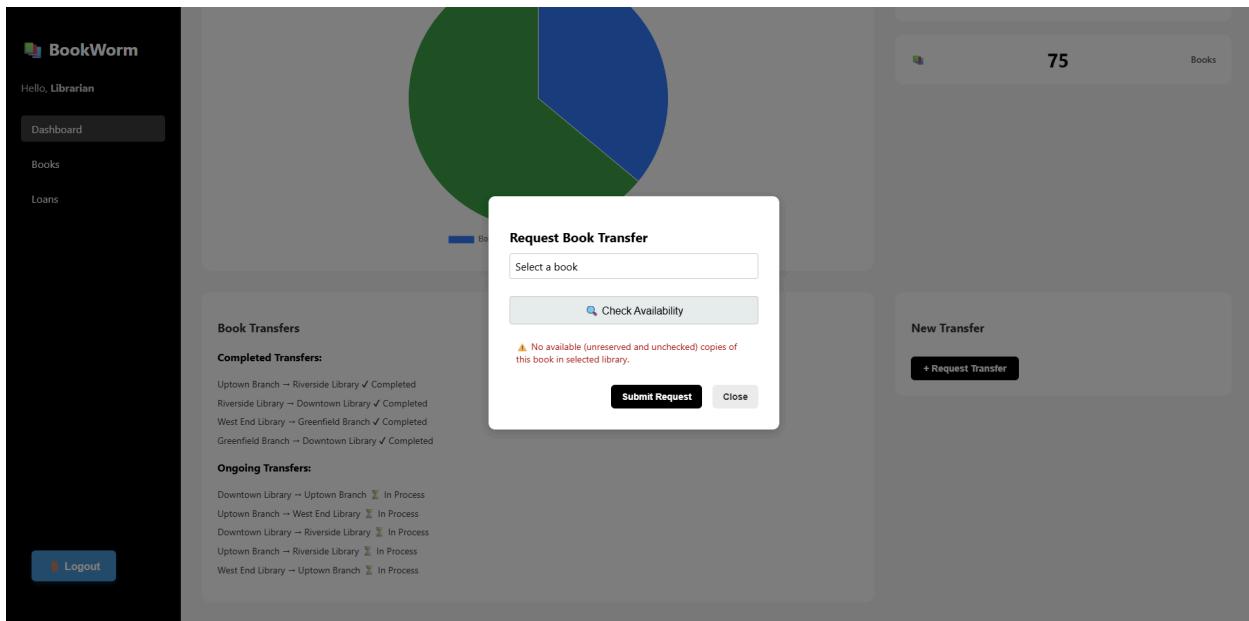
Dashboard izgled stranice kod bibliotekara kao i statistika vezana za njegovu biblioteku.



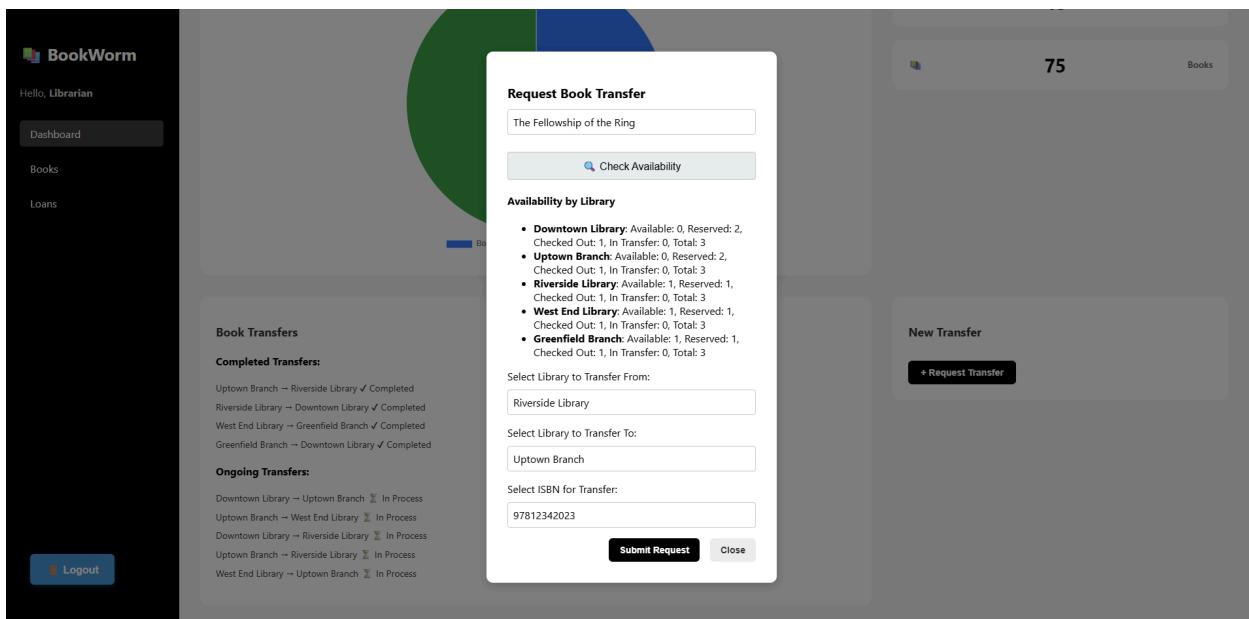
Statistika vezana za transfere između biblioteka.



Kada želimo pokrenuti transfer između biblioteka i ukoliko ne izaberemo biti jednu knjigu izaći će nam poruka o greški.



Kada odaberemo sve potrebne opcije i pritisnemo *Submit Request* kreirat će se transfer.



Prikaz svih knjiga u biblioteci u kojoj je prijavljen bibliotekar.

The screenshot shows the 'Book Management' page. On the left is a sidebar with 'Hello, Librarian' and navigation links for 'Dashboard', 'Books' (which is selected), and 'Loans'. A 'Logout' button is at the bottom. The main area has a header 'Book Management' and a '+ Add Book' button. Below is a table with columns: ISBN, Title, Author, Genre, Available, and Action. The table lists several books like 'Harry Potter and the Sorcerer's Stone', 'The Fellowship of the Ring', and 'Animal Farm' with their respective details and actions.

ISBN	Title	Author	Genre	Available	Action
97812342000	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342001	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342015	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342016	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342030	Animal Farm	George Orwell	Dystopian	No	
97812342031	Animal Farm	George Orwell	Dystopian	No	
97812342045	Pride and Prejudice	Jane Austen	Romance	No	
97812342046	Pride and Prejudice	Jane Austen	Romance	No	
97812342047	Pride and Prejudice	Jane Austen	Romance	Yes	
97812342060	The Shining	Stephen King	Horror	No	
97812342061	The Shining	Stephen King	Horror	No	
97812342062	The Shining	Stephen King	Horror	No	
97812342032	Animal Farm	George Orwell	Dystopian	No	
97812342002	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	

Prikaz prozora za dodavanje knjiga.

The screenshot shows the 'Book Management' page with an 'Add Book' modal open. The modal has fields for 'ISBN' (empty) and 'Select user' (empty). It includes 'Add' and 'Cancel' buttons. The background table is the same as the previous screenshot, listing books and their details.

ISBN	Title	Author	Genre	Available	Action
97812342000	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342001	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342015	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342016	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342030	Animal Farm	George Orwell	Dystopian	No	
97812342031	Animal Farm	George Orwell	Dystopian	No	
97812342045	Pride and Prejudice	Jane Austen	Romance	No	
97812342046	Pride and Prejudice	Jane Austen	Romance	No	
97812342047	Pride and Prejudice	Jane Austen	Romance	Yes	
97812342060	The Shining	Stephen King	Horror	No	
97812342061	The Shining	Stephen King	Horror	No	
97812342062	The Shining	Stephen King	Horror	No	
97812342032	Animal Farm	George Orwell	Dystopian	No	
97812342002	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	

Prikaz greške u slučaju da bibliotekar želi da posudi knjigu koje nema u biblioteci.

The screenshot shows the Book Management interface. On the left, there's a sidebar with 'Hello, Librarian' and links for 'Dashboard', 'Books' (which is selected), and 'Loans'. A 'Logout' button is at the bottom. The main area has a title 'Book Management' and a table with columns: ISBN, Title, Author, Genre, Available, and Action. An alert box in the center says 'localhost:3000 says This book is currently not available for loan.' with an 'OK' button. The table data includes books like Harry Potter and the Sorcerer's Stone, The Fellowship of the Ring, Animal Farm, etc., with various authors and genres.

Prikaz prozora za posudbu knjiga knjige koje ima u biblioteci.

The screenshot shows the Book Management interface with a loan dialog box overlaid. The dialog is titled 'Loan Book: Pride and Prejudice' and contains fields for 'Select user' (with a dropdown menu), 'dd/mm/yyyy' for the loan date, and another 'dd/mm/yyyy' for the return date. It has 'Loan' and 'Cancel' buttons. The background table is the same as the previous screenshot, listing books and their details. The 'Books' link in the sidebar is highlighted.

Prikaz knjiga koje su posuđene i čiji je rok za vraćanje idalje uredu.

The screenshot shows the BookWorm application interface. On the left is a dark sidebar with a logo, 'Hello, Librarian', and navigation links: 'Dashboard', 'Books', and 'Loans' (which is highlighted). On the right is a table titled 'Loans' with two tabs: 'Borrowed Books' (selected) and 'Overdue Borrowers'. The table has columns: ID, ISBN, Title, User, Loan Date, Due Date, and Action. The data shows 14 entries, all from user 'Emma Johnson', all with 'Loan Date' as '2025-03-10' and 'Due Date' as '2025-03-20'. The 'Action' column contains small blue icons. At the bottom right of the table is a small text 'Emma Johnson'.

ID	ISBN	Title	User	Loan Date	Due Date	Action
1	97812342000	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
2	97812342003	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
3	97812342006	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
4	97812342009	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
5	97812342012	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
6	97812342015	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
7	97812342018	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
8	97812342021	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
9	97812342024	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
11	97812342030	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	
12	97812342033	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	
13	97812342036	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	
14	97812342030	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	

Prikaz stranice za vraćanje knjige koja je posuđena.

The screenshot shows the BookWorm application interface, similar to the previous one but with a modal dialog. The dialog is titled 'Confirm Return' and asks 'Are you sure the user has returned the book?'. It has two buttons: 'Yes, Return' (highlighted in red) and 'Cancel'. The background table of loans is partially visible behind the dialog.

ID	ISBN	Title	User	Loan Date	Due Date	Action
1	97812342000	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
2	97812342003	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
3	97812342006	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
4	97812342009	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
5	97812342012	Harry Potter and the Sorcerer's Stone	Emma Johnson	2025-03-10	2025-03-20	
6	97812342015	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
7	97812342018	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
8	97812342021	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
9	97812342024	The Fellowship of the Ring	Emma Johnson	2025-03-10	2025-03-20	
11	97812342030	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	
12	97812342033	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	
13	97812342036	Animal Farm	Emma Johnson	2025-03-10	2025-03-20	

Prikaz knjiga koje su posuđene i nisu vraćene u roku.

The screenshot shows the BookWorm application interface. On the left is a dark sidebar with the BookWorm logo, a 'Hello, Librarian' message, and navigation links for Dashboard, Books, and Loans. The Loans link is highlighted with a blue background. At the bottom of the sidebar is a 'Logout' button. The main content area has a light gray header with the title 'Loans'. Below the header are two tabs: 'Borrowed Books' and 'Overdue Borrowers', with 'Overdue Borrowers' being the active tab and highlighted in black. The main body is a table with the following columns: ID, User, Title, Due Date, and Days Late. The data in the table is as follows:

ID	User	Title	Due Date	Days Late
1	Emma Johnson	Harry Potter and the Sorcerer's Stone	2025-03-20	91
2	Emma Johnson	Harry Potter and the Sorcerer's Stone	2025-03-20	91
3	Emma Johnson	Harry Potter and the Sorcerer's Stone	2025-03-20	91
4	Emma Johnson	Harry Potter and the Sorcerer's Stone	2025-03-20	91
5	Emma Johnson	Harry Potter and the Sorcerer's Stone	2025-03-20	91
6	Emma Johnson	The Fellowship of the Ring	2025-03-20	91
7	Emma Johnson	The Fellowship of the Ring	2025-03-20	91
8	Emma Johnson	The Fellowship of the Ring	2025-03-20	91
9	Emma Johnson	The Fellowship of the Ring	2025-03-20	91
11	Emma Johnson	Animal Farm	2025-03-20	91
12	Emma Johnson	Animal Farm	2025-03-20	91
13	Emma Johnson	Animal Farm	2025-03-20	91
14	Emma Johnson	Animal Farm	2025-03-20	91
15	Emma Johnson	Animal Farm	2025-03-20	91

14.4. Admin - Izgled stranice

Admin ima dashboard isti kao i bibliotekar zajedno sa svim funkcionalnostima. Osim toga ima i opciju za pregled svih knjiga u biblioteci u kojoj je on zadužen. Na sljedećem prozoru možemo vidjeti opciju za uređivanje knjiga.

The screenshot shows the Book Management page of the BookWorm application. On the left, there's a sidebar with 'Hello, Admin' and navigation links for 'Dashboard' and 'Books'. A 'Logout' button is at the bottom. The main area has a title 'Book Management' and a table of books. The table columns are ISBN, Title, Author, Genre, Available, and Action. An 'Add Book' button is in the top right. A modal window titled 'Edit Book' is open over the table, showing fields for ISBN (97812342010), Title (Harry Potter and the Sorcerer's Stone), and Author (J.K. Rowling). It also shows 'Fantasy' under 'Genre' and 'No' under 'Available'. At the bottom of the modal are 'Update' and 'Cancel' buttons.

ISBN	Title	Author	Genre	Available	Action
97812342009	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342010	Harry Potter and the Sorcerer's Stone	J.K. Rowling	Fantasy	No	
97812342024	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342025	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	No	
97812342026	The Fellowship of the Ring	J.R.R. Tolkien	Fantasy	Yes	
97812342039	Animal Farm		Dystopian	No	
97812342040	Animal Farm		Dystopian	No	
97812342041	Animal Farm		Dystopian	Yes	
97812342054	Pride and Prejudice	Jane Austen	Romance	No	
97812342055	Pride and Prejudice	Jane Austen	Romance	No	
97812342056	Pride and Prejudice	Jane Austen	Romance	Yes	
97812342069	The Shining	Stephen King	Horror	No	
97812342070	The Shining	Stephen King	Horror	No	
97812342071	The Shining	Stephen King	Horror	Yes	

14.5. Superadmin - Izgled stranice

Prikaz prozora za prikaz svih knjiga u cijelom sistemu.

The screenshot shows the 'Book Management' page of the BookWorm application. On the left is a dark sidebar with navigation links: 'Dashboard', 'Books' (which is selected), 'Libraries', and 'Users'. A 'Logout' button is at the bottom of the sidebar. The main area has a title 'Book Management' and a '+ Add Book' button. Below is a table with columns: ID, Title, Description, Language, Year, Pages, Author, Genre, and Action. The table contains five rows of book data:

ID	Title	Description	Language	Year	Pages	Author	Genre	Action
1	Harry Potter and the Sorcerer's Stone	A young wizard's journey begins.	English	1997	320	J.K. Rowling	Fantasy	
2	The Fellowship of the Ring	The first part of The Lord of the Rings trilogy.	English	1954	423	J.R.R. Tolkien	Fantasy	
3	Animal Farm	A satirical allegory about totalitarianism.	English	1945	112	George Orwell	Dystopian	
4	Pride and Prejudice	A classic romance novel.	English	1813	279	Jane Austen	Romance	
5	The Shining	A psychological horror novel.	English	1977	447	Stephen King	Horror	

Prikaz prozora za uređivanje knjiga.

The screenshot shows the 'Book Management' page with an 'Edit Book' dialog box overlaid on the list of books. The dialog box has fields for Title, Description, Language, Year, Pages, Author, and Genre. The 'Title' field contains 'Harry Potter and the Sorcerer's Stone', 'Description' contains 'A young wizard's journey begins.', 'Language' is 'English', 'Year' is '1997', 'Pages' is '320', 'Author' is 'J.K. Rowling', and 'Genre' is 'Fantasy'. At the bottom of the dialog are 'Update' and 'Cancel' buttons. The background table remains the same as in the previous screenshot.

Prikaz prozora za brisanje određene knjige.

The screenshot shows the Book Management section of the BookWorm application. On the left, a sidebar menu includes options like Dashboard, Books (which is selected), Libraries, and Users. A central table lists five books with columns for ID, Title, Description, Language, Year, Pages, Author, Genre, and Action (edit and delete icons). A modal dialog box titled "Confirm Deletion" is overlaid on the table, asking "Are you sure you want to delete this book?" with "Yes, Delete" and "Cancel" buttons.

ID	Title	Description	Language	Year	Pages	Author	Genre	Action
1	Harry Potter and the Sorcerer's Stone	A young wizard's journey begins.	English	1997	320	J.K. Rowling	Fantasy	
2	The Fellowship of the Ring	The first part of The Lord of the Rings trilogy.	English	1954	423	J.R.R. Tolkien	Fantasy	
3	Animal Farm	A satirical allegory about totalitarianism.	English	1945	112	George Orwell	Dystopian	
4	Pride and Prejudice	A classic novel by Jane Austen.	English	1813	279	Jane Austen	Romance	
5	The Shining	A psychological horror novel by Stephen King.	English	1977	447	Stephen King	Horror	

Prikaz svih biblioteka u sistemu.

The screenshot shows the Library Management section of the BookWorm application. The sidebar menu is identical to the Book Management one. The central table lists five libraries with columns for ID, Name, Contact, Location, and Action (edit and delete icons). The libraries listed are Downtown Library, Uptown Branch, Riverside Library, West End Library, and Greenfield Branch.

ID	Name	Contact	Location	Action
1	Downtown Library	555-1010	5th Avenue	
2	Uptown Branch	555-2020	Main Street 42	
3	Riverside Library	555-3030	River Road 11	
4	West End Library	555-4040	Sunset Blvd 88	
5	Greenfield Branch	555-5050	Green St 19	

Prikaz uređivanja opcija neke biblioteke.

The screenshot shows the BookWorm Library Management interface. On the left is a dark sidebar with navigation links: Dashboard, Books, Libraries (which is selected and highlighted in grey), and Users. A 'Logout' button is at the bottom of the sidebar. The main content area has a title 'Library Management' and a table with columns: ID, Name, Contact, Location, and Action. There are five rows of library data. A modal window titled 'Edit Library' is open over the table, containing fields for Name (Downtown Library), Contact (555-1010), and Location (5th Avenue). Below the modal are 'Update' and 'Cancel' buttons. In the top right corner of the main area, there is a '+ Add Library' button.

ID	Name	Contact	Location	Action
1	Downtown Library	555-1010	5th Avenue	
2	Uptown Branch	555-2020	Main Street 42	
3	Riverside Library		River Road 11	
4	West End Library		Sunset Blvd 88	
5	Greenfield Branch	555-1010	Green St 19	

Prikaz prozora za brisanje određene biblioteke.

The screenshot shows the BookWorm Library Management interface. The sidebar and main content area are identical to the previous screenshot. A modal window titled 'Confirm Deletion' is open over the table. It contains the text 'Are you sure you want to delete this branch?' and two buttons: 'Yes, Delete' (in a black box) and 'Cancel'. The rest of the table and interface elements are visible in the background.

ID	Name	Contact	Location	Action
1	Downtown Library	555-1010	5th Avenue	
2	Uptown Branch	555-2020	Main Street 42	
3	Riverside Library	555-3030	River Road 11	
4	West End Library		Sunset Blvd 88	
5	Greenfield Branch		Green St 19	

Prikaz svih korisnika (sa svih rolama) u sistemu.

ID	First Name	Last Name	Email	Phone	Role	Library	Action
1	Emma	Johnson	emma.johnson@gmail.com	555-1111	Downtown Library	USER	
2	Liam	Williams	liam.williams@gmail.com	555-1112	Uptown Branch	USER	
3	Olivia	Brown	olivia.brown@gmail.com	555-1113	Riverside Library	USER	
4	Noah	Jones	noah.jones@gmail.com	555-1114	West End Library	USER	
5	Ava	Garcia	ava.garcia@gmail.com	555-1115	Greenfield Branch	USER	
6	William	Martinez	will.martinez@gmail.com	555-1116	Downtown Library	USER	
7	Sophia	Lopez	sophia.lopez@gmail.com	555-1117	Uptown Branch	USER	
8	James	Gonzalez	james.gonzalez@gmail.com	555-1118	Riverside Library	USER	
9	Lucas	Clark	librarian1@gmail.com	555-2221	Downtown Library	LIBRARIAN	
10	Mia	Lewis	librarian2@gmail.com	555-2222	Greenfield Branch	LIBRARIAN	
11	Benjamin	Hall	admin1@gmail.com	555-3331	West End Library	ADMIN	
12	Isabella	Allen	admin2@gmail.com	555-3332	Riverside Library	ADMIN	
13	Charlotte	King	superadmin@gmail.com	555-9999		SUPERADMIN	

Prikaz dodavanja novog korisnika u sistem.

ID	First Name	Last Name	Email	Phone	Role	Library	Action
1	Emma	Johnson	emma.johnson@gmail.com	555-1111	Downtown Library	USER	
2	Liam	Williams			Uptown Branch	USER	
3	Olivia	Brown			Riverside Library	USER	
4	Noah	Jones			West End Library	USER	
5	Ava	Garcia			Greenfield Branch	USER	
6	William	Martinez			Downtown Library	USER	
7	Sophia	Lopez			Uptown Branch	USER	
8	James	Gonzalez			Riverside Library	USER	
9	Lucas	Clark	librarian1@gmail.com	555-2221	Downtown Library	LIBRARIAN	
10	Mia	Lewis	librarian2@gmail.com	555-2222	Greenfield Branch	LIBRARIAN	
11	Benjamin	Hall	admin1@gmail.com	555-3331	West End Library	ADMIN	
12	Isabella	Allen	admin2@gmail.com	555-3332	Riverside Library	ADMIN	
13	Charlotte	King	superadmin@gmail.com	555-9999		SUPERADMIN	

Prikaz prozora za uređivanje određenog korisnika.

The screenshot shows the BookWorm application's User Management page. On the left is a dark sidebar with navigation links: Dashboard, Books, Libraries, and Users (which is highlighted). At the bottom of the sidebar is a Logout button. The main content area has a title "User Management" and a table of user data. A modal window titled "Edit User" is open over the table, containing fields for First Name (Emma), Last Name (Johnson), Email (emma.johnson@gmail.com), Phone (555-1111), Role (USER), Library (Downtown Library), and Action (edit and delete icons). Below the modal is an "Update" button. The table lists 13 users with columns for ID, First Name, Last Name, Email, Phone, Role, Library, and Action. The users are: 1. Emma Johnson, 2. Liam Williams, 3. Olivia Brown, 4. Noah Jones, 5. Ava Garcia, 6. William Martinez, 7. Sophia Lopez, 8. James Gonzalez, 9. Lucas Clark, 10. Mia Lewis, 11. Benjamin Hall, 12. Isabella Allen, and 13. Charlotte King.

ID	First Name	Last Name	Email	Phone	Role	Library	Action
1	Emma	Johnson	emma.johnson@gmail.com	555-1111	USER	Downtown Library	
2	Liam	Williams			USER	Uptown Branch	
3	Olivia	Brown			USER	Riverside Library	
4	Noah	Jones			USER	West End Library	
5	Ava	Garcia			USER	Greenfield Branch	
6	William	Martinez			USER	Downtown Library	
7	Sophia	Lopez			USER	Uptown Branch	
8	James	Gonzalez			USER	Riverside Library	
9	Lucas	Clark			LIBRARIAN	Downtown Library	
10	Mia	Lewis	librarian1@gmail.com	555-2222	LIBRARIAN	Greenfield Branch	
11	Benjamin	Hall	admin1@gmail.com	555-3331	ADMIN	West End Library	
12	Isabella	Allen	admin2@gmail.com	555-3332	ADMIN	Riverside Library	
13	Charlotte	King	superadmin@gmail.com	555-9999	SUPERADMIN		

Prikaz prozora za brisanje određenog korisnika.

The screenshot shows the BookWorm application's User Management page. The sidebar and table structure are identical to the previous screenshot. A modal window titled "Confirm Deletion" is centered over the table, asking "Are you sure you want to delete this user?". It contains three buttons: "Yes, Delete", "Delete", and "Cancel". The table data is the same as in the previous screenshot, listing 13 users.

ID	First Name	Last Name	Email	Phone	Role	Library	Action
1	Emma	Johnson	emma.johnson@gmail.com	555-1111	USER	Downtown Library	
2	Liam	Williams	liam.williams@gmail.com	555-1112	USER	Uptown Branch	
3	Olivia	Brown	olivia.brown@gmail.com	555-1113	USER	Riverside Library	
4	Noah	Jones			USER	West End Library	
5	Ava	Garcia			USER	Greenfield Branch	
6	William	Martinez			USER	Downtown Library	
7	Sophia	Lopez			USER	Uptown Branch	
8	James	Gonzalez	james.gonzalez@gmail.com	555-1118	USER	Riverside Library	
9	Lucas	Clark	librarian1@gmail.com	555-2221	LIBRARIAN	Downtown Library	
10	Mia	Lewis	librarian2@gmail.com	555-2222	LIBRARIAN	Greenfield Branch	
11	Benjamin	Hall	admin1@gmail.com	555-3331	ADMIN	West End Library	
12	Isabella	Allen	admin2@gmail.com	555-3332	ADMIN	Riverside Library	
13	Charlotte	King	superadmin@gmail.com	555-9999	SUPERADMIN		

15. Zaključak

Kroz ovaj projekat uspješno je implementiran kompletan distribuirani sistem za upravljanje bibliotekama korištenjem mikroservisne arhitekture. Svaki mikroservis ima jasno definisanu odgovornost, dok su međusobna komunikacija, sigurnost i razmjena podataka riješeni upotrebom modernih pristupa poput Spring Cloud Eureka, Spring Cloud Gateway, JWT autentifikacije, kao i RabbitMQ-a za asinhronu komunikaciju. Time je postignuta modularnost, skalabilnost i otpornost sistema, što predstavlja važnu osnovu za stabilan i dugoročno održiv softverski proizvod.

Razvijen je API Gateway koji služi kao jedinstvena ulazna tačka za sve korisničke zahtjeve, uz pravilno konfigurisano rutiranje i povezivanje putem service discovery mehanizma. Autentifikacija je realizovana decentralizovano koristeći JWT tokene sa podrškom za role-based pristup, dok je asinhrona komunikacija implementirana korištenjem saga obrasca i RabbitMQ queue-ova kako bi se obezbijedila konzistentnost u slučaju grešaka. Sistem je u potpunosti dockerizovan, sa Dockerfile i docker-compose.yml konfiguracijama koje omogućavaju jednostavno pokretanje i skaliranje cijelog okruženja.

Na kraju, kroz frontend aplikaciju demonstrirane su funkcionalnosti sistema, uključujući proces rezervacije, posudbe, validacije, transfera i prikaz grešaka, čime je potvrđena funkcionalna ispravnost svih komponenti. Implementirani sistem ne samo da obezbjeđuje sigurnost i efikasnost u radu s podacima, već i pruža temelj za buduće proširenje – kako u funkcionalnom, tako i u infrastrukturnom smislu.