

*Projektni zadatak iz nastavnog predmeta CS310 Skripting jezici*

Online biblioteka

Mentor: Student:

Nikola Dimitrijević Dušan Stojiljković 5835

*U Nišu, 20.08.2025*

Sadržaj

[1. Uvod 3](#_Toc206551777)

[1.1. Opis teme projekta 3](#_Toc206551778)

[1.2. Cilj projekta i funkcionalnosti koje će biti implementirane 3](#_Toc206551779)

[1.3. Kratak pregled tehnologija i alata koji će biti korišćeni 4](#_Toc206551780)

[2. Planiranje i dizajn sistema 5](#_Toc206551781)

[2.1. Funkcionalni zahtevi 5](#_Toc206551782)

[2.2. Nefunkcionalni zahtevi 6](#_Toc206551783)

[2.3. Use-case dijagrami 7](#_Toc206551784)

[3. Razvoj korisničkog interfejsa 8](#_Toc206551785)

[3.1. Opis tehnologija za stilizaciju 8](#_Toc206551786)

[3.2. Pregled ključnih strana 9](#_Toc206551787)

[3.2.1. Početna strana 9](#_Toc206551788)

[3.2.2. Login i registracija 9](#_Toc206551789)

[3.2.3. Početna strana korisnika 10](#_Toc206551790)

[3.2.4. Profil 11](#_Toc206551791)

[3.2.5. Porudžbine 12](#_Toc206551792)

[3.2.6. Admin panel za upravljanje korisnicima 13](#_Toc206551793)

[3.2.7. Admin panel za upravljanje knjigama 13](#_Toc206551794)

[4. Implementacija funkcionalnosti 14](#_Toc206551795)

[4.1. Korisničke funkcionalnosti 15](#_Toc206551796)

[4.2. Administratorske funkcionalnosti 17](#_Toc206551797)

[5. Backend 19](#_Toc206551798)

[5.1. Struktura projekta 19](#_Toc206551799)

[5.2. Konfiguracija (.env) 19](#_Toc206551800)

[5.3. Modeli 20](#_Toc206551801)

[5.3.1. User 20](#_Toc206551802)

[5.3.2. Book 20](#_Toc206551803)

[5.3.3. Order 20](#_Toc206551804)

[5.3.4. Activity 20](#_Toc206551805)

[5.4. Middleware 21](#_Toc206551806)

[5.5. Paginacija, pretraga i sortiranje 21](#_Toc206551807)

[5.6. Rute i kontroleri 21](#_Toc206551808)

[5.6.1. Auth 21](#_Toc206551809)

[5.6.2. Users (admin osim update-self) 21](#_Toc206551810)

[5.6.3. Books 21](#_Toc206551811)

[5.6.4. Orders 22](#_Toc206551812)

[5.7. Bezbednost 22](#_Toc206551813)

[5.8. Standardni oblici grešaka 22](#_Toc206551814)

[6. Zaključak 23](#_Toc206551815)

[6.1. Pregled postignutih rezultata 23](#_Toc206551816)

[6.2. Izazovi i prepreke tokom rada na projektu 23](#_Toc206551817)

[6.3. Mogućnosti za buduća unapređenja 24](#_Toc206551818)

[7. Prilozi 24](#_Toc206551819)

[8. Literatura 24](#_Toc206551820)

# Uvod

## Opis teme projekta

Ovaj projekat predstavlja web aplikaciju “Biblioteka” razvijenu u React Vite okruženju. Cilj je omogućiti tri nivoa korisnika:**guest** (pregled knjiga), **user** (iznajmljivanje, lični profil, favoriti) i **admin** (CRUD nad korisnicima i knjigama). Aplikacija koristi Tailwind CSS za stilizaciju i node.js sa mongodb kao REST API za skladištenje podataka o korisnicima, knjigama i porudžbinama. ​

## Cilj projekta i funkcionalnosti koje će biti implementirane

Cilj projekta jeste izrada responzivne web‑aplikacije koja omogućava upravljanje elektronskom evidencijom knjiga i korisnika u maloj biblioteci, pri čemu su podržane tri uloge: guest, user i admin. Aplikacija je realizovana kao frontend u React Vite okruženju uz korišćenje Tailwind CSS za stilizaciju, dok je backend rađen u node.js sa mongodb bazom podataka u kojoj će se pamtiti sve potrebne informacije za funkcionalnost aplikacije.

Implementirane funkcionalnosti obuhvataju:

* **Guest:**

mogućnost pregleda liste knjiga i detalja svake knjige, bez potrebe za autentifikacijom (komponente HomePage.jsx i BookDetailPage.jsx) .

* **User:**

Registracija i prijava naloga (RegisterPage.jsx, LoginPage.jsx) sa klijentskom validacijom i čuvanjem session‑a u AuthContext .

Pregled i izmena ličnih podataka, promena lozinke (ProfilePage.jsx) ​ProfilePage.

Iznajmljivanje knjiga, vraćanje knjiga i praćenje istorije porudžbina (BookCard.jsx, OrdersPage.jsx) .

Obeležavanje omiljenih knjiga (favorites) uz toggle funkciju (AuthContext.jsx) ​AuthContext.

* **Admin:**

Posmatranje posebnog admin panela (AdminDashboard.jsx), upravljanje korisnicima (AdminUsersPage.jsx) i knjigama (AdminBooksPage.jsx) s potpunim CRUD operacijama.

Prilikom brisanja korisnika automatsko vraćanje i brisanje svih povezanih porudžbina te oslobađanje iznajmljenih knjiga.

## Kratak pregled tehnologija i alata koji će biti korišćeni

**React 18 + Vite** kao JavaScript biblioteka za izgradnju korisničkog interfejsa i brzi development server/bundler. Aplikacija je inicijalizovana preko Vite alata, što omogućava trenutno osvežavanje (HMR) i brzu kompilaciju ​App-a.

**Tailwind CSS** za util‑class pristup stilizaciji, responsivni dizajn i lakše upravljanje stanjima komponenti (hover, focus). Svi JSX fajlovi koriste Tailwind klase (npr. bg-gray-100, rounded-2xl) kako bi se ubrzao razvoj i smanjio broj prilagođenih CSS fajlova ​HomePage.

**React Router DOM** za deklarativno upravljanje rutama, uključujući zaštićene rute (ProtectedRoute.jsx) i linkovanje između stranica (NavLink, useNavigate) ​Navbar.

**Axios** (paket axios) za HTTP pozive prema REST API‑ju, enkapsulisan u apiService.js sa podešenom baznom URL adresom (env varijabla ili http://localhost:3000) ​apiService.

**UUID** (uuid paket) za generisanje jedinstvenih ID‑eva prilikom kreiranja korisnika i porudžbina direktno u klijentskom kodu (AdminUsersPage.jsx, BookCard.jsx) .

**React Context API** za globalno skladištenje session‑a korisnika (komponenta AuthContext.jsx) uz metode login, register, logout i toggleFavorite ​AuthContext.

**Lucide React** ikone (Star, StarOutline) za vizuelno označavanje favorita i stilizovanih dugmadi u karticama knjiga (BookCard.jsx) ​BookCard.

**Git & npm** za verzionisanje koda i upravljanje zavisnostima; ključne skripte u package.json obuhvataju serve:api za pokretanje json‑servera i fill:covers za batch update korica knjiga.

**TypeScript** na frontendu (TSX/TS fajlovi: ProfilePage.tsx, BookCard.tsx, AuthContext.tsx, apiService.ts), tipovi u types.ts.

**Custom hooks & kontekst:** useAuth (sesija, favorites, profil), ProtectedRoute (zaštita ruta).

**Node.js + Express** za backend (controllers/\*, routes/\*, middleware/\*).

**MongoDB + Mongoose** (modeli: User, Book, Order, Activity; util: parsePagination).

**JWT autentikacija** (authMiddleware) i role zaštita (roleMiddleware), token u Authorization: Bearer ….

**bcryptjs** za heš lozinki (i pri registraciji i pri admin-kreiranju korisnika).

**CORS i dotenv** za konfiguraciju okruženja (.env).

**Aktivnosti/Audit log** preko Activity modela (npr. BOOK\_CREATED, ORDER\_RETURNED, …).

**Snapshot knjige** u porudžbini (Order.bookSnapshot) za stabilnu istoriju i prikaz nakon brisanja knjige.

# Planiranje i dizajn sistema

## Funkcionalni zahtevi

1. **Pregled kataloga knjiga (Guest)**

* Lista svih knjiga s paginacijom po 8 stavki (HomePage.jsx).
* Detaljan prikaz pojedinačne knjige sa slikom, autorom, godinom, opisom (BookDetailPage.jsx) ​.

1. **Autentikacija i upravljanje nalogom (User)**

* Registracija novog korisnika sa validacijom imena, email‑a i lozinke (RegisterPage.jsx) ​.
* Prijava postojećeg korisnika sa greškama za neispravne kredencijale (LoginPage.jsx) ​.
* Izmena ličnih podataka i promena lozinke nakon potvrde stare lozinke (ProfilePage.jsx) ​.
* Obeležavanje i uklanjanje favorita (toggleFavorite u AuthContext.jsx) .
* Iznajmljivanje i vraćanje knjiga (User)
* Evidencija porudžbina (resurs /orders): kreiranje prilikom iznajmljivanja, ažuriranje returnedAt pri vraćanju (OrdersPage.jsx).

1. **Administratorski interfejs (Admin)**

* Dashboard: navigacija na stranice za upravljanje korisnicima i knjigama (AdminDashboard.jsx) ​.
* Upravljanje korisnicima: CRUD nad korisnicima; pri brisanju korisnika vraćanje i brisanje svih porudžbina, oslobađanje knjiga i čišćenje favorite (AdminUsersPage.jsx).
* Upravljanje knjigama: CRUD nad knjigama s generisanjem novog ID-ja kao max+1, validacijom naslova/autor/godine i čišćenjem porudžbina te favorite prilikom brisanja (AdminBooksPage.jsx).

## Nefunkcionalni zahtevi

1. **Performanse**

* Prvo učitavanje (HomePage) ≤ **2s** na lokalnom okruženju (Chrome, hladan cache), uključujući GET /api/books?page=1&limit=12.
* Svi sledeći navigacioni prelazi ≤ **500ms** (SPA).
* Paginacija i pretraga su **server-side**; odgovori sadrže samo traženu stranicu (items, total, page, pageSize).
* MongoDB indeksi za najčešće upite: Book(title, author, year), Book(rentedBy), Order(userId, createdAt), Order(bookId).

2. **Responsivnost**

* UI je u potpunosti prilagodljiv (mobilni ≥ 360px do desktop ≥ 1440px) korišćenjem Tailwind util-klasa i grid/flex layouta.
* Kritične komponente (Navbar, kartice knjiga, tabele admina) imaju breakpoint-specifične klase.

3. **Sigurnost**

* **JWT Bearer** autentikacija; token se čuva u localStorage, interceptor ga dodaje u Authorization header.
* **Role-based** autorizacija (user/admin) na backendu (middleware) i **ProtectedRoute** na frontendu.
* Lozinke se čuvaju **bcrypt** heširane; admin kreiranje korisnika takođe hešuje.
* Validacija i sanitizacija ulaza (kontroleri + Mongoose runValidators).
* CORS ograničen na VITE\_API\_URL; .env tajne varijable kroz dotenv.

4. **Pouzdanost podataka**

* Trajni storage u **MongoDB**; porudžbine čuvaju **bookSnapshot** (naslov/autor/godina/slika) radi stabilne istorije čak i nakon brisanja knjige.
* Pri brisanju knjige: zatvaranje aktivnih porudžbina (returnedAt), postavljanje status="book\_deleted", uklanjanje bookId, snapshot ostaje.
* Aktivnosti (audit) se beleže u Activity modelu.

5. **Odrzavanje i kvalitet koda**

* Modularna arhitektura: kontroleri, modeli, middleware, utili; na frontu stranice/komponente/hook-ovi (useAuth).
* Dosledan API format: liste vraćaju { data: { items, total, page, pageSize } }.

6. **Skalabilnost**

* Odvojeni frontend i backend servisi; horizontalno skaliranje backenda moguće (stateless, JWT).
* Paginacija/filtriranje/sortiranje na serveru; dodati indekse po potrebi.
* Mogućnost kasnijeg uvođenja cache-a (npr. Redis) za frekventne upite (GET /books).

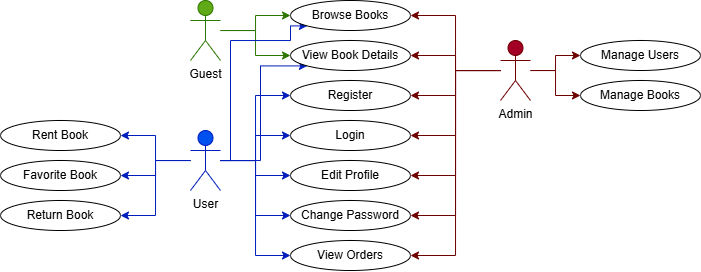
7. **Upotrebljivost i pristupačnost**

* Jasne poruke o greškama (401/403/404/409/500) i validacija formi na klijentu.
* Kontrast i stanja elemenata (hover/focus/disabled) kroz Tailwind.
* Alt tekstovi za slike omota; fallback slike kada nema covera.

8. **Posmatranje i logovanje**

* Backend loguje ključne događaje (kreiranje/vraćanje/brisanje) kroz Activity; server logovi za greške (stack trace).
* HTTP kodovi su semantički tačni radi lakše dijagnostike.

## Use-case dijagrami



**Guest (zeleno)**

* Može da pregleda listu knjiga
* Može da pogleda detalje o knjizi)

**User (plavo, nasleđuje Guest)**

Sve što Guest radi, plus:

* Registracija naloga
* Prijava
* Izmena ličnih podataka
* Promena lozinke
* Označava knjige kao omiljene
* Iznajmljuje knjige
* Vraća iznajmljene knjige
* Pregleda sopstvene porudžbine

**Admin (crveno, nezavisni akter)**

Nakon prijave u Admin panel može da:

* Dodaje/izmenjuje/briše korisnike
* Dodaje/izmenjuje/briše knjige

# Razvoj korisničkog interfejsa

## Opis tehnologija za stilizaciju

Za izradu i održavanje konzistentnog, responzivnog i brzog korisničkog interfejsa odabrana je **Tailwind CSS**, umesto klasičnog pisanja CSS‑a ili biblioteka poput Styled Components.

* **Utility‑first pristup**

Tailwind omogućava direktno pisanje stilova unutar JSX‑a kroz preddefinisane klase (npr. bg-gray-100, rounded-2xl, px-4 py-2) koje kombinuju sve što je potrebno za izgled komponenti .

* **Brza iteracija i JIT kompilacija**

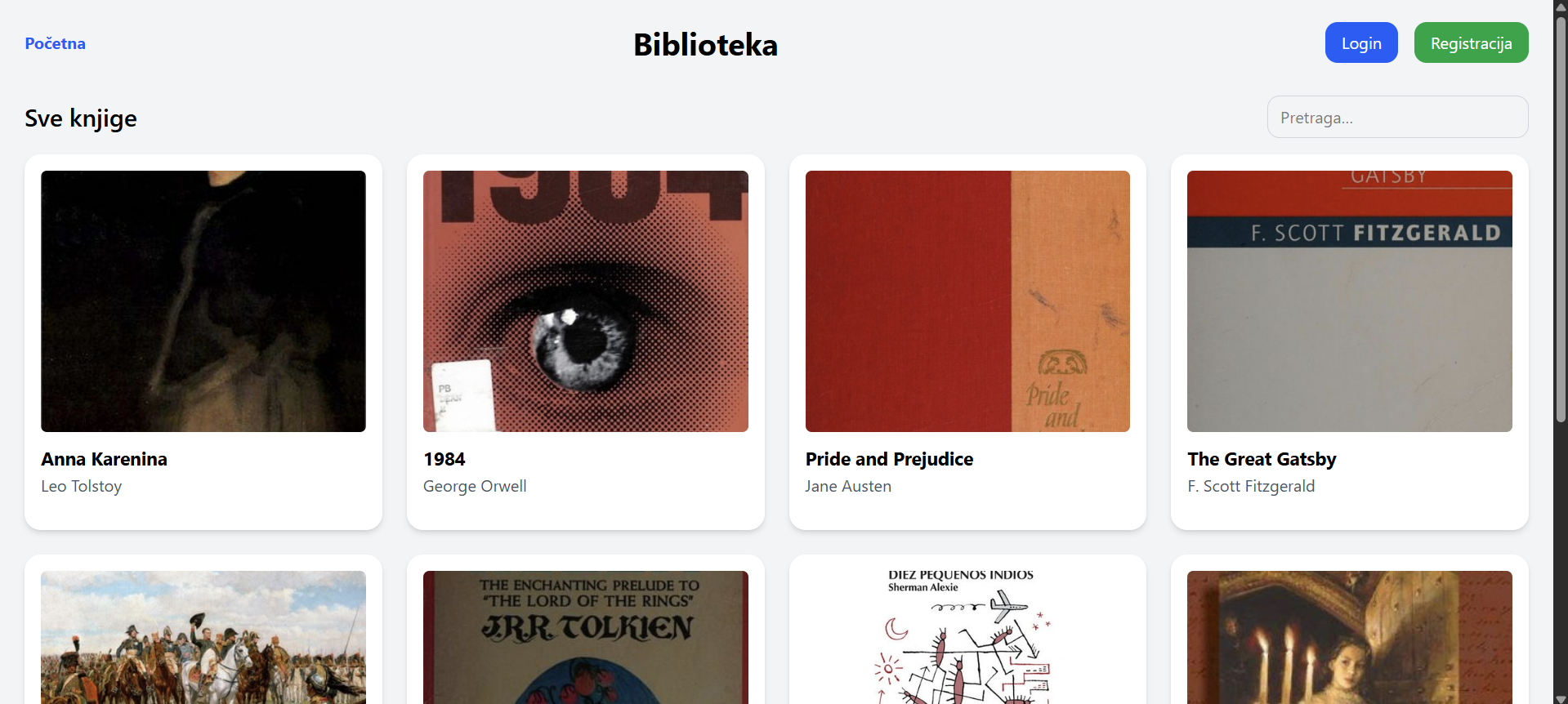
Vite + Tailwind JIT značajno ubrzavaju razvoj: klase se generišu u realnom vremenu i u konačnom bundle‑u ostaju samo one koje su zaista korišćene.

* **Responsivni dizajn “out‑of‑the‑box”**

Kroz Tailwind‑ove breakpoint util-klase (sm:, md:, lg:) lako se postiže prilagodljiv raspored (grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4) na različitim ekranima.

## Pregled ključnih strana

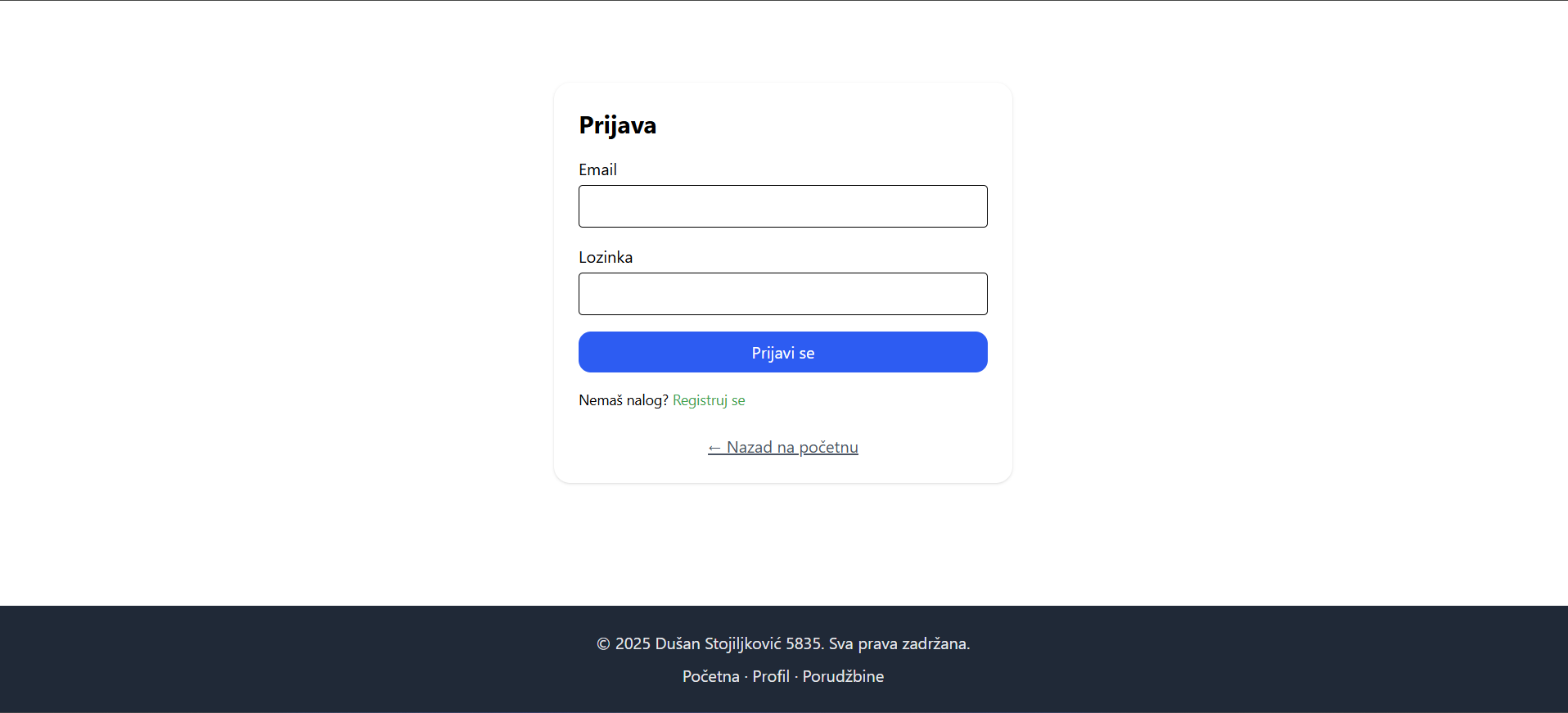
### Početna strana



Slika 1 – početna strana

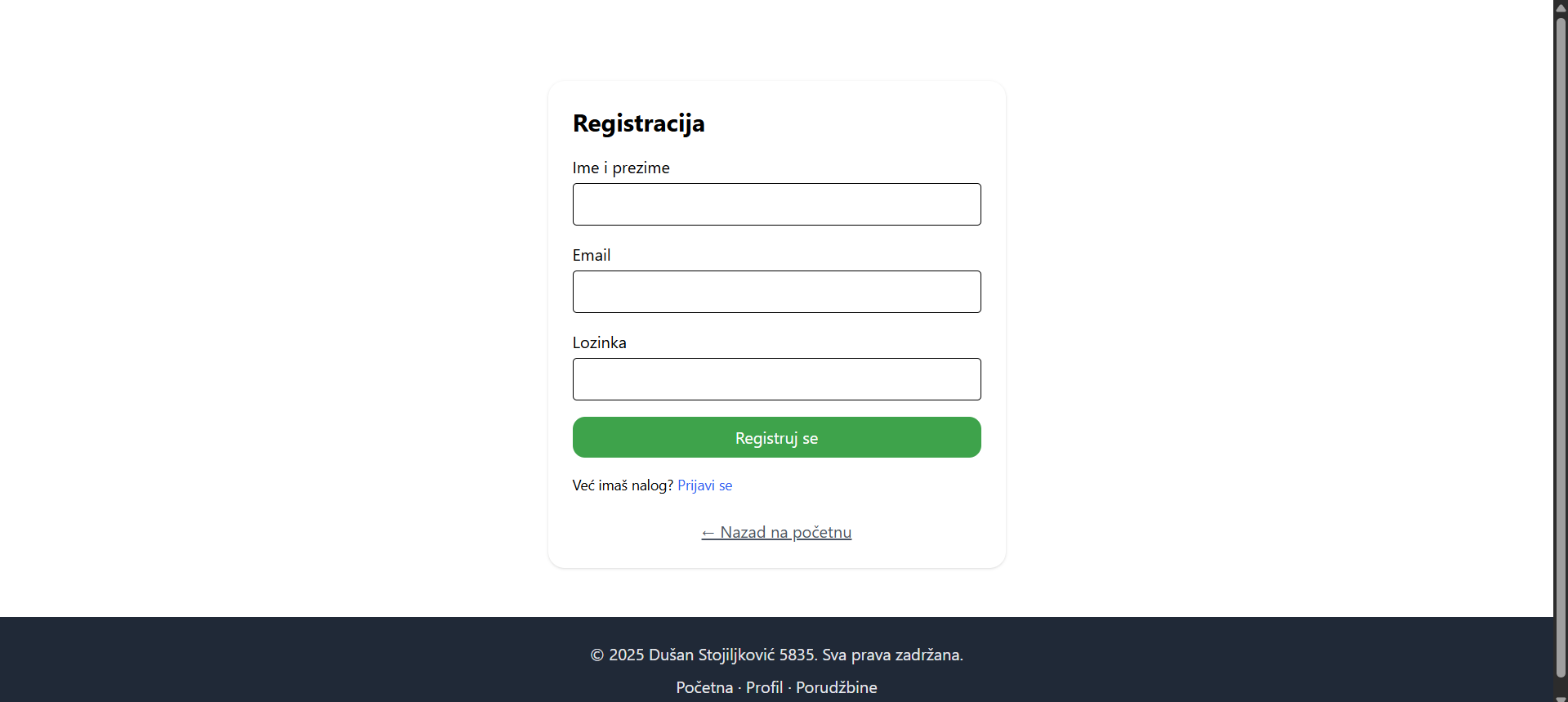
Početna strana na kojoj guest može da pregleda sve dostupne knjige u biblioteci, pretraži knjige po autoru, godini izdanja ili nazivu knjige i da se uloguje ili registruje.

### Login i registracija



Slika 2 – Stranica za prijavu

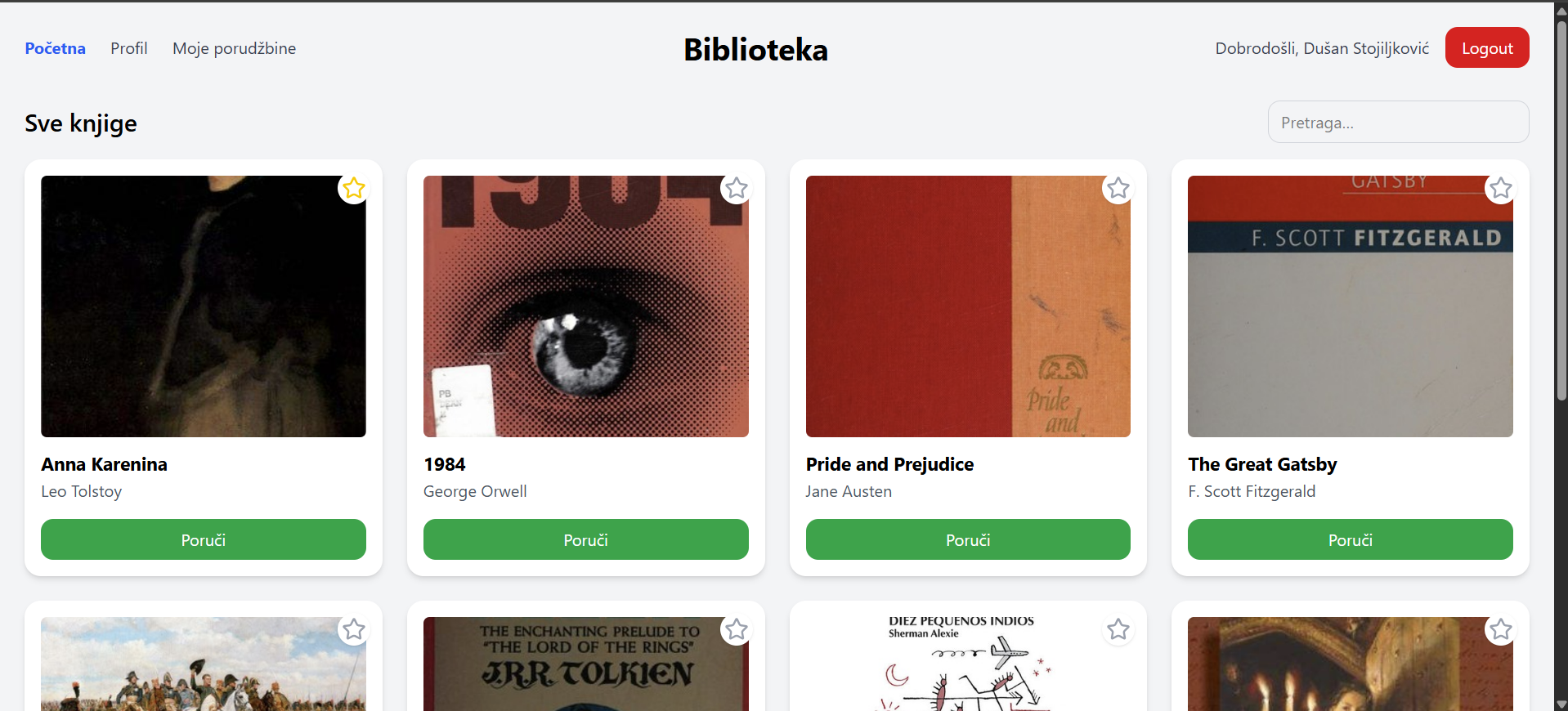
Na stranici za prijavu korisnik unosi svoje kredencijale i na osnovu svoje uloge biva redirektovan na odgovarajuću stranicu.



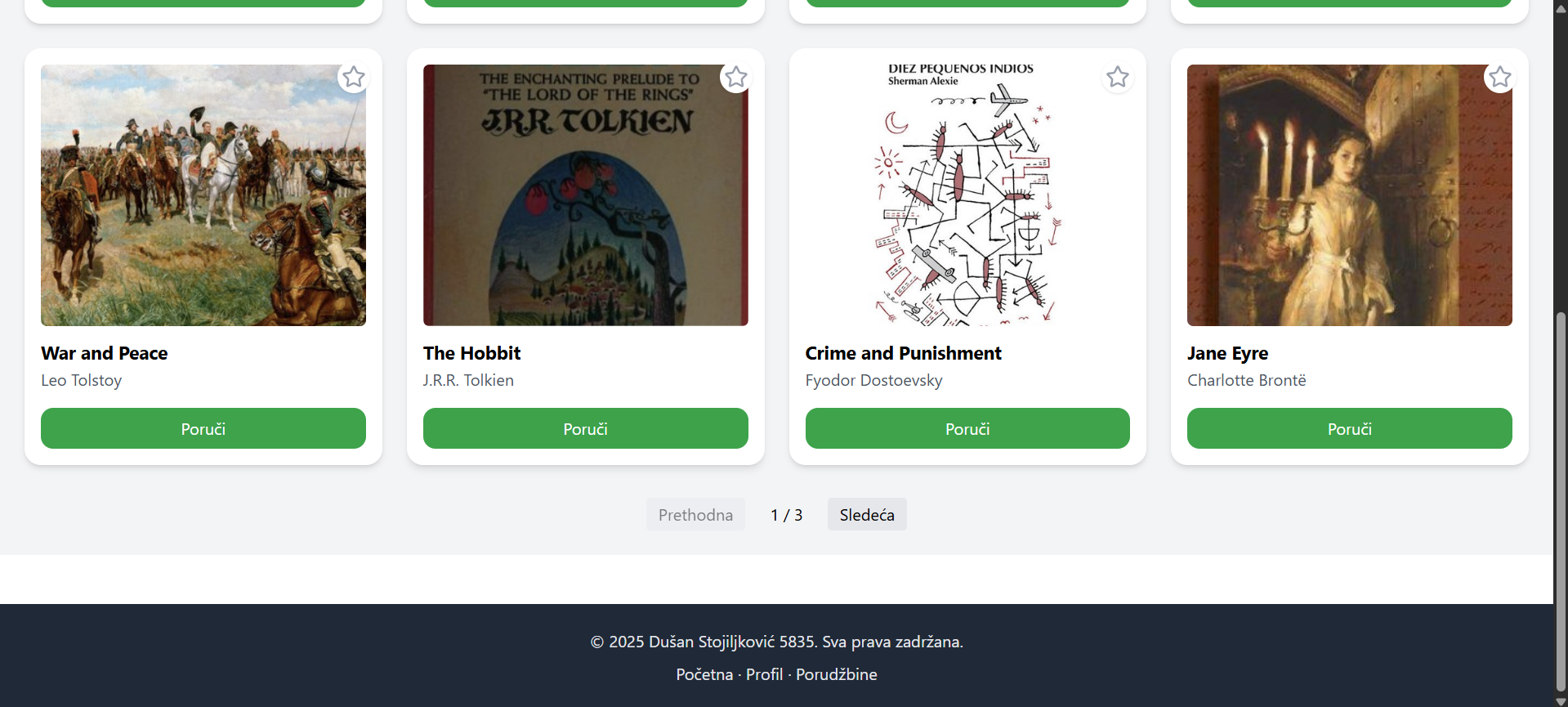
Slika 3 – Stranica za registraciju

Korisnik popunjava polja podacima i registruje svoj nalog. Podaci moraju biti validni (najmanje 6 karaktera za ime i prezime, validan format email adrese, password mora imati minimalno 6 karaktera, ne možemo registrovati postojeću email adresu). Ako korisnik unese neku nevalidnu informaciju biće obavešten porukom na ekranu.

### Početna strana korisnika

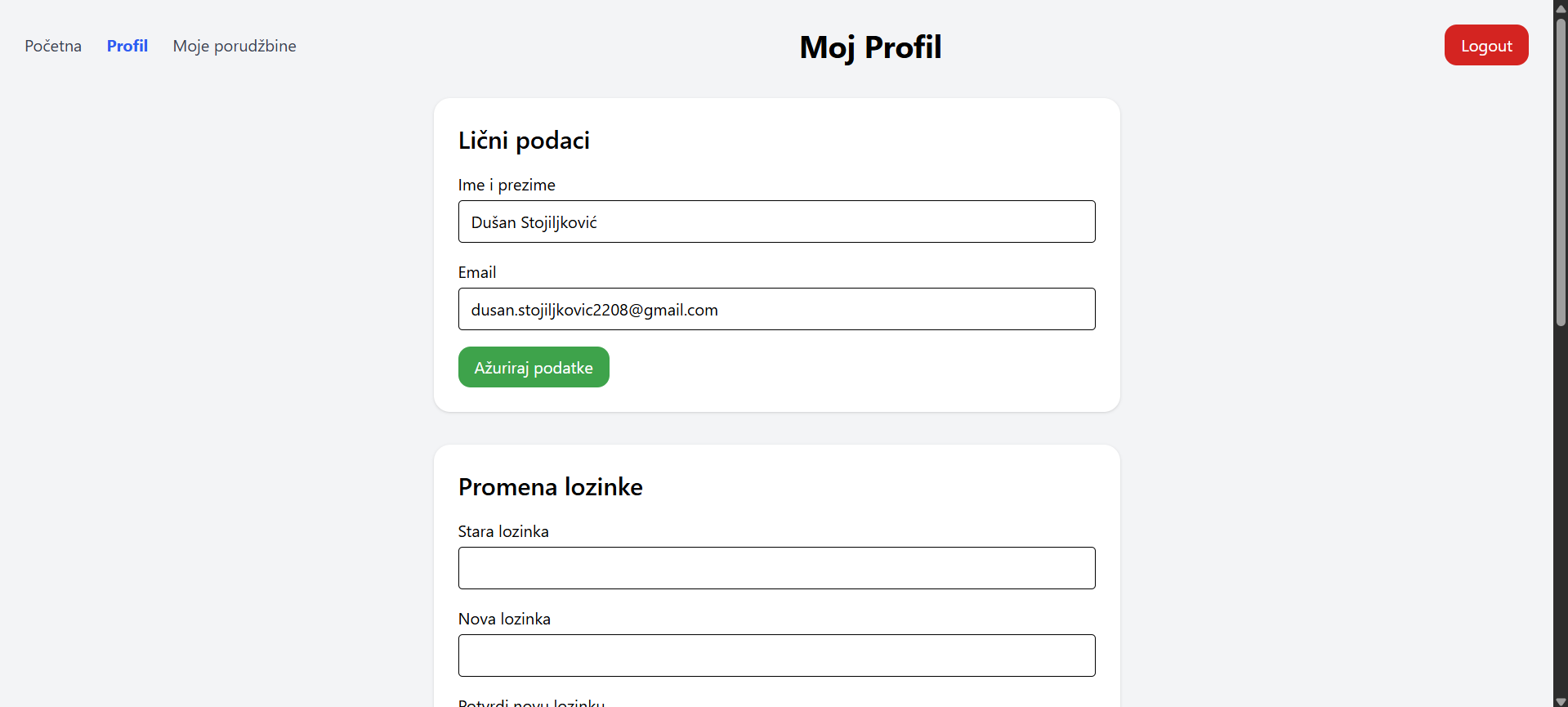


Slika 4 – Početna strana korisnika

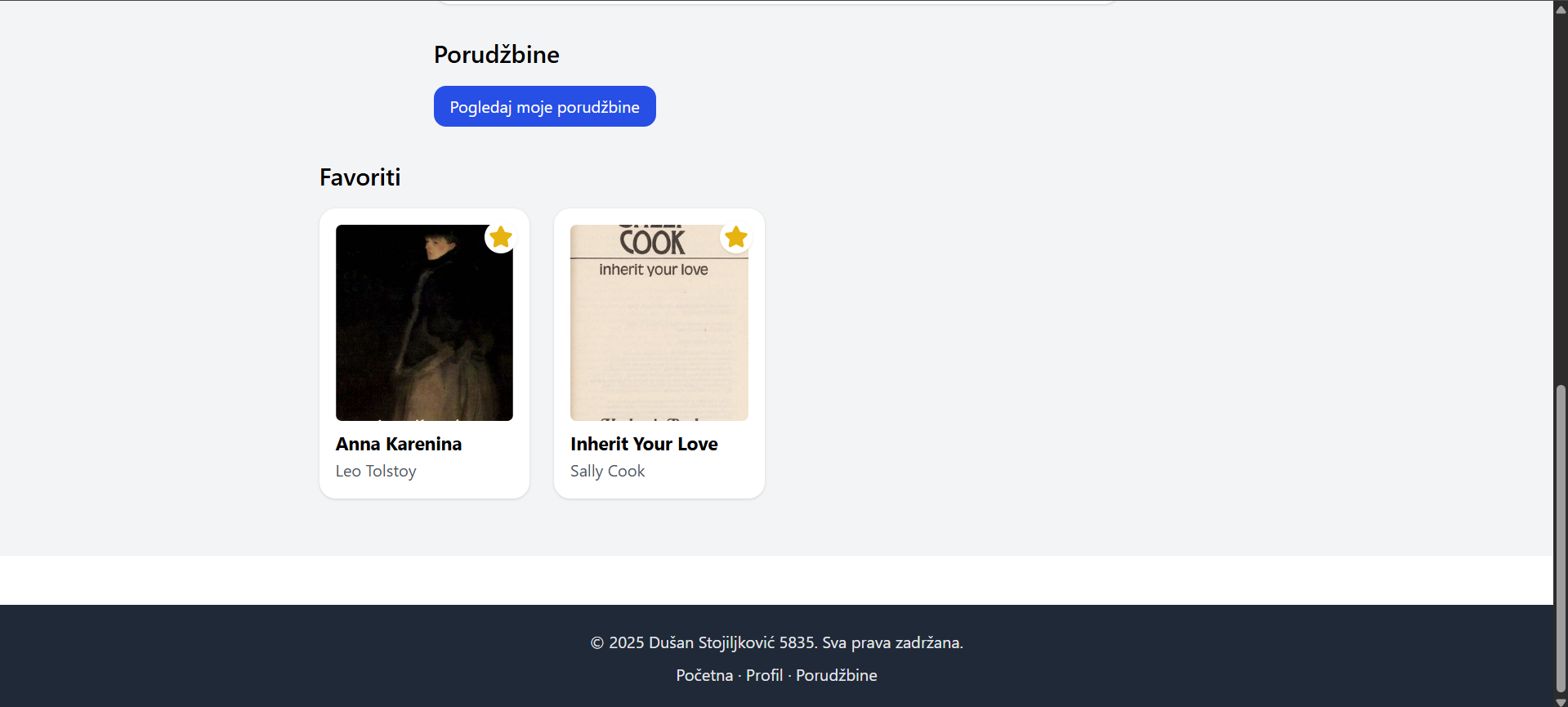
Za razliku od guesta, korisnik može dodavati knjige u omiljene klikom na zvezdicu u gornjem desnom uglu knjige i ima dugme za poručivanje knjige. Takođe ima dve dodatke kartice u navigacionom meniju a to su Profil i Moje porudžbine. Na slici 5 možemo videti da na početnoj strani prikazujemo 8 knjiga i paginacijom smo omogućili da korisnik može da pregleda knjige na drugim stranama.

Slika 5 - Paginacija

### Profil



Slika 6 – Profil



Slika 7 - Favoriti

Na ovoj stranici možemo da izmenimo lične podatke i lozinku i da pregledamo sve knjige koje smo dodali kao omiljene.

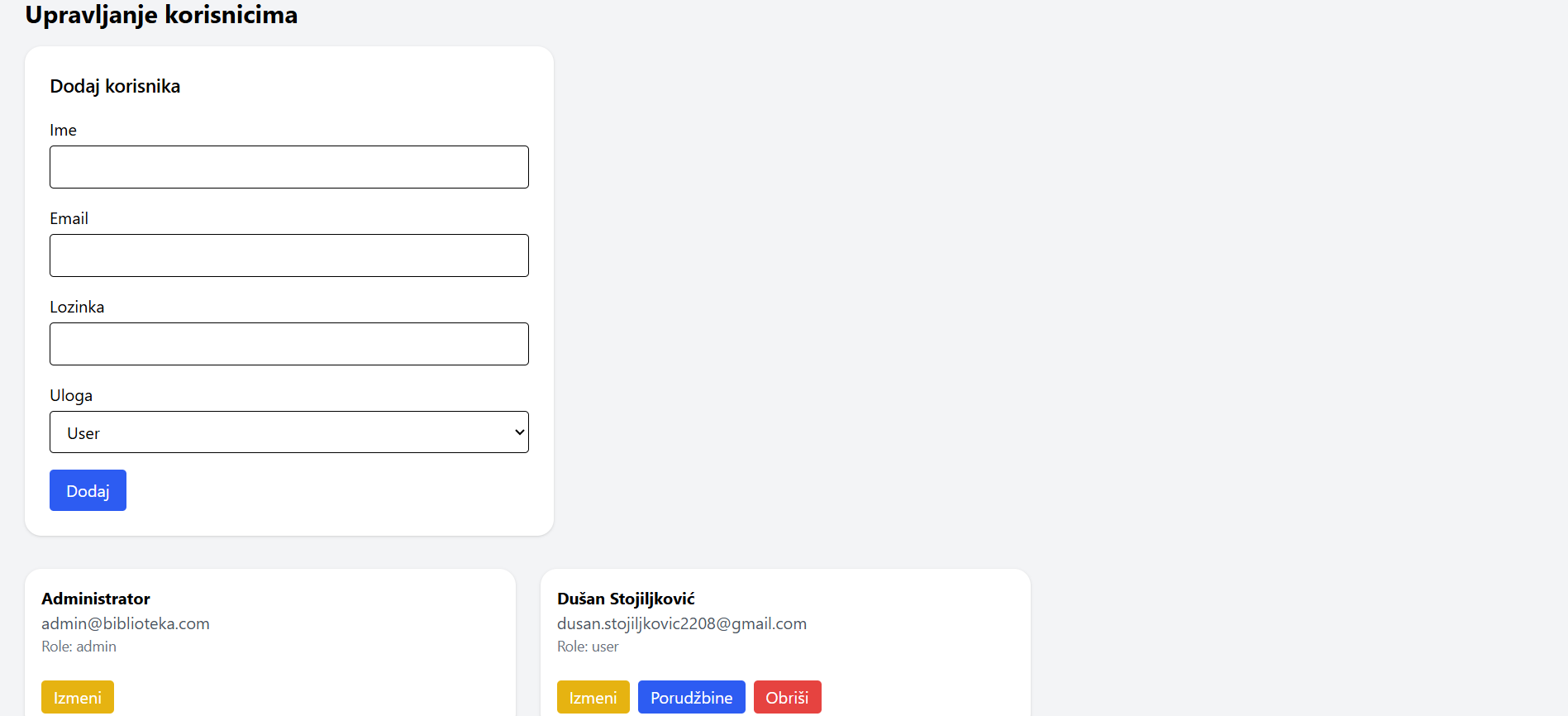
### Porudžbine



Slika 8 – Porudžbine

Na ovoj strani možemo videti sve naše aktivne porudžbine i dugme za povratak knjige biblioteci kao i istoriju knjiga koje smo poručivali.

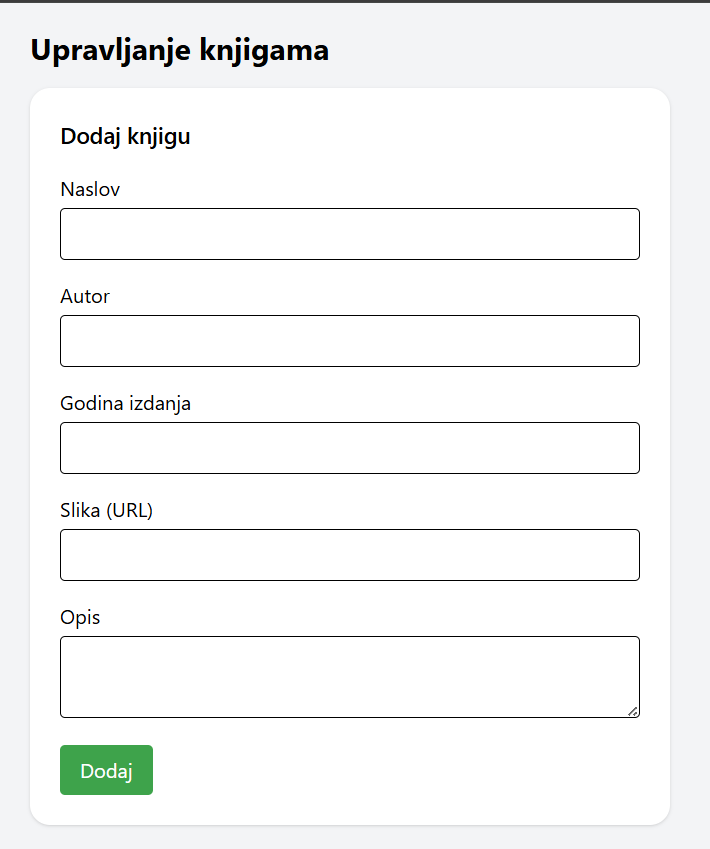
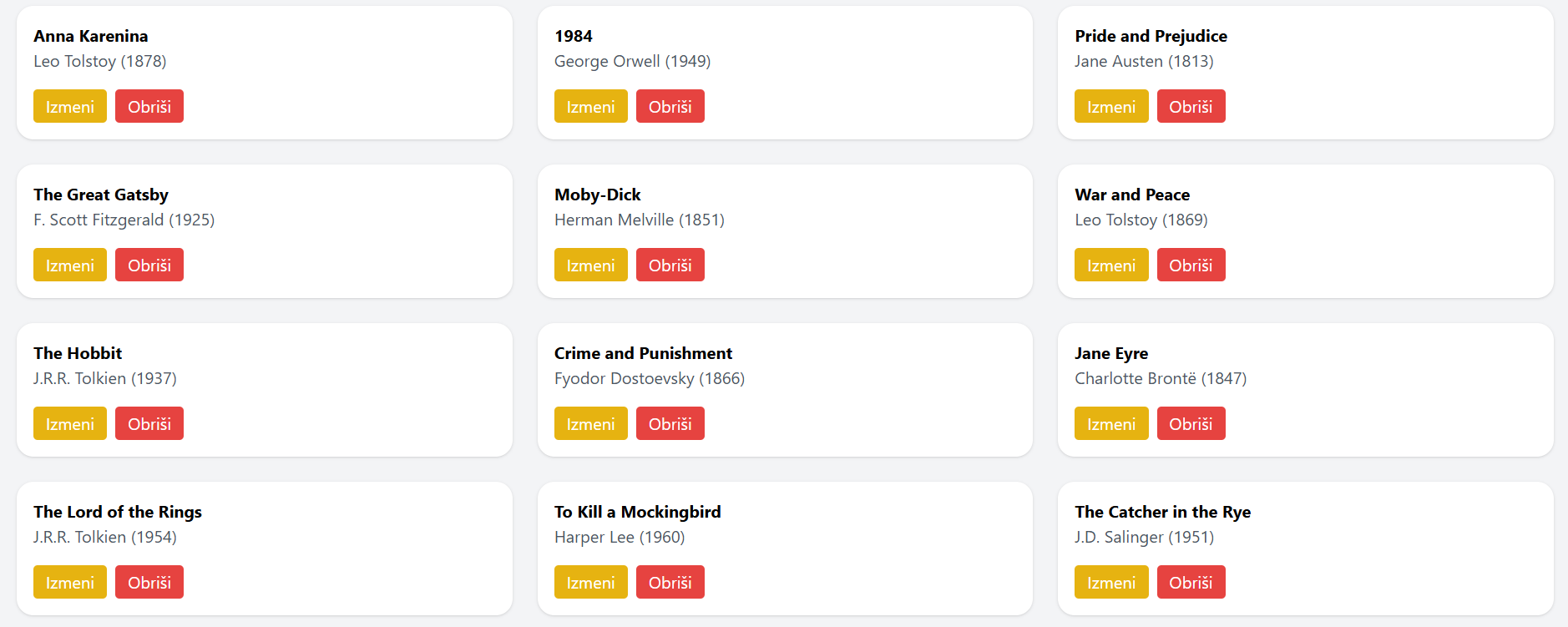
### Admin panel za upravljanje korisnicima



Slika 8 – Panel za upravljanje korisnicima

Na ovom panelu možemo obavljati CRUD operacije nad korisnicima i videti sve aktivne porudžbine kao i istoriju porudžbina za svakog korisnika. Admin može promeniti i svoje kredencijale ali ne može menjati svoju ulogu jer uvek mora postajati barem jedan admin nalog i ne može brisati sopstveni nalog.

### Admin panel za upravljanje knjigama

 Na ovoj strajici admin može vršiti CRUD operacije nad knjigama. Brisanje knjige se vrši kaskadno tako da će se ta knjiga obrisati i iz favorita korisnika i iz aktivnih porudžbina kao i iz istorije porudžbina. Nakon dodavanja nove knjige možemo sami uneti URL slike omota ili pokrenuti skriptu iz comman prompt-a komandom **npm run fill:covers** koja će pretražiti sajt **openlibrary** po nazivu knjige i ako pronađe sliku za tu knjigu ažurirati db.json dokument.

Slika 9 – Panel za upravljanje knjigama

# Implementacija funkcionalnosti

## Korisničke funkcionalnosti

Registracija se obavlja u komponenti RegisterPage.jsx. Nakon klijentske validacije imena, email‑a i lozinke (validate()), poziva se metoda register iz konteksta za autentikaciju.



Slika 10 – Registracija kod

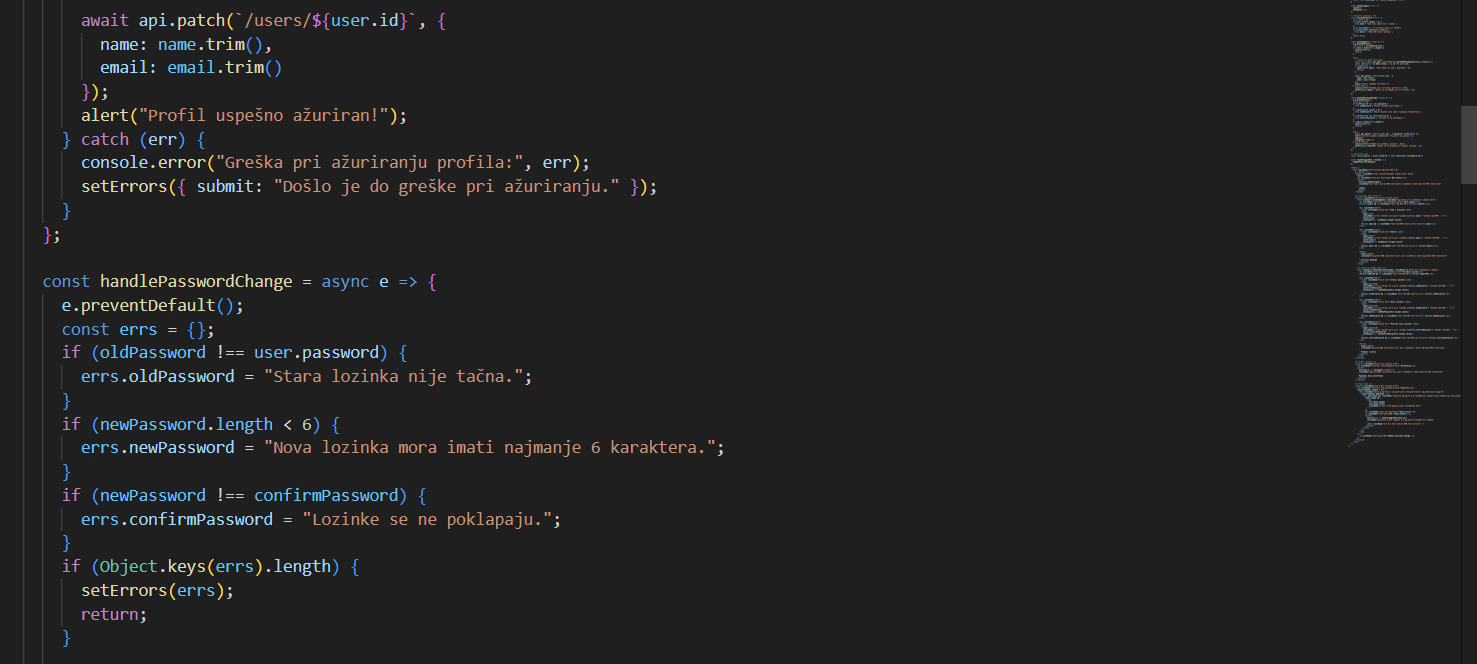
Prijava je implementirana u LoginPage.jsx, gde se poziva login(email,password).



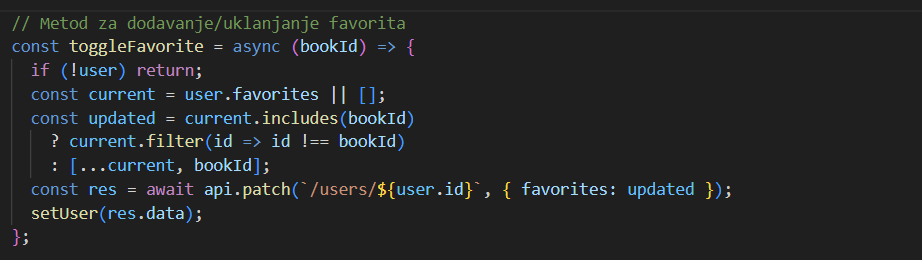
Slika 11 – Prijava kod

Profil korisnika (ProfilePage.jsx) prikazuje i ažurira lične podatke.

Promena lozinke vrši se drugim PATCH pozivom, uz proveru stare lozinke iz stanja user.password.

Slika 12 – Ažuriranje ličnih podataka i lozinke

Favoriti: toggleFavorite(bookId) u AuthContext.jsx ažurira niz favorites



Slika 13 – Ažuriranje omiljenih knjiga

Iznajmljivanje knjiga u BookCard.jsx



Slika 14 – Iznajmljivanje knjiga

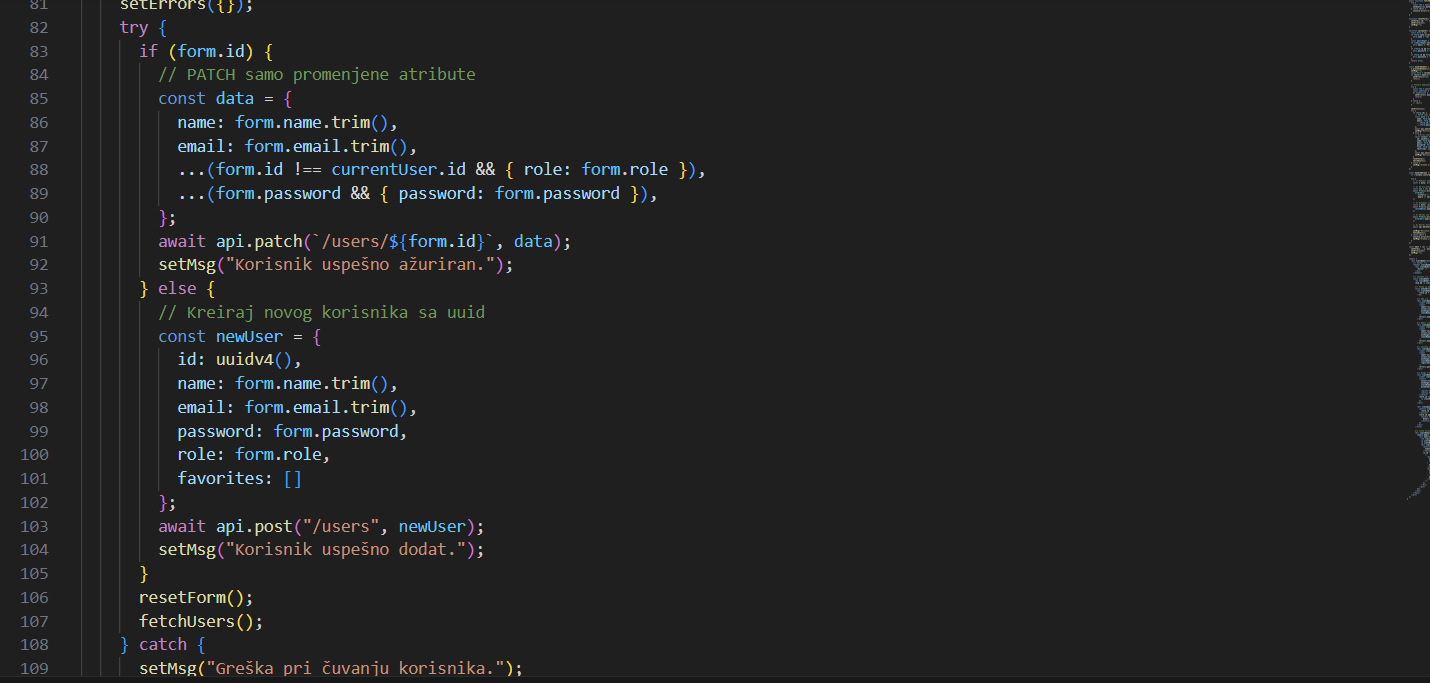
Vraćanje knjiga u OrdersPage.jsx



Slika 15 – Vraćanje knjige

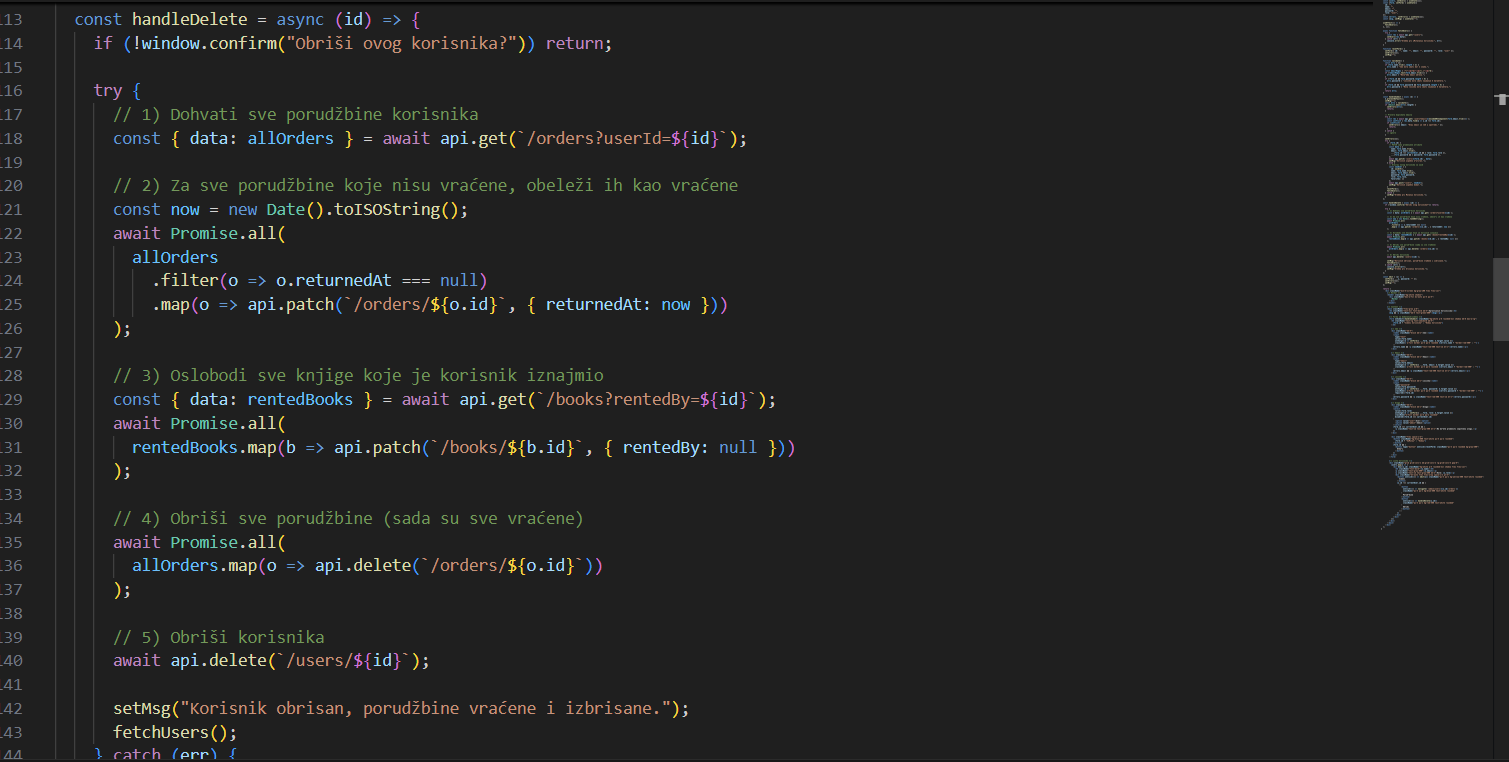
## Administratorske funkcionalnosti

Dodavanje i ažuriranje korisnika u AdminUsersPage.jsx



Slika 16 – Dodavanje i ažuriranje korisnika

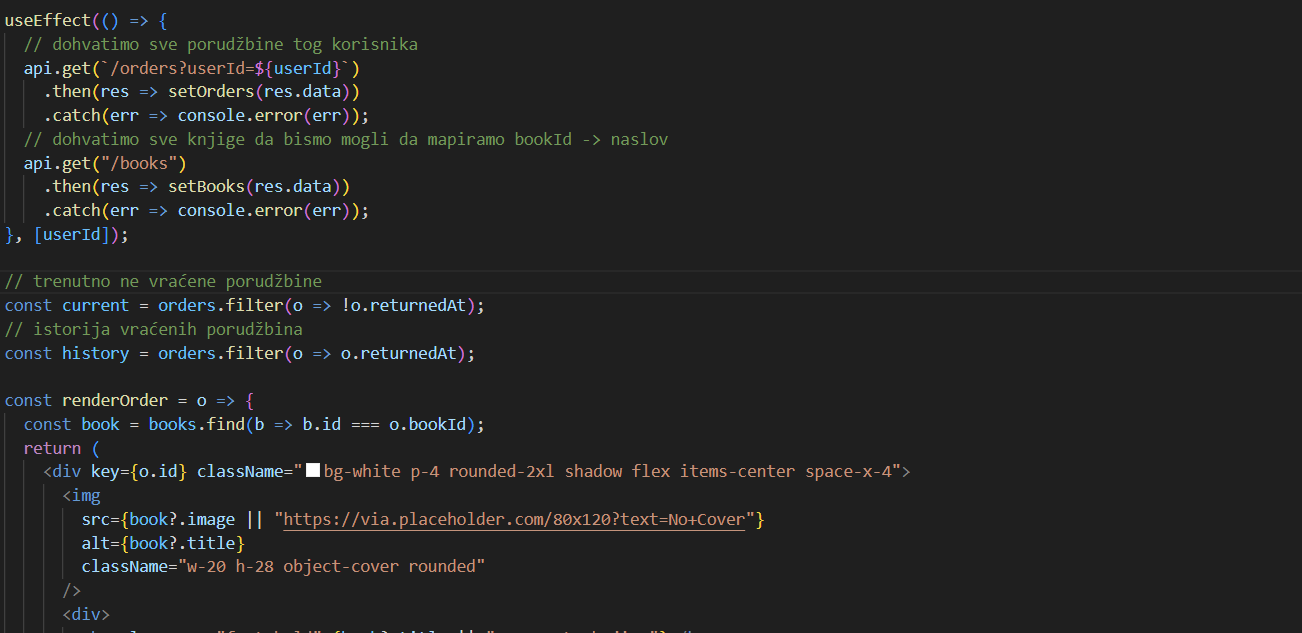
Brisanje korisnila u AdminUsersPage.jsx



Slika 17 – Brisanje korisnika

Funkcionalnost za dodavanje, brisanje i ažuriranje knjiga je potpuno ista kao i za korisnika.

Prikaz svih porudžbina korisnika u AdminUserOrdersPage.jsx



Slika 18 – Prikaz svih porudžbina

# Backend

Backend je REST API zasnovan na **Node.js/Express** + **MongoDB/Mongoose**. Autentikacija je preko **JWT Bearer** tokena. Uloge su **user** i **admin**.

## Struktura projekta

biblioteka-backend/

├─ controllers/

│ ├─ authController.js

│ ├─ usersController.js

│ ├─ booksController.js

│ └─ ordersController.js

├─ middleware/

│ ├─ authMiddleware.js

│ └─ roleMiddleware.js

├─ models/

│ ├─ User.js

│ ├─ Book.js

│ ├─ Order.js

│ └─ Activity.js

├─ routes/

│ ├─ auth.routes.js

│ ├─ users.routes.js

│ ├─ books.routes.js

│ ├─ activities.routes.js

│ └─ orders.routes.js

├─ utils/

│ └─ pagination.js

├─ validation/

│ ├─ auth.validation.js

│ ├─ user.validation.js

│ ├─ book.validation.js

│ └─ order.validation.js

├─ scripts/

│ ├─ addCovers.mjs

│ └─ seed.js

├─ server.js

└─ .env

## Konfiguracija (.env)

PORT=4000

MONGO\_URI=mongodb://localhost:27017/biblioteka

JWT\_SECRET=supersecretchangeme

JWT\_EXPIRES\_IN=7d

CORS\_ORIGIN=http://localhost:5173

## Modeli

### User

 name (String), email (String, unique), password (bcrypt hash), role ("user"|"admin"), favorites: [ObjectId(Book)]

 Lozinka se **uvek hešuje** i pri registraciji i pri admin kreiranju.

### Book

* title, author, year (Number), image (URL ili null), description, rentedBy: ObjectId(User) | null.

### Order

 userId: ObjectId(User)

 bookId: ObjectId(Book) | null (može postati *null* ako je knjiga obrisana)

 bookSnapshot: { id, title, author, year, image }

 status: "active" | "returned" | "canceled" | "book\_deleted"

 rentedAt, returnedAt|null

 **Snapshot** se upisuje u trenutku kreiranja porudžbine.

### Activity

* type (npr. BOOK\_CREATED, ORDER\_RETURNED), userId?, meta?, createdAt.

## Middleware

 **authMiddleware**: čita Authorization: Bearer <jwt>, verifikuje token, puni req.user.

 **roleMiddleware("admin")**: dozvoljava pristup samo adminima.

## Paginacija, pretraga i sortiranje

 parsePagination(req.query):

* page (default 1), limit (npr. 12), proračun skip
* sort format: field:dir (npr. title:asc, year:desc)

 **/books**: server-side pretraga q nad title|author (regex, i), sortiranje i paginacija.

 Sve liste vraćaju **uniformni odgovor**:

{ "data": { "items": [...], "total": 123, "page": 1, "pageSize": 12 } }

## Rute i kontroleri

### Auth

* **POST /api/auth/register** { name, email, password } → 201 { data: { token, ...user } }  
  Greške: 409 ako email postoji.
* **POST /api/auth/login** { email, password } → 200 { data: { token, ...user } }
* **GET /api/auth/me** → 200 { data: user } (zahteva Bearer)

### Users (admin osim update-self)

* **GET /api/users** (paginacija)
* **POST /api/users** (admin) – heš lozinke pre upisa
* **PATCH /api/users/:id** – admin i/ili vlasnik; može { favorites: string[] }
* **DELETE /api/users/:id** (admin)

### Books

* **GET /api/books** – q, page, limit, sort
* **GET /api/books/:id**
* **POST /api/books** (admin)
* **PATCH /api/books/:id** (admin)
* **DELETE /api/books/:id** (admin)  
  Pre brisanja:
  1. popuni bookSnapshot gde ga nema,
  2. **zatvori** aktivne porudžbine (returnedAt=now, status="book\_deleted"),
  3. unset bookId u porudžbinama,
  4. obriši knjigu, zabeleži Activity.

### Orders

* **GET /api/orders** – user vidi svoje; admin ?userId=.  
  Odgovor dopunjen poljima:
  + displayBook: objekat za prikaz (živa knjiga ako postoji, u suprotnom snapshot)
  + isActive: true ako status="active" i returnedAt=null
* **POST /api/orders** { bookId } → kreira porudžbinu:
  + set book.rentedBy=user,
  + upiše bookSnapshot,
  + vrati displayBook + isActive: true.  
    Greška: 409 ako je knjiga već iznajmljena.
* **PATCH /api/orders/:id/return** – vlasnik ili admin; set returnedAt, status="returned", book.rentedBy=null (ako knjiga postoji).

## Bezbednost

 JWT potpisan tajnom iz .env; obavezni Authorization: Bearer.

 Lozinke **bcrypt.hash** (10 salt rounds).

 CORS: dozvoli samo frontend origin.

 Validacije: Mongoose runValidators: true u update-ima + osnovne provere u kontrolerima.

## Standardni oblici grešaka

 401 Unauthorized (npr. bez tokena)

 403 Forbidden (rola/tuđi resurs)

 404 Not found

 409 Conflict (npr. dupli email, knjiga već iznajmljena)

 500 Internal Server Error (neočekivane greške)

Odgovor:{ "error": "Opis greške", "details": "opciono" }

# Zaključak

## 6.1. Pregled postignutih rezultata

Projekat **„Biblioteka”** uspešno realizuje planirane funkcionalnosti za tri nivoa korisnika uz stabilan backend i konzistentnu istoriju podataka.

* **Guest**: pregled kataloga, detalji knjige.
* **User**: registracija i prijava (JWT), uređivanje profila i lozinke, favoriti (dodavanje/uklanjanje), iznajmljivanje i vraćanje knjiga, pregled aktivnih zaduženja i istorije.
* **Admin**: CRUD nad korisnicima i knjigama, pregled porudžbina po korisniku, server-side pretraga/sort/paginacija.
* **Backend**: Node.js/Express + MongoDB/Mongoose, **JWT** autentikacija, **role-based** autorizacija, heš lozinki (bcrypt), dosledan format odgovora (liste: { data: { items, total, page, pageSize } }).
* **Porudžbine (robustnost)**: uveden **Order.bookSnapshot** (naslov/autor/godina/slika u trenutku kreiranja) + čišćenje pri brisanju knjige (zatvaranje aktivnih porudžbina returnedAt, status="book\_deleted", unset bookId).
* **Frontend**: React 18 + Vite, React Router, Tailwind. HomePage sa **pretragom, sortiranjem (naslov/autor/godina) i paginacijom** (server-side).

## 6.2. Izazovi i prepreke tokom rada na projektu

* **Konzistentnost podataka pri brisanju knjige**: inicijalno su porudžbine ostajale sa „mrtvom“ referencom. Rešeno snapshot-om, zatvaranjem aktivnih porudžbina i uklanjanjem bookId.
* **401 pri loginu korisnika kreiranih iz admin panela**: lozinke su bile snimane kao plaintext. Rešeno heširanjem u createUser + migraciona skripta za postojeće naloge.
* **500 pri brisanju uz transakcije**: lokalni Mongo bez replica-seta ne podržava transakcije. Implementirana varijanta brisanja **bez sessions/transactions**.
* **„Nepoznat naslov“ posle poručivanja**: frontend je spajao order sa listom knjiga. Rešeno dodatkom displayBook u /orders i korišćenjem bookSnapshot kao fallbacka.
* **Paginacija i pretraga**: prebačeno na **server-side** (stabilne performanse, dosledni rezultati).
* **Favoriti i dozvole**: rešeni 403/ID problemi (ispravno slanje Authorization i korišćenje validnog user.\_id).
* **Kvalitet omota knjiga bez API ključeva**: eliminisani nepouzdani izvori, pooštren matching (score) i uveden **SVG fallback** da se izbegnu pogrešne slike.
* **UI upozorenja (React keys)**: dodati stabilni key atributi u listama.

## 6.3. Mogućnosti za buduća unapređenja

* **Domena biblioteke**: rokovi, penali, rezervacije, obaveštenja (email/push), „librarian“ rola.
* **Bezbednost**: refresh tokeni, reset lozinke preko emaila, rate-limit/helmet, audit dashboard.
* **Pretraga i performanse**: full-text (Atlas Search), dodatni indeksi, keš (Redis) za popularne upite.
* **Podaci i tokovi**: soft-delete za knjige/korisnike + restore; bulk import/export (CSV), „review mode“ za omote.
* **DevOps**: Docker Compose, CI/CD, seed skripte, staging okruženje; (po želji) replica-set i transakcije.
* **UI/UX**: tamni režim, pristupačnost (a11y), mikro-animacije, i18n, poboljšan „optimistic UI“.

# Prilozi

Link ka GitHub repozitorijumu: <https://github.com/work-dusan/biblioteka---react---node.js.git>

# Literatura

1. <https://react.dev/learn> [20.08.2025.]
2. <https://tailwindcss.com/docs/installation/using-vite> [20.08.2025.]
3. <https://www.w3schools.com/REACT/DEFAULT.ASP> [20.08.2025.]
4. <https://nodejs.org/docs/latest/api/> [20.08.2025.]
5. <https://www.w3schools.com/nodejs/> [20.08.2025.]
6. <https://www.youtube.com/watch?v=f2EqECiTBL8> [20.08.2025.]