Seminarski rad

Tema: E biblioteka

Sadržaj

[Verbalni opis 3](#_Toc76398007)

[Opis slučajeva korišćenja 4](#_Toc76398008)

[Opis procesa – dijagrami sekvenci 7](#_Toc76398009)

[SK1: Kreiranje seminarskog rada 7](#_Toc76398010)

[SK2: Kreiranje prijave 9](#_Toc76398011)

[Konceptualni model 12](#_Toc76398012)

[Specifikacija REST API – ja 13](#_Toc76398013)

[Opis korišćenih tehnologija 16](#_Toc76398014)

[React 16](#_Toc76398015)

[Express 16](#_Toc76398016)

[Axois 17](#_Toc76398017)

[Mysql 17](#_Toc76398018)

[TypeORM 17](#_Toc76398019)

[Reprezentativni delovi koda 18](#_Toc76398020)

[Link ka udaljenom repozitorijumu 23](#_Toc76398021)

# Verbalni opis

Potrebno je napraviti veb aplikaciju biblioteke.

Aplikacija treba da prepozna 2 vrste korisnika:

* Običnog korisnika
* Administratora

Da bi korisnik radio sa aplikacijom, potrebno je da se uloguje ili, ukoliko nema svoj nalog, da se registruje.

Običan korisnik može da pretražuje, gleda i naručuje knjige.

Administrator ima sve funkcionalnosti kao i običan korisnik, a pored toga može da kreira, menja i briše knjige i prati i menja stanje narudžbina.

Prilikom analize sistema za koji se pravi aplikacija, prepoznati su sledeći domenski objekti:

* Knjiga
* Žanr
* Autor
* Korisnik
* Komentar

Svaku knjigu je napisao jedan ili više autora, a jedan autor može napisati više knjiga.

Knjiga može spadati u više žanrova, dok u jednom žanru može pripadati više knjiga.

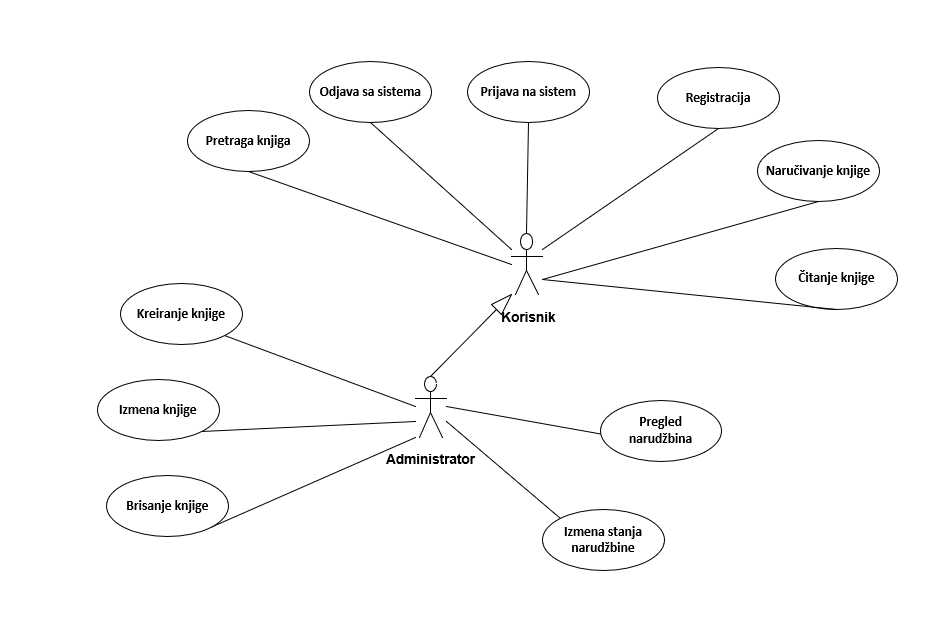
Svaka korpa se vezuje za jednog korisnika dok jedan korisnik može imati više korpi.

Svaka korpa ima listu svojih stavki.

Stavka ima korpu kojoj pripada, knjigu koja se naručuje kao i količinu knjiga koje se naručuju.

# Opis slučajeva korišćenja

U sistemu smo prepoznali sledeće slučajeve korišćenja:

* Prijava na sistem
* Odjava sa sistema
* Registrovanje na sistem
* Pretraga knjiga
* Kreiranje knjige
* Izmena knjige
* Brisanje knjige
* Naručivanje knjiga
* Čitanje knjige
* Izmena stanja narudžbine
* 

Slika 1 Dijagram slučajeva korišćenja

## 

**SK1: Slučaj korišćenja – Kreiranje knjige**

**Aktori SK**

Administator

**Učesnici SK**

Administator i sistem

**Preduslov: Sistem je uključen i administator je ulogovan pod svojom šifrom. Sistem prikazuje administatoru formu za rad sa knjigom.**

**Osnovni scenario SK**

1. Administator unosi vrednosti za kreiranje knjige. (APUSO)
2. Administator poziva sistem da kreira knjigu sa datim podacima. (APSO)
3. Sistem kreira knjigu sa datim podacima. (SO)
4. Sistem prikazuje administatoru poruku: „Sistem je usepšno kreirao knjigu“. (IA)

Alternativna scenarija

* 1. Ukoliko sistem ne može da kreira knjigu on prikazuje administatoru objavu: „Sistem ne može da kreira knjigu“.(IA)

**SK2: Slučaj korišćenja – Pretraga knjiga**

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov: Sistem je uključen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje korsiniku formu za rad sa knjigom.**

**Osnovni scenario SK**

1. Korisnik unosi vredonost po kojoj pretražuje knjigu. (APUSO)
2. Korisnik poziva sistem da nađe knjigu po zadatoj vrednosti. (APSO)
3. Sistem traži knjigu po zadatoj vrednosti. (SO)
4. Sistem prikazuje korsiniku knjigu. (IA)

Alternativna scenarija

* 1. Ukoliko sistem ne može da nađe objavu on prikazuje korsiniku objavu: „Sistem ne može da nađe knjigu“.(IA)

**SK3: Slučaj korišćenja – Naručivanje knjiga**

**Aktori SK**

Korisnik

**Učesnici SK**

Korisnik i sistem

**Preduslov: Sistem je uključen, korisnik je ulogovan pod svojom šifrom i korisnik je odabrao proizvode koje želi da naruči. Sistem prikazuje korsiniku formu za rad sa korpom.**

**Osnovni scenario SK**

1. Korisnik unosi telefonski broj i adresu. (APUSO)
2. Korisnik poziva sistem da sačuva korpu sa datim podacima. (APSO)
3. Sistem čuva korpu sa datim podacima. (SO)
4. Sistem čuva podatake za svaku stavku korpe.(SO)
5. Sistem prikazuje korsiniku poruku: „Sistem je usepšno sačuvao korpu“. (IA)

Alternativna scenarija

* 1. Ukoliko sistem ne može da sačuva korpu on prikazuje korsiniku objavu: „Sistem ne može da sačuva korpu“.(IA) prekida se izvršenje SK
  2. Ukoliko sistem ne može da sačuva neku stavku on briše sve podatke o prethodno sačuvanim stavkama i korpi i prikazuje korisniku poruku: „Sistem ne može da sačuva korpu“. (IA)

# Opis procesa – dijagrami sekvenci

## SK1: Kreiranje knjige



Slika 2 SK1: Osnovni scenario



Slika 3 SK1: Alternativni scenario

## SK2: Pretraga knjiga

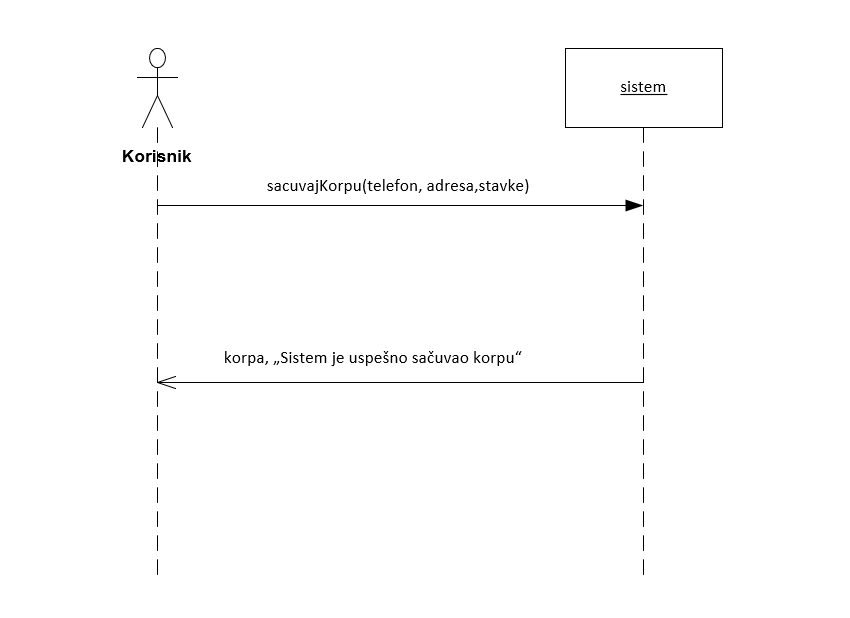


Slika 4 SK2: Osnovni scenario

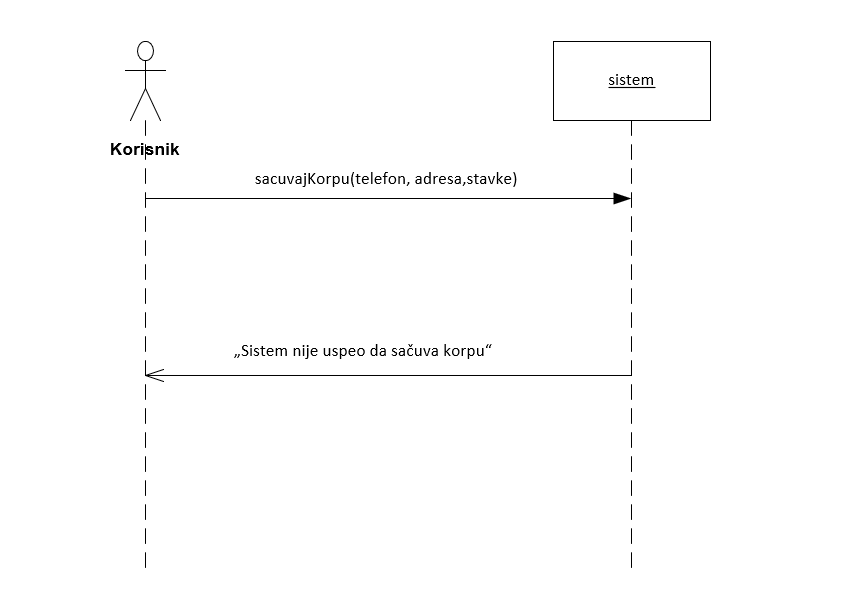


Slika 5 SK2: Alternativni scenario

## SK3: Naručivanje knjiga

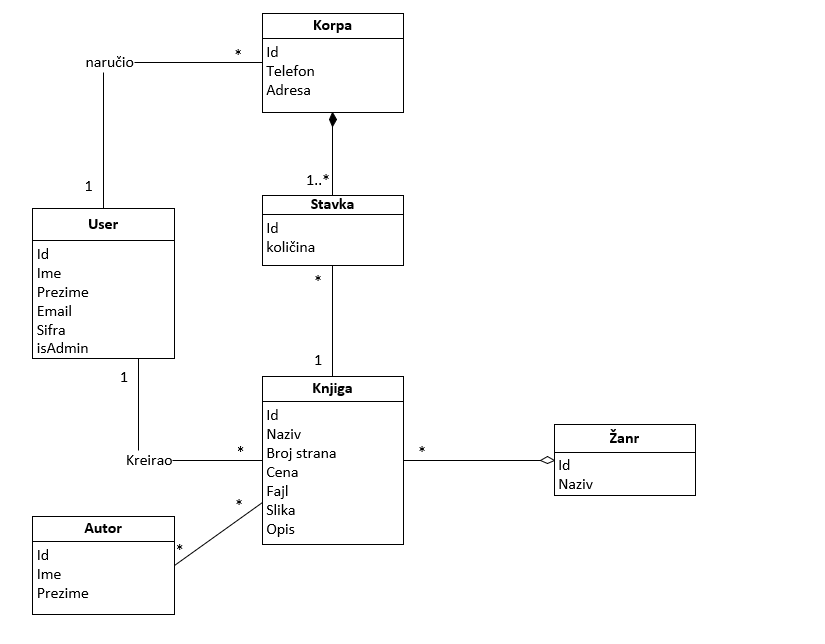


Slika 6 SK3 Osnovni scenario



Slika 7 SK3 Alternativni scenario

# Konceptualni model



Slika 8 Konceptualni model

# Specifikacija REST API – ja

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve knjige |
| HTTP metoda | GET |
| URL | /knjiga |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [    ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve žanrove |
| HTTP metoda | GET |
| URL | /topic |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [    ] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Kreiraj knjigu |
| HTTP metoda | POST |
| URL | /knjiga |
| URL parametri | (nema ) |
| HTTP body parametri | Primer: |
| Format HTTP body parametara | application/json |
| Izlazni parametri | Primer: |
| Format izlaznih parametara | application/json |

# 

# Opis korišćenih tehnologija

Prilikom izrade aplikacije korišćene su sledeće tehnologije:

* React – kreiranje poslovne logike na strani klijenta
* Express – kreiranje servera
* Axios – biblioteka za pravljenje AJAX zahteva
* Typeorm – objektno – relacioni maper
* Mysql – baza podatakaReact

## React

React predstavlja biblioteka za kreiranje korisničkog interfejsa i poslovne logike na klijentskoj strani. React služi za kreiranje SPA web aplikacija odnosno aplikacija koje imaju samo jednu html stranu i kojima glavnu ulogu ima javascript.

Glavni koncept u React – je komponenta.

Komponenta predstavlja nešto što može da se iscrta kao jedna celina.

Celokupni prikaz aplikacije dobija se kompozicijom raznih elemenata u okviru jedne glavne komponente koja se, najčešće, naziva App.

Komponente mogu definisane kao funkcija i kao klasa.

Svaka komponenta ima svoje atribute koje je dobila komponente roditelja i svoje stanje koje je ona sama kreirala. Komponenta može da menja svoje stanje ali ne može da menja svoje atribute. Prilikom promene stanja poziva se posebna funkcija koja to radi nakon čega se prikaz na ekranu automatski menja na svim mestima koja su se iscrtavala na osnovu tog stanja.

## Express

Predstavlja „framework“ za izradu serverske strane u Node.js okruženju. Za razliku od većine ostalih servera, Express ne podržava višenitno programiranje. Problem konkurentnosti on rešava pomoću asinhronog programiranja i „event loop“ – a.

Express aplikacije su zamišljene da rade kao REST servisi, a obrada zahteva klijenta vrši se na sledeći način:

* Pristiže zahtev klijenta
* Izvršavaju se funkcije koje su definisane da se izvrše nad svakim zahtevom („middleware“)
* Zahtev se rutira na osnovu svog URL – a
* Izvršava se funkcija pridodata za dati URL
* Šalje se odgovor korisniku

## Axois

Axios je biblioteka koja pojednostavljuje rad sa AJAX zahtevima. Axios omogućava kreiranje svih vrsta http zahteva kao i kreiranje zaglavlja. Kao odgovor sa servera, axios vraća korisniku originalni odgovor sa pridruženim metapodacima koje je axios izgenerisao.

## Mysql

Relaciona baza podataka koja je besplatna za korišćenje.

## TypeORM

Objektno – relacion maper u javascript – u. Pravila prevođenja vrše se anotacijom klasa u programskom kodu. Sva konfiguracija nalazi se fajlu ormconfig.json. TypeORM omogućava rad sa bazom na 2 načina:

* Preko repozitorijuma (Repository pattern)
* Preko menadžera

U prvom slučaju olakšan je rad sa bazom ali su mogućnost kontrole toka i prilagođavanje rada ograničeni dok je u drugom omogućena mnogo veća kontrola i kastomizacija na račun komplikovanijeg koda.

# Reprezentativni delovi koda

Login putanja

 app.post('/login', async (req, res) => {

        const { email, password } = req.body;

        const users = await getRepository(User).find({

            where: {

                password,

                email

            }

        });

        if (users.length === 0) {

            res.status(400).send('User doesn\'t exist');

        } else {

            (req.session as any).user = users[0];

            req.session.save();

            res.json(users[0]);

        }

    })

Podaci o putanji

export interface Route {

    method: 'get' | 'post' | 'patch' | 'delete',

    route: string,

     actions: ((req: Request, res: Response, next?: any) => void |

Promise<void>)[]

}

Entitet knjiga

@Entity()

export class Knjiga {

    @PrimaryGeneratedColumn()

    id?: number;

    @Column()

    naziv: string

    @Column()

    brojStrana: number;

    @Column({ nullable: true })

    fajl?: string;

    @Column()

    slika: string;

    @ManyToOne(t => Zanr, { eager: true })

    zanr: Zanr

    @ManyToMany(t => Autor, { eager: true })

    @JoinTable({ name: 'autorstvo' })

    autori: Autor[]

    @ManyToOne(t => User, { eager: true })

    kreirao: User

    @Column()

    cena: number

    @Column()

    opis: string

}

Prebacivanje fajlova:

{

    method: 'post',

    route: '/knjiga',

    actions: [isAdmin, upload, renameFile('image'), renameFile('file'), kreirajKnjigu]

}

Kreiranje knjige:

export async function kreirajKnjigu(req: Request, res: Response) {

    const user = (req.session as any).user as User;

    const data = req.body as KnjigaDTO;

    const insertRes = await getManager().save(Knjiga, {

        autori: JSON.parse(data.autori).map(id => {

            return {

                id

            }

        }),

        zanr: {

            id: data.zanr

        },

        brojStrana: data.brojStrana,

        cena: data.cena,

        naziv: data.naziv,

        slika: data.image,

        fajl: data.file,

        opis: data.opis,

        kreirao: {

            id: user.id

        }

    })

    res.json(await getRepository(Knjiga).findOne(insertRes.id));

}

Rutiranje na klijentskoj strani

<BrowserRouter>

      <Navbar user={user} logout={logout} />

      {

        user ? (

          <Switch>

            <Route path='/knjiga/:id'>

              <KnjigaPrikaz dodajUKorpu={dodajUKorpu} getKnjiga={id => {

                return knjige.find(elem => elem.id === id)

              }} />

            </Route>

            <Route path='/knjige'>

              <KnjigePage zanrovi={zanrovi} autori={autori} knjige={knjige} />

            </Route>

            {

              user.isAdmin && (

                <Route path='/dashboard'>

                  <Dashboard kreiraj={kreirajKnjigu} izmeni={izmeniKnjigu} obrisi={obrisiKnjigu} autori={autori} zanrovi={zanrovi} knjige={knjige} />

                </Route>

              )

            }

            <Route path='/korpa'>

              <KorpaPage naruci={naruci} izmeniStavku={izmeniStavku} obrisiStavku={obrisiStavku} stavke={stavke} />

            </Route>

            <Route path='/'>

              <HomePage knjige={knjige.slice(0, 3)} />

            </Route>

          </Switch>

        ) : (

          <Grid columns='16'>

            <Grid.Row centered className='loginRed'>

              <Grid.Column className='loginSlika' width='9'>

              </Grid.Column>

              <Grid.Column width='6'>

                <Switch>

                  <Route path='/register'>

                    <RegisterPage onSubmit={register} />

                  </Route>

                  <Route path='/'>

                    <LoginPage onSubmit={login} />

                  </Route>

                </Switch>

              </Grid.Column>

            </Grid.Row>

          </Grid>

        )

      }

    </BrowserRouter>

Pozivanje API – ja.

export async function obrisiKnjiguService(id: number) {

    await axios.delete(SERVER + '/knjiga/' + id);

}

export async function izmeniKnjiguService(data: KnjigaDTO, id: number) {

    const res = await axios.patch(SERVER + '/knjiga/' + id, data);

    return res.data as Knjiga;

}

export async function kreirajKnjiguService(data: FormData) {

    const res = await axios.post(SERVER + '/knjiga', data, {

        headers: {

            'Content-type': 'multipart/form-data'

        }

    })

    return res.data as Knjiga;

}

# Link ka udaljenom repozitorijumu