Seminarski rad

Tema: E – prodavnica

Milica Stanković

Sara Bošković

Sadržaj

[Verbalni opis 3](#_Toc68516297)

[Opis slučajeva korišćenja 4](#_Toc68516298)

[Opis procesa – dijagrami sekvenci 5](#_Toc68516299)

[Konceptualni model 6](#_Toc68516300)

[Specifikacija REST API – ja 7](#_Toc68516301)

[Reprezentativni delovi koda 9](#_Toc68516302)

[Link ka udaljenom repozitorijumu 10](#_Toc68516303)

# Verbalni opis

Potrebno je kreirati online prodavnicu odeće. Prodavnici mogu pristupiti samo ulogovani korisnici.

Korisnik može da gleda i naručuje proizvode. Kada korisnik naruči određeni proizvod, on će biti smešten u korpu. Korisnik može staviti više proizvoda u istu korpu. Takođe, korisnik može odrediti količinu proizvoda koju želi da kupi.

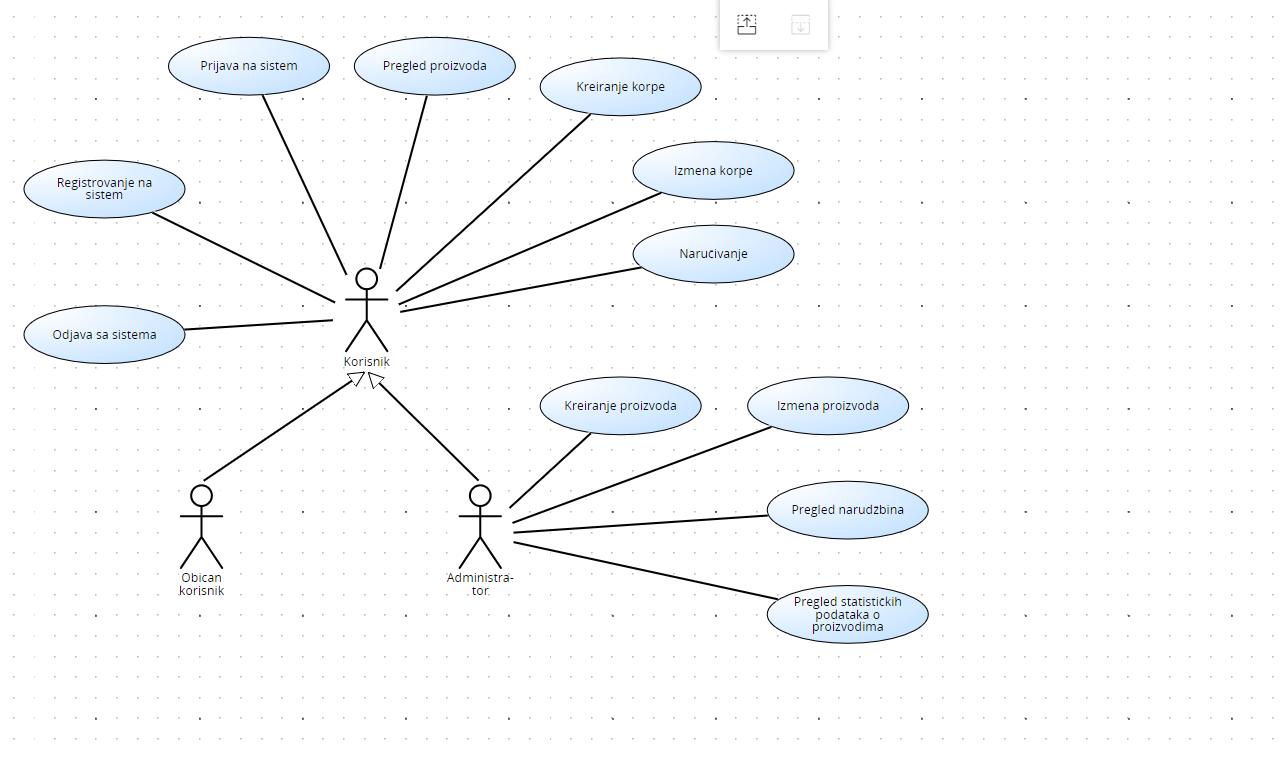
Kako bi izvršio naruđžbinu, korisnik mora da unese podatke o kontakt telefonu i adresi dostave.

U sistemu postoji i specijalan korisnik, administrator. On ima sve mogućnosti kao i običan korisnik, s tim što može da prati sve naručene pošiljke. Takođe, on može i kreirati i menjati podatke o proizvodima.

# Opis slučajeva korišćenja

U sistemu smo prepoznali sledeće slučajeve korišćenja:

* Prijava na sistem
* Registrovanje na sistem
* Odjava sa sistema
* Pregled proizvoda
* Kreiranje korpe
* Izmena korpe
* Naručivanje
* Kreiranje proizvoda
* Izmena proizvoda
* Pregled narudžbina
* Pregled statističkih podataka o proizvodima



Slika 1 Dijagram slučajeva korišćenja

## 

**SK1: Slučaj korišćenja – Kreiranje proizvoda**

**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 proizvodom.**

**Osnovni scenario SK**

1. Korisnik unosi vrednosti za kreiranje proizvoda. (APUSO)
2. Korisnik poziva sistem da kreira proizvod sa datim podacima. (APSO)
3. Sistem kreira proizvod sa datim podacima. (SO)
4. Sistem prikazuje korsiniku poruku: „Sistem je usepšno kreirao proizvod“. (IA)

Alternativna scenarija

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

**SK2: Slučaj korišćenja – Naručivanje**

**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**

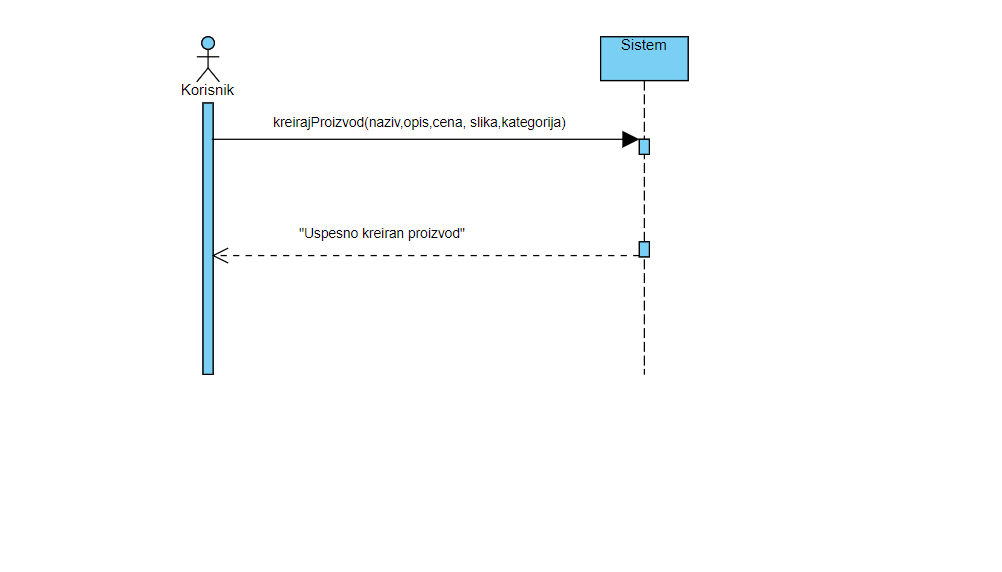
1. Korisnik unosi telefonski broj i adresu. (APUSO)
2. Korisnik poziva sistem da sačuva korpu sa datim podacima. (APSO)
3. Sistem čuva korpu sa datim podacima. (SO)
4. Sistem čuva podatake za svaku stavku korpe.(SO)
5. Sistem prikazuje korsiniku poruku: „Sistem je usepšno sačuvao korpu“. (IA)

Alternativna scenarija

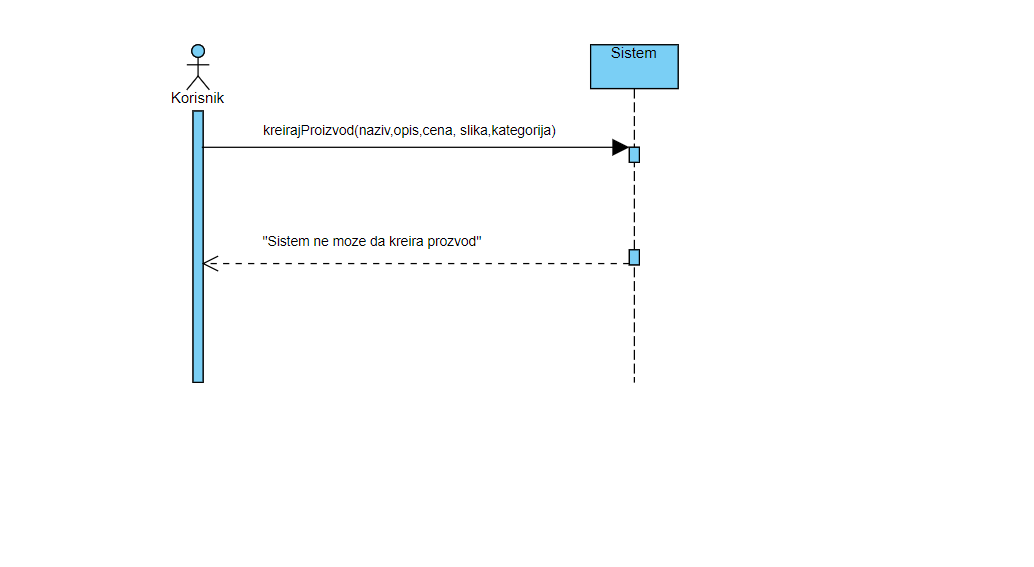
* 1. Ukoliko sistem ne može da sačuva korpu on prikazuje korsiniku objavu: „Sistem ne može da sačuva korpu“.(IA) prekida se izvršenje SK
  2. Ukoliko sistem ne može da sačuva neku stavku on briše sve podatke o prethodno sačuvanim stavkama i korpi i prikazuje korisniku poruku: „Sistem ne može da sačuva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

# Opis procesa – dijagrami sekvenci

## Kreiranje proizvoda



Slika 2

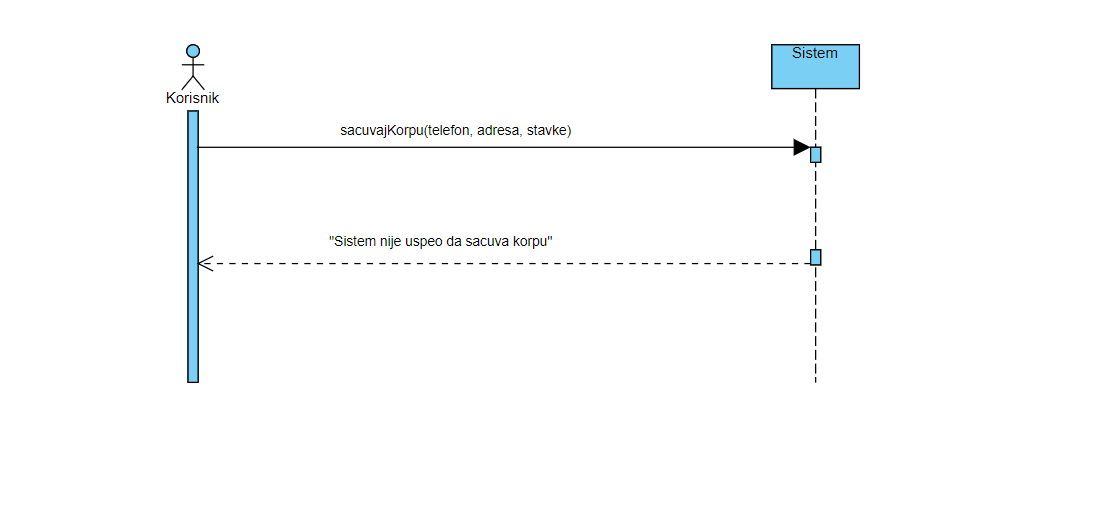


Slika 3

## Čuvanje korpe

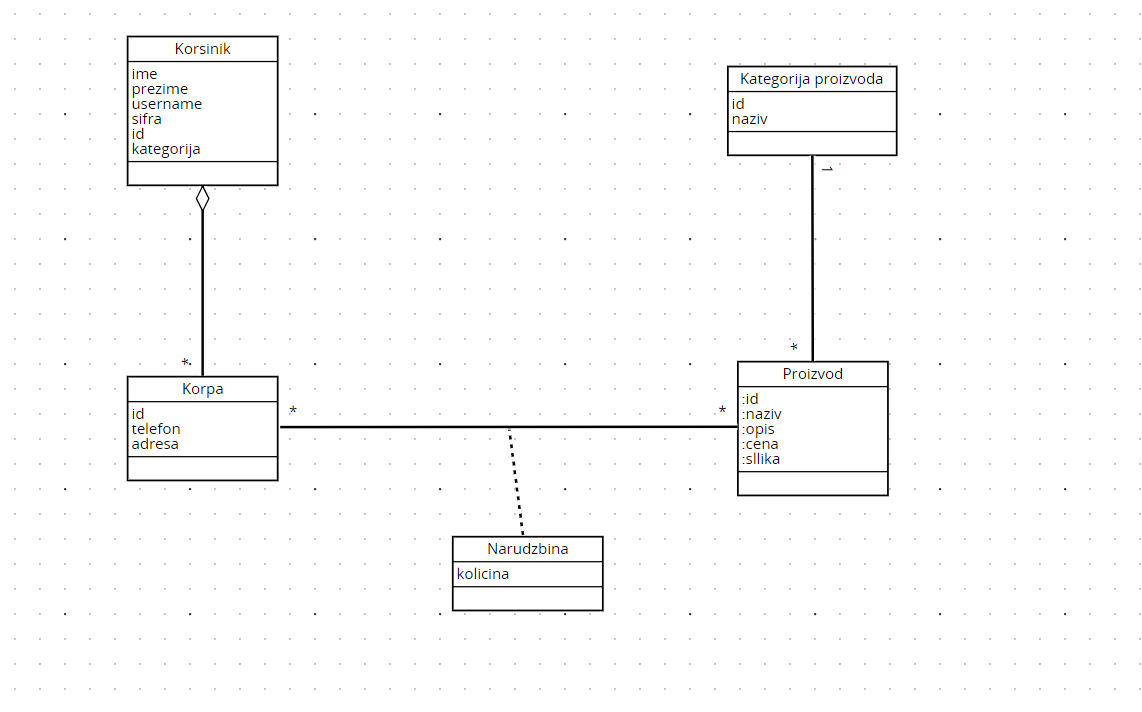
## 

Slika 4



Slika 5

# Konceptualni model



Slika 6 Konceptualni model

# Specifikacija REST API – ja

|  |  |
| --- | --- |
| Opis funkcije | Vrati sve korpe |
| HTTP metoda | GET |
| URL | /cart |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [  {  "id":2,  "adress":"Beogradska 7",  "phone":"0628916761",  "user":{  "id":1,  "firstName":"a",  "lastName":"a",  "category":"admin",  "username":"a"  },  "items":[  {  "product":{  "id":5,  "name":"afd",  "description":"fadsg",  "picture":"slika.jpg",  "price":345,  "productCategory":{  "id":2,  "name":"farmerke"  }  },  "ammount":1  }]  }  ] |
| Format izlaznih parametara | application/json |
| Opis funkcije | Vrati sve kategorije |
| HTTP metoda | GET |
| URL | /category |
| URL parametri | (nema) |
| HTTP body parametri | (nema) |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  [{  "id":1,  "name":"haljine"  },  {  "id":2,  "name":"farmerke"  },  {  "id":3,  "name":"majice"  }] |
| Format izlaznih parametara | application/json |

|  |  |
| --- | --- |
| Opis funkcije | Izmeni proizvod |
| HTTP metoda | PATCH |
| URL | /product/:id |
| URL parametri | Id proizvoda |
| HTTP body parametri | Primer:  {  “name”:”majica”,  “description”:”neki opis”,  “productCategory”:2  } |
| Format HTTP body parametara | (nema) |
| Izlazni parametri | Primer:  {  “id”:2,  “name”:”majica”,  “description”:”neki opis”,  “productCategory”:{  “id”:2,  “name”:”majice”  },  “price”:35,  “picture”:”slika.jpg”  } |
| Format izlaznih parametara | application/json |

# Reprezentativni delovi koda

 const key = fs.readFileSync('./key.pem', 'utf8');

    const cert = fs.readFileSync('./cert.pem', 'utf8');

    // create express app

    const app = express();

    app.use(cors({

        credentials: true,//protiv xss napada

        methods: ['GET', 'POST', 'PATCH', 'DELETE'],

        origin: 'http://localhost:3000'

    }));

    app.use(express.json());

    app.use(session({

        secret: 'adsfgdhtydafsjtiuyi',

        resave: false,

        saveUninitialized: false,

        cookie: {

            sameSite: 'none',

            secure: true,

            maxAge: 1000 \* 60 \* 10,//10min

            httpOnly: true,

        }

    }))

    // register express routes from defined application routes

    app.post('/product', upload.single('file'), ProductController.create)

    Routes.forEach(route => {

        app[route.method](route.route, (req: Request, res: Response) => {

            route.controller[route.action](req, res);

        });

    });

    // setup express app here

    // ...

    // start express server

    const server = https.createServer({

        key: key,

        cert: cert,

    }, app)

    server.listen(process.env.PORT || 4000, () => console.log('app is listening'))

export interface Route {

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

    route: string,

    action: string,

    controller: GenericController

}

@Entity()

export default class Order {

    @Column()

    ammount: number

    @ManyToOne(type => Product, { eager: true, primary: true, onDelete: 'RESTRICT' })

    product: Product;

    @ManyToOne(type => Cart, c => c.items, { eager: true, primary: true, onDelete: 'CASCADE' })

    cart: Cart;

}

async create?(request: Request, response: Response): Promise<any> {

        const user = (request.session as any).user as User | undefined;

        if (!user) {

            response.sendStatus(403);

            return;

        }

        const cart = request.body as Partial<Cart>;

        cart.user = user;

        getManager().transaction(async (manager) => {

            const createdCart = await manager.save(Cart, { adress: cart.adress, executed: false, phone: cart.phone, user: user });

            const prepared = cart.items.map(element => {

                return { ...element, cart: createdCart };

            });

            await manager.save(Order, prepared);

            response.sendStatus(200);

        })

    }

useEffect(() => {

    (async function () {

      try {

        const resUser = await axios.post(SERVER\_URL + '/check', {

        }, { withCredentials: true });

        setUser(resUser.data);

      } catch (error) {

      }

      try {

        const resProduct = await axios.get(SERVER\_URL + '/product')

        setProducts(resProduct.data);

        const resCat = await axios.get(SERVER\_URL + '/category');

        setCategories(resCat.data);

        const cartsRes = await axios.get(SERVER\_URL + '/cart')

        setCarts(cartsRes.data);

      } catch (error) {

        console.log(error.response);

      }

    })().then(val => {

      setLoading(false);

    })

  }, [])

    <BarChart

                        className='whiteBackground'

                        compact

                        width={1200}

                        height={500}

                        data={props.products.map(product => {

                            const res: any = {

                                product: product.name

                            }

                            let ammount = 0;

                            let totalPrice = 0;

                            for (let cart of props.carts) {

                                for (let item of cart.items) {

                                    totalPrice += item.ammount \* item.product.price;

                                    if (item.product.id === product.id) {

                                        ammount += item.ammount;

                                    }

                                }

                            }

                            const total = ammount \* product.price / totalPrice;

                            res.ammount = ammount;

                            res.total = total \* 100;

                            return res;

                        })}

                    >

                        <XAxis name='Product' dataKey="product" >

                            <Label value='Products' offset={0} position="insideBottom" />

                        </XAxis>

                        <YAxis label={{ value: '', angle: -90, position: 'insideLeft', fontSize: 14 }} minTickGap={1} />

                        <Tooltip />

                        <CartesianGrid strokeDasharray="1 1" stroke="#f5f5f5" />

                        <Bar name='Ammount ordered' dataKey="ammount" fill="blue" />

                        <Bar name='% in cart price' dataKey="total" fill="#83ca9d" />

                    </BarChart>

# Link ka udaljenom repozitorijumu