**Seminarski rad**

Tema: E – prodavnica

Studenti:

Ana Katić 117/17

Stefan Kujović 132/17

Aleksandar Marković 358/17

Sadržaj

[Verbalni opis 3](#_Toc76549455)

[Opis slučajeva korišćenja 4](#_Toc76549456)

[Opis procesa – dijagrami sekvenci 7](#_Toc76549457)

[Kreiranje proizvoda 7](#_Toc76549458)

[Čuvanje korpe 8](#_Toc76549459)

[Konceptualni model 9](#_Toc76549460)

[Specifikacija REST API – ja 10](#_Toc76549461)

[Opis korišćenih tehnologija 13](#_Toc76549462)

[React 13](#_Toc76549463)

[Express 13](#_Toc76549464)

[Axois 14](#_Toc76549465)

[Mysql 14](#_Toc76549466)

[TypeORM 14](#_Toc76549467)

[Korisničko uputstvo 15](#_Toc76549468)

[Reprezentativni delovi koda 20](#_Toc76549469)

# Verbalni opis

Kreiraćemo online prodavnicu odeće kojoj 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.

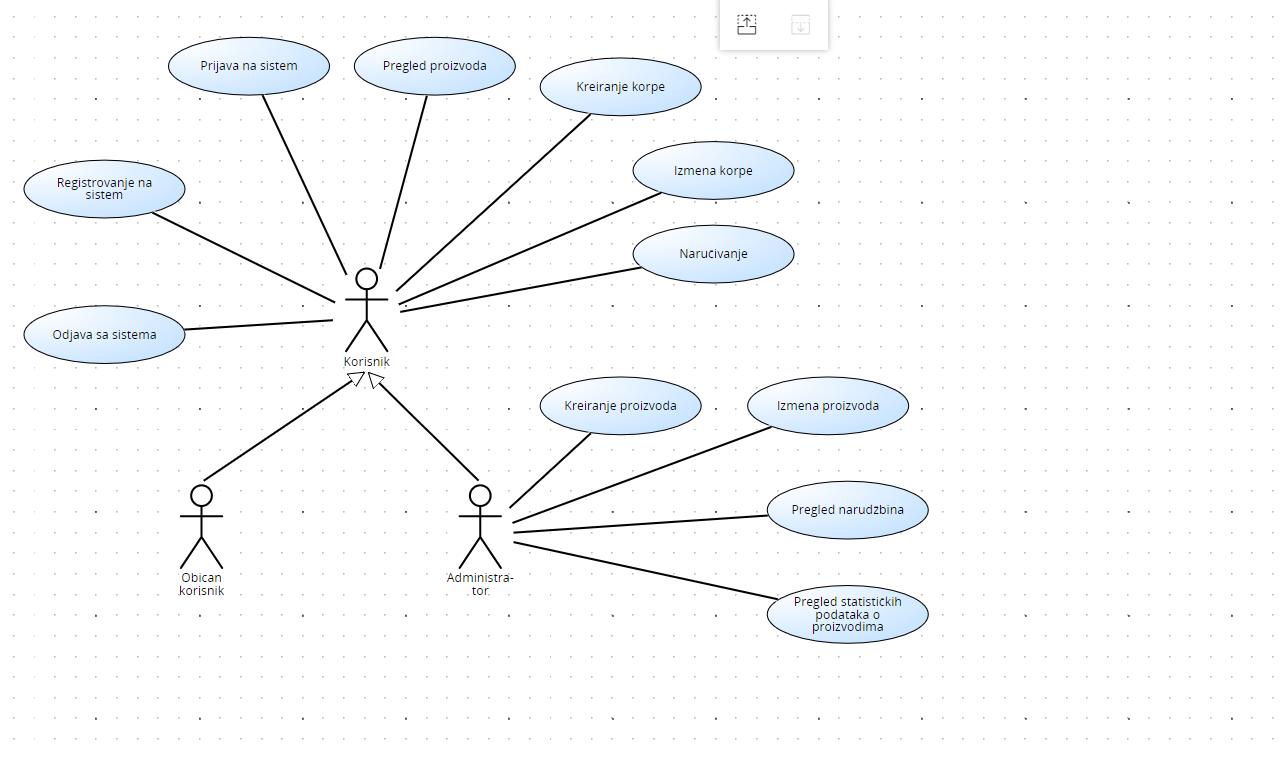
U sistemu postoji i specijalan korisnik, administrator. On ima sve mogućnosti kao i običan korisnik, s tim što može da prati sve porudzbine. Takođe, on može i kreirati i menjati podatke o proizvodima.

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

# 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**

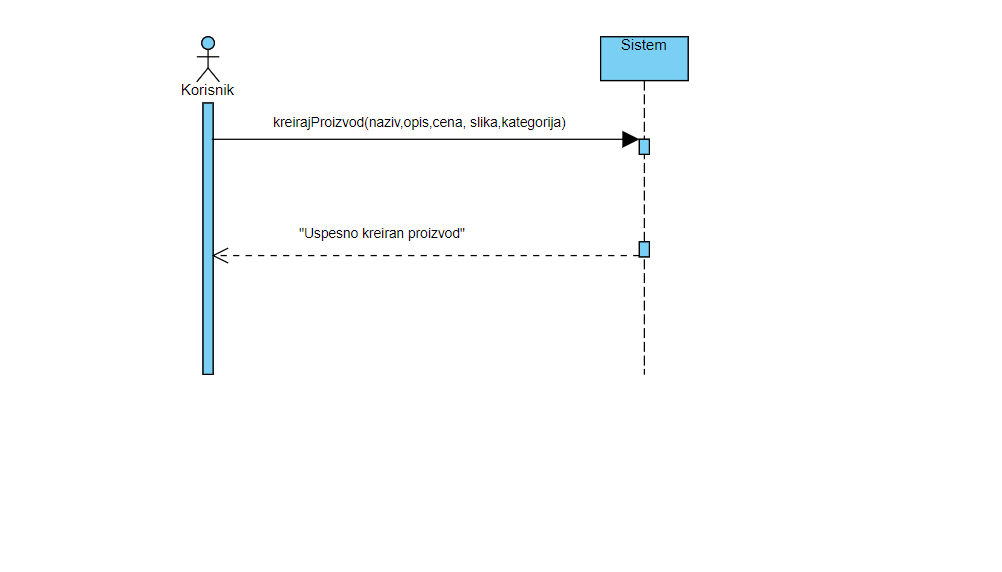
1. Korisnik unosi telefonski broj i adresu. (APUSO)
2. Korisnik poziva sistem da sačuva korpu sa datim podacima. (APSO)
3. Sistem čuva korpu sa datim podacima. (SO)
4. Sistem čuva podatake za svaku stavku korpe.(SO)
5. Sistem prikazuje korsiniku poruku: „Sistem je usepšno sačuvao korpu“. (IA)

Alternativna scenarija

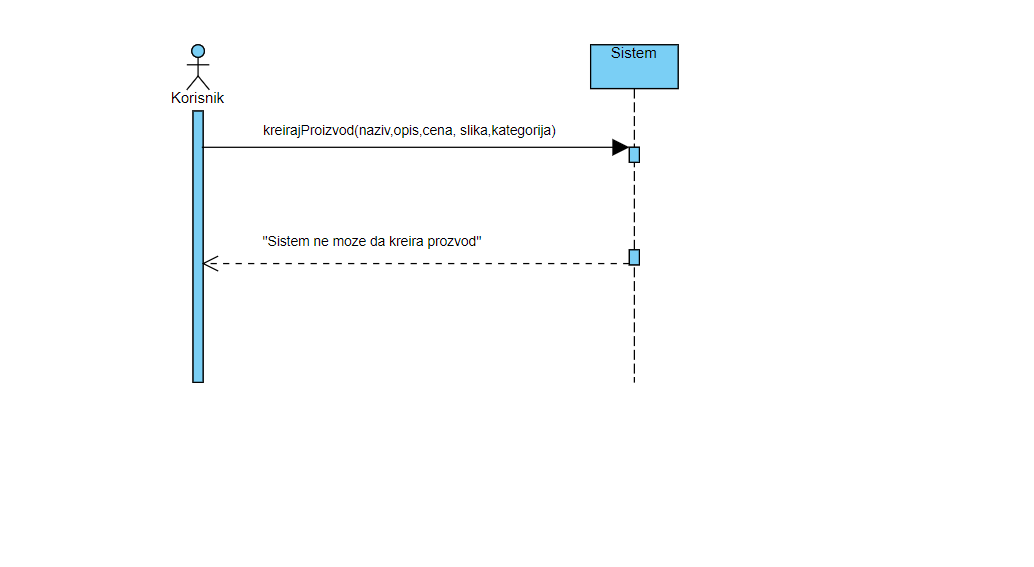
* 1. Ukoliko sistem ne može da sačuva korpu on prikazuje korsiniku objavu: „Sistem ne može da sačuva korpu“.(IA) prekida se izvršenje SK
  2. Ukoliko sistem ne može da sačuva neku stavku on briše sve podatke o prethodno sačuvanim stavkama i korpi i prikazuje korisniku poruku: „Sistem ne može da sačuva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

# Opis procesa – dijagrami sekvenci

## Kreiranje proizvoda

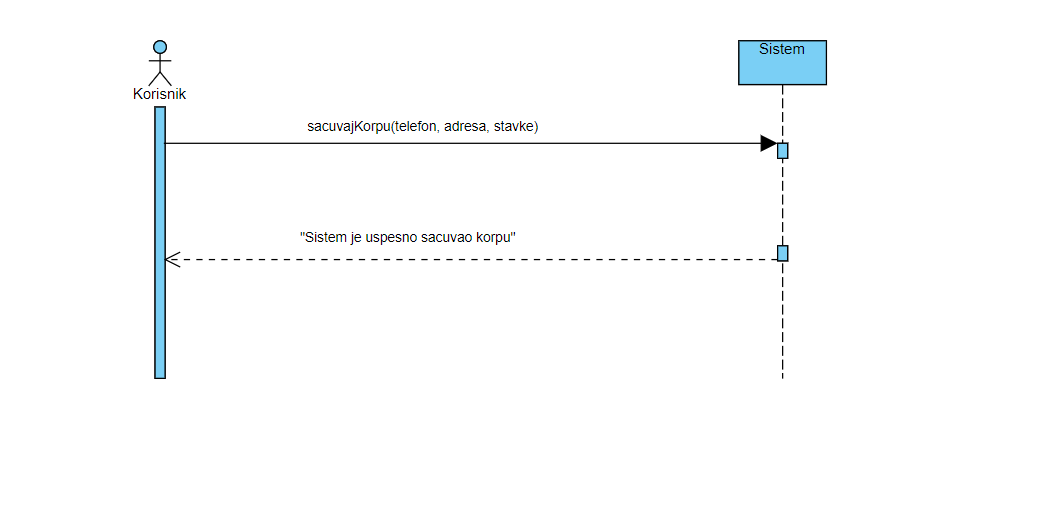


Slika 2

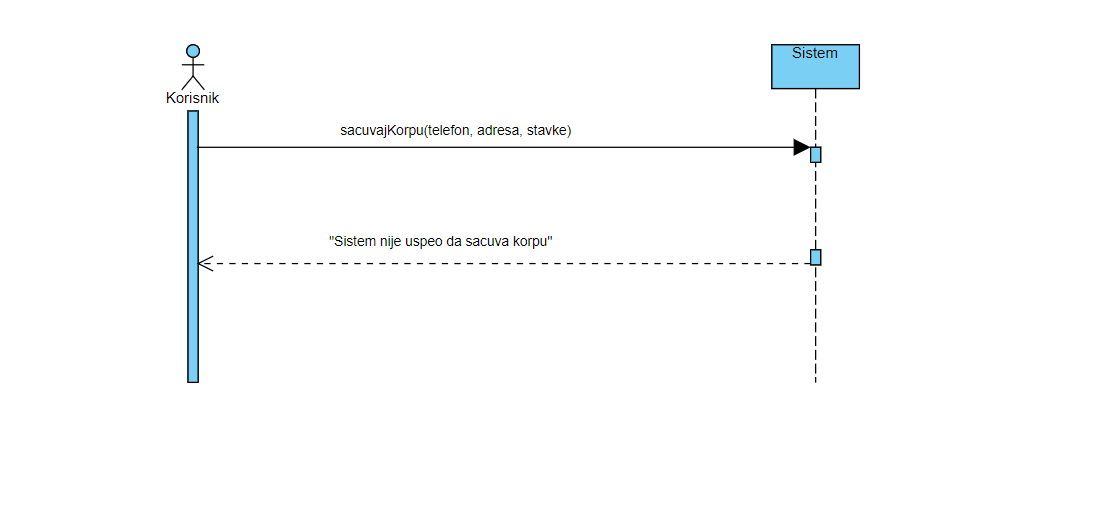


Slika 3

## Čuvanje korpe

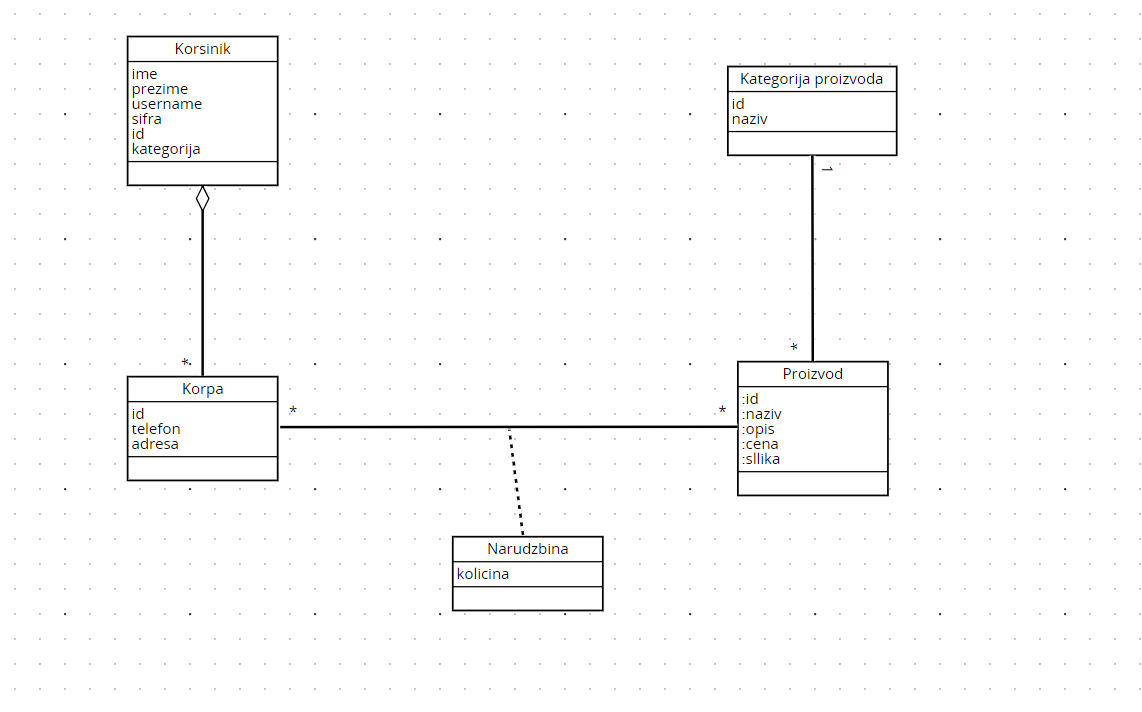


Slika 4



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 |

## 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.

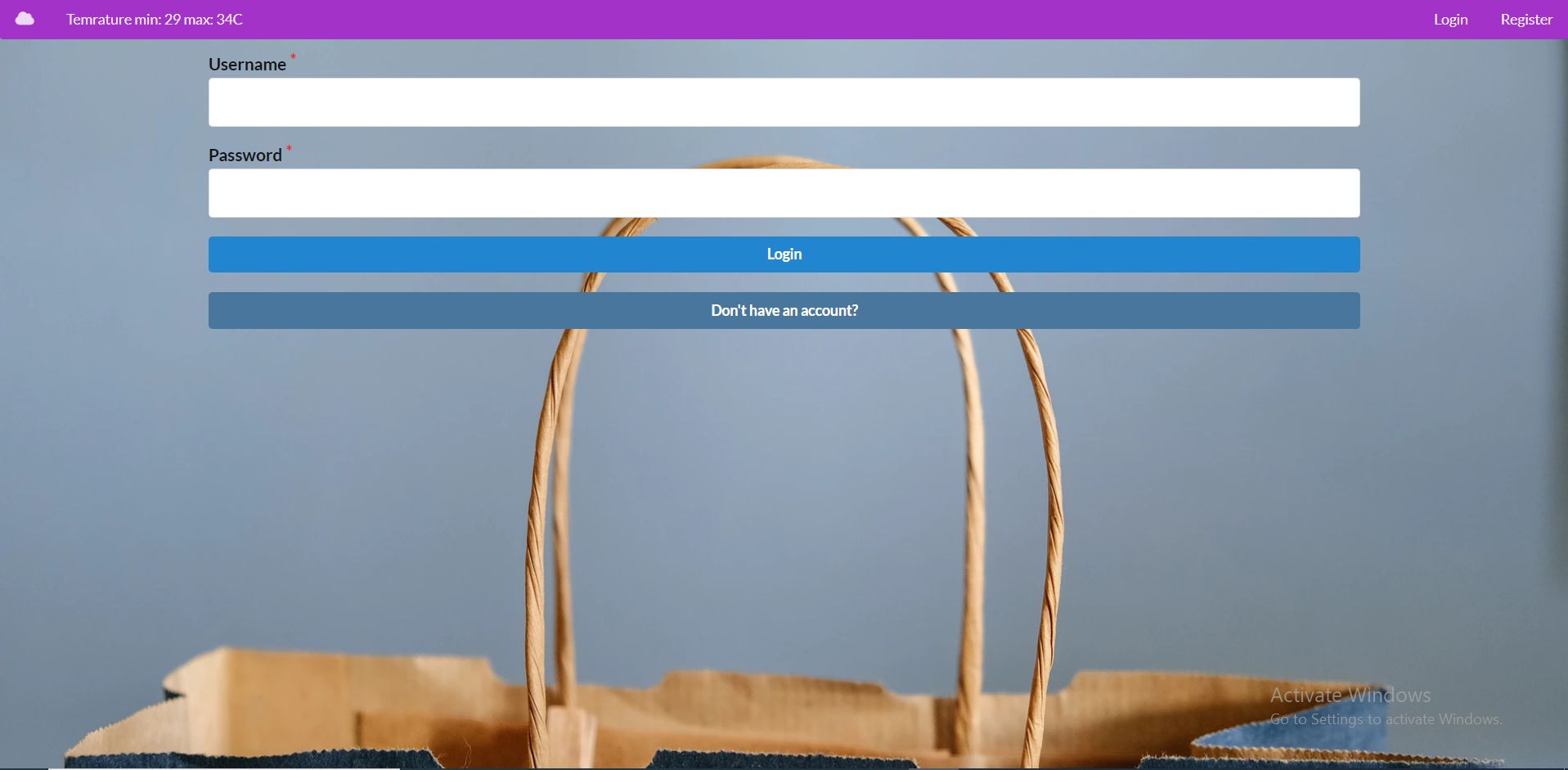
## TypeORM

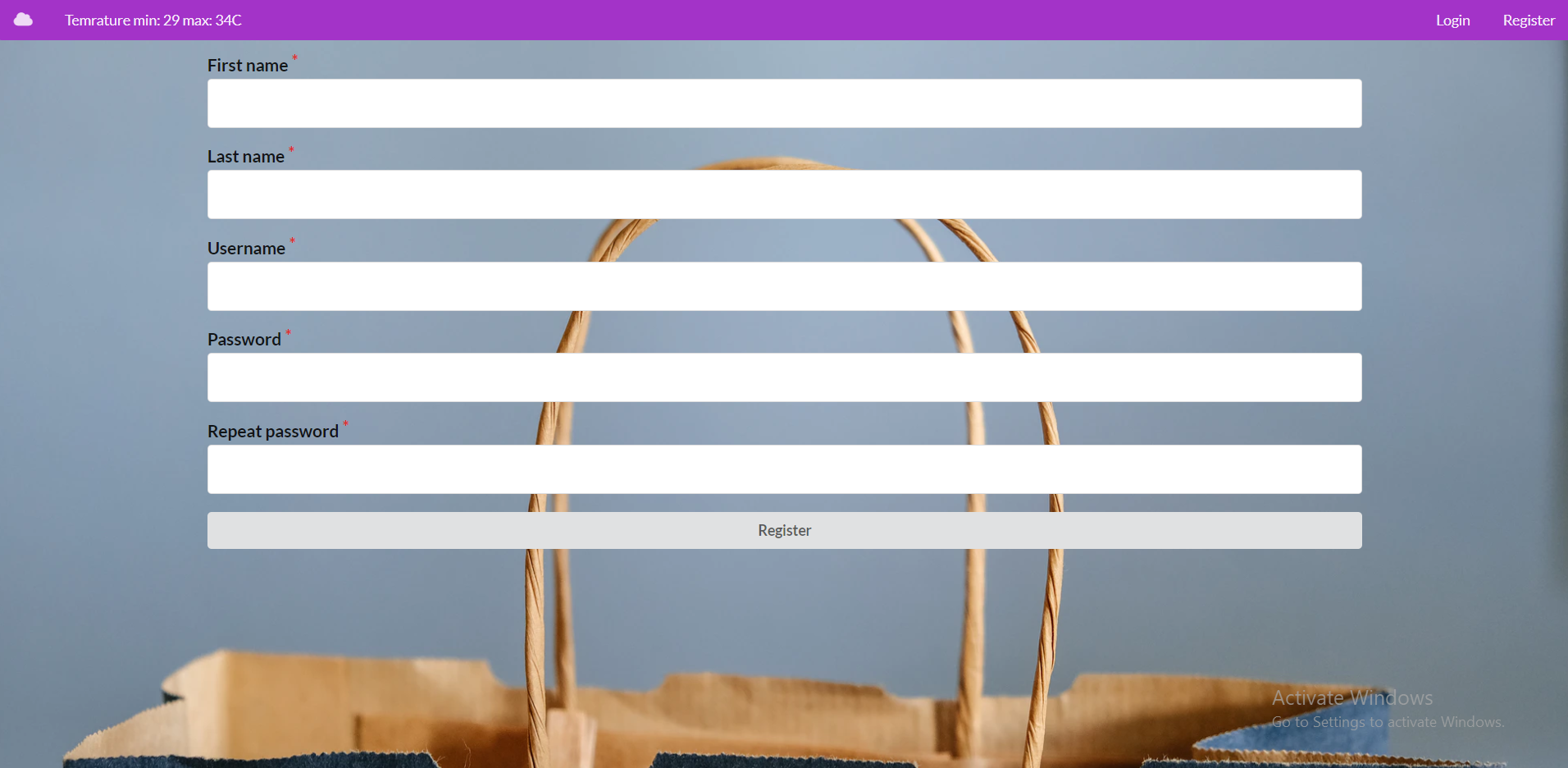
Objektno – relacion maper u javascript – u. Pravila prevođenja vrše se anotacijom klasa u programskom kodu. Sva konfiguracija nalazi se fajlu ormconfig.json. TypeORM omogućava rad sa bazom na 2 načina:

* 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.

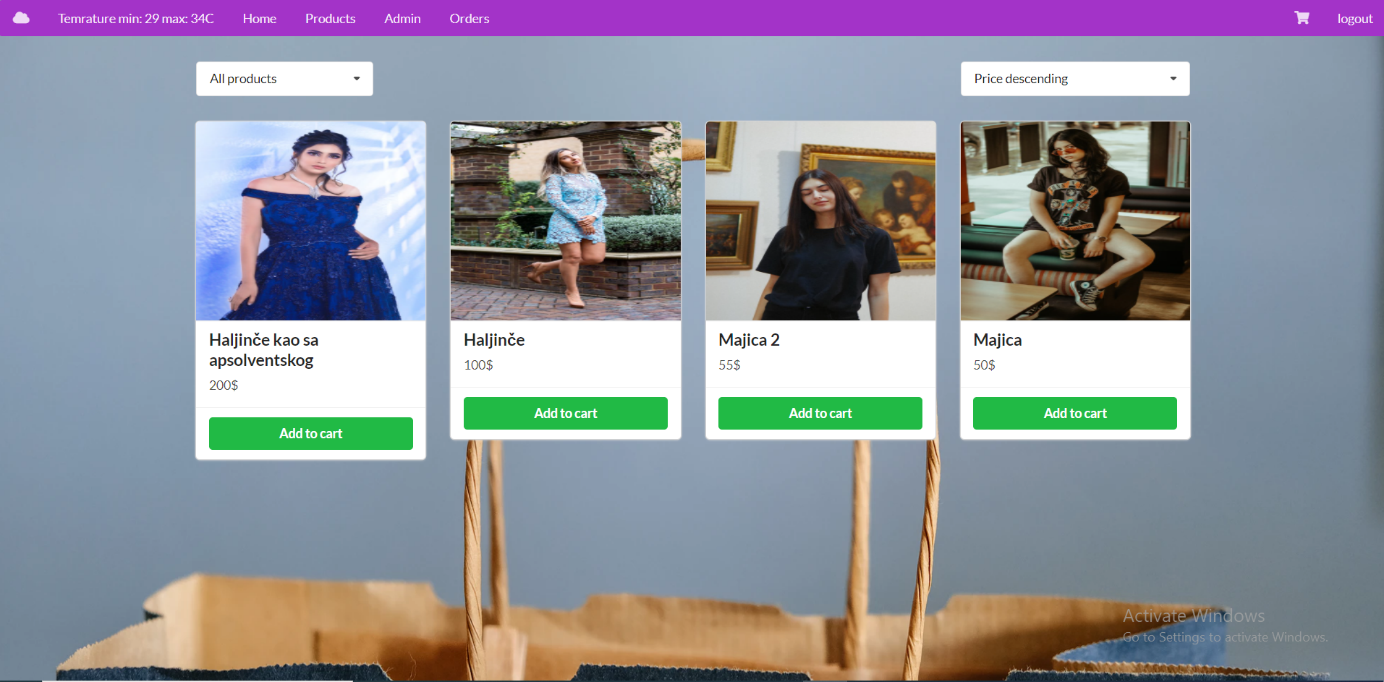
# Korisničko uputstvo

**Prijava na sistem**

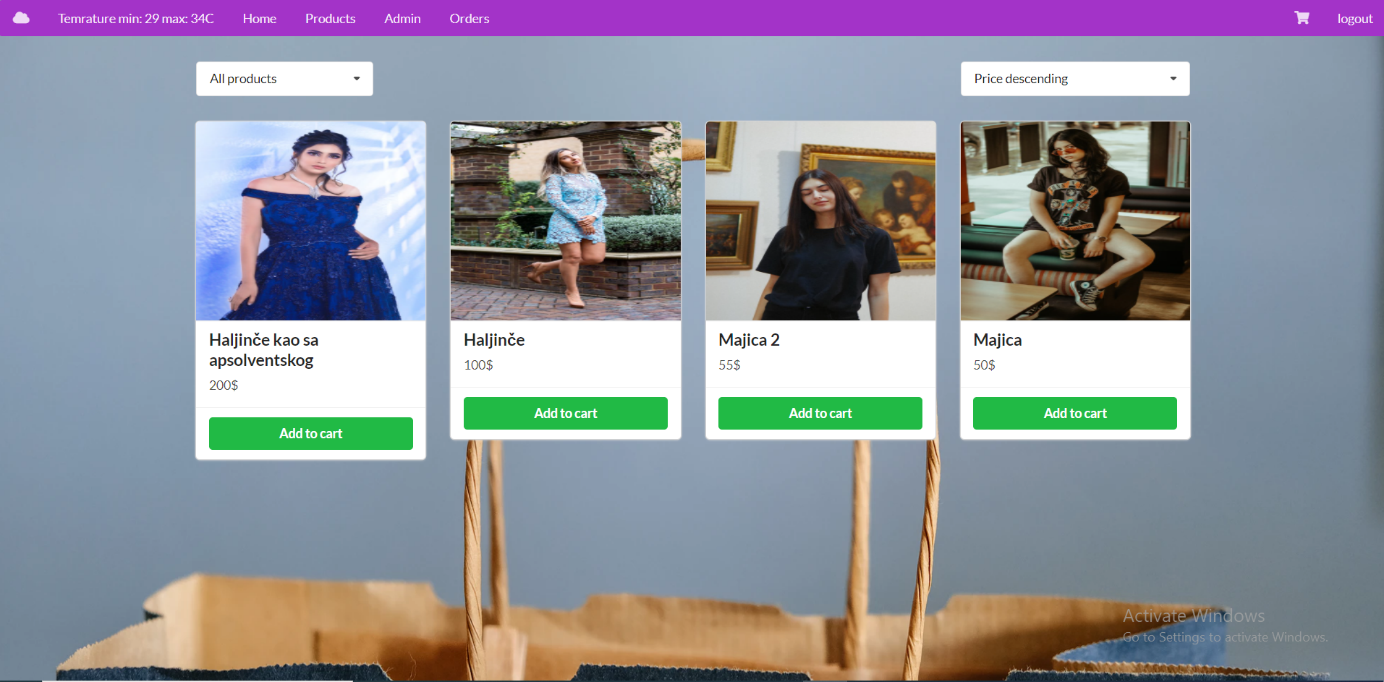
**Registrovanje na sistem**

**Odjava sa sistema**

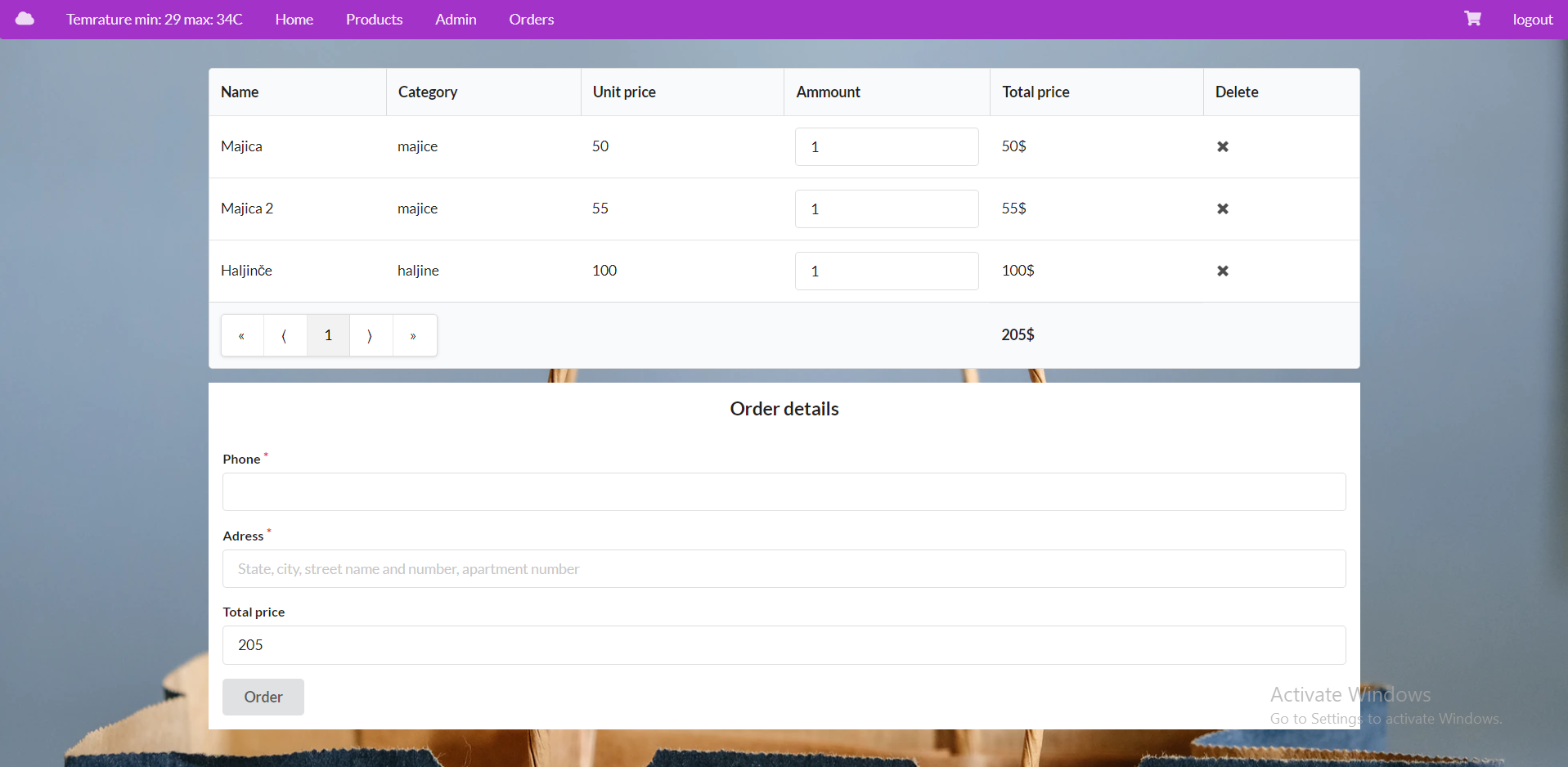


**Pregled proizvoda**

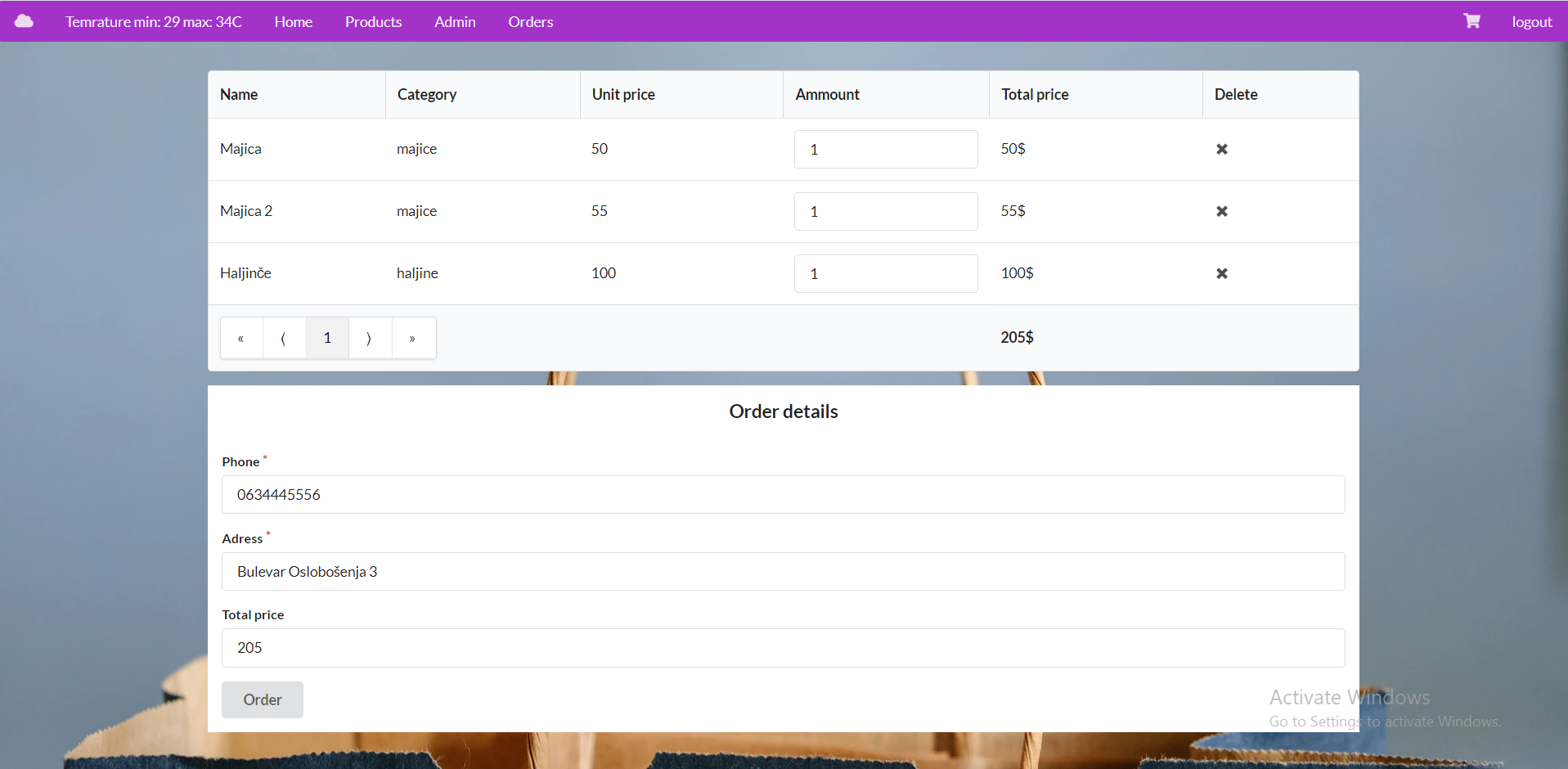
**Kreiranje korpe**

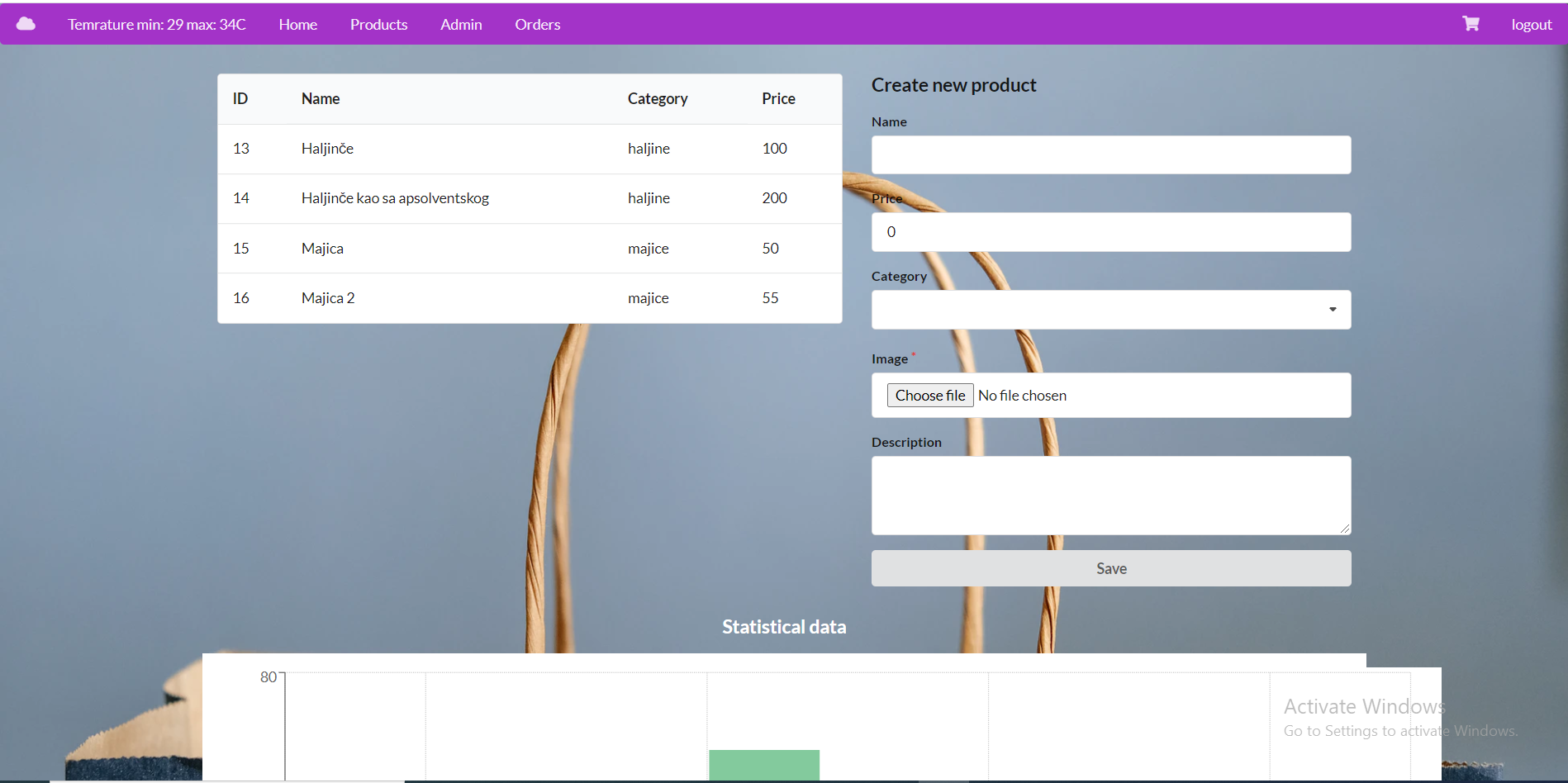


**Izmena korpe**

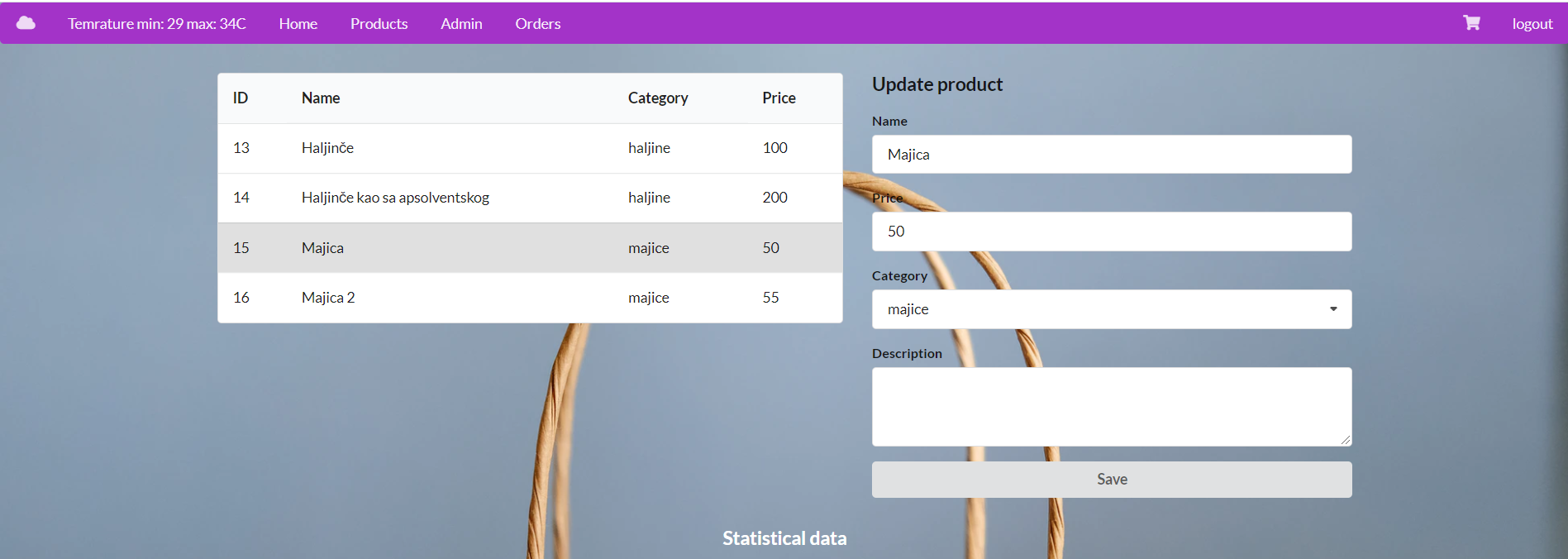


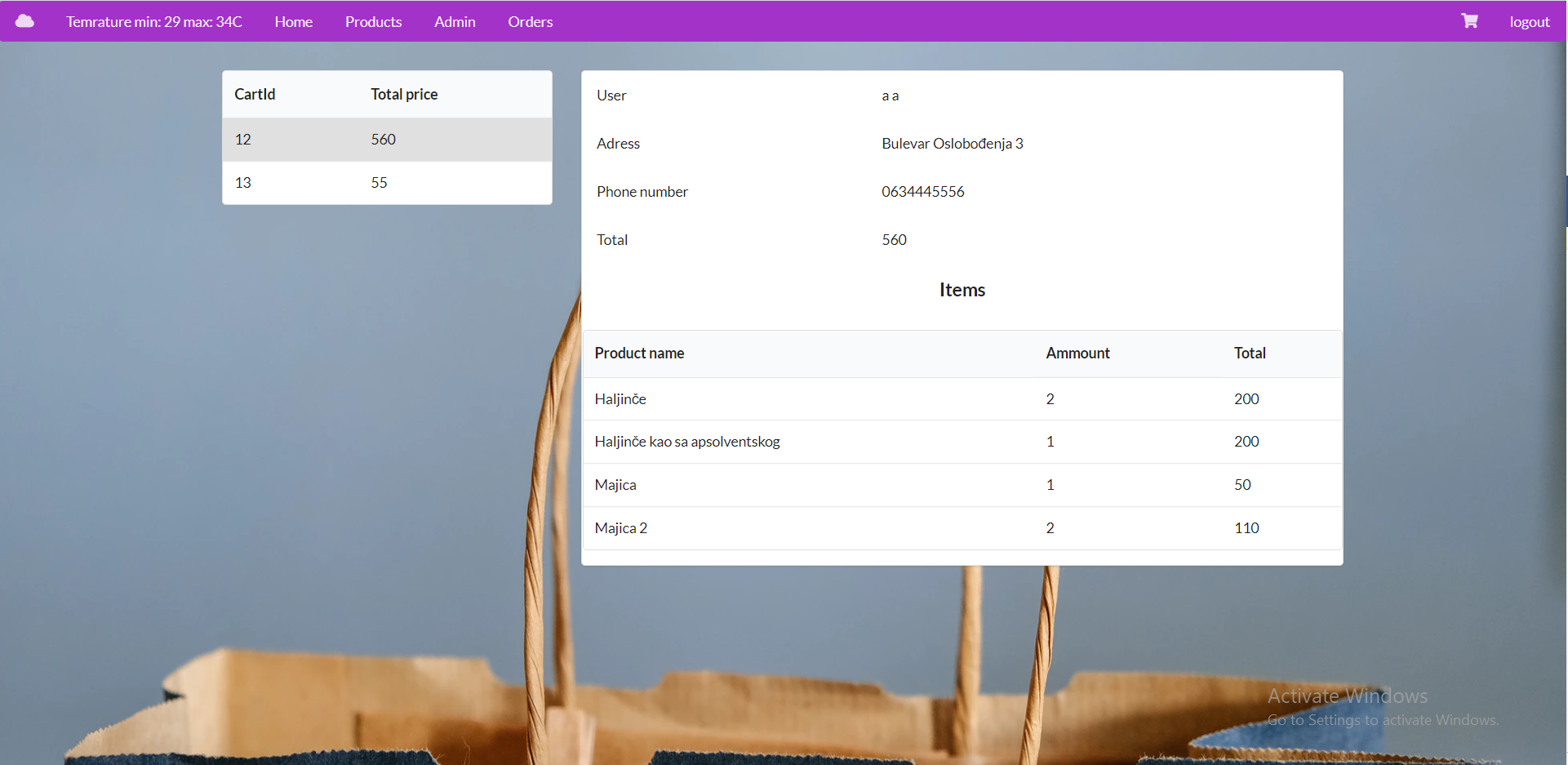
**Naručivanje**

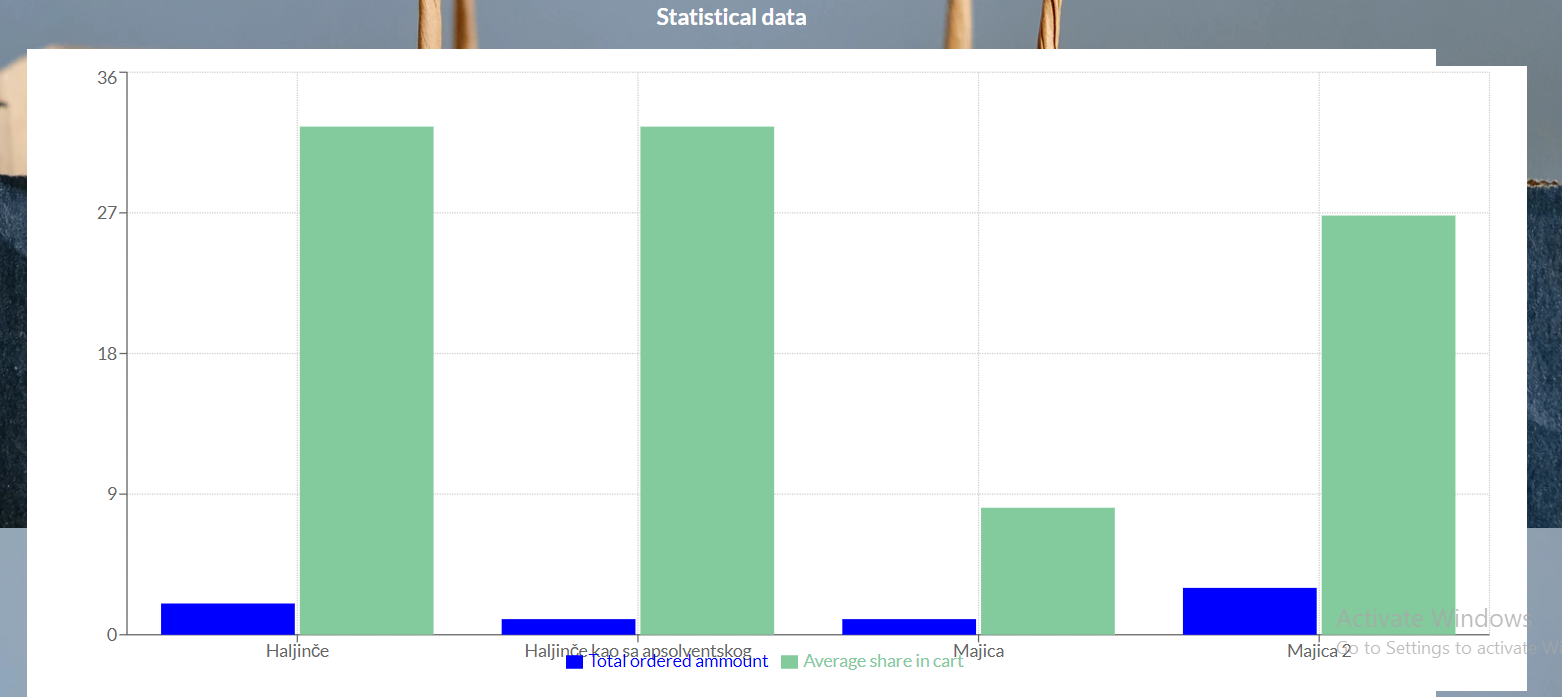


**Kreiranje proizvoda**

**Izmena proizvoda**



**Pregled narudžbina**

**Pregled statističkih podataka o proizvodima**

# 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 repozitorijumu

https://github.com/katana29/eprodavnica.git