Razvoj web aplikacije za internetsku trgovinu pomoću tehnologija Spring Boot, React i MongoDB

Krušvar, Dominik

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij računarstva

Završni rad

Razvoj web aplikacije za internetsku trgovinu pomoću tehnologija Spring Boot, React i MongoDB

Rijeka, rujan 2024.

Dominik Krušvar

0069093323

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Prijediplomski sveučilišni studij računarstva

Završni rad

Razvoj web aplikacije za internetsku trgovinu pomoću tehnologija Spring Boot, React i MongoDB

Mentor: Izv. prof. dr. sc. Marko Gulić

Rijeka, rujan 2024.

Dominik Krušvar

0069093323

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 21.03.2024.

Zavod: Zavod za računarstvo

Predmet: Razvoj web aplikacija

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Dominik Krušvar (0069093323)**

Studij: Sveučilišni prijediplomski studij računarstva (1035)

Zadatak: **Razvoj web aplikacije za internetsku trgovinu pomoću tehnologija Spring Boot, React i MongoDB / Development of a web application for web store using Spring Boot, React and MongoDB technologies**

Opis zadatka:

Razviti web aplikaciju za internetsku trgovinu. Aplikacija treba imati odvojeni administracijski dio i korisnički dio. Administrator mora imati mogućnost dodavanja proizvoda i informacija o njima kao i mogućnost upravljanje narudžbama. Također, administrator mora imati mogućnost analize prodajnih podataka pomoću grafikona. Registrirani korisnik mora imati mogućnost upisivanja svojih podataka kao i pretraživanje proizvoda s obzirom na odabrane filtere, te mogućnost naručivanja i kupnju odabranih proizvoda. Nadalje, treba implementirati razmjenu poruka između administratora i korisnika u realnom vremenu. Za razvoj poslužiteljskog dijela web aplikacije treba koristiti Spring Boot zajedno s MongoDB sustavom za upravljanje bazama podataka. Za razvoj klijentskog dijela aplikacije treba koristiti React knjižicu za razvoj korisničkog sučelja uz učinkovito renderiranje aplikacije na uređajima s različitim veličinama zaslona.

Rad mora biti napisan prema Uputama za pisanja diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 22.03.2024.

Predsjednik povjerenstva za
završni ispit:
prof. dr. sc. Miroslav Joler

Mentor:
doc. dr. sc. Marko Gulić

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio završni rad pod nazivom Razvoj web aplikacije za internetsku trgovinu pomoću tehnologija Spring Boot, React i MongoDB iz kolegija Razvoj web aplikacija uz mentorstvo izv. prof. dr. sc. Marka Gulića.

Rijeka, rujan 2024.

_ _ _ _ _ _ _ _ _

Dominik Krušvar

Zahvala

Zahvaljujem se profesoru Marku Guliću na mentorstvu i pomoći tijekom izrade završnog rada te obitelji na podršci tijekom studiranja.

SADRŽAJ

1 UVOD

U ovom radu predstavljena je web aplikacija za internetsku trgovinu. Aplikacija se temelji na REST arhitekturi. RESTful aplikacije koriste HTTP zahtjeve za dohvaćanje i obradu podataka te metode kao što su GET, POST, PATCH, DELETE za dohvaćanje, stvaranje, promjenu te brisanje sadržaja [1]. Aplikacija ima odvojen administracijski i korisnički dio. Administrator ima mogućnost samostalnog uređivanja stranice tako što može dodavati, uređivati i brisati proizvode. Također ima mogućnost odobravanja pristiglih narudžbi te pregled i analizu statistike prodaje proizvoda uz pomoć nekoliko grafikona. Korisnik ima mogućnost registracije te uređivanja osobnih podataka koji se spremaju za daljnju kupnju. Nakon prve registracije korisnik može kupovati proizvode koji su dostupni u prodaji. Prilikom kupovine može se služiti sa nekoliko filtera kako bi lakše pronašao željeni proizvod. U svakom trenutku može pogledati svoju trenutnu košaricu te cijelu svoju povijest narudžbi. Ukoliko je korisniku potrebna pomoć pri kupnji može se obratiti administratoru te razmjenjivati poruke s njim u realnom vremenu. Sve ove mogućnosti su razvijene pomoću poznatih i modernih tehnologija u ovom području računarstva.

Za razvoj poslužiteljskog dijela aplikacije korišten je Spring Boot. To je moderan i moćan okvir za razvoj aplikacija koji omogućuje jednostavnije postavljanje i pokretanje web aplikacije u samo nekoliko koraka [2]. Za upravljanje bazom podataka korišten je MongoDB sustav [3]. To je dokumentno orijentirana baza podataka koja sprema podatke u oblaku. Spring Boot ima prilično dobru podršku za korištenje MongoDB sustava te se ove dvije tehnologije dobro usklađuju i tako omogućuju izradu modernih, brzo rastućih web aplikacija. Za razvoj klijentskog dijela aplikacije korišten je React. To je knjižnica koja olakšava razvoj interaktivnih web aplikacija podjelom aplikacije na više komponenti koje se kasnije međusobno povezuju i nadopunjuju. Uz React je korišten TypeScript koji zahtjeva preciznije definiranje tipova varijabli kako bi se kod lakše održavao i kako bi se lakše mogle otkriti pogreške [4]. Za stiliziranje komponenti korišten je Material UI koji nudi već gotove primjere komponenti koje se kasnije dodatno mogu prilagođavati korisničkim željama. Za povezivanje i komunikaciju između poslužiteljskog i klijentskog dijela aplikacije korišten je Axios. To je knjižnica kompatibilna s Reactom koja olakšava slanje HTTP zahtjeva te obradu odgovora sa poslužiteljske strane [5].

Ovaj rad se sastoji od nekoliko poglavlja. U drugom poglavlju detaljnije su opisane tehnologije korištene za izradu projekta spomenute u prošlom odlomku. U trećem poglavlju detaljnije su opisane funkcionalnosti i mogućnosti cjelokupne aplikacije iz perspektive korisnika i iz perspektive administratora. U četvrtom poglavlju detaljno je opisana i objašnjena funkcionalnost komunikacije administratora i korisnika u realnom vremenu kroz primjere iz programskog koda same aplikacije. U posljednjem poglavlju iznesen je zaključak.

2 TEHNOLOGIJE

2.1 Spring Boot

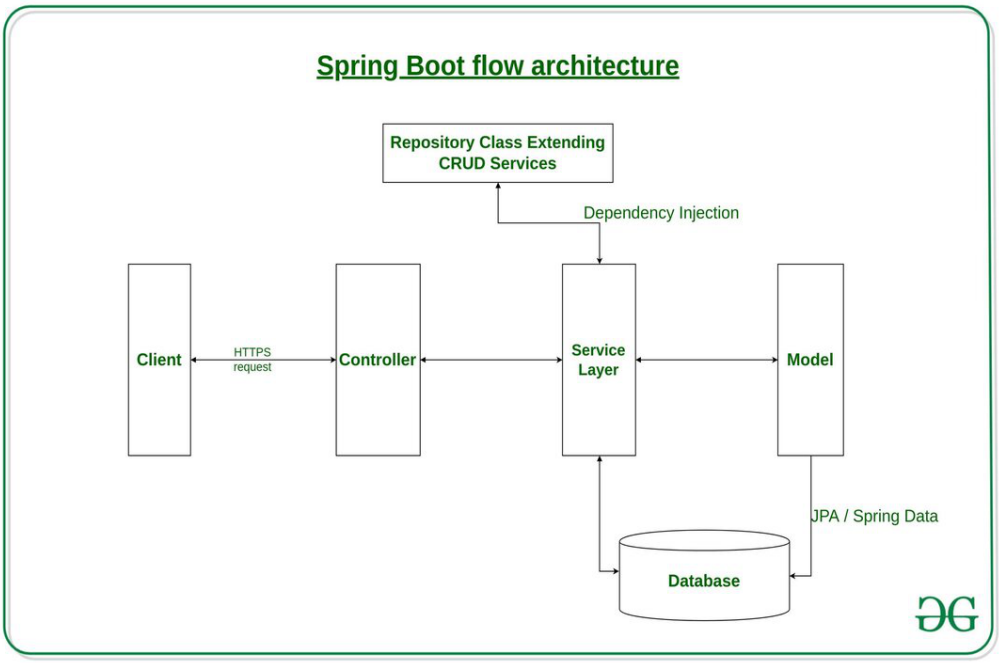
Spring Boot je radni okvir otvorenog koda baziran na Javi. Zahvaljujući jednostavnoj i brzoj mogućnosti inicijalizacije cijelog sustava postao je veoma popularan u svijetu web aplikacija. Prvi puta je predstavljen 2014. godine s ciljem ubrzanja i olakšanja dosadašnjeg načina konfiguracije aplikacija baziranih na Javi. Prilično brzo je prihvaćen jer je donio mnogo benefita od kojih je glavni brzina konfiguracije te tako osigurao programerima više vremena za rad na samoj logici aplikacije, a manje za postavljanje sustava [2].

2.1.1 Karakteristike Spring Boota

Spring Boot koristi slojevitu arhitekturu u kojoj svaki sloj može komunicirati sa ostalim slojevima iznad ili ispod sebe. Ovakva arhitektura doprinosi boljoj modularnosti te lakšem održavanju i testiranju aplikacije. Svaki sloj ima specifične uloge i koristi različite tehnologije dostupne unutar Spring Boota kako bi se osigurala skalabilnost aplikacije. Sastoji se od 4 sloja, a to su prezentacijski (eng. presentation), poslovni (eng. business), konzistentni (eng. persistance) te sloj baze podataka (eng. database). Prezentacijski sloj sastoji se od pogleda (eng. view) odnosno predstavlja sloj zadužen za prikaz podataka koje će aplikacija koristiti. Njegove zadaće su procesuiranje HTTP zahtjeva te provjera autentičnosti poslanih zahtjeva. Ujedno i pretvara JSON (eng. JavaScript Object Notation) objekte u Java objekte odnosno podatke primljene sa klijentske strane aplikacije pretvara u podatke razumljive poslužiteljskoj strani aplikacije. Sljedeći sloj je poslovni i njegove zadaće su implementacija poslovnih pravila i logike koju aplikacija pruža, provjera ispravnosti podataka te autorizacija odnosno upravljanje pravima pristupa određenim funkcionalnostima i resursima. Sljedeći sloj je konzistentni te su njegove zadaće upravljanje logikom spremanja podataka odnosno pristup bazi podataka te mapiranje objekata u tip koji se koristi u bazi podataka. Posljednji sloj je sloj baze podataka i predstavlja stvarnu bazu podataka u koju se spremaju potrebni podaci.

Ovakva arhitektura omogućuje jednostavnu i jasnu definiciju protoka podataka u Spring Bootu. Prvi korak je prihvaćanje HTTP zahtjeva koji klijentska strana aplikacije pošalje. Taj zahtjev se nakon primitka prosljeđuje u Kontroler (eng. Controller) koji ga

mapira na odgovarajuću metodu te prosljeđuje zaduženom servisu. Servis (eng. Service) sadrži poslovnu logiku aplikacije, odnosno zadužen je za validaciju i transformaciju podataka . Sljedeći korak je pristup bazi podataka koji je moguć preko perzistentnog sloja. Nakon što se obave potrebne operacije s bazom podataka, podaci ponovno dolaze do Kontrolera koji vraća korisniku odgovor u obliku potvrde, greške ili konkretnih podataka. Na Slici 2.1 prikazan je proces protoka podataka opisan iznad.

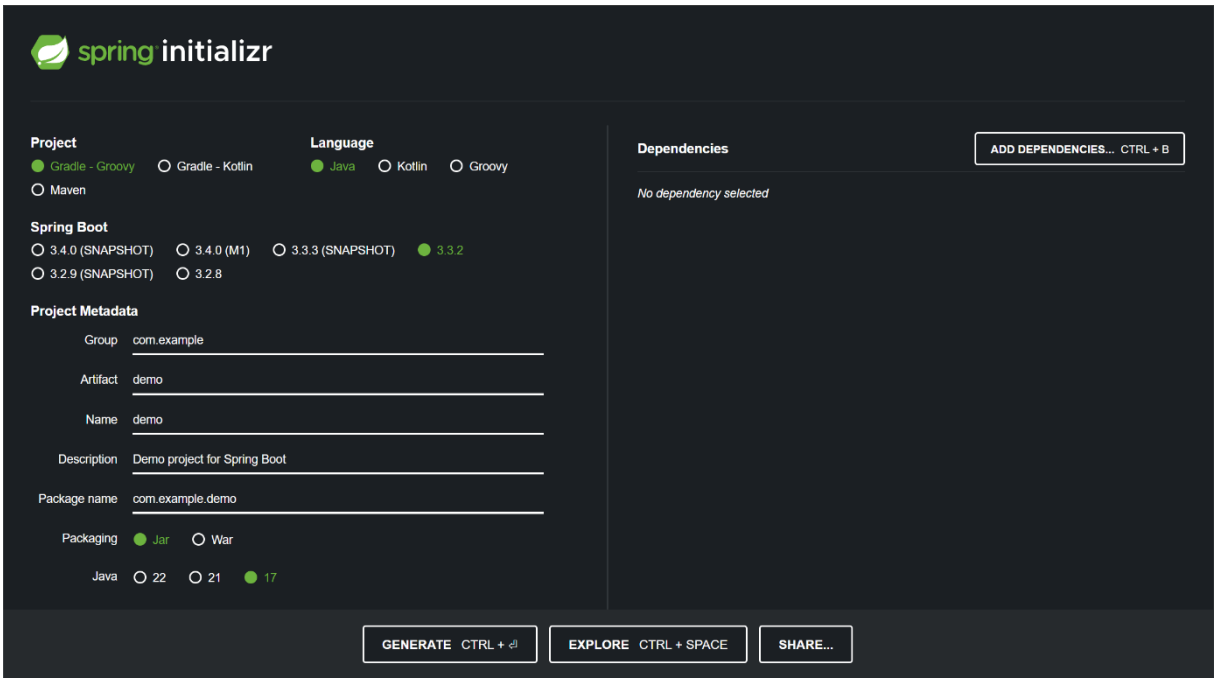


Slika 2.1: Prikaz toka podataka u Spring Bootu [6]

2.1.2 Inicijalizacija projekta

Brzina i lakoća inicijalizacije i konfiguracije projekta bitna je prednost Spring Boota u odnosu na slične konkurentske tehnologije. Spring Boot tako nudi alat pod nazivom *Spring Initializr* koji je dostupan na webu, u komandnoj liniji te u nekoliko integriranih razvojnih okruženja kao što su Intellij IDEA, Eclipse te Visual Studio Code. Sve verzije alata pružaju iste mogućnosti te je tako za ovaj projekt korištena inicijalizacija putem web stranice. Nakon dolaska na stranicu potrebno je odabrati vrstu projekta, odnosno vrstu alata koji će se koristiti za izgradnju, implementaciju i podizanje projekta. Odabir je moguć između alata Maven i Gradle, a u ovom projektu je korišten Maven. Uz to odabran je i programski jezik Java kao što je već ranije spomenuto. Sljedeći korak je odabir verzije Spring Boota. Najsigurniji odabir je posljednja stabilna verzija kako bi se

osigurala podrška za sve potrebne funkcionalnosti. Nakon ovog odabira potrebno je ispuniti nekoliko podataka o projektu. Potrebno je definirati grupu, artefakt, ime, opis projekta te ime paketa u kojem će se projekt nalaziti. Sljedeća stavka je dodavanje zavisnosti (eng. dependencies) koje su potrebne u projektu. Zavisnosti se uvijek mogu i kasnije, ručno dodavati u projekt, no *Spring Initializr* nudi ovu mogućnost i ovdje kako bi nakon kreiranja projekta programer odmah mogao započeti s radom. Neke od mogućih zavisnosti koje su ujedno i korištene u ovom projektu su *Spring Web* koji omogućuje izradu web aplikacija temeljenih na REST arhitekturi, *Spring Security* koji omogućuje provjeru autentičnosti korisnika, *Spring Data MongoDB* koji omogućuje povezivanje sa MongoDB bazom podataka te *Lombok* koji olakšava i automatizira pisanje koda. Nakon izvršenja svih ovih odabira potrebno je još samo generirati *zip* datoteku koja sadrži cijelu konfiguraciju te ju je moguće preuzeti i jednostavno pokrenuti u odabranom integriranom razvojnom okruženju. Na Slici 2.2 prikazan je početni prozor web aplikacije *Spring Initializr* opisan u ovom odlomku [7].



Slika 2.2: Početni prozor Spring Initializra [7]

2.1.3 Spring Security

Spring Security je iznimno moćan i fleksibilan okvir koji pruža mogućnost osiguranja aplikacija i kontrolu pristupa u Java aplikacijama. Glavne prednosti, kao i kod samog Spring Boota, su lakoća konfiguracije te velika mogućnost različitih odabira i

načina autentifikacije i autorizacije korisnika [8]. Spring Security je moguće dodati u projekt putem ranije spomenutih zavisnosti. Glavni korak je kreiranje konfiguracijske klase u kojoj je potrebno podesiti adrese koje smiju pristupati aplikaciji, načine autentifikacije korisnika, kao što je JWT (eng. JSON Web Token), te prava pristupa autoriziranih korisnika. JWT je obliku dugačkog niza nasumičnih slova i brojki te predstavlja način provjere autentičnosti korisnika. Prilikom slanja određenog zahtjeva poslužiteljskom djelu aplikacije korisnik s klijentskog dijela dobiva svoj JWT koji poslužiteljski dio kreira. Taj JWT će vrijediti neki određeni period vremena te će za to vrijeme korisnik biti autoriziran i moći pristupati resursima uz slanje tog koda u zaglavlju svakog zahtjeva. Nakon isteka tog perioda, korisnik mora ponovno dobiti novi JWT od poslužiteljskog dijela za nastavak komunikacije. Provjera prava pristupa moguća je uz oznaku *@PreAuthorize* koja se dodaje prije određenog resursa. Unutar ove oznake moguće je provjeravati ima li korisnik ulogu koja je potrebna za pristup traženim resursima. Spring Security ima ugrađene metode zaštite od uobičajenih ranjivosti kao što su CSRF (eng. Cross-Site Request Forgery) napadi, fiksacija sesije (eng. Session Fixation) te omogućuje podršku za komunikaciju putem HTTPS protokola.

2.1.4 Spring Data MongoDB

Spring Data MongoDB omogućuje lakše povezivanje i komunikaciju sa MongoDB bazom podataka. Nudi repozitorij *MongoRepository* koji sadrži već integrirane metode osnovnih operacije kao što su dodavanje, uređivanje i brisanje podataka iz baze podataka. Za zahtjevnije pretraživanje baze podataka dostupna je i anotacija *@Query* koja omogućuje pisanje punih upita za bazu podataka. Nakon dodavanja ranije spomenute zavisnosti potrebna je konfiguracija Spring Boota kako bi mogao komunicirati sa bazom podataka. Ta konfiguracija se odvija u *application.properties* datoteci koja je sastavni dio svakog projekta i prikazana je na Slici 2.3. Potrebno je unijeti naziv baze podataka te poveznicu za konekciju koju je moguće pronaći unutar MongoDB sustava. Podaci koji se nalaze unutar vitičastih zagrada smatraju se tajnima stoga su smješteni unutar *.env* datoteke koja se ne objavljuje prilikom dodavanja koda na platformu poput GitHuba.

```
1  spring.data.mongodb.database=${mongo_database}
2  spring.data.mongodb.uri=mongodb+srv://${mongo_user}:${mongo_password}@${mongo_cluster}
3
```

Slika 2.3: Application.properties datoteka

2.2 React

React je JavaScript knjižnica otvorenog koda namijenjena izradi korisničkih sučelja aplikacija. Prvi puta je predstavljen 2013. godine od strane tvrtke Facebook Inc. Trenutno ga održava tvrtka Meta uz pomoć individualnih programera i kompanija. React je iznimno popularna tehnologija za izradu dinamičkih web stranica i koriste ga milijuni programera svakog dana. Karakteristike poput virtualnog DOM-a, komponentnog pristupa, jednosmjernog toka podataka i deklarativnog tipa programiranja postavile su novi standard izrade modernih web aplikacija [4].

2.2.1 Postavljanje projekta

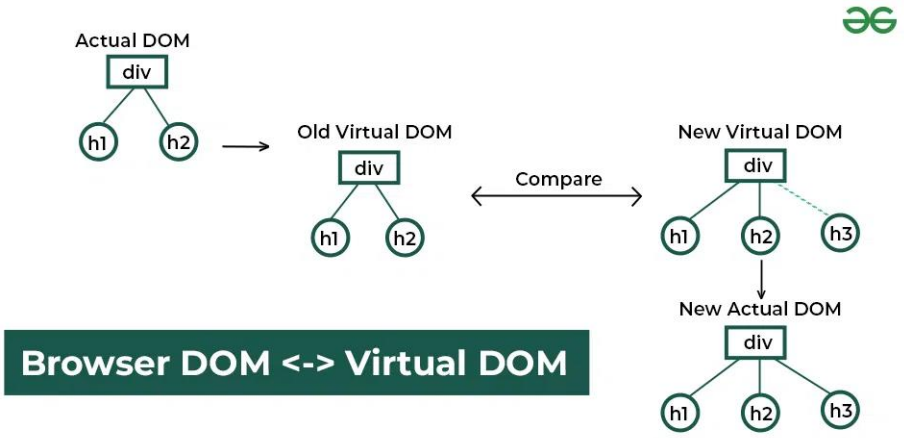
Za početak razvijanja React aplikacije potrebno je postaviti okruženje. Prvi korak je instalacija Node.js okoline koja omogućuje izvršavanje JavaScript koda izvan web preglednika. Uz Node.js potreban je i npm (eng. Node Packet Manager) koji služi kao upravitelj paketa te omogućuje instaliranje potrebnih knjižnica izrađenih od strane ostalih programera. Odnosno olakšava i smanjuje potrebu za pisanjem vlastitih programskih rješenja za svaki mogući problem. Nakon preuzimanja ovih alata potrebno je u terminalu izvršiti naredbu *npx create-react-app naziv_projekta* za stvaranje novog projekta.

2.2.2 Komponente

Jedna od najvažnijih karakteristika Reacta je njegov komponentni pristup. Komponente predstavljaju osnovnu građevnu jedinicu i ključne su za organizaciju i strukturu svake React aplikacije. Svaka komponenta predstavlja određeni dio korisničkog sučelja ili funkcionalnosti te sadrži svoju logiku i stanje. Komponente mogu biti jednostavne poput tipke ili polja za prikaz teksta, ali i složene poput navigacijske trake ili obrasca za unos podataka. Komponentni pristup omogućuje razvoj modularnog koda koji se može jednostavnije održavati i ponovno upotrebljavati. Komponente se mogu koristiti bilo gdje u aplikaciji te međusobno ugnijezditi kako bi stvorile kompleksniju funkcionalnost. Mogu primati podatke ili parametre putem svojih svojstava *props* što omogućuje komunikaciju između roditeljskih (eng. parent) i dječjih (eng. child) komponenata.

2.2.3 Virtualni objektni model dokumenta

Virutalni objektni model dokumenta (eng. Virtual Document Object Model, VDOM) jedna je od novosti koje je React uveo u svijet izrade korisničkih sučelja i pomaže u optimizaciji i efikasnosti ažuriranja korisničkog sučelja. Tradicionalni DOM (eng. Document Object Model) predstavlja strukturu HTML dokumenta kao stablo objekata. Svaki element web stranice je čvor u tome stablu stoga je manipulacija takvom strukturom prilično spora i zahtjeva puno resursa. Virtualni DOM predstavlja kopiju pravog DOM-a koja se nalazi u memoriji. Svaka promjena koja se napravi na stranici ažurira se najprije u virtualnom DOM-u. Nakon toga React uspoređuje staru verziju virtualnog DOM-a sa ažuriranom te pronalazi minimalan skup promjena potrebnih da bi se stvarni DOM sinkronizirao sa novim virtualnim DOM-om. React tako ažurira samo potrebne elemente stranice u kojima se dogodila promjena, a ne cijelu stranicu. Na Slici 2.4 prikazano je funkcioniranje Virtualnom DOM-a te utjecaj virtualnog DOM-a na stvarni DOM [9].



Slika 2.4: Funkcioniranje Virtualnog DOM-a [9]

2.2.4 Jednosmjerno povezivanje podataka

Jednosmjerno povezivanje podataka u Reactu odnosi se na način na koji se podaci prenose iz roditeljskih u dječje komponente. Odnosno naziva se jednosmjernim jer podaci putuju iz roditeljske komponente prema djetetu, ali ne mogu putovati u obrnutom smjeru. Podaci se prosljeđuju putem svojstava (eng. props) te kada se neko stanje podataka promjeni, komponente koje koriste te podatke moraju se ponovno učitati kako bi prikazale ažurirane podatke. Prednosti ovakvog načina povezivanja podataka su lakše praćenje promjena stanja jer je uvijek poznato gdje informacije idu, jasna i jednostavna struktura

aplikacije te kontrola stanja od strane roditeljske komponente koja upravlja podacima koje šalje dječjoj [10].

2.2.5 React kuke

React kuke (eng. hooks) su značajno unaprijedile način funkcioniranja Reacta u odnosu na njegove početke. Omogućuju dijeljenje zahtjevnije komponente u manje funkcije bazirane po sadržaju te smanjuju potrebu za pisanjem kompliciranih klasa za upravljanje određenim komponentama. Najčešće korištene React kuke su *useState* i *useEffect* koje olakšavaju rad sa stanjima i efektima u funkcijskim komponentama [11].

Kuka *useState* omogućuje definiranje varijable stanja (eng. state variable) koja ostaje sačuvana i nakon završetka rada neke funkcije. Sastoji se varijable koja čuva određeno stanje te funkcije za ažuriranje tog stanja. Za deklariranje ovakve varijable potrebno je kuki poslati inicijalno stanje te varijablu koja može biti broj, riječ, nula ili neki objekt. Kasnije se vrijednost te varijable može jednostavno mijenjati pozivom na *setState()* funkciju. Na Slici 2.5 prikazan je ovdje opisan način definiranje varijable stanja. Podatak *count* predstavlja varijablu koja čuva stanje brojača, a *setCount* je funkcija za ažuriranje stanja toga brojača. Početno stanje zadano je u zagradi nakon *useState* i u ovom slučaju to je broj nula [12].

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"
  const [count, setCount] = useState(0);
```

Slika 2.5: Primjer korištenja kuke useState [12]

Kuka *useEffect* omogućuje dohvaćanje podataka iz vanjskog izvora, ažuriranje objektnog modela dokumenta (eng. DOM) te postavljanje brojača (eng. timer) za neku određenu akciju. Olakšava preciznu kontrolu kada i kako se neki efekt treba izvršiti pomoću zavisnosti te osigurava pravilno oslobađanje resursa putem povratnih funkcija. Kuka *useEffect* izvršava se nakon prvog učitavanja stranice te nakon svakog sljedećeg osvježavanje stranice. Moguće je dodati zavisnost, odnosno neku varijablu, te kada se stanje te varijable promjeni ponovno će se izvršiti *useEffect* kuka. Na Slici 2.6 prikazano je korištenje *useEffect* kuke. Prvi dio koda odnosi se na *useState* kuku iz prethodnog

odlomka. U nastavku je kuka *useEffect* koja pročita zadnju vrijednost varijable *count* te ažurira naslov dokumenta. Prilikom svakog osvježavanje stranice ova brojka će se ponovno pročitati i po potrebi promijeniti [13].

```
import React, { useState, useEffect } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });
```

Slika 2.6: Primjer korištenja kuke useEffect [13]

2.2.6 TypeScript

React je izvorno JavaScript knjižnica za izradu korisničkih sučelja. TypeScript predstavlja jezik koji prevodi (eng. compile) JavaScript kod i može se reći kako je TypeScript nadskup (eng. superset) JavaScripta jer nudi sve mogućnosti kao i JavaScript uz još neke svoje dodatke. TypeScript dodaje potrebu za deklaracijom tipa podatka koji se koristi što pomaže kod provjere i traženja potencijalnih grešaka u kodu. Također prevodilac programa (eng. compiler) provjerava i traži greške prije samog pokretanja aplikacije što nije slučaj kod JavaScripta. Nudi sučelja i tipove za definiranje strukture objekata koji se koriste, mogućnost modula za enkapsulaciju koda koji olakšavaju njegovo ponovno korištenje te moderne tipove podataka kao što su unije, presjeci i mape. Potencijalne loše strane su potreba za učenjem dodatnog jezika pošto mu se sintaksa razlikuje od JavaScripta te nedostupnost i nekompatibilnost nekih dodatnih knjižnica. U ovom projektu je korišten Typescript uz React. Za pokrenuti ovakvu vrstu React projekta potrebno je u terminalu upisati naredbu *npx create-react-app naziv_projekta --template typescript*. TypeScript se uvijek može i naknadno dodati u već postojeći projekt [14].

2.2.7 Recharts

Recharts je knjižnica za vizualizaciju podataka u React aplikacijama koja omogućuje kreiranje različitih tipova grafova i dijagrama. Koristi React komponente za kreiranje grafova te omogućuje deklarativan pristup, odnosno potrebno je samo unijeti

podatke, a Recharts će se pobrinuti za izgled grafa. Pruža širok raspon grafova kao što su linijski, stupčasti, tortni, radijalni te mnogo drugi. Komponente grafova je moguće stilizirati i prilagođavati potrebama aplikacije te im dodavati animacije. Recharts je moguće dodati u projekt pomoću naredbe *npm install recharts*. Veoma je intuitivan za korištenje te je u ovom projektu korišteno nekoliko grafova koji će biti prikazani u trećem poglavlju [15].

2.3 MongoDB

MongoDB je dokumentna baza podataka namijenjena brzo rastućim i skalabilnim aplikacijama. Predstavljena je 2007. godine i trenutno je veoma popularna u svijetu web aplikacija baš zbog svoje fleksibilnosti i jednostavnosti. Pruža podršku za većinu modernih tehnologija te omogućuje izravno pokretanje bez potrebe za dodatnim konfiguracijama.

Dokumentna baza podataka označava da se podaci ne spremaju u stupce i redove kao kod SQL (eng. Structured Query Language) baza već u obliku dokumenata predstavljenih kao BSON (eng. Binary Representation Of Data). Ovakav tip baze podataka je izrazito fleksibilan te omogućuje variranje u strukturi podataka, odnosno spremanje dokumenata koji su samo djelomično ispunjeni podacima. MongoDB baza podataka je izrazito skalabilna i lako se može horizontalno povećavati te uspješno i brzo baratati velikom količinom podataka. Ne zahtjeva unaprijed definiranu shemu te se struktura baze podataka može naknadno mijenjati bez ikakvih posljedica ili potrebnih dodatnih radnji. Pruža mogućnosti implementacije (eng. deployment) na razne servise u *oblaku* (eng. Cloud) kao što su AWS, Azure i Google Cloud kroz MongoDB Atlas ili lokalno korištenje pomoću MongoDB Compass. MongoDB Compass je besplatan interaktivni alat za analiziranje, pregled i optimizaciju podataka spremljenih u bazi podataka. Omogućuje pregled spremljenih dokumenata i tipova podataka kako bi se moglo provjeriti ispravnost rada koda koji je zadužen za operacije sa bazom podataka. Pruža i mogućnosti pronalaska podataka spremljenih u bazu pomoću upita (eng. query). Spajanje ove baze podataka sa Spring Bootom moguće je pomoću ranije spomenute zavisnosti *Spring Data MongoDB* te je odmah nakon inicijalnog povezivanja baza podataka spremna za prihvat i spremanje podataka [3].

2.4 Axios

Axios je HTTP klijent za node.js i preglednike (eng. browser) temeljen na obećanjima (eng. promise-based). On je izomorfan što znači da se može izvoditi i u preglednicima i na node.js platformi koristeći isti kod. Na strani poslužitelja (eng. server-side) Axios koristi node.js http modul, dok na strani klijenta (na pregledniku) koristi XMLHttpRequests. Omogućuje presretanje HTTP zahtjeva i odgovora što znači da je moguće manipulirati odgovorom poslužiteljske strane prije nego stigne do korisnika ili aplikacije. Podržava asinkrone zahtjeve pomoću Promise API-a. Asinkroni zahtjevi omogućuju da klijentska strana aplikacije nastavi svoj rad nakon pokretanja zahtjeva te se vraća na zahtjev tek prilikom dobivanja odgovora. Automatski pretvara podatke u zahtjevima i odgovorima u JSON, FormData, ili URL enkodirane oblike. Omogućuje otkazivanje zahtjeva te postavljanje vremenskih ograničenja ukoliko se zahtjev ne može izvršiti. Serijski pretvara upitne parametre (eng. query parameters) u odgovarajući format i podržava ugniježđene unose. Pruža zaštitu od CSRF napada na klijentskoj strani. Axios je moguće preuzeti te dodati u projekt naredbom *npm install axios* [5].

Primjer sa Slike 2.7 koristi Axios za izvršavanje HTTP POST zahtjeva prema URL adresi */user*. Početni dio koda *axios.post* omogućuje slanje POST zahtjeva te u tijelu zahtjeva sadrži podatke *firstName* sa vrijednosti Fred te *lastName* sa vrijednosti Flinstone te se oni šalju poslužiteljskoj strani. Dio koda *.then(function (response))* dodaje rukovatelja za uspješan odgovor poslužiteljske strane. Kada poslužitelj uspješno obradi zahtjev i vrati odgovor biti će pozvan dio koda u nastavku ove funkcije, odnosno ispisati će se odgovor u konzolu. Dio koda koji započinje sa *.catch(function (error))* dodaje rukovatelja za slučaj kada dođe do pogreške tijekom izvršavanje zahtjeva na poslužiteljskoj strani. Ukoliko dođe do pogreške izvršiti će se dio koda koji se nalazi u nastavku ove funkcije, odnosno ispisati će se greška u konzolu.

```
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Slika 2.7: Primjer korištenja Axiosa [5]

2.5 Material UI

Material UI je React knjižnica komponenti otvorenog koda koja implementira Googleove smjernice za Material Design. Razvio ju je Google 2014. godine i danas je jedna od najpopularnijih knjižnica ovog tipa. Pruža velik skup komponenti i stilova koji olakšavaju izradu privlačnih i responzivnih korisničkih sučelja. Koristi princip mreže (eng. grid) za postavljanje i raspored komponenti u prostoru. Nudi velik broj raznovrsnih komponenti kao što su tipke, kartice, dijalozi, navigacija te čitav skup ikona za sve moguće potrebe. Omogućuje jednostavno tematiziranje i prilagodbu komponenti kako bi bolje odgovarale određenom stilu aplikacije. Svakoj komponenti je moguće nadjačavanje trenutnog stila pomoću ključne riječi *sx* uz dodavanje klasičnih CSS argumenata. Još jedna dobra strana je responzivnost komponenti. Komponente su dizajnirane tako da se jednostavno prilagođavaju raznim uređajima, od mobitela do velikih monitora, bez da programer o tome mora voditi računa i dodavati posebne mogućnosti i baviti se CSS naredbama kako bi sve bilo pravilno uređeno. Material UI je moguće dodati u projekt instalacijom paketa pomoću naredbe *npm install @mui/material*. Nakon instalacije, moguće je implementirati bilo koju komponentu pomoću naredbe *import* plus ime određene komponente iz baze komponenata [16].

2.6 IntelliJ IDEA

IntelliJ IDEA je integrirano razvojno okruženje napisano u Javi. Pretežno služi za razvoj softvera u jezicima Java, Groovy i Kotlin te sličnim koji se temelje na korištenju Java virtualnog stroja (eng. Java virtual machine) iako podržava i druge jezike kao što su JavaScript i Python. Razvijen je od strane JetBrainsa, ranije poznatog kao IntelliJ, te je prva verzija nastala 2001. godine. Spada u platforme otvorenog koda, iako nudi i plaćenu verziju. Za većinu programera i osnovna verzija pruža dovoljno mogućnosti te nema potrebe za plaćanjem dodatnog iznosa kako bi otključali još neke dodatne mogućnosti koje se nikad neće ni iskoristiti u praksi. IntelliJ IDEA jedan je od najmoćnijih i najpopularnijih integriranih razvojnih okruženja današnjice. Omogućuje automatsko dovršavanje koda i preporuke za refaktoriranje, razne alate za testiranje te laganu integraciju raznih radnih okvira i alata. Pruža odličnu podršku za Spring Boot i TypeScript te jednostavno povezivanje i verzioniranje koda pomoću platforme Git stoga je u ovom projektu korišten za razvoj poslužiteljskog i klijentskog dijela aplikacije [17].

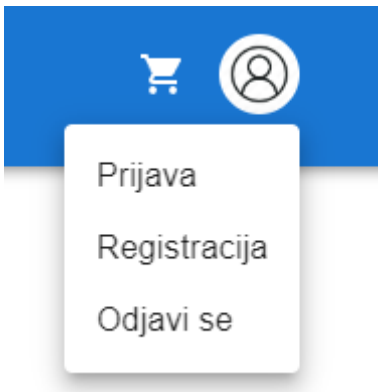
3 OPIS APLIKACIJE

U sklopu projekta razvijena je web aplikacija za internetsku trgovinu. Aplikacija se dijeli na administracijski i korisnički dio, neke funkcionalnosti su zajedničke, dok neke pripadaju samo jednoj razini autoriteta u aplikaciji. U nastavku će biti detaljnije opisane i prikazane sve funkcionalnosti ove web aplikacije.

Prilikom dolaska na web stranicu moguć je pregled većine funkcionalnosti bez potrebe za autentifikacijom kao korisnik. Međutim za dodavanje proizvoda u košaricu te kasnije naručivanje i kupnju tih proizvoda, što je jedna od glavnih stavki aplikacije za internetsku trgovinu, potrebno se registrirati korisničkim podacima stoga će najprije biti opisan proces same autentifikacije.

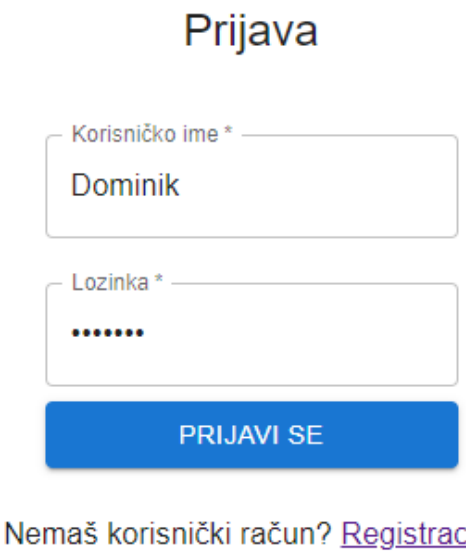
3.1 Autentifikacija

Autentifikacija korisnika moguća je pritiskom na ikonu korisničkog računa koja se nalazi u gornjem desnom kutu aplikacije i prikazana je na Slici 3.1. Nakon otvaranja ikone pojavljuje se padajući izbornik u kojem je moguć odabir opcije *Prijava* za ponovnu prijavu u već postojeći račun, *Registracija* za izradu novog korisničkog računa te *Odjavi se* za odjavu sa trenutno prijavljenog računa. Autentifikacija je moguća i pritiskom na ikonu za dodavanje proizvoda u košaricu koja se nalazi na svakom proizvodu i biti će prikazana u daljnjem tekstu kod opisa proizvoda. Ukoliko je korisnik već registriran, ikona će dodati proizvod u košaricu umjesto da vodi korisnika na pogled za autentifikaciju.



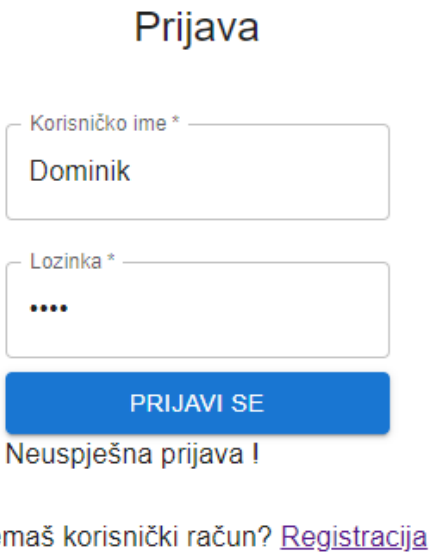
Slika 3.1: Prikaz ikone korisničkog računa i padajućeg izbornika

Odabirom opcije *Prijava* otvara se novi pogled prikazan na Slici 3.2. Prilikom prijave potrebno je upisati odabrano korisničko ime i lozinku te pritisnuti tipku *Prijavi se*. Ispod tipke za prijavu nalazi se poveznica na pogled za registraciju ukoliko korisnik još nema kreiran korisnički račun pomoću kojeg bi se mogao prijaviti.



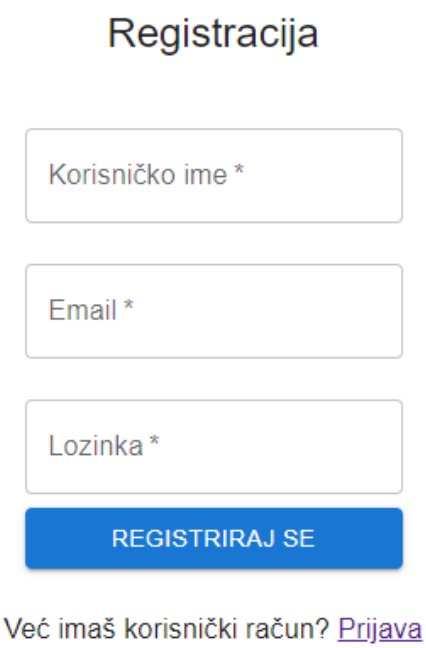
Slika 3.2: Prikaz pogleda za prijavu korisnika

Za uspješnu prijavu potrebno je ispuniti oba polja ispravnim podacima te se nakon toga korisnik usmjeruje na stranicu s ponudom. Ukoliko je jedan od podataka kriv i ne podudara se s ranije spremljenim podacima, korisnik će biti obaviješten napomenom *Neuspješna prijava !* kao što je vidljivo na Slici 3.3.



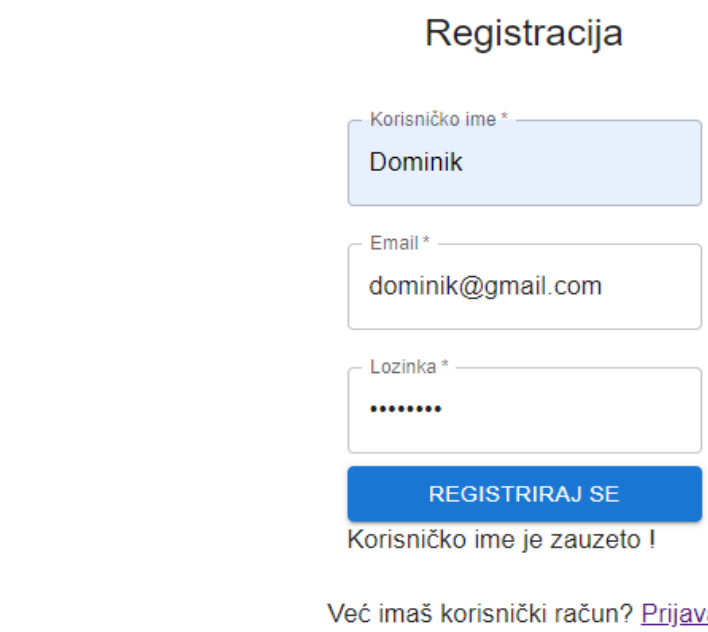
Slika 3.3: Prikaz neuspješne prijave

Ukoliko korisnik još nema kreiran račun može to napraviti odabirom opcije *Registracija* iz padajućeg izbornika. Nakon odabira ove opcije otvara se novi pogled prikazan na Slici 3.4. Prilikom registracije potrebno je unijeti korisničko ime, adresu e-pošte i lozinku te pritisnuti tipku *Registriraj se*. Ispod tipke za registraciju nalazi se poveznica na pogled za prijavu ukoliko korisnik već ima postojeći korisnički račun.



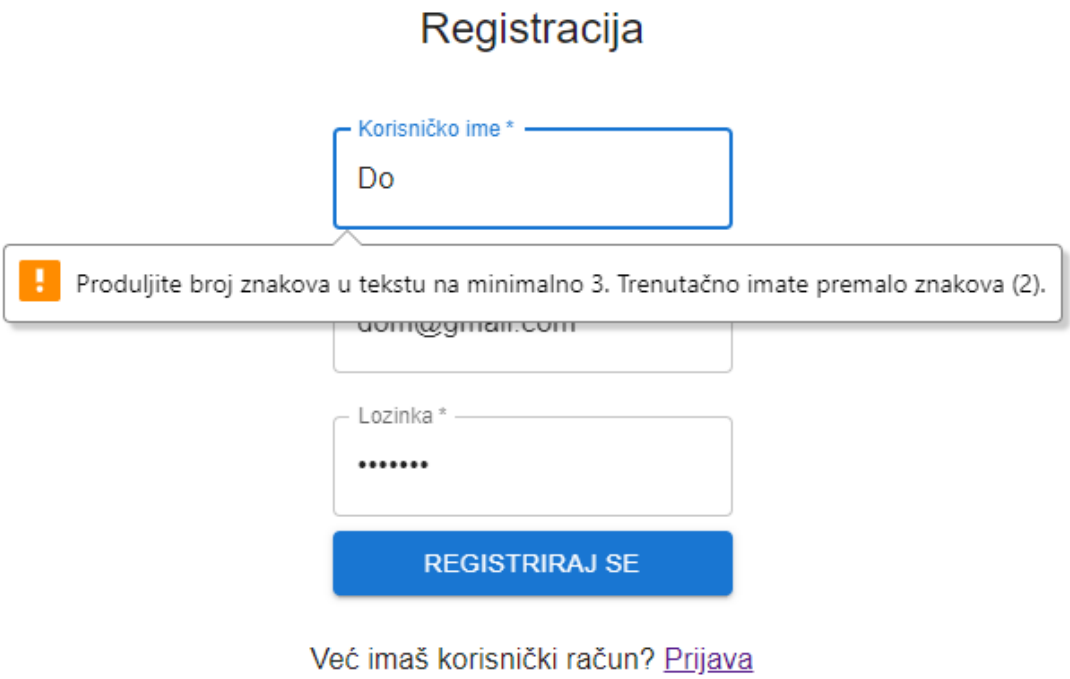
Slika 3.4: Prikaz pogleda za registraciju

Za uspješnu registraciju potrebno je ispuniti sva tri polja pošto su označena kao obavezna. Ukoliko ih korisnik ne ispuni pojaviti će se upozorenje kako je potrebno ispuniti ovo polje za nastavak. Također, korisnik mora odabrati autentično korisničko ime i adresu e-pošte koji se još ne koriste u aplikaciji od strane nekog drugog korisnika. Ukoliko su korisničko ime ili adresa e-pošte zauzeti, aplikacija će prikazati upozorenje kako je određeni podatak već zauzet kao što je prikazano na Slici 3.5 za primjer već korištenog korisničkog imena.



Slika 3.5: Prikaz neuspješne registracije

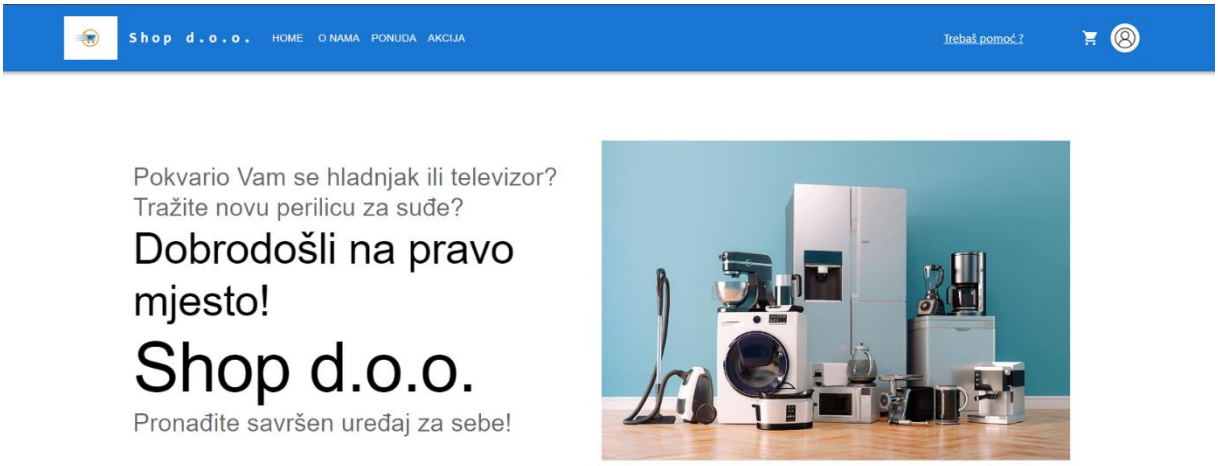
Za podatke korisničko ime i lozinka postavljena su ograničenja broja znakova. Za korisničko ime je potrebno unijeti minimalno tri, a maksimalno dvadeset znakova. Za lozinku je potrebno unijeti minimalno šest, a maksimalno četrdeset znakova. Ukoliko ovi uvjeti nisu ispunjeni aplikacija će prikazati upozorenje kao što je prikazano na Slici 3.6.



Slika 3.6: Poruka upozorenja za neispravno unesen podatak

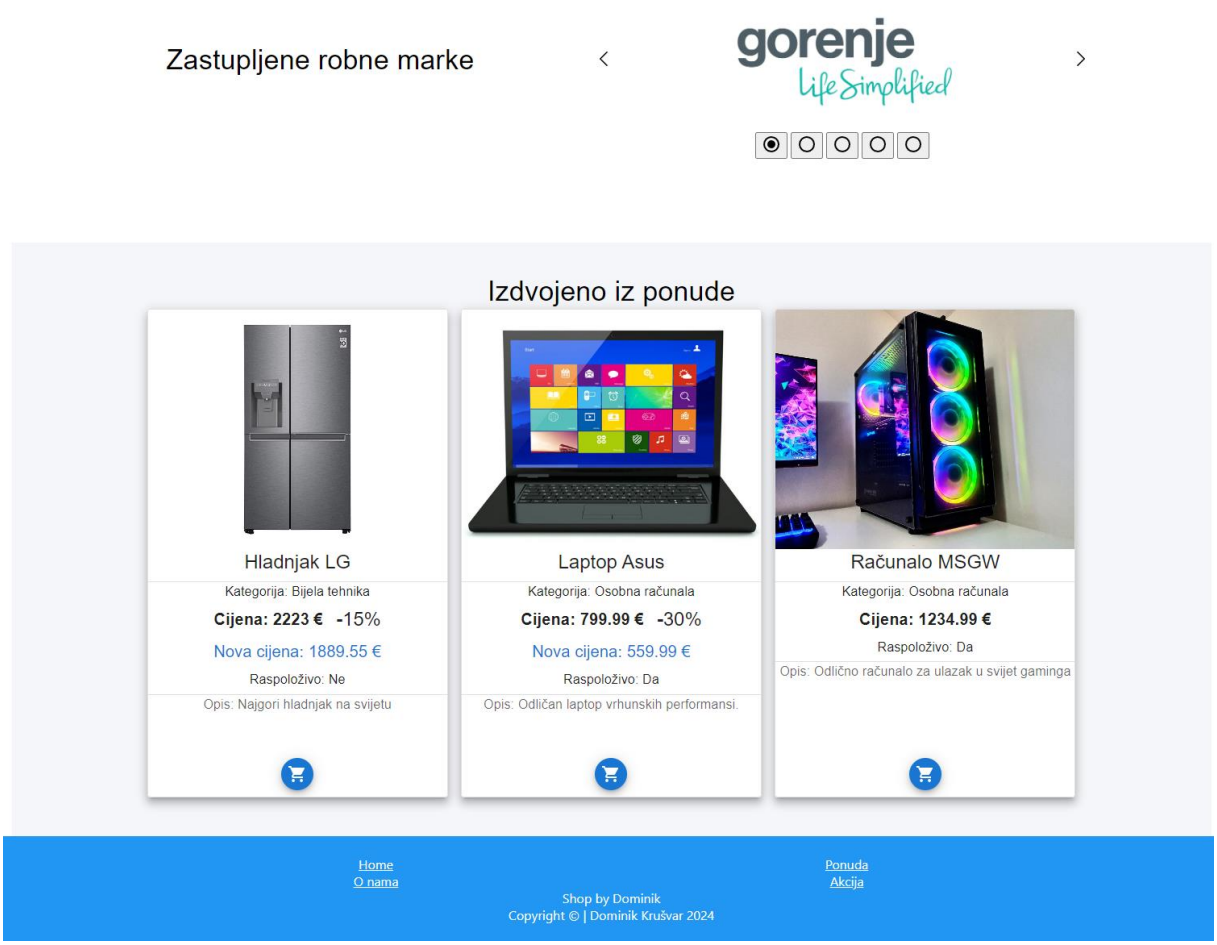
3.2 Pogledi kroz aplikaciju za korisnika

Prilikom dolaska na web stranicu internetske trgovine korisniku se otvara početna (eng. home) stranica prikazana na Slici 3.7 i Slici 3.8. Na vrhu stranice nalazi se navigacijska traka koja sadrži logo i naziv trgovine, poveznice na ostale poglede unutar aplikacije koji će kasnije u tekstu biti detaljnije opisani, poveznica na komunikaciju administratora i korisnika pod nazivom *Trebaš pomoć ?*, ikona košarice te ranije spomenuta ikona korisničkog računa koja omogućuje autentifikaciju. Ispod navigacijske trake nalazi se tekst koji opisuje internetsku trgovinu te prigodna slika. U realnim uvjetima ovaj dio stranice bio bi dizajniran u dogovoru sa predstavnikom internetske trgovine kojoj je stranica namijenjena.



Slika 3.7: Prikaz početne stranice (1)

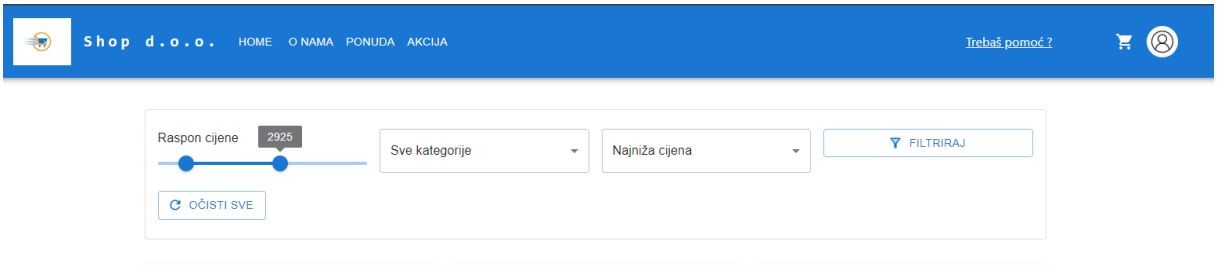
U nastavku početne stranice nalazi se galerija slika koja sadrži zastupljene robne marke u toj internetskoj trgovini. Galerija se, pomoću strelica, može listati u lijevu i desnu stranu te je ispod slike prikazano na kojoj se slici korisnik trenutno nalazi te ukupno koliko slika ima. Prikaz galerije je na vrhu Slike 3.8. Ispod galerije prikazana su tri proizvoda izdvojena iz ponude kako bi privukla korisnike na kupnju u ovoj internetskoj trgovini. Proizvodi će detaljnije biti opisani u nastavku teksta vezano uz pogled *Ponuda*. Na dnu stranice se nalazi zaglavlje koje sadrži poveznice na sve poglede slično kao i u navigacijskoj traci te podaci o kreatoru web stranice.



Slika 3.8: Prikaz početne stranice (2)

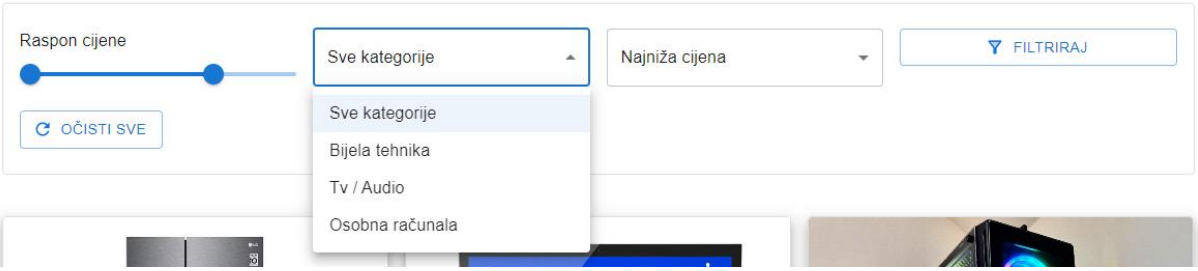
Pogled pod nazivom *O nama* sadrži osnovne informacije o internetskoj trgovini. Sastoji se od nekoliko tekstova koji opisuju internetsku trgovinu, njezin cilj te zaposlenike. Uz svaki tekst dolazi i slika koja prikazuje opisane stavke. Ovaj pogled ne pruža nikakve funkcionalnosti aplikaciji osim izgleda stoga neće biti prikazan slikom.

Sljedeći pogled naziva se *Ponuda* te sadrži sve proizvode koji su dostupni u ponudi internetske trgovine. Na samom vrhu pogleda, ispod ranije spomenute navigacijske trake, nalazi se odvojeni dio stranice predviđen filterima za pretragu proizvoda prikazan na Slici 3.9.



Slika 3.9: Prikaz filtera za pretragu proizvoda

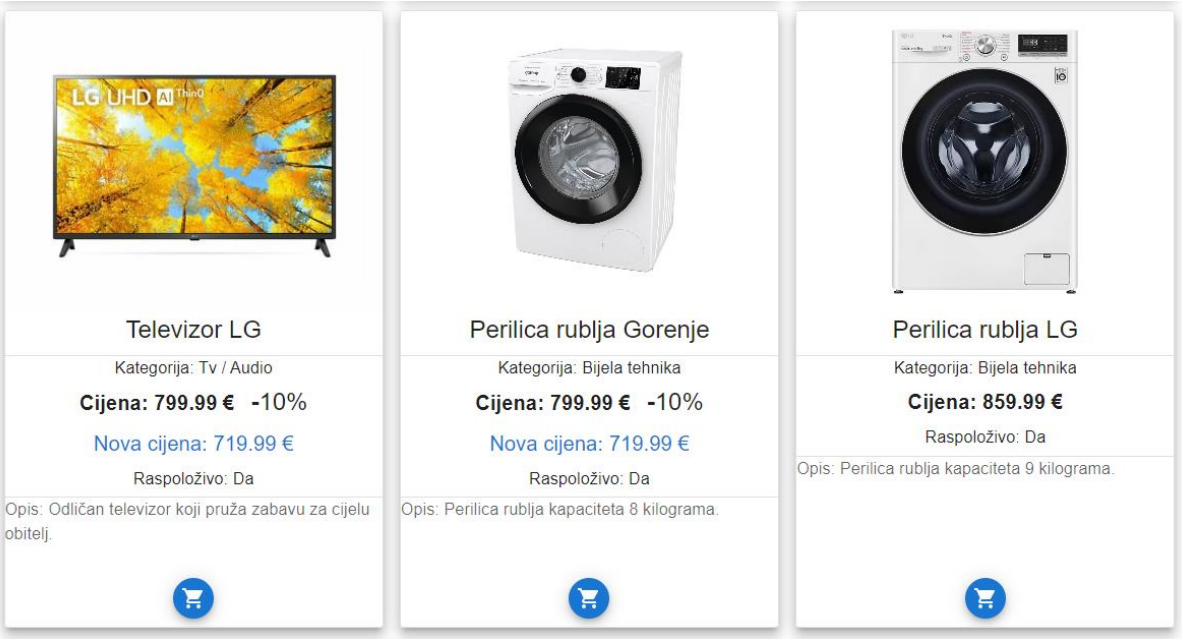
Prvi mogući filter je po rasponu cijene proizvoda koji ulaze u pretragu. Sastoji se od dvije točke koje se pomiču lijevo-desno nakon što korisnik klikne mišem na njih. Prilikom pomicanja ili samo prelaska mišem preko točke korisniku se prikazuje iznos, odnosno vrijednost koju točka ima u tom trenutku. Sljedeći mogući filter je odabir kategorije proizvoda koji se prikazuju. Sastoji se od padajućeg izbornika unutar kojeg je moguće odabrati željenu kategoriju proizvoda i prikazan je na Slici 3.10. Ponuđene su kategorije *Bijela tehnika*, *Tv/Audio* te *Osobna računala*. Treći mogući filter sortira proizvode prema cijeni uzlazno ili silazno, odnosno od najniže cijene prema najvišoj ili od najviše cijene prema najnižoj, kako bi kupci mogli lakše usporediti nekoliko proizvoda sa sličnom cijenom. Nakon što korisnik odabere željene opcije potrebno je pritisnuti tipku pod nazivom *Filtriraj* kako bi se ponuda promijenila te kako bi bili prikazani samo proizvodi koji zadovoljavaju odabrane parametre. Moguće je odabrati vrijednost za sva tri filtera, ali i samo pojedinačnu opciju. Pritiskom na tipku *Očisti sve* poništavaju se sve filtrirane vrijednosti te se korisniku ponovno prikazuje početni raspored proizvoda.



Slika 3.10: Prikaz padajućeg izbornika za odabir kategorije

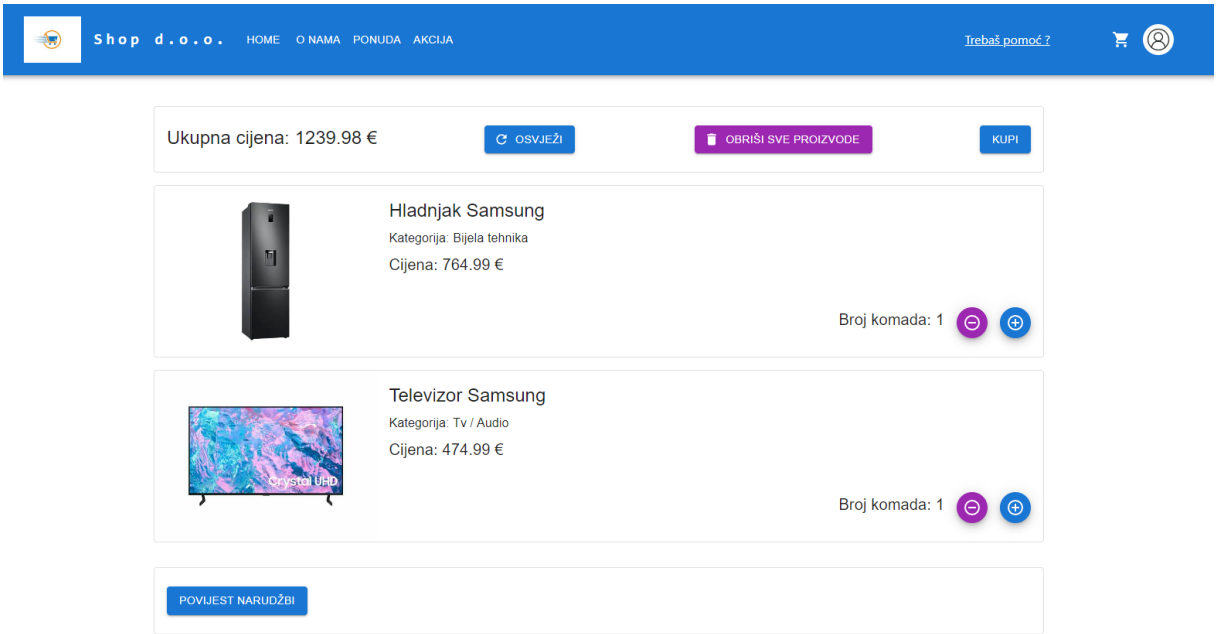
Nakon prostora za filtere poredani su proizvodi koje nudi internetska trgovina. Izgled nekoliko proizvoda prikazan je na Slici 3.11. Svaki proizvod predstavlja jednu kartičnu komponentu (eng. Card Component) preuzetu iz Material UI knjižnice komponenti [16]. Na vrhu svake kartice nalazi se pripadajuća slika i naziv proizvoda. Nakon toga dolaze još neke informacije o proizvodu, a to su kategorija kojoj pripada, cijena proizvoda te dostupnost u ponudi označena sa *Raspoloživo*. Ukoliko je proizvod na akciji, pored cijene će biti određeni postotak popusta, a ispod cijene će biti prikazana nova cijena nakon izračunatog popusta istaknuta plavom bojom kako bi privukla pozornost kupca za taj proizvod. Primjer proizvoda na akciji vidljiv je na Slici 3.11 za prva dva prikazana proizvoda dok je treći proizvod primjer za proizvod koji nije na akciji. Ispod ovih informacija nalazi se i detaljniji opis proizvoda koji će korisniku pružiti još dodatnih informacija o navedenom proizvodu. Na samom dnu proizvoda

nalazi se ikona košarice. Prilikom prijelaza mišem preko ove ikone pojavljuje se poruka *Dodaj u košaricu !*, a pritiskom na ovu ikonu korisnik dodaje proizvod u košaricu.



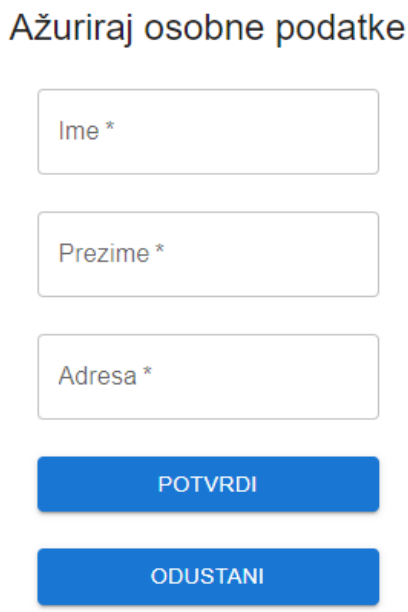
Slika 3.11: Prikaz proizvoda

Nakon što korisnik doda određene proizvode u košaricu i spreman je za kupnju potrebno je izvršiti još nekoliko koraka. Pritiskom na ikonu košarice koja se nalazi na navigacijskoj traci otvara se pogled prikazan na Slici 3.12.



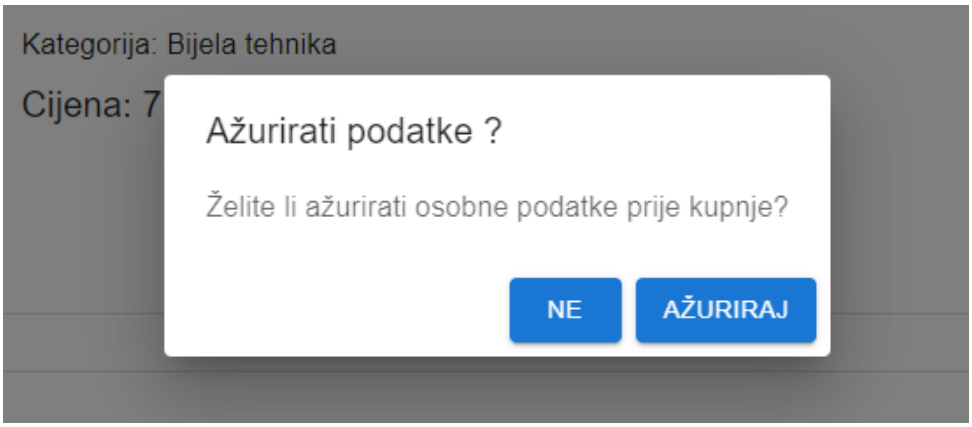
Slika 3.12: Prikaz pogleda košarice

Na samom vrhu pogleda, ispod navigacijske trake, prikazana je ukupna cijena košarice, tipka *Osvježi*, tipka *Obriši sve proizvode* te tipka *Kupi*. Pritiskom na tipku *Osvježi* ažurira se sadržaj i cijena košarice te se miču proizvodi kojima je odabrana količina nula jer je korisnik ipak odlučio da ih ne želi kupiti. Pritiskom na tipku *Obriši sve proizvode* brišu se svi proizvodi koji su prisutni u košarici te se ukupna cijena ažurira na nulu. Pritiskom na tipku *Kupi* otvara se pogled prikazan na Slici 3.13 ukoliko je korisniku ovo prva kupnja te je potrebno upisivati korisničke podatke. Podaci koje je potrebno upisati su ime, prezime i adresa te se pritiskom tipke *Potvrdi* oni spremaju u korisnikov profil za daljnju kupovinu. Pritiskom tipke *Odustani* korisnik će biti vraćen na pogled košarice.



Slika 3.13: Pogled za unos osobnih podataka

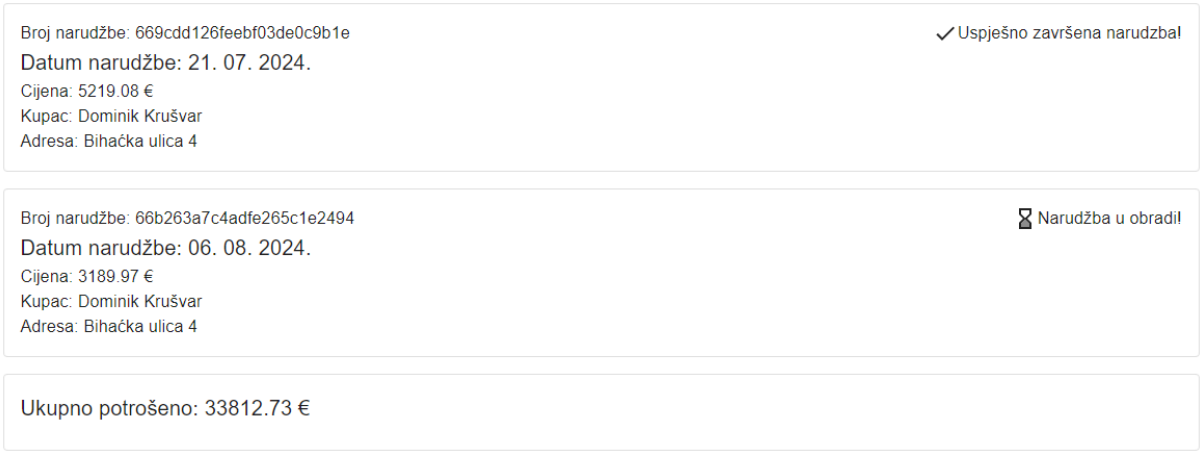
Ukoliko je korisnik već ranije kupovao i unio korisničke podatke prilikom pritiska tipke *Kupi* otvoriti će se skočni prozor prikazan na Slici 3.14 sa upitom o ažuriranju postojećih korisničkih podataka. Ako korisnik odabere opciju *Ne* kupnja će biti izvršena pomoću ranije unesenih korisničkih podataka. Ukoliko odabere opciju *Ažuriraj* biti će preusmjeren na pogled sa Slike 3.13 te ovdje može unijeti nove podatke o imenu, prezimenu i adresi koji će biti spremljeni za sljedeće kupnje.



Slika 3.14: Skočni prozor sa upitnikom za ažuriranje podataka

U nastavku pogleda košarice nalaze se proizvodi koje je korisnik odabrao i dodao u košaricu. Za svaki proizvod je prikazana slika, naziv proizvoda, kategorija kojoj pripada te cijena jednog primjerka. Na desnoj strani prikazan je broj komada koji se trenutno nalaze u košarici te dvije tipke. Prva tipka ljubičaste boje s ikonom minusa služi za uklanjanje jednog proizvoda iz košarice. Druga tipka plave boje s ikonom plusa služi za dodavanje jednog proizvoda u košaricu. Nakon svakog pritiska jedne od ovih tipki prisutna je odgoda (eng. delay) prilikom koje se tipka ne može pritisnuti kako bi se spriječile potencijalne pogreške koje mogu nastati zbog pre brzog stiskanja ove tipke. Ako aplikacija ne uspije na vrijeme obraditi svaki od zahtjeva moglo bi doći do razlike u broju komada prikazanih na ekranu i stvarnom broju spremljenom u aplikaciji.

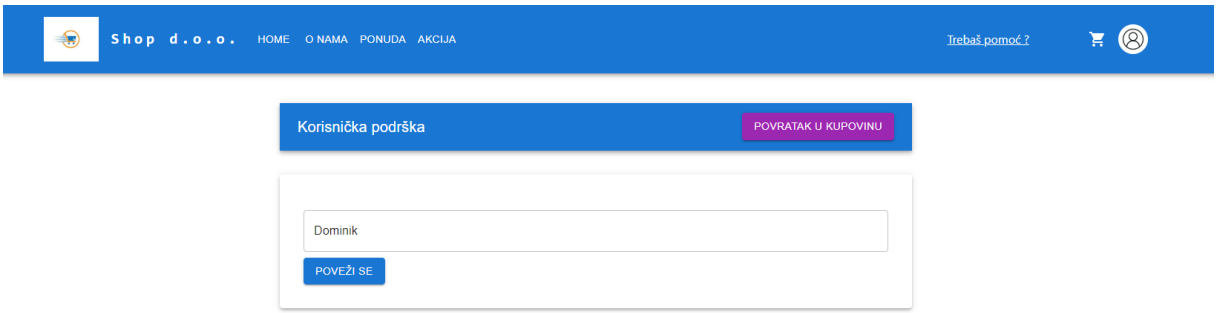
Na dnu pogleda košarice nalazi se tipka pod nazivom *Povijest narudžbi*. Prilikom pritiska ove tipke otvara se novi pogled prikazan na Slici 3.15. Unutar ovog pogleda nalaze se sve narudžbe koje je korisnik kreirao u ovoj aplikaciji. Za svaku narudžbu prikazan je njen jedinstveni broj, datum kada je izvršena, iznos te narudžbe te uneseni podaci o kupcu koje korisnik unosi na pogledu sa Slike 3.13. Na desnoj strani nalazi se podatak o izvršenju narudžbe. Narudžba može imati dva statusa i to su *Narudžba u obradi!* i *Uspješno završena narudžba!*. Svaka kreirana narudžba automatski dobiva status *Narudžba u obradi!*. Kako bi se status promijenio u *Uspješno završena narudžba!* potrebno je odobrenje administratora, odnosno djelatnika internetske trgovine, koje će biti detaljnije opisano u dijelu teksta posvećenom administracijskom dijelu aplikacije. Na dnu ovog pogleda nalazi se podatak koji prikazuje ukupni iznos koji je korisnik potrošio na sve narudžbe koje su uspješno izvršene.



Slika 3.15: Pogled povijesti narudžbi

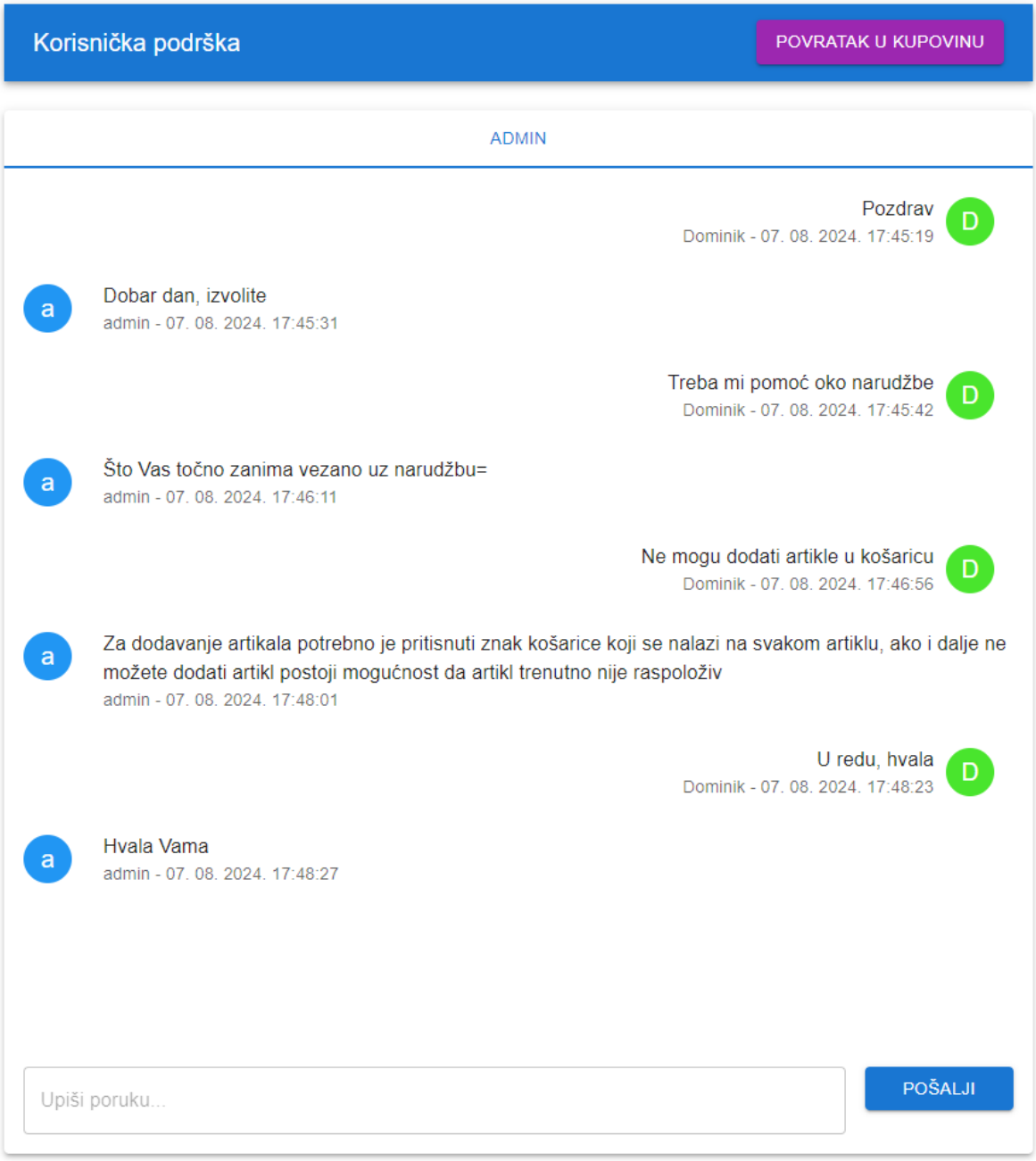
Pogled pod nazivom *Akcija* sadrži istaknute proizvode koji su na akciji te popust iznosi minimalno 15 posto. Izgled proizvoda i funkcionalnost dodavanja u košaricu je ista kao i na pogledu *Ponuda* opisanom u prijašnjem tekstu.

Ako korisniku zatreba pomoć tijekom kupnje proizvoda ili ga zanima neka informacija o internetskoj trgovini, može komunicirati s administratorom u realnom vremenu. Za ovu mogućnost potrebno je pritisnuti poveznicu koja se nalazi na navigacijskoj traci pod nazivom *Trebaš pomoć ?*. Nakon toga se otvara pogled prikazan na Slici 3.16. U gornjem desnom kutu nalazi se tipka *Povratak u kupovinu* koja korisnika vraća na pogled *Ponuda* ako ipak nema potrebu za komunikacijom sa administratorom. Tekstni okvir u sredini pogleda zahtjeva od korisnika unos imena prije nego može započeti komunikaciju. Ukoliko je korisnik trenutno prijavljen sa korisničkim računom, aplikacija će automatski ispuniti ovo polje za korisnika kao što je slučaj na Slici 3.16. Nakon unosa imena potrebno je još pritisnuti tipku *Poveži se* kako bi se otvorio novi pogled koji omogućuje komunikaciju.



Slika 3.16: Pogled za povezivanje sa korisničkom podrškom

Nakon što se korisnik povezao, otvara se pogled prikazan na Slici 3.17. Na samom vrhu i dalje se nalazi tipka *Povratak u kupovinu* kako bi korisnik mogao izaći iz razgovora i vratiti se u kupovinu u bilo kojem trenutku. Ispod toga nalazi se kartica *ADMIN* u kojoj se pojavljuju poruke nakon što administrator odgovori korisniku. Poruke koje korisnik šalje organizirane su na desnoj strani, dok su administratorove poruke na lijevoj strani. Svaka poruka sadrži ime pošiljatelja te datum i vrijeme u koje je poslana. Na dnu pogleda nalazi se tekstni okvir za unos poruke, a pored njega je tipka *Pošalji* koju je potrebno pritisnuti za poslati poruku.

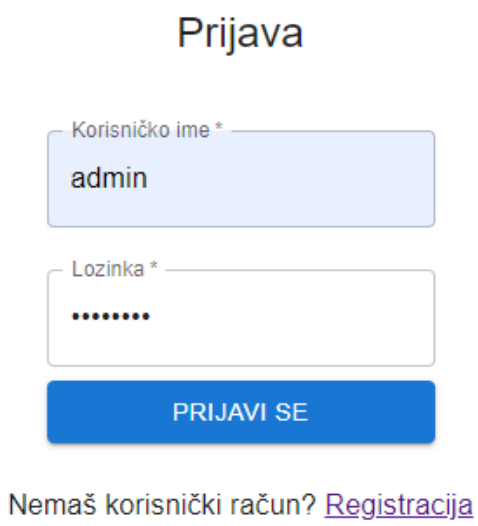


Slika 3.17: Pogled za razgovor korisnika i administratora

3.3 Pogledi kroz aplikaciju za administratora

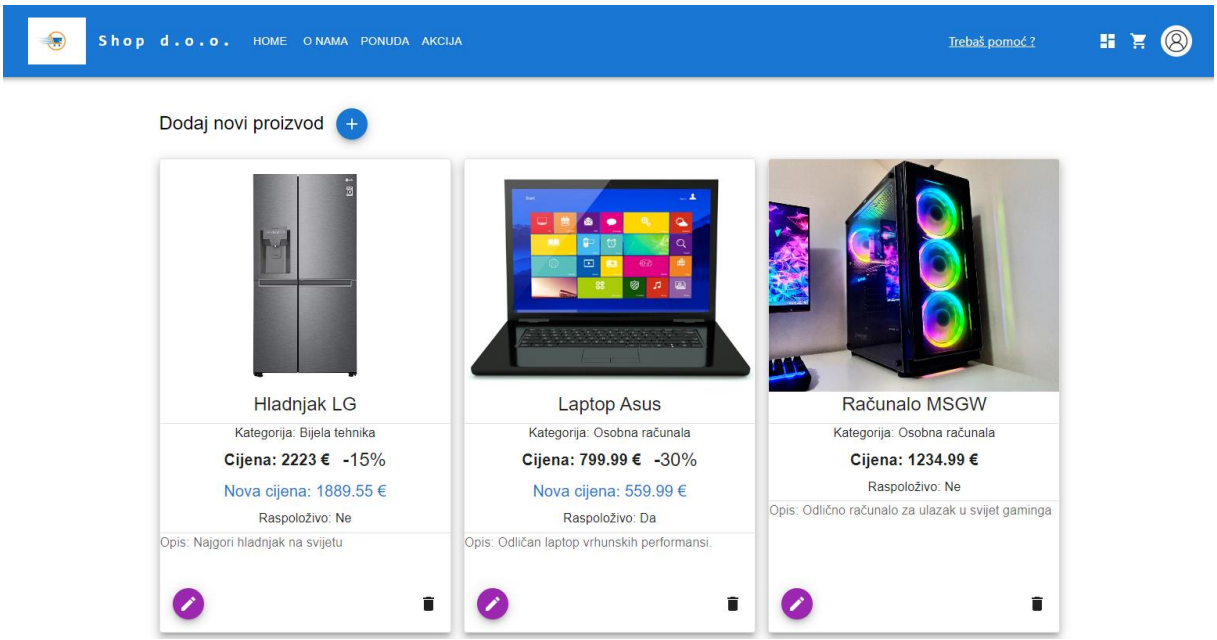
Vlasnik internetske trgovine kojoj je web aplikacija namijenjena ima ulogu administratora web aplikacije. S tom ulogom dolazi mogućnost samostalnog uređivanja nekih dijelova aplikacije vezanih uz proizvode, mogućnost pregleda statistike prodaje te razgovor s korisnikom u sklopu korisničke podrške.

Prilikom dolaska na web stranicu potrebno se prijaviti korisničkim imenom *admin* te dodijeljenom lozinkom koju se dobije od kreatora same web aplikacije. Za ovu ulogu ne postoji oblik registracije kojim je moguće kreirati još jedan ovakav račun kako se neki drugi slučajni korisnici ne bi mogli pojaviti u ovoj ulozi. Postupak prijave isti je kao i kod prijave korisnika putem ikone korisničkog računa koja se nalazi na navigacijskoj traci te potom odabira opcije *Prijava*. Ova ikona prikazana je u ranijem tekstu na Slici 3.1. Nakon odabira opcije *Prijava* otvara se pogled za prijavu te se unose pripadajući podaci prikazani na Slici 3.18.



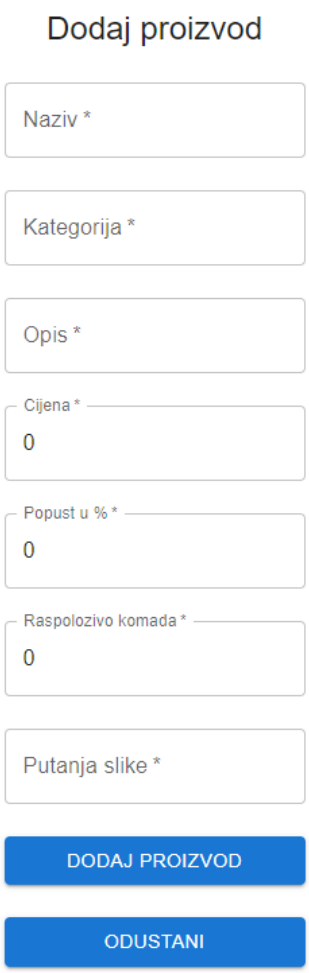
Slika 3.18: Pogled za prijavu administratora

Pogled početne stranice (eng. home), pogledi *O nama* i *Akcija* jednaki su za korisnika i za administratora te nije moguće nikakvo uređivanje. Pogled *Ponuda* pruža mogućnost dodavanja novih proizvoda, ažuriranja informacija o postojećim proizvodima te mogućnost uklanjanja trenutno prikazanih proizvoda. Ovaj pogled prikazan je na Slici 3.19. Prva mogućnost je dodavanje novih proizvoda i nalazi se na vrhu pogleda, odmah ispod navigacijske trake. Označena je sa *Dodaj novi proizvod* te je dodavanje moguće pritiskom na plavu ikonu koja sadrži znak plus.



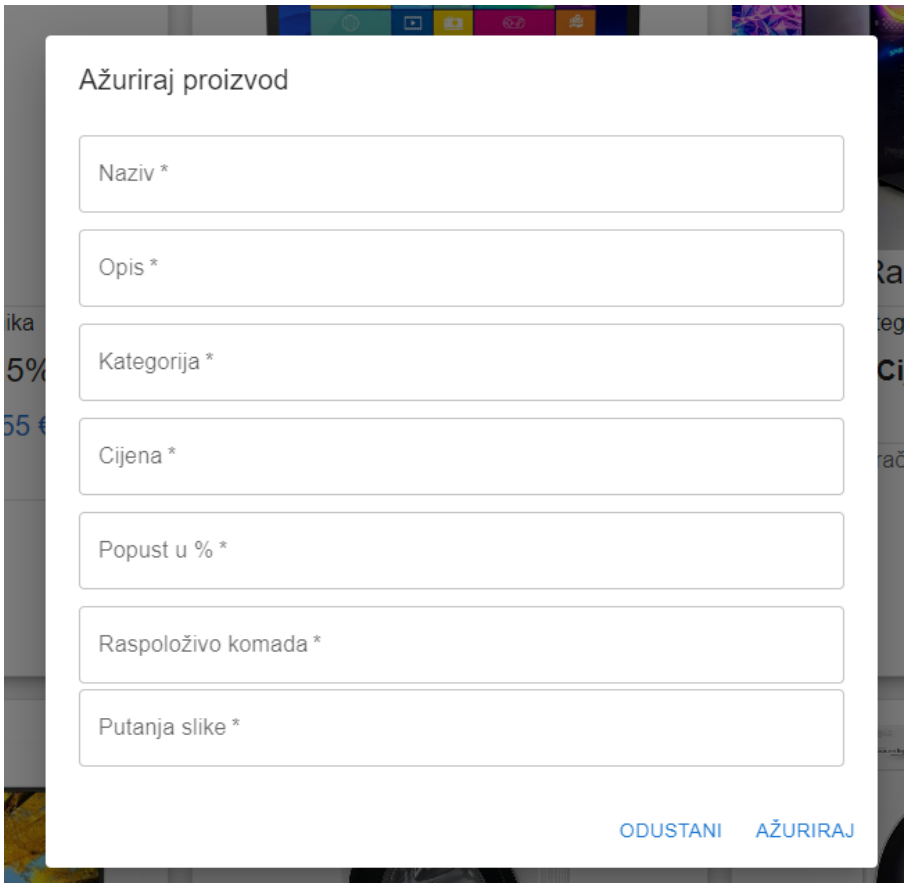
Slika 3.19: Pogled Ponuda za administratora

Nakon pritiska na ikonu otvara se pogled za dodavanje proizvoda prikazan na Slici 3.20. Pogled se sastoji od forme za upis podataka koje proizvod treba sadržavati. Svi su podaci obavezni za unos te nije moguće dodati proizvod bez da su sva polja ispunjena. Podaci koje je potrebno unijeti su naziv proizvoda, kategorija kojoj proizvod pripada, kratki opis koji korisniku pruža dodatne informacije o tom proizvodu te cijena proizvoda u eurima. Polje za popust je također obavezno i unosi se postotak popusta, no može se unijeti vrijednost nula, te će tako za proizvod biti prikazana samo unesena originalna cijena. Polje za broj raspoloživih komada također može imati vrijednost nula ako proizvod još nije dostupan, pa se taj podatak može kasnije ažurirati o čemu nešto više u sljedećem odlomku. Posljednji podatak koji je potrebno unijeti je putanja do slike proizvoda. Ukoliko ta putanja nije dostupna umjesto slike će biti prikazan zamjenski tekst koji opisuje tu sliku. Nakon što su sva polja ispunjena potrebno je pritisnuti tipku *Dodaj proizvod* kako bi se proizvod pojavio u ponudi. Ako se ipak odustane od dodavanja novog proizvoda, potrebno je pritisnuti tipku *Odustani* te se tako izlazi iz pogleda za dodavanje proizvoda i vraća na pogled *Ponuda*.



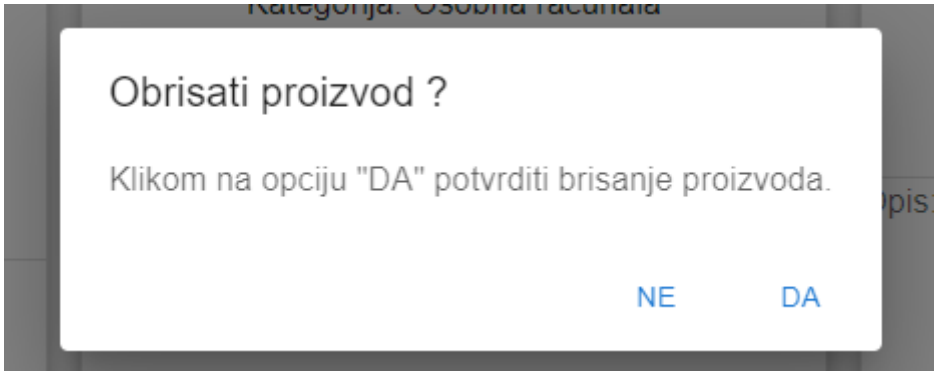
Slika 3.20: Pogled za dodavanje proizvoda

Ako se s vremenom promijene neke informacije o proizvodu ili je potrebno dodati popust i ažurirati broj raspoloživih komada, to je moguće pomoću pogleda za ažuriranje proizvoda prikazanog na Slici 3.21. Otvara se u obliku dijaloga nakon pritiska na ljubičastu ikonu sa znakom olovke koja se nalazi u donjem lijevom kutu svakog proizvoda. Moguće je promijeniti sve informacije o proizvodu. Tako je moguće ažurirati naziv proizvoda, opis, kategoriju kojoj pripada, promijeniti cijenu proizvoda te dodati ili ukloniti popust za taj proizvod. Nabavom novih proizvoda mijenja se i broj dostupnih komada pa je tako moguće ažurirati i taj podatak. Ukoliko dođe do promjene putanje slike ili je proizvod promijenio izgled, moguće je ažurirati i dodati novu putanju slike kako bi bio prikazan najnoviji proizvod. U ovoj formi nije potrebno ispuniti sva polja pri ažuriranju proizvoda nego je moguće ispuniti samo ono polje čiji sadržaj se želi ažurirati, dok će sva ostala polja i dalje zadržati prijašnju vrijednost. Za potvrdu ažuriranja vrijednosti potrebno je pritisnuti tipku *Ažuriraj* koja se nalazi u donjem desnom kutu dijaloga.



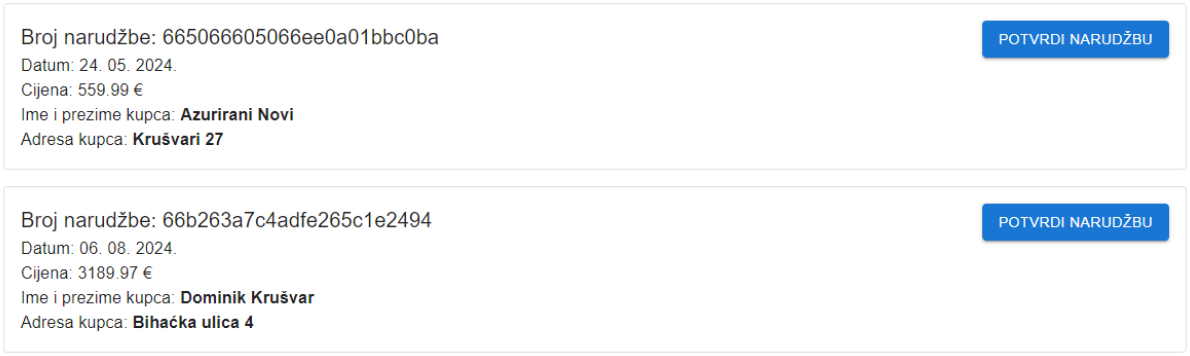
Slika 3.21: Dijalog za ažuriranje proizvoda

Treća mogućnost vezana uz proizvod je brisanje proizvoda. To je moguće pritiskom na crnu ikonu kante za smeće koja se nalazi u donjem desnom kutu svakog proizvoda. Nakon pritiska ikone otvara se skočni prozor za potvrdu brisanja artikla kako administrator ne bi slučajno uklonio proizvod bez znanja o tome. Skočni prozor prikazan je na Slici 3.22 te sadrži pitanje *Obrisati proizvod ?*. Potrebno je odabrati opciju *DA* kako bi proizvod stvarno bio obrisan. Odabirom opcije *NE* slijedi povratak na pogled *Ponuda*.



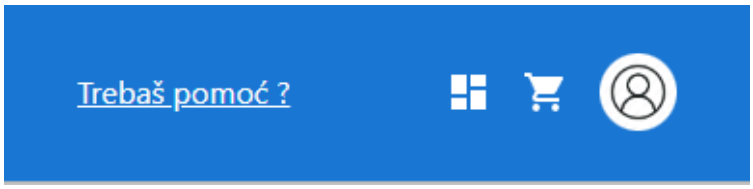
Slika 3.22: Skočni prozor za potvrdu brisanja proizvoda

Administrator ima mogućnost potvrde korisnikove narudžbe kako bi narudžba dobila status *Uspješno završena narudžba!*. Ova funkcionalnost je dostupna pritiskom na ikonu košarice koja se nalazi na navigacijskoj traci u gornjem desnom kutu. Nakon pritiska na ikonu otvara se pogled potvrdu narudžbe prikazan na Slici 3.23. Za svaku narudžbu je prikazan jedinstveni broj narudžbe, datum kada je kreirana, cijena te narudžbe te podaci o kupcu. Podaci o kupcu sadrže ime i prezime kupca te adresu. Ovi podaci se mogu iskoristiti ukoliko internetska trgovina ima mogućnost dostave narudžbi. Na desnoj strani svake narudžbe nalazi se tipka *Potvrdi narudžbu* koju je potrebno pritisnuti kako bi narudžba bila uspješno završena. Dok god se narudžba ne potvrdi od strane administratora kupac neće vidjeti narudžbu kao završenu, nego će ta narudžba biti u statusu obrade.



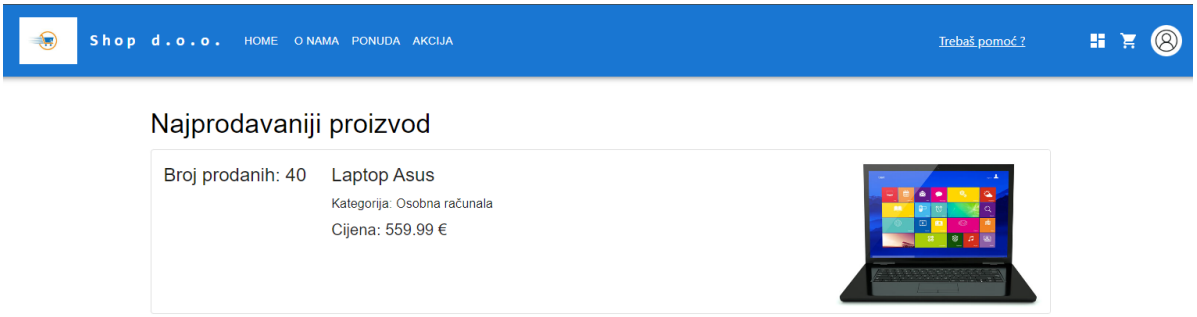
Slika 3.23: Pogled za odobravanje narudžbi

Prilikom prijave kao administrator u gornjem desnom kutu navigacijske trake pojavljuje se jedna nova ikona koja korisniku nije dostupna. Prikazana je na Slici 3.24 i nalazi se slijeva ikoni košarice. Ova ikona otvara pogled u kojem administrator može vidjeti statistiku prodaje proizvoda u internetskoj trgovini.



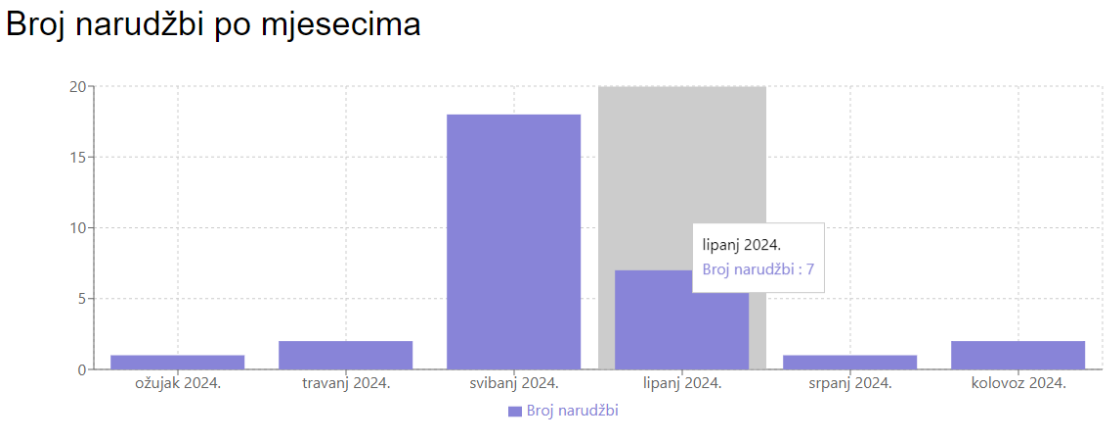
Slika 3.24: Prikaz dodatne ikone na navigacijskoj traci za administratora

Na samom vrhu pogleda nalazi se najprodavaniji proizvod u internetskoj trgovini. Ovdje se računaju samo proizvodi iz narudžbi koje su potvrđene od strane administratora i imaju status *Uspješno završena narudžba!*. Na lijevoj strani je ispisan broj prodanih komada najprodavanijeg proizvoda. Pored toga se nalazi naziv proizvoda, kategorija kojoj pripada te cijena sa već uračunatim popustom ukoliko je popust dostupan. Na desnoj strani prikazana je slika tog proizvoda. Najprodavaniji proizvod prikazan je na Slici 3.25. Ovakav tip podatka može biti koristan vlasniku internetske trgovine kako bi vidio što ljudi najviše žele i kolika je cijena koja je najpristupačnija većini kupaca.



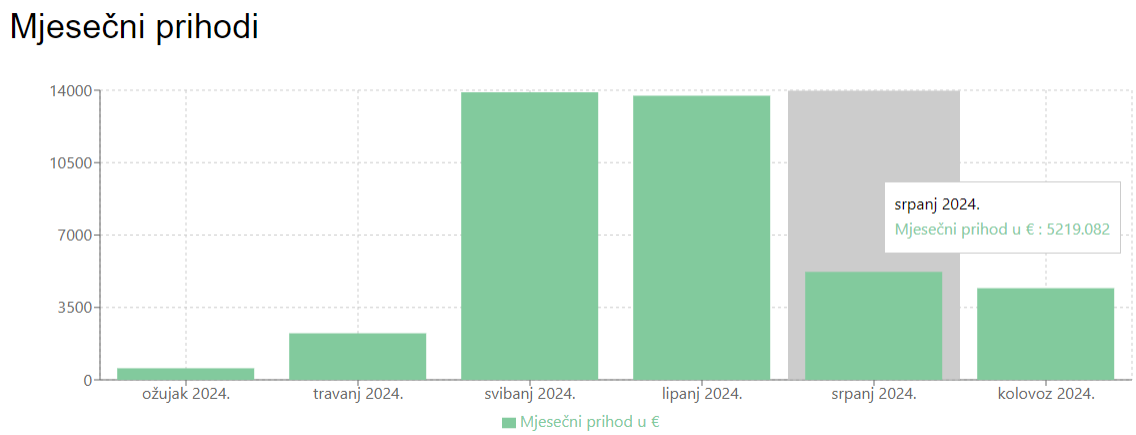
Slika 3.25: Prikaz najprodavanijeg proizvoda

Sljedeći podatak kojeg administrator može vidjeti je prikaz broja narudžbi po mjesecima. Prikazan je u obliku grafa kreiranog pomoću knjižnice grafova Recharts [15]. S lijeve strane se nalazi os s brojevima narudžbi, a na dnu mjeseci u godini. Na grafu su prikazani svi mjeseci u kojima je ostvarena bilo kakva prodaja te se prelaskom mišem preko stupca može vidjeti točan broj narudžbi u tom mjesecu. Ovaj graf je prikazan na Slici 3.26 i može detaljno prikazati administratoru kako internetska trgovina posluje u određenom mjesecu u godini te kada bi možda bilo potrebno dodati neke akcije ili ponude kako bi se prodaja pokušala poboljšati.



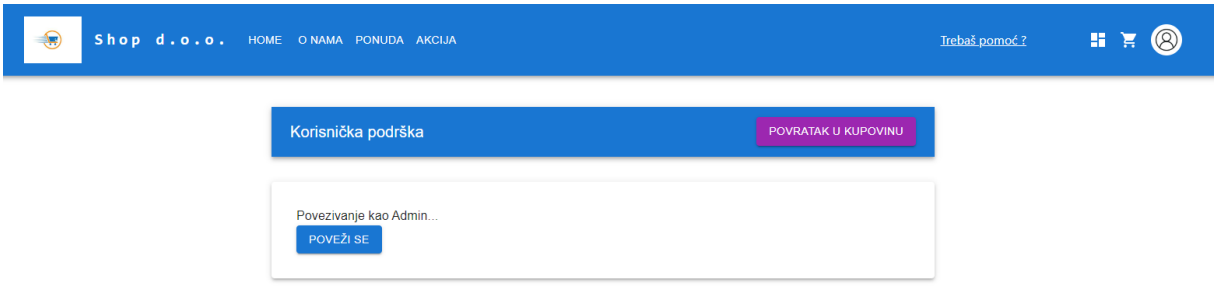
Slika 3.26: Graf broja narudžbi po mjesecima

Sljedeći podatak koji je dostupan administratoru je podatak o mjesečnim prihodima internetske trgovine. Prikazan je u obliku još jednog grafa izrađenog pomoću knjižnice Recharts [15]. Na lijevoj strani je os s iznosima u eurima, a na dnu mjeseci u godini. Na grafu su prikazani mjeseci u kojima je ostvarena prodaja te se prelaskom mišem preko stupca može vidjeti točan iznos koji je zarađen u tom mjesecu. Graf je prikazan na Slici 3.27 i može poslužiti administratoru za lakšu vizualizaciju zarade i mjesečnih prihoda te usporedbu prihoda koje je zabilježio u računovodstvu i prihoda koje je zabilježila aplikacija.



Slika 3.27: Graf mjesečnih prihoda

Za pružanje korisničke podrške služi ista poveznica koja se nalazi na navigacijskoj traci pod nazivom *Trebaš pomoć ?*. Pritiskom na poveznicu otvara se pogled za povezivanje prikazan na Slici 3.28 u kojem aplikacija automatski prepoznaje kako je riječ o administratoru i nudi tipku *Poveži se* bez potrebe za upisivanjem imena. Nakon povezivanja otvara se pogled za razgovor između administratora i korisnika ranije prikazan na Slici 3.17.



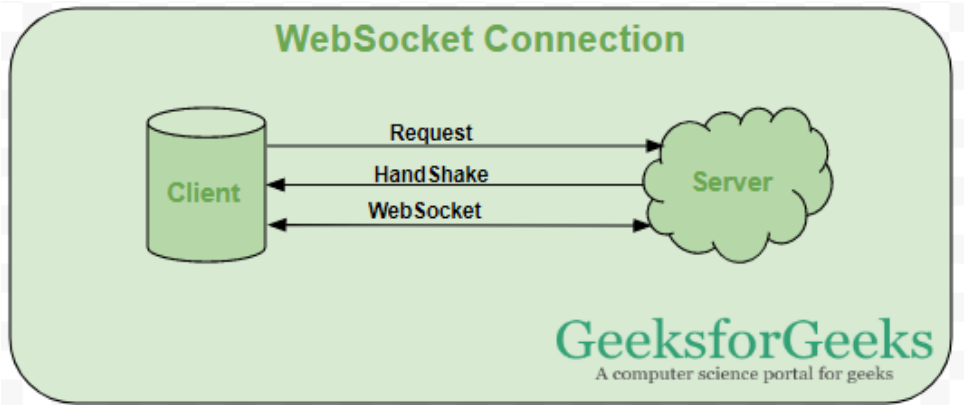
Slika 3.28: Pogled za povezivanje u korisničku podršku kao administrator

4 RAZVOJ I IMPLEMENTACIJA SPECIFIČNE APLIKACIJSKE FUNKCIONALNOSTI

4.1 Komunikacija između korisnika i administratora

4.1.1 WebSocket komunikacija

Funkcionalnost komunikacije između korisnika i administratora u realnom vremenu omogućena je korištenjem internetskih utičnica (eng. web socket). WebSocket je tehnologija koja omogućuje dvosmjernu komunikaciju između klijenta i poslužitelja preko dugotrajno otvorenog priključka koji se naziva utičnica (eng. socket). Prilikom prvog zahtjeva za komunikacijom otvara se komunikacija između klijenta i poslužitelja te ostaje otvorena sve dok jedan od njih ne odluči prekinuti vezu. Za razliku od HTTP protokola, kod kojeg klijent mora slati zahtjev za svaki odgovor od poslužitelja, WebSocket omogućuje poslužitelju da pošalje podatke klijentu čim su dostupni bez potrebe za slanjem novih zahtjeva. Ovakav način komunikacije prikazan je na Slici 4.1. Prvi korak za komunikaciju je slanje zahtjeva (eng. Request) poslužitelju, nakon toga poslužitelj potvrđuje komunikaciju (eng. HandShake) te je komunikacija uspostavljena. WebSocket komunikacija najčešće se koristi kod aplikacija za razgovor, aplikacija koje zahtijevaju osvježenje u realnom vremenu te raznih igara [18].



Slika 4.1: Proces uspostave komunikacije kod WebSocketa [18]

4.1.2 Postavljanje poslužiteljske strane aplikacije

Spring Boot nudi podršku za WebSocket komunikaciju putem STOMP (eng. Simple Text Oriented Messaging Protocol) protokola. Ovaj protokol je jednostavan za implementaciju i razumijevanje te koristi naredbe kao što su *CONNECT*, *SEND* i *SUBSCRIBE* za uspostavu

komunikacije [19]. Za korištenje WebSocket komunikacije na poslužiteljskoj strani aplikacije, u Spring Bootu, prvo je potrebno dodati zavisnost u *pom.xml* datoteku u kojoj se nalaze sve zavisnosti korištene u projektu. Na Slici 4.2 prikazana je zavisnost koju je potrebno dodati i naziva se *spring-boot-starter-websocket* uz odabir posljednje stabilne verzije ove zavisnosti.

Slika 4.2: Zavisnost potrebna za WebSocket komunikaciju

Nakon dodavanja zavisnosti slijedi pisanje konfiguracijske klase koja služi za postavljanje WebSocketa, definiranje krajnjih točaka komunikacije te konfiguraciju posrednika (eng. message broker) koji omogućuje dvosmjernu komunikaciju između klijenta i poslužitelja u stvarnom vremenu. Ova klasa prikazana je na Slici 4.3.

Na samom vrhu se nalaze anotacije *@Configuration* i *@EnableWebSocketMessageBroker*. Anotacija *@Configuration* označava kako je ovo konfiguracijska klasa i da Spring Boot prepoznaje ovu klasu kao izvor definicija. Anotacija *@EnableWebSocketMessageBroker* omogućuje WebSocket podršku uz STOMP protokol te automatski konfigurira potrebne komponente kako bi se omogućila komunikacija.

Metoda *registerStompEndpoints()* definira krajnje točke na koje se klijenti mogu povezivati. Kako bi se klijenti povezali na ovu utičnicu moraju pristupiti krajnjoj točki na url adresi */ws* i to je definirano pomoću naredbe *registry.addEndpoint(„/ws“)*. U nastavku se još nalazi naredba *setAllowedOriginPatterns(„*“)* koja omogućuje CORS (eng. Cross-Origin Resource Sharing) komunikaciju, odnosno aplikacija može primati zahtjeve za spajanje na WebSocket iz bilo kojeg izvora. Naredba *withSockJS()* omogućuje SockJS podršku koja osigurava kompatibilnost s preglednicima koji ne podržavaju WebSocket protokol.

Metoda *configureMessageBroker()* služi za postavljanje i konfiguraciju posrednika poruka te definira prefikse za odredišta poruka unutar aplikacije. Naredba *registry.setApplicationDestinationPrefixes(„/app“)* definira prefiks */app* za sve poruke koje će biti usmjerene prema metodama označenim s anotacijom *@MessageMapping* unutar aplikacije. Svaki zahtjev poslan na ovu destinaciju biti će obrađen unutar aplikacije te procesuiran prema definiranoj destinaciji. Naredba *registry.enableSimpleBroker(„/user“)* aktivira jednostavni

posrednik poruka koji obrađuje i distribuira poruke klijentima. Prefiks */user* označava da će jednostavni posrednik poruka slati poruke klijentima na destinacije koje počinju s ovim prefiksom.

Naredba *registry.setUserDestinationPrefix(„/user“)* postavlja prefiks */user* za destinacije specifične za pojedinačne korisnike. Ova funkcionalnost omogućuje slanje poruka direktno određenim korisnicima, na temelju njihove sesije ili identiteta [20].

```
Dominik Krušvar
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    no usages  Dominik Krušvar
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry){
        registry.addEndpoint( ...paths: "/ws").setAllowedOriginPatterns("*").withSockJS();
    }

    no usages  Dominik Krušvar
    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.setApplicationDestinationPrefixes("/app");
        registry.enableSimpleBroker( ...destinationPrefixes: "/user");
        registry.setUserDestinationPrefix("/user");
    }
}
```

Slika 4.3: Konfiguracijska klasa WebSocket komunikacije

4.1.3 Postavljanje klijentske strane aplikacije

Za postavljanje WebSocket komunikacije u Reactu potrebno je dodati SockJS i stomp knjižnice u projekt. SockJS knjižnica omogućuje WebSocket komunikaciju između klijenta i poslužitelja uz dodatnu podršku za slične mehanizme komunikacije ukoliko preglednik ne podržava WebSocket komunikaciju. Posebno je korisna za aplikacije široke primjene gdje neće uvijek biti dostupna podrška za WebSocket komunikaciju. Za dodavanje u projekt potrebna je instalacija pomoću naredbe *npm install sockjs-client* [21]. Stomp također omogućuje WebSocket komunikaciju u React aplikacijama. Stomp pruža strukturiran način za slanje i primanje poruka kod poslužitelja koji rade pomoću STOMP protokola, kao što je to slučaj kod Spring Boota. Za dodavanje u projekt potrebna je instalacija pomoću naredbe *npm install @stomp/stompjs* [22].

4.1.4 Uspostavljanje komunikacije

Prilikom dolaska na pogled (Slika 3.16) kao korisnik ili na pogled (Slika 3.28) kao administrator, potrebno se povezati u WebSocket komunikaciju. Pri dolasku na stranicu pokreće se *useEffect* kuka prikazana na Slici 4.4 koja dohvaća korisničko ime sa web pohrane preglednika te postavlja to ime u korisničke podatke za komunikaciju.

```
useEffect( effect: () : void => {
    const username = AuthService.getUsername();
    if (username) {
        setUserData( value: (prevState : UserData ) : {connected: boolean, message: string, receivername: string, username: any} => ({
            ...prevState,
            username: username,
        }));
    }
}, deps: []);
```

Slika 4.4: Kuka useEffect za dohvat podataka

U kodu prikazanom na Slici 4.5 provjerava se radi li se o administratoru ili korisniku. Naredba *AuthService.getUsername()* dohvaća podatke o trenutno prijavljenoj osobi sa web pohrane unutar preglednika te provjerava odgovara li korisničko ime imenu *admin*. Ukoliko ime odgovara prikazat će se poruka *Povezivanje kao Admin…* te tipka *Poveži se* pomoću koje je moguće pristupiti WebSocket komunikaciji.

```
<Paper elevation={3} style={{ padding: '2rem', marginTop: '2rem' }}>
    {(AuthService.getUsername() === "admin") ? (
        <Typography>Povezivanje kao Admin...</Typography>
    ) : (
        <TextField
            fullWidth
            id="user-name"
            placeholder="Unesi ime"
            value={userData.username}
            onChange={handleUsername}
            margin="normal"
            variant="outlined"
        />
    )}
    <Button variant="contained" color="primary" onClick={registerUser}>
        Poveži se
    </Button>
</Paper>
```

Slika 4.5: Kod za identifikaciju trenutno prijavljene osobe

Kada korisnik pristupi ovom pogledu prikazuje se tekstni okvir za upis imena. Ukoliko je korisnik trenutno prijavljen u aplikaciju, tekstni okvir će biti već ispunjen njegovim imenom, ako korisnik nije prijavljen tekstni okvir će sadržavati poruku *Unesi ime*. Prilikom upisa imena aktivira se funkcija *handleUsername()* prikazana na Slici 4.6 koja sprema uneseno ime u korisničke podatke koji će se koristiti prilikom ove WebSocket komunikacije.

```
1 usage  Dominik Krušvar
const handleUsername = (event: React.ChangeEvent<HTMLInputElement>) : void  => {
    setUserData( value: { ...userData, username: event.target.value });
};
```

Slika 4.6: Funkcija za obradu unesenog imena korisnika

Prilikom pritiska tipke *Poveži se* poziva se funkcija *registerUser()* koja poziva funkciju *connect()* prikazanu na Slici 4.7 koja omogućuje uspostavljanje WebSocket komunikacije s poslužiteljem. Otvara se nova SockJS veza prema krajnjoj točki definiranoj u poslužiteljskom dijelu aplikacije i stvara STOMP klijent koji se koristi za razmjenu poruka preko WebSocketa. Ukoliko je veza uspješno uspostavljena poziva se funkcija *onConnected()*, a ako je došlo do pogreške poziva se funkcija *onError()* koja ispisuje grešku u konzolu.

```
1 usage  Dominik Krušvar
const connect = () : void  => {
    let Sock : WebSocket  = new SockJS( url: 'http://localhost:8080/ws');
    stompClient = over(Sock);
    stompClient.connect({}, onConnected, onError);
};
```

Slika 4.7: Funkcija za spajanje sa poslužiteljskim dijelom aplikacije

Funkcija *onConnected()* ažurira stanje korisničkih podataka, odnosno mijenja status povezan iz stanja laž (eng. false) u stanje istina (eng. true). STOMP klijent se pretplaćuje na kanal */user/{username}/private* gdje *{username}* predstavlja korisničko ime. Poruke poslane na ovaj kanal dostavljaju se korisniku. Druga pretplata je na kanal */admin* koji se koristi za poruke koje administrator šalje korisnicima. Nakon toga se poziv funkcija *userJoin()*. Funkcija *onConnected()* prikazana je na Slici 4.8.

Slika 4.8: Funkcija za pretplatu na kanal za komunikaciju

Funkcija *userJoin()* stvara objekt tipa *ChatMessage* koji sadrži informacije o imenu pošiljatelja, datum, poruku te status. Ispunjeni su samo ime pošiljatelja te status sa porukom *JOIN* dok su ostala dva podatka prazna. Ova poruka služi kako bi obavijestila poslužiteljski dio aplikacije da se taj korisnik pridružio komunikaciji. Naredba *stompClient.send()* šalje poruku poslužiteljskom dijelu aplikacije na odredište */app/message* pri čemu poruku pretvara u JSON oblik prije slanja. Ova funkcija prikazana je na Slici 4.9 te je nakon ovog koraka komunikacija uspješno uspostavljena.

Slika 4.9: Funkcija za slanje poruke o pridruženju komunikaciji

4.1.5 Slanje i primanje poruka

Nakon uspješnog uspostavljanja komunikacije otvara se pogled prikazan na Slici 3.17. Za izgled toga pogleda zaslužan je kod sa Slike 4.10. U prvoj liniji koda provjerava se je li osoba povezana u razgovor, odnosno jesu li ispunjeni ranije opisani koraci. Ostatak koda sastoji se od Material UI komponenata kao što su podloga *Paper*, prostor za komponente *Box*, kartica *Tab*, lista korisnika *List* i tipka za slanje *Button* [16]. Dio koda unutar komponenti je skriven pošto se ovaj pogled sastoji od velikog broja mogućnosti te se tako i kod sastoji od prilično

mnogo linija kako bi sve izgledalo i funkcioniralo dobro. Na dnu ovog koda nalazi se mogućnost slanja poruke pomoću tipke i mogućnost slanja poruke pritiskom tipke *Enter* na tipkovnici. Pritiskom jedne od ovih tipki poziva se funkcija *sendValue()* ukoliko korisnik šalje poruku administratoru ili funkcija *sendPrivateValue()* ukoliko administrator šalje poruku korisniku.

```
{userData.connected ? (
    <Paper elevation={3} sx={{ mt: 3 }}>
        <Box display="flex" flexDirection="column" height="80vh">
            <Tabs value={tab} onChange={handleTabChange} indicatorColor="primary" textColor="primary" variant="fullWidth">
                {[...privateChats.keys()].map((name : string ) => (
                    <Tab key={name} label={name} value={name} />
                ))}
            </Tabs>
            <Box flexGrow={1} overflow="auto">
                <List>
                    {(privateChats.get(tab) || []).map((chat : ChatMessage , index : number ) => (
                        <ListItem
                            key={index}
                            alignItems="flex-start"
                            style={{...}}
                        ...>
                    ))}
                </List>
            </Box>
            <Box p={2}>
                <Grid container spacing={2}>
                    <Grid item xs={10}...>
                    <Grid item xs={2}>
                        <Button
                            fullWidth
                            variant="contained"
                            color="primary"
                            onClick={tab === "Admin" ? sendValue : sendPrivateValue}
                        >
                            Pošalji
                        </Button>
```

Slika 4.10: Material UI komponente

Ove dvije funkcije za slanje poruka su prilično slične pa će ovdje biti opisana i prikazana samo funkcije *sendValue()* koja šalje poruku administratoru. Funkcija je prikazana na Slici 4.11 i na samom početku provjerava postoji li *stompClient*, odnosno je li veza sa WebSocketom uspostavljena. Nakon što je utvrđeno da veza postoji, stvara se novi objekt tipa *ChatMessage* koji sadrži podatke o imenu pošiljatelja, imenu primatelja (koji je u ovom slučaju *admin)*, tekst poruke, trenutno vrijeme kada se poruka šalje te status *MESSAGE* koji označava kako je ovo standardna poruka. Nakon kreiranja objekta dohvaćaju se trenutne poruke u razgovoru s administratorom ili se kreira novi razgovor ukoliko još nema poruka. Nova poruka se dodaje na kraj razgovora te se ažurirana verzija razgovora sprema pomoću naredbe *setPrivateChats()*. Naredbom *stompClient.send()* šalje se poruka prema poslužiteljskom dijelu aplikacije na putanju */app/private-message* u JSON obliku. Nakon slanja poruke resetira se podatak trenutno

spremljen u poruci i postavlja na prazan skup znakova kako bi korisnik mogao nastaviti pisati nove poruke.

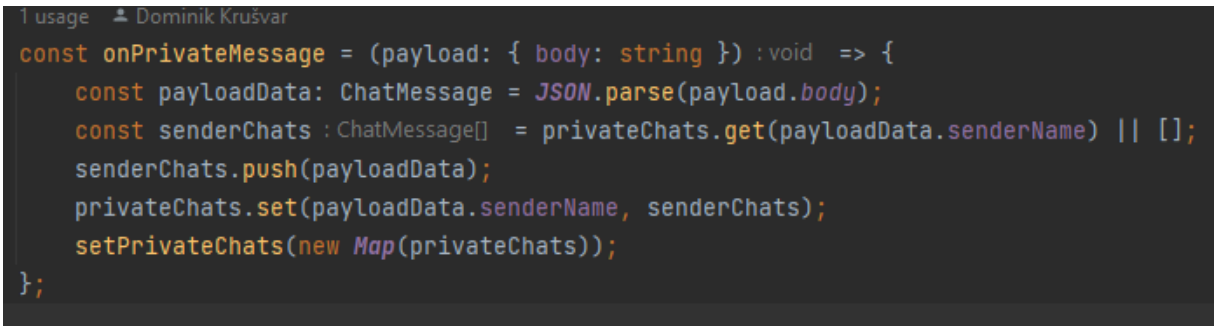
```
2 usages  Dominik Krušvar *
const sendValue = () : void  => {
    if (stompClient) {
        const chatMessage: ChatMessage = {
            senderName: userData.username,
            receiverName: "admin",
            message: userData.message,
            date: new Date().toISOString(),
            status: "MESSAGE"
        };

        const newChats : ChatMessage[]  = privateChats.get("admin") || [];
        newChats.push(chatMessage);
        privateChats.set("admin", newChats);
        setPrivateChats(new Map(privateChats));

        stompClient.send("/app/private-message", {}, JSON.stringify(chatMessage));
        setUserData( value: { ...userData, message: "" });
    }
}
```

Slika 4.11: Funkcija za slanje poruke prema administratoru

Za slanje i obradu poruka na poslužiteljskom dijelu aplikacije služi Kontroler (eng. controller) prikazan na Slici 4.12. U prvoj liniji koda nalazi se anotacija *@Controller* što označava da ova klasa upravlja zahtjevima, u ovom slučaju WebSocket komunikacijom. Anotacija *@CrossOrigin* omogućuje prihvaćanje zahtjeva s adrese koju koristi klijentska strana aplikacije. Unutar kontrolera se pomoću anotacije *@Autowired* dodaje klasa *SimpMessagingTemplate* koja u Spring Bootu služi za slanje poruka prema WebSocket klijentima. Anotacija *@MessageMapping* povezuje dolazne poruke s odredištem */app/private-message* s metodom *receiveMessage()* unutar kontrolera koja je zadužena za njihovu obradu. Unutar ove metode prihvaća se poruka i mapira na objekt tipa *Message*. Pomoću metode *convertAndSendToUser()*, uz dodavanje argumenata ime pošiljatelja, odredišta */private* te teksta poruke, poruka se šalje primatelju.

Slika 4.12: Kontroler za slanje i obradu poruka

Nakon što poslužiteljski dio aplikacije odradi svoj zadatak, poruka se ponovno vraća u klijentski dio aplikacije te se ovdje prihvaća pomoću dvaju funkcija. Funkcija *onPrivateMessage()* služi za prihvaćanje poruka poslanih od strane korisnika, a funkcija *onAdminMessage()* za prihvaćanje poruka poslanih od strane administratora. Funkcije su veoma slične. U nastavku će biti opisana funkcija za prihvaćanje poruka poslanih od strane korisnika (Slika 4.13). Prvi korak je dekodiranje primljenog podatka u JSON obliku i pretvaranje u JavaScript objekt tipa *ChatMessage*. Nakon toga se dohvaća trenutni niz poruka od tog pošiljatelja ili se stvara novi niz ukoliko je ovo prva poruka. Primljena poruka se dodaje u niz te se ažurira mapa dosad primljenih poruka. U posljednjoj liniji koda poziva se naredba *setPrivateChats()* sa novom kopijom mape razgovora što signalizira Reactu da ponovno učita komponentu razgovora i prikaže najnoviju poruku.



Slika 4.13: Funkcija za prihvaćanje poruka poslanih od strane korisnika

5 ZAKLJUČAK

U ovom radu predstavljena je web aplikacija za internetsku trgovinu temeljena na REST arhitekturi. Aplikacija je podijeljena na administracijski i korisnički dio. Administrator ima mogućnost kreiranja, uređivanja i brisanja proizvoda, u svakom trenutku može vidjeti statistiku prodaje u internetskoj trgovini te može odobravati korisnikove narudžbe. S druge strane, korisnik nakon registracije može dodavati u košaricu i kasnije kupovati proizvode, filtrirati proizvode prema svojim željama, pregledati svoju povijest narudžbi te uređivati svoje osobne podatke. Ukoliko korisniku zatreba pomoć pri kupnji, može se obratiti administratoru porukom te komunicirati s njim u realnom vremenu.

Spring Boot radni okvir je iznimno popularan u svijetu web aplikacija. U ovom projektu je korišten zbog svoje univerzalnosti, jednostavnosti te produktivnosti. Sama inicijalizacija i postavljanje projekta se odvijaju brzo, pruža kvalitetnu podršku za dodatne mogućnosti kao što su autentifikacija korisnika, sigurnost te povezivanje s bazom podataka. Uz veliku zajednicu programera koji koriste Spring Boot lakše je pronaći odgovore na sva pitanja i riješiti potencijalne zastoje i probleme u projektu.

React knjižnica korištena je za izradu klijentskog dijela web aplikacije. Karakteristike poput komponentnog pristupa, virtualnog DOM-a te jednosmjernog povezivanje podataka temelji su Reacta te ga čine veoma popularnom i moćnom knjižnicom. Uz pomoć *useState* i *useEffect* kuka moguće je upravljati stanjem varijabli te ažuriranjem aplikacije i korisničkog sučelja. Korištenje TypeScripta uz React dodaje potrebu za definiranjem tipova varijabli i daje dodatnu mogućnost provjere koda prije izvršavanja. Za uređivanje izgleda stranice korištena je React knjižnica komponenti Material UI koja nudi velik skup komponenata i stilova za izradu modernih web stranica.

MongoDB sustav za upravljanje bazom podataka iznimno je kompatibilan sa Spring Bootom. Baza podataka je dokumentna što omogućuje fleksibilnost kod spremanja podataka koji variraju u strukturi. Također, moguće je proširivanje i dodavanje novih podataka u bazu bez zahtjevnih preinaka i mijenjana strukture baze podataka.

Za daljnji razvoj i poboljšanje ove aplikacije bilo bi potrebno poboljšati sustav registracije i prijave korisnika. Trebalo bi dodati provjeru valjanosti adrese e-pošte tako što korisnik treba potvrditi registraciju unutar svoje e-pošte. Potrebno je dodati i

mogućnost resetiranja lozinke ukoliko je korisnik zaboravio prijašnju kako ne bi morao raditi novi račun samo zbog zaboravljene lozinke. Jedna od bitnih stavki web aplikacija za internetsku trgovinu je i mogućnost kartičnog plaćanje kod kupnje proizvoda. U ovom projektu ta mogućnost nije dodana zbog iznimnih sigurnosnih mjera koje trebaju biti poduzete kako ne bi došlo do gubitka ovako vrijednih podataka. Nadalje, moguće je i razvijanje umjetne inteligencije koja bi zamijenila administratora i uspješno pružala korisničku podršku u razgovoru s korisnikom za većinu često postavljanih pitanja.

LITERATURA

[1] aws.amazon, What is a RESTful API?, s interneta, https://aws.amazon.com/what-is/restful-api/, 21. srpanj 2024

[2] spring.io, Web Applications, s interneta, https://spring.io/web-applications, 21. srpanj 2024.

[3] MongoDB, Why Use MongoDB and When to Use It?, s interneta, https://www.mongodb.com/resources/products/fundamentals/why-use-mongodb, 21. srpanj 2024.

[4] React, React s interneta, https://react.dev/, 21. srpanj 2024.

[5] Axios, Getting Started, s interneta, https://axios-http.com/docs/intro, 21. srpanj 2024.

[6] geeksforgeeks, Spring Boot - Architecture, s interneta, https://www.geeksforgeeks.org/spring-boot-architecture/, 22. srpanj 2024.

[7] Spring Initializr, s interneta, https://start.spring.io/, 22. srpanj 2024.

[8] spring.io, Spring Security, s interneta, https://spring.io/projects/spring-security, 23. srpanj 2024.

[9] geeksforgeeks, ReactJS Virtual DOM, s interneta, https://www.geeksforgeeks.org/reactjs-virtual-dom/, 27. srpanj 2024.

[10] P. Mondal, React Data Binding, s interneta, https://blog.stackademic.com/react-data-binding-a-comprehensive-guide-to-one-way-and-two-way-binding-6fb945add5ed, 27. srpanj 2024.

[11] React, Introducing Hooks, s interneta, https://legacy.reactjs.org/docs/hooks-intro.html, 27. srpanj 2024.

[12] React, Using the State Hook, s interneta, https://legacy.reactjs.org/docs/hooks-state.html, 27. srpanj 2024.

[13] React, Using the Effect Hook, s interneta, https://legacy.reactjs.org/docs/hooks-effect.html, 27. srpanj 2024.

[14] AppsDevPro, React with Typescript vs JavaScript, s interneta https://www.appsdevpro.com/blog/react-with-typescript-vs-javascript/, 31. srpanj 2024.

[15] F. Akinyemi, How to create a chart in React with Recharts, s interneta, https://dev.to/femi_akinyemi/how-to-create-a-chart-in-react-with-recharts-2b58, 31. srpanj 2024.

[16] MaterialUI, Move faster with intuitive React UI tools, s interneta, https://mui.com/, 31. srpanj 2024.

[17] IntelliJ IDEA, Features overview, s interneta, https://www.jetbrains.com/idea/features/, 10. kolovoz 2024.

[18] geeksforgeeks, What is web socket and how it is different from the HTTP?, s interneta, https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/ , 11. kolovoz 2024.

[19] geeksforgeeks, STOMP Protocol, s interneta, https://www.geeksforgeeks.org/stomp-protocol/, 11. kolovoz 2024.

[20] spring.io, Using WebSocket to build an interactive web application, https://spring.io/guides/gs/messaging-stomp-websocket, 11. kolovoz 2024.

[21] npm, sockjs, s interneta, https://www.npmjs.com/package/sockjs, 11. kolovoz 2024.

[22] npm, @stomp/stompjs, s interneta, https://www.npmjs.com/package/@stomp/stompjs, 11. kolovoz 2024.

POPIS SLIKA

SAŽETAK

U ovom radu predstavljena je web aplikacija za internetsku trgovinu koja ima administracijski i korisnički dio. Administrator ima mogućnost dodavanja proizvoda i uređivanja informacija o njima, potvrđivanja narudžbi te pregleda statistike prodaje proizvoda pomoću grafikona. Korisnik nakon registracije može uređivati svoje podatke te filtrirati i kupovati proizvode dostupne u internetskoj trgovini. U svakom trenutku moguća je razmjena poruka između administratora i korisnika u realnom vremenu. Za razvoj poslužiteljskog dijela aplikacije korišten je Spring Boot uz sustav za upravljanje bazom podataka MongoDB. Za razvoj klijentskog dijela aplikacije korišten je React uz pomoć Material UI-a za uređivanje izgleda web stranice.

Ključne riječi: web aplikacija, internetska trgovina, Spring Boot, React, MongoDB,

ABSTRACT

This paper presents a web application for online shopping, divided into an administrative and a user section. The administrator has the ability to add products and edit their information, confirm orders, and view product sales statistics using charts. After registering, the user can edit their information and filter and purchase products available in the web store. At any time, real-time message exchange between the administrator and the user is possible. The server-side of the application was developed using Spring Boot with MongoDB as the database management system. The client-side of the application was developed using React with the help of Material UI for styling the website.

Keywords: web application, web store, Spring Boot, React, MongoDB