WEB APLIKACIJA ZA PRODAJU ELEKTRONIČKIH PROIZVODA

Batistić, Ilario

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split / Sveučilište u Splitu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:228:919457

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-13**

Repository / Repozitorij:

Repository of University Department of Professional Studies

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

ILARIO BATISTIĆ

Z A V R Š N I R A D

Web aplikacija za prodaju elektroničnih proizvoda

Split, rujan 2023.

SVEUČILIŠTE U SPLIT

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

Predmet: Informacijski sustavi

Z A V R Š N I R A D

Kandidat: Ilario Batistić

Naslov rada: Web aplikacija za prodaju elektroničnih proizvoda

Mentor: mr. sc. Karmen Klarin, viši predavač

Split, rujan 2023.

Sadržaj

SAŽETAK 1
SUMMARY 2
1. **Uvod** 3
2. **Korištene Tehnologije** 4
2.1. TypeScript 4
2.2. React 4
2.3. Redux Toolkit 5
2.4. Material UI 6
2.5. Java 6
2.6. Spring Boot 7
2.7. MySQL 8
2.8. JSON Web Token(JWT) 9
3. **Izrada baze podataka** 10
3.1. Entiteti i relacije 10
4. **Izrada aplikacije** 12
4.1 Izrada sustava za autentifikacija i autorizaciju 12
4.1.1 Početna stranica 12
4.1.2 Registracija kupaca 14
4.1.3 Prijava korisnika 15
4.1.4 Korisnikov profil i odjava 17
4.2 Izrada administracijskog sučelja 18
4.2.1 Upravljanje korisnika 18
4.2.2 Pregled adresa 21
4.2.3 Upravljanje marki 22
4.2.4 Upravljanje kategorija 24
4.2.5 Upravljanje proizvoda 26
4.3 Izrada sučelja za kupce 30
4.3.1 Pregled proizvoda 30
4.3.2 Dodavanje proizvode u košaru i popis želja 31
4.3.3 Kupnja proizvoda i plaćanje narudžbe 34
4.3.4 Pregled narudžbi, popis želja i potvrda dostave narudžbe 35
5. **Zaključak** 37
LITERATURA 37

SAŽETAK

Spark Mart je *web shop* aplikacija specijalizirana za prodaju raznih elektroničnih uređaja. Omogućava dodavanje različitih marki i kategorija, korisnika, narudžbi i proizvoda. Proizvodi su povezani s markama i kategorijama, dok su narudžbe povezane s korisnikom i proizvodom. Za uspješnu implementaciju *backend* koristi se Java i Spring Boot, a za *frontend* TypeScript i React. Za bazu podataka koristi se MYSQL.

Web aplikacija omogućava sve CRUD (engl. *Create, Read, Update, Delete*) operacije na entitetima, uz to omogućava korisniku da proizvode stavi na liste želja i da iste kupi, čime se uspješno kreira narudžba. Kada se narudžba isporuči, korisnik mora potvrditi dostavu naručenih proizvoda, a kada je potvrđen primitak svih proizvoda narudžba se uspješno zatvara.

Ključne riječi: React, Spring Boot, MYSQL, Web Shop, Java

SUMMARY

Web Application for the Sale of Electronic Products

Spark Mart is a web shop application that specializes in the sale of a variety of different electronic devices. It enables the creation of new users, addresses, brands, categories, products, and orders. The products are associated with brands and categories, while the orders are linked to users and products. For the backend, Java and Spring Boot were utilized, and for the frontend, TypeScript and React were employed. MYSQL serves as the chosen database system.

The web app allows for the execution of all CRUD (*Create*, *Read*, *Update*, *Delete*) operations on entities. Furthermore, users can add products to their wishlists and make purchases, which automatically generates corresponding orders. After the order is dispatched, users confirm the receipt of each product they ordered. Once all products are confirmed as delivered, the order is automatically closed.

Keywords: React, Spring Boot, MYSQL, Web Shop, Java

1. Uvod

Spark Mart je *online web shop* aplikacija koja je specijalizirana za prodaju elektroničnih uređaja iz različitih kategorija: televizija, laptop, PC dijelovi, PC dodaci, kamere, dronovi, eksterna memorija.

Cilj je bila izrada *Web* aplikacije koju bi klijenti mogli koristiti kao poslovnu aplikaciju. Spark Mart je savršena prilika za upotrebu nove tehnologije i nadogradnju znanja temeljenom na već korištenim tehnologijama. Spark Mart se sastoji od tri dijela: *frontend*, *backend* i baza podataka (engl. *database*). Za *frontend* se koristi React inicijaliziran uz pomoć Vitea, ali JavaScript nije korišten, nego TypeScript sa SWC kompajlerom.

Za *backend* se koristi Java i njezin najpopularniji i najkorišteniji *web framework*, Spring Boot. Spring Boot je inicijaliziran pomoću internetskog alata koji se zove Spring Initializer. Za bazu podataka se koristi MYSQL radi njezine lagane integracije sa Spring Boot *frameworkom* koji automatski generira shemu za Spark Mart bazu podataka.

Poglavlje Korištene tehnologije opisuje sve tehnologije koje su bile iskorištene tijekom razvoja *web* aplikacije. Opisuje osnovnu povijest tehnologije, što te tehnologije nude i zašto su korištene. Poglavlje Izrada aplikacija opisuje poduzete korake za razvoj *web* aplikacije. Sastoji se od tri potpoglavlja: izrada sustava za autentifikaciju i autorizaciju, izrada administracijskog sučelja i izrada sučelja za kupce.

Svako potpoglavlje sadržava dodatna poglavlja koje opisuju dijelove *web* aplikacije. Poglavlje zaključak sadržava sveukupna mišljenja o *web* aplikaciji i nove značajke koje se mogu dodati aplikaciji.

2. Korištene tehnologije

2.1. TypeScript

TypeScript je programski jezik otvorenog kôda koji je razvio Microsoft 2012. godine. Stvoren je kao nadskup JavaScripta, što znači da je bilo koji kôd napisan u TypeScriptu valjan i u JavaScriptu. No, TypeScript donosi značajne dodatke koji proširuju i unapređuju postojeće mogućnosti JavaScripta, čime se pruža bolje iskustvo razvojnim stručnjacima.

Ključna karakteristika TypeScripta je njegov tipizirani sustav. Tijekom svog životnog ciklusa, jedna JavaScript varijabla može sadržavati različite tipove podataka, što može uzrokovati greške koje nije lako ispraviti. No, TypeScript omogućava definiranje varijabli sa snažno definiranim tipovima podataka. Strogo definirani tipovi varijabli omogućuju ranu detekciju i otkrivanje mogućih grešaka vezanih uz tipove varijabli (na primjer, pokušaj korištenja "*length*" nad varijablom koja sadrži samo cijeli broj).

TypeScript sadrži već unaprijed definirane tipove poput broja, niza, logičkog tipa i slično, no, omogućava i stvaranje potpuno novih tipova koji korisnicima pomažu pri programiranju u TypeScriptu. Bitno je napomenuti da je TypeScript uvijek usklađen s najnovijim verzijama ECMAScripta te je kompatibilan s različitim internetskim preglednicima i okvirima.

Zbog toga je migracija projekta s JavaScripta na TypeScript relativno jednostavna. Zbog svoje korisnosti, TypeScript je postao izrazito popularan među JavaScript programerima. Često je podržan u okvirima poput Reacta ili Vuea, ili čak postaje ekskluzivna opcija za pisanje kôda u određenim okvirima, kao što je slučaj s Angularom [1].

2.2. React

React je programerska biblioteka otvorenog kôda koju je kreirao Facebook. Namijenjena je pojednostavljivanju procesa izrade interaktivnih korisničkih sučelja (engl. *user interface*) za *web* i mobilne aplikacije. Od svog službenog predstavljanja 2013. godine,

React je stekao slavu zbog svoje arhitekture zasnovane na komponentama, efikasnog modela za prikaz i naglaska na ponovnoj upotrebljivosti i održavanju.

React aplikacija sastoji se od komponenata, koji su elementi korisničkog sučelja osmišljeni za ponovnu upotrebu.

Svaka komponenta može sadržavati vlastito stanje, predstavljajući dinamične podatke koji se mogu mijenjati po potrebi, te *props* (svojstva), konstantne vrijednosti koje se prenose od nadređene (roditeljske) komponente do podređene (dječje) komponente. Za efikasno ažuriranje stvarnog Modela objekata dokumenta (engl. *Document Object Model*), React koristi Virtualni DOM. Kada se stanje aplikacije promijeni, React stvara virtualnu reprezentaciju korisničkog sučelja, izračunavajući minimalne potrebne promjene za ažuriranje stanja.

Zbog svoje popularnosti, React sadrži mnogo vanjskih paketa koji omogućavaju različite funkcionalnosti, uključujući usmjeravanje (*React Router*), upravljanje stanjem (*Redux*), upravljanje obrascima (*React Hook Form*), unaprijed izrađene komponente (*React Bootstrap*, MUI, itd.) [2].

2.3. Redux Toolkit

Redux Toolkit je skupina alata napravljena za pojednostavljenje korištenja Redux biblioteke. Redux je biblioteka koja se često koristi u Reactu za upravljanje kompleksnim stanjima i tok podataka. Ali u Reduxu je ponekad potrebno pisati *boilerplate* kôd koji treba pratiti određene standarde pa je Redux Toolkit stvoren da se taj problem riješi.

Jedan od glavnih dodataka u Redux Toolkitu je „*configureStore*" funkcija koja olakšava proces implementiranja i održavanja globalnih stanja kroz sve komponente u React aplikaciji. Redux Toolkit organizira sve podatke uz pomoću odreska (engl. *slice*), *slice* je izolirani komad koji sadržava vlastita stanja, podatke i funkcije koje se koriste za različite svrhe.

Redux Toolkit u sebi ima ugrađenu podršku za asinkrono programiranje uz pomoć „*createAsyncThunk*" koji pojednostavljuje proces stvaranja asinkronih događaja kao što je

npr. dohvaćanje podataka. Uspješno pokreće radnje stvaranjem posebnih signala tijekom njihovog izvođenja, koji su potrebni za različite faze u asinkronoj operaciji [3].

2.4. Material UI

Material UI je popularan dizajnerski okvir i biblioteka komponenata otvorenog kôda koja se koristi za izradu korisničkih sučelja *web* aplikacija. Google je službeno predstavio Material UI 2014. godine i usklađuje se s Googleovim Material Designom.

Glavni cilj Material UI-a je pružiti obiman skup već stvorenih komponenata koje se mogu koristiti za izradu suvremenih, reaktivnih i atraktivnih korisničkih sučelja, a da pri tome nije potrebno sve graditi iz početka. Material UI nudi raznolike komponente, uključujući dugme, obrasce, navigaciju, kartice, modalne prozore i druge.

Material UI je izgrađen na temelju Reacta, što ga čini lako integriranim u bilo koju React aplikaciju. Za stilizaciju komponenata koristi se JSS (engl. CSS *in* JS) ili Emotion, što omogućava razvojnim stručnjacima prilagodbu komponenata prema njihovim potrebama, uz istodobno poštivanje principa Material Designa. Material UI uživa veliku popularnost jer omogućava programerima izradu atraktivne i suvremene *web* stranice [4].

2.5. Java

Java je svestrani, objektno orijentirani i visokorazinski programski jezik koji je razvio Sun Microsystems 1996. godine. Dizajniran je da bude platformski neovisan i omogućava programerima pisanje kôda koji je upotrebljiv na različitim operativnim sustavima.

Načelo "piši jednom, pokreni bilo gdje" postiže se putem JVM-a (engl. *Java Virtual Machine*). Java prikriva kompleksnost s raznim slojevima apstrakcije i omogućava programerima fokus na visokorazinsko programiranje. Bitna značajka Jave je njezin sakupljač smeća (engl. *garbage collector*), koji automatski upravlja memorijom aplikacije i oslobađa je kada je to potrebno, omogućavajući programerima da se ne moraju brinuti o curenju memorije. Programi napisani u Javi sastoje se od razreda koji sadrže polja koja predstavljaju podatke, a instanca Java klase naziva se objektom.

Njezina podrška za višenitnost i konkurenciju omogućuje programerima stvaranje aplikacija koje mogu učinkovito rukovati s više zadataka istovremeno, poboljšavajući performanse u scenarijima gdje se zadaci mogu izvoditi paralelno. Međutim, upravljanje višenitnim aplikacijama može biti izazovno, jer zahtijeva pažljivu sinkronizaciju i koordinaciju kako bi se spriječili trkaći uvjeti i blokade.

Popularnost Jave dovela je do njezinog usvajanja u raznim područjima, uključujući razvoj *weba*, mobilnih aplikacija, poslužiteljskih aplikacija, znanstveno računanje itd. Njezina široka primjena, zajedno sa njezinom zrelošću, dokumentacijom i podrškom zajednice, čini je izborom za mnoge programere i organizacije [5].

2.6. Spring Boot

Spring Boot je moćan i široko korišten okvir otvorenog kôda koji je dizajniran kako bi pojednostavio proces razvoja produkcijom spremnih, skalabilnih i visoko performansnih aplikacija koristeći Spring Framework. Spring Boot je 2014. godine stvorio Pivotal tim (sada dio VMwarea).

Glavni cilj Spring Boota je minimiziranje konfiguracije projekta na temelju Springa i omogućavanje programerima da lako kreiraju aplikacije bez znatnog napora. Spring Boot smanjuje količinu standardnog kôda i postavljanje konfiguracija, jer automatski konfigurira većinu stvari prema unaprijed definiranim postavkama. Jedna od ključnih prednosti Spring Boota je ugrađena podrška za web.

Početno, u sebi već sadrži ugrađene poslužitelje poput Tomcata, Jettyja i Undertowa, što eliminira potrebu za posebnom konfiguracijom vanjskih *web* poslužitelja. Zahvaljujući ovom automatskom postavljanju, programer može brzo početi implementirati poslovnu logiku aplikacije.

Spring Boot nudi široku paletu startera. Starteri su unaprijed konfigurirani predlošci koji uključuju često korištene ovisnosti i postavke za određene uporabne slučajeve, kao što su *web* aplikacije, pristup podacima, komunikacija itd. Springov ekosustav je izuzetno razvijen i omogućava jednostavnu integraciju u Spring Bootu.

Podržava različite izvore podataka, uključujući relacijske baze podataka i No SQL. Spring Security omogućava zaštitu aplikacije od različitih vrsta napada. Osim sigurnosti i baza podataka, Spring pruža podršku za izradu *unit* testova pomoću JUnita, migraciju baza podataka uz pomoć Flyway migracija i ugrađeni ORM (Mapiranje objekata i relacija) putem Hibernatea [6].

2.7. MySQL

MySQL je opsežno korišten sistem za upravljanje relacijskim bazama podataka (RDBMS). Razvio ga je švedski entitet MySQL AB, a sada je u vlasništvu tvrtke Oracle. MySQL slijedi relacijski model koji organizira podatke u tablice s redovima i stupcima, omogućujući korisnicima definiranje odnosa među elementima podataka.

Temeljna snaga MySQL-a leži u izvanrednoj izvedbi koja se postiže putem različitih optimizacijskih tehnika i indeksirajućih mehanizama. Za učinkovito dohvaćanje podataka, MySQL koristi strukturu indeksiranja B-stabla, omogućujući brze operacije čitanja i pisanja, čak i u obimnim skupovima podataka. Nadalje, MySQL nudi različite pogone za pohranu podataka, kao što su InnoDB i MyISAM.

Pouzdanost i integritet podataka osigurani su značajkama kao što su transakcije, usklađenost s ACID (engl. *Atomicity, Consistency, Isolation, Durability*) standardom te mehanizmi za sigurnosne kopije i obnovu podataka. Transakcije osiguravaju da niz operacija nad bazom podataka bude ili u potpunosti završen ili potpuno poništen u slučaju kvarova, održavajući dosljednost baze podataka.

Posebno se ističe skalabilnost MySQL-a, koja omogućuje rukovanje velikim skupovima podataka i istovremenim korisnicima uz primjenu tehnika poput replikacije, shardiranja i klasteriranja. MySQL je popularan izbor za *web* aplikacije, sustave za upravljanje sadržajem, e-trgovinu, dugoročno pohranjivanje podataka itd. Kompatibilan je s različitim programskim jezicima, uključujući Java, PHP, Python i drugi [7].

2.8. JSON Web Token(JWT)

JWT (engl. *JSON Web Token*) je kompaktna i jedinstvena metoda za predstavljanje informacija između dvije strane na siguran način. JWT je široko korišteni otvoreni standard (RFC 7519) za autentifikaciju i autorizaciju na temelju tokena u *web* aplikacijama. JWT-ovi su osobito popularni u suvremenom *web* razvoju zbog svoje jednostavnosti, svestranosti i sposobnosti sigurnog prijenosa podataka između strana.

JWT Token se sastoji od tri glavna dijela: zaglavlje, teret (engl. *payload*) i potpis (engl. *signature*). Zaglavlje obično sadrži metapodatke o tokenima, poput algoritma koji se koristi za njihovo potpisivanje. Teret sadrži tvrdnje (engl. *claims*), koje su izjave o subjektu (poput korisnika) i dodatne podatke. Tvrdnje mogu uključivati identifikatore korisnika, vrijeme isteka, izdavača i više.

Potpis se stvara kodiranjem zaglavlja, tereta i tajnog ključa koristeći navedeni algoritam. Taj potpis osigurava integritet tokena i sprječava manipulaciju. Jedna od glavnih značajka JWT je njihova kompaktnost. Zato što se sve potrebne informacije već nalaze u tokenu, poslužitelj ne treba pohranjivati informacije o sesiji, što JWT čini idealnim za mikro servise i API-eve(engl. *Application Programming Interface*).

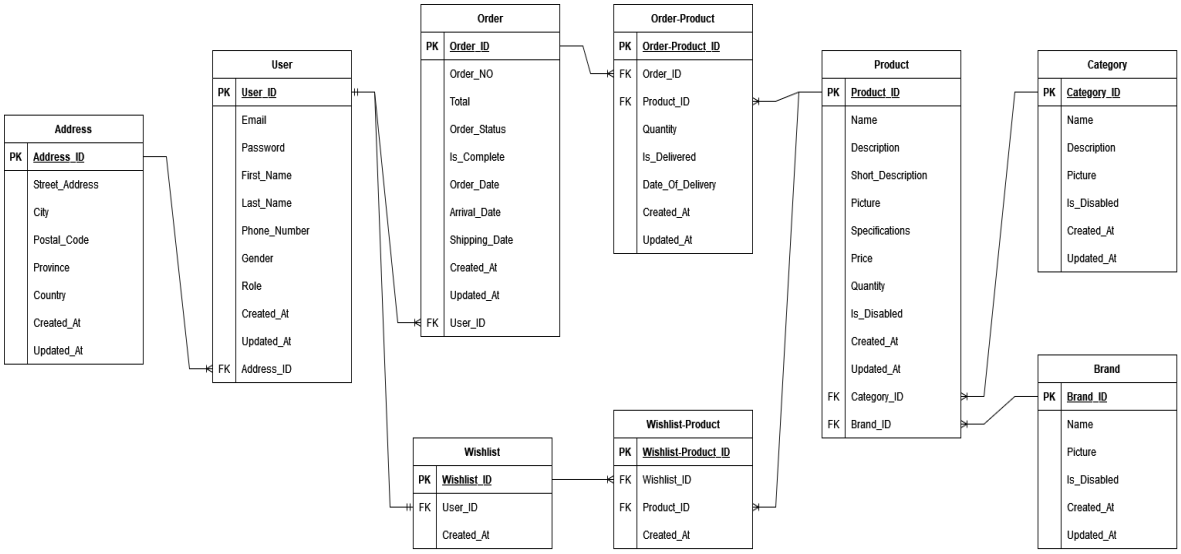
Ali to znači da su sve informacije unutar tokena vidljive svima onima koji mu imaju pristup, pa je zato potrebno šifrirati sve osjetljive podatke ili potpuno izbjegavati njihovo dodavanje u token. JWT-ovi se često koriste za autentifikaciju i autorizaciju. Kada se korisnik prijavi, poslužitelj generira JWT s relevantnim korisničkim informacijama i potpisuje ga tajnim ključem. Klijent tada sprema ovaj JWT, obično u kolačiću preglednika ili lokalnom spremištu.

Za svaki daljnji zahtjev, klijent uključuje JWT u zaglavlje zahtjeva, omogućavajući poslužitelju da provjeri autentičnost korisnika i dodijeli pristup autoriziranim resursima. Međutim, iako JWT-ovi nude fleksibilnost i praktičnost, moraju se pažljivo koristiti. Ako se ne koriste ispravno, JWT-ovi mogu biti predstavljati sigurnosne rizike, poput curenja tokena i napada ponovnog izvođenja [8].

3. Izrada baze podataka

3.1. Entiteti i relacije

Baza podataka za Spark Mart sadržava sljedeće entitete: Adresa (engl. *Address*), Marka (engl. *Brand*), Kategorija (engl. *Category*) Narudžba (engl. *Order*), Proizvod (engl. *Product*), Korisnik (engl. *User*), Popis želja (engl. *Wishlist*) i dva dodatna entiteta koji se koriste za više na više kardinalnu vezu: Narudžba-Proizvod (engl. *OrderProduct*) i Popis želja-Proizvod (engl. *WishlistProduct*).



Slika 1: Schema baze podataka

Adresa se sastoji od sljedećih polja: id, adresa stanovanja, grad, poštanski broj, provincija, država, kolekcija stranih ključeva koji predstavljaju instance entiteta. Korisnik i dva polja namijenjena za praćenje datuma stvaranje instance entiteta i praćenje datuma promjena instance entiteta. Marka se sastoji od sljedećih polja: id, naziv marke, naziv slike, polje koji prati je li instanca entiteta isključena, kolekcija stranih ključeva koji predstavljaju instance entiteta Proizvod i dva polja namijenjena za praćenje datuma stvaranje instance entiteta i praćenje datuma promjena instance entiteta.

Kategorija se sastoji od sljedećih polja: id, naziv kategorije, opis, naziv slike, polje koji prati je li instanca entiteta isključena, kolekcija stranih ključeva koji predstavljaju

instance entiteta Proizvod i dva polja namijenjena za praćenje datuma stvaranja instance entiteta i praćenje datuma promjena instance entiteta.

Narudžba se sastoji od sljedećih polja: broj narudžbe, sveukupna cijena narudžbe, polje za praćenje je li narudžba dovršena, stanje narudžbe, datum narudžbe, datum isporuke, datum dostave, strani ključ za instancu entiteta Korisnik, kolekcija stranih ključeva koji predstavljaju instance entiteta Narudžba-Proizvod i dva polja namijenjena za praćenje datuma stvaranja instance entiteta i praćenje datuma promjena instance entiteta.

Proizvod se sastoji od sljedećih polja: id, naziv proizvoda, opis proizvoda, kratki opis proizvoda, naziv slike, tehnički podaci proizvoda, cijena, količina, polje za praćenje je li instanca entiteta isključena, strani ključ za instancu entiteta Marka, strani ključ za instancu entiteta Kategorija, kolekcija stranih ključeva koji predstavljaju instance entiteta Narudžba-Proizvod, kolekcija stranih ključeva koji predstavljaju instance entiteta Popis želja-Proizvod i dva polja namijenjena za praćenje datuma stvaranja instance entiteta i praćenje datuma promjena instance entiteta.

Korisnik se sastoji od sljedećih polja: id, elektronična pošta, lozinka, ime, prezime, broj mobitela, spol, uloga, polje za praćenje je li instanca entiteta isključena, strani ključ za instancu entiteta Adresa, strani ključ za instancu entiteta Popis želja, kolekcija stranih ključeva koji predstavljaju instance entiteta Narudžba i dva polja namijenjena za praćenje datuma stvaranja instance entiteta i praćenje datuma promjena instance entiteta. Popis želja se sastoji od sljedećih polja: id, polje namijenjeno za praćenje datuma stvaranja instance entiteta, strani ključ za instancu entiteta Korisnik i kolekcija stranih ključeva koji predstavljaju instance entiteta Popis želja-Proizvod.

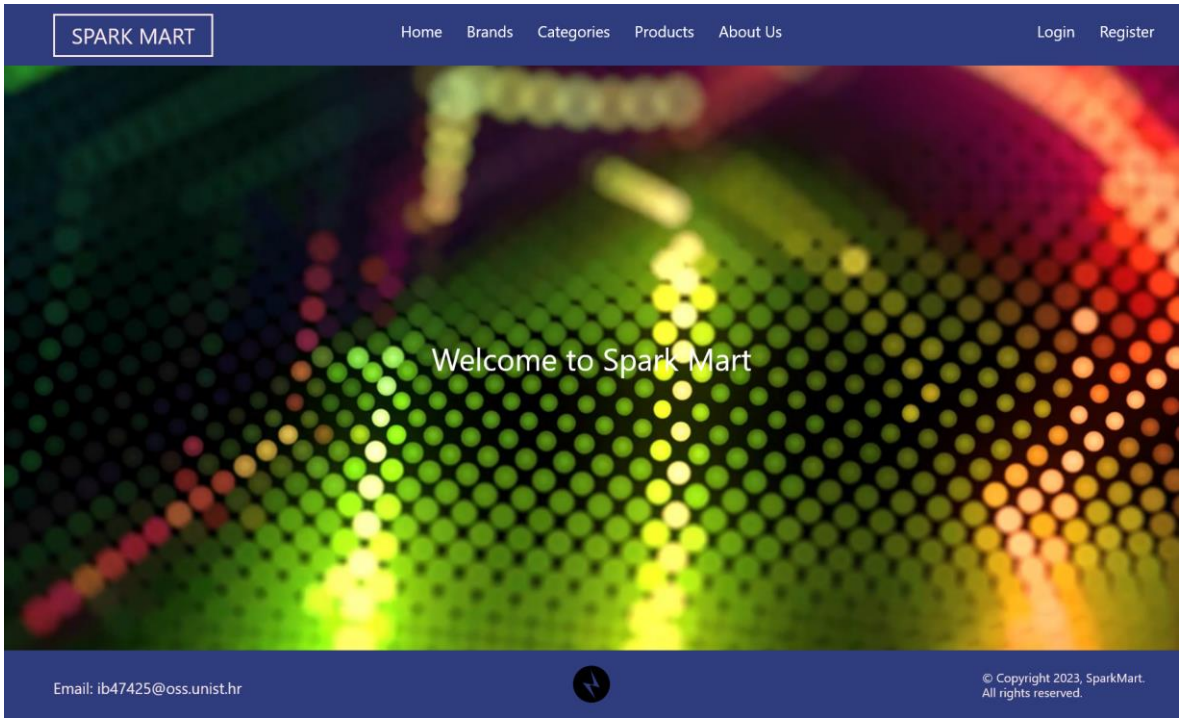
Narudžba-proizvod se sastoji od sljedećih polja: id, količina kupljenog proizvoda, polje za praćenje je li proizvod dostavljen, datum dostave, strani ključ za instancu entiteta Proizvod, strani ključ za instancu entiteta Narudžba i dva polja namijenjena za praćenje datuma stvaranja instance entiteta i praćenje datuma promjena instance entiteta. Popis želja-Proizvod se sastoji od sljedećih polja: id, datum stvaranje instanca entiteta, strani ključ za instancu entiteta Popis želja i strani ključ za instancu entiteta Proizvod.

4. Izrada aplikacije

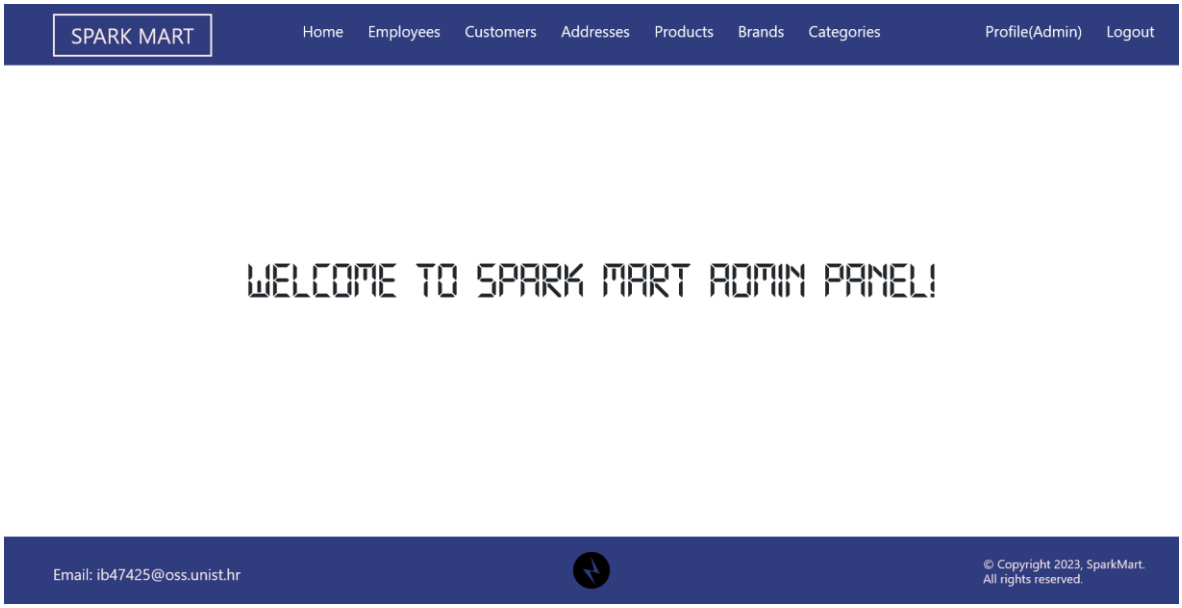
4.1 Izrada sustava za autentifikacija i autorizaciju

4.1.1 Početna stranica

Početna stranica je podijeljena na tri dijela: dio za kupce, dio za zaposlenike i dio za neprijavljene korisnike. Dio za neprijavljene korisnike se prikazuje svim osobama koji nisu prijavljene (Slika 2), dok se dio za zaposlenike prikazuje isključivo osobama s ulogom zaposlenika ili administratora (Slika 3).

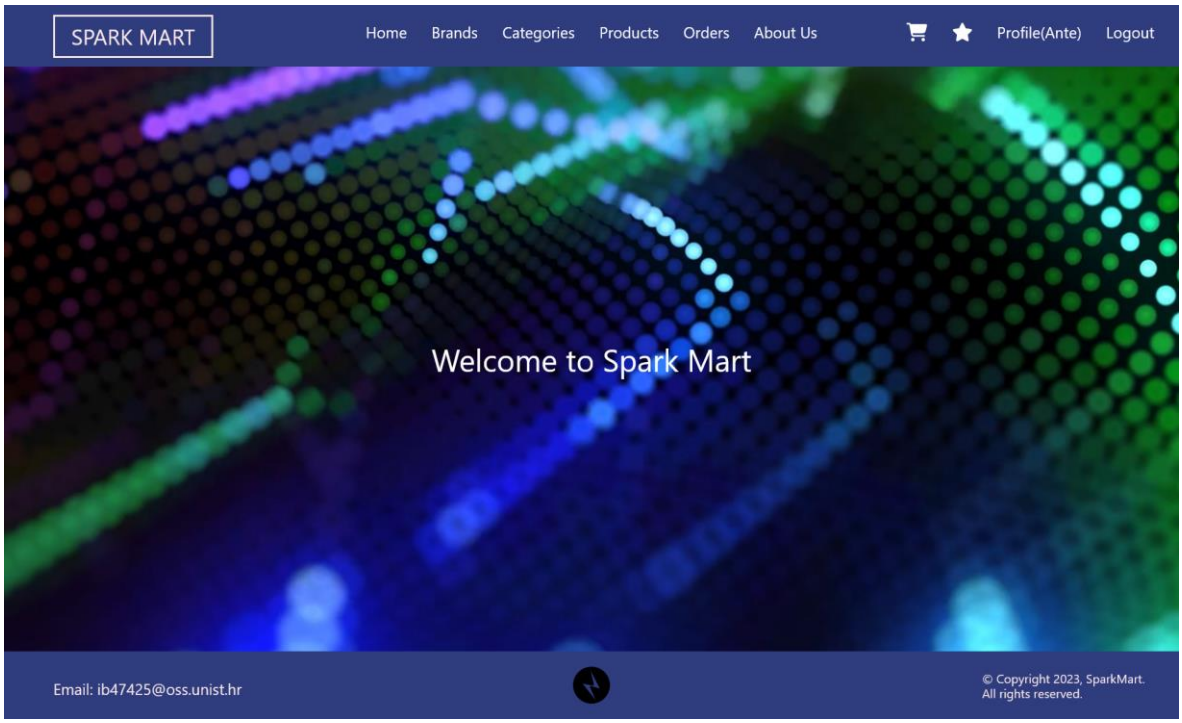


Slika 2: Početna stranica za neprijavljene korisnike



Slika 3: Početna stranica za zaposlenike

Početna stranica za kupce se prikazuje isključivo za korisnike koji imaju ulogu kupca (Slika 4).

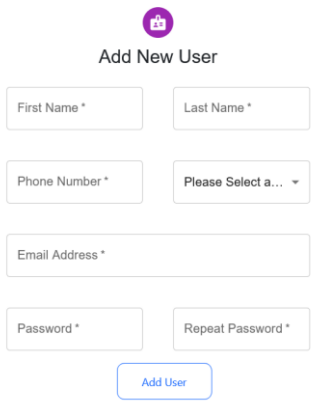


Slika 4: Početna stranica za kupce

Nakon uspješne autentifikacije zaposlenika ili administratora, korisniku se prikazuje bijela pozadina s tekstom "Dobrodošli u Spark Mart Admin Panel!". U zaglavlju stranice dolazi do promjene s navigacije za kupce na navigaciju za zaposlenike. Nakon promjene zaglavlja, korisnik može koristiti navigaciju kako bi prelazio s jedne stranice na drugu, može posjetiti svoj profil u kojemu se ispisuju osobne informacije ili informacije o adresi stanovanja, odjavom izlazi iz profila.

4.1.2 Registracija kupaca

Registracija kupaca može se provesti tako da se pristupi stranici za registraciju. Na toj stranici nalazi se obrazac koji mora sadržavati sljedeća polja: ime, prezime, elektronička pošta, broj mobitela, spol, lozinka i ponovljena lozinka (Slika 5).



Slika 5: Registracija novih korisnika

Kada korisnik ispuni obrazac, provjeravaju se uneseni podaci i ako nešto nije ispravno popunjeno prikazuje se greška. Prije nego što se podaci pošalju, provjerava se postoji li već korisnik s navedenom elektroničkom poštom ili brojem mobitela. Ako su svi podaci ispravni, stvara se novi JSON (engl. *JavaScript Object Notation*) i šalje se funkciji za registraciju novih korisnika (Ispis 1).

```
const config = {
            headers: {
                  'Content-Type': 'application/json',
            },
}
await axios.post(
        `${backendURL}/auth/register`,
        { firstName, lastName, phoneNumber,
         email, password, gender, role },
        config
)
```

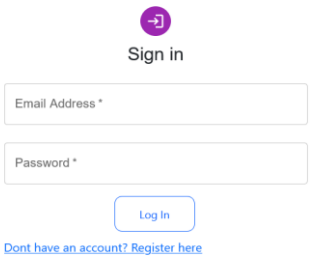
Ispis 1: Kôd za registraciju korisnika

Funkcija koristi "*createAsyncThunk*" iz Redux Toolkita kako bi stvorila asinkroni zahtjev prema *backendu*. Kad zahtjev stigne do *backenda* putem određene API točke, pretvara se u DTO (engl. *Data Trasfer Object*) i prenosi se servisu za autentifikaciju. U servisu se provode dodatne provjere, stvara se nova instanca klase "Korisnik".

Svi se podaci prenose na novu instancu i dodjeljuje joj se prazna adresa. Nakon toga nova instanca korisnika sprema se u bazu podataka. Nakon uspješnog spremanja korisnika, generira se novi token i vraća se natrag *frontendu*.

4.1.3 Prijava korisnika

Prijava korisnika može se provesti tako da se pristupi stranici za prijavu Na toj stranici nalazi se polje za elektroničku poštu i lozinku.



Slika 6: Prijava korisnika

Kada korisnik ispuni obrazac, provjeravaju se svi uneseni podaci kako bi se osigurala njihova ispravnost prije nego što se zahtjev pošalje *backendu*. U slučaju da su svi podaci valjani, stvara se novi JSON objekt koji se prosljeđuje funkciji za prijavu korisnika (Ispis 2). Funkcija koristi "*createAsyncThunk*" iz Redux Toolkita kako bi stvorila asinkroni zahtjev prema *backendu*. Kada zahtjev stigne do *backenda* putem određene API točke, konvertira se u DTO i šalje se servisu za autentifikaciju.

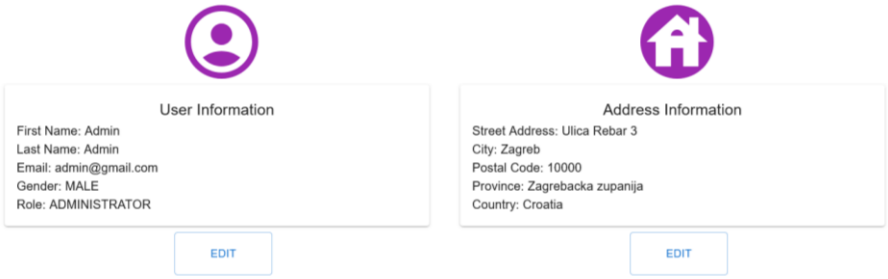
Ako prijava korisnika ne uspije, korisnik će vidjeti poruku o grešci pri prijavi. Međutim, ako je prijava uspješna, generira se JWT token koji se vraća natrag u *frontend*. U *frontendu* se JWT token pohranjuje u lokalnoj memoriji (engl. *local storage*) kako bi se koristio za autentifikaciju korisnika u svim budućim zahtjevima. Nakon što se token dohvati, šalje se natrag u *backend* gdje se koristi za dohvaćanje korisničkih podataka. Kada se podaci korisnika dohvate, stvara se Redux Toolkit *slice* gdje se pohranjuje token i informacije o korisniku.

```
const config = {
            headers: {
             'Content-Type': 'application/json',
            },
        };
const { data } = await axios.post(
      `${backendURL}/auth/login`,
      { email, password },
          config
);
localStorage.setItem('userToken', String(data.token));
const getConfig = {
      headers: { Authorization: `Bearer ${data.token}` }
};
const userInfo = await axios.get(`${backendURL}/auth/get-user-info`,
                                    getConfig
);
return userInfo.data;
```

Ispis 2: Prijava korisnika

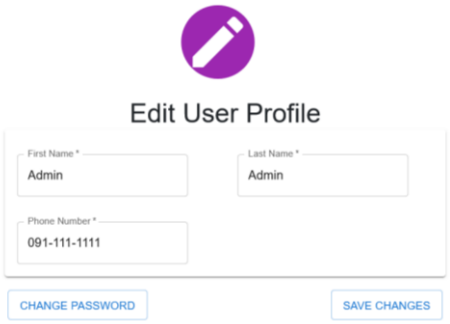
4.1.4 Korisnikov profil i odjava

U Profil korisnika se može pristupiti tako da se pritisne dugme na kojem piše profil i u zagradama ime korisnika. Stranica sadržava informacije o korisniku i informacije o adresi stanovanja (Slika 7).



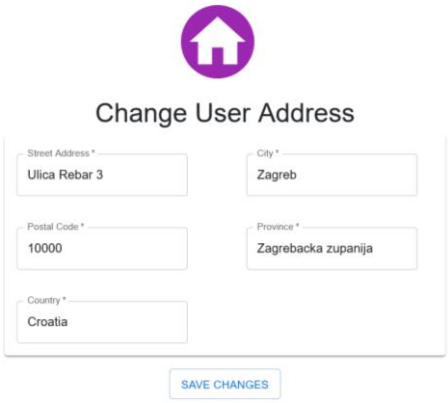
Slika 7**:** Profilna stranica korisnika

Ako korisnik želi promijeniti svoje informacije ili adresu, samo mora pritisnuti dugme za promjenu određene informacije. Pritiskanje dugme za promjenu podataka korisnika vodi do stranice za promjenu podataka korisnika (Slika 8).



Slika 8: Promjena podataka korisnika

Osim što omogućava promjenu podataka korisnika, dugme će korisnika odvesti do stranice za mijenjanje lozinke. Pritiskanjem dugme za promjenu korisnikove adrese, moguće je na za to predviđenoj stranici unijeti novu adresu (Slika 9). Odjava korisnika se izvodi pritiskanjem dugme za odjavu (engl. *logout*) u zaglavlju stranice (Slika 3).



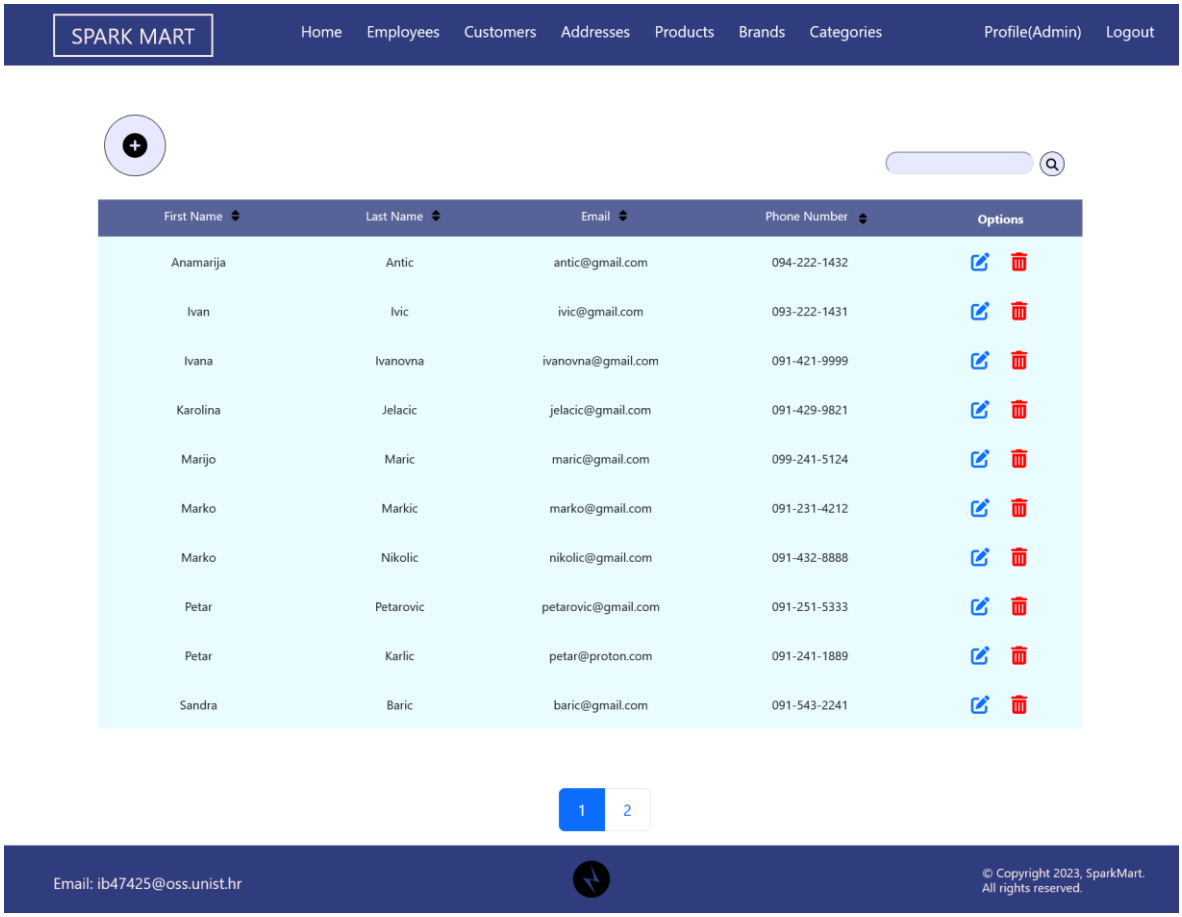
Slika 9: Unos nove adrese

4.2 Izrada administracijskog sučelja

4.2.1 Upravljanje korisnika

Administracijska stranica za upravljanje korisnicima dostupna je isključivo administratoru. Administracijska stranica za korisnike podijeljena je na dva dijela: stranicu za kupce i stranicu za zaposlenike. Kada se otvori stranica za korisnike, ovisno o trenutačnoj ulozi korisnika, prikazat će se svi korisnici odgovarajuće uloge (Slika 10). Stranica omogućava paginaciju, sortiranje i filtriranje.

Administrator može dodati nove zaposlenike i kupce, uređivati korisničke podatke i brisati korisnike. Stranica za kupce dodatno pruža mogućnost pregleda njihovih narudžbi i popisa želja. Za dodavanje novih korisnika služi stranica za dodavanje korisnika. Obrazac na njoj identičan je onome na stranici za registraciju (Slika 5).



Slika 10: Administratorska stranica za korisnike

Ovisno o trenutnoj stranici s koje se pristupa, na stranici za dodavanje novog korisnika dodat će se polje uloge u JSON objekt prije nego što se podaci pošalju. To polje uloge može biti ili "zaposlenik" ili "kupac". Nakon što korisnik ispuni obrazac, uneseni podaci se pohranjuju u JSON obliku i šalju na *backend*.

```
var user = userRepository.findById(uuid)
      .orElseThrow(() -> new UserNotFoundException("ERROR: User by
given ID not found."));
if(!user.getOrders().isEmpty()|| !user.getWishlist().getProducts().
isEmpty()) {
     user.setDisabled(true);
     userRepository.save(user);
} else {
     userRepository.delete(user);
}
```

Ispis 3: Brisanje ili isključivanje korisnika

Prije slanja zahtjeva *backendu*, obavezno se dodaje JWT token u zaglavlju zahtjeva. Ako token nedostaje, zahtjev će odmah biti odbijen. Kada zahtjev stigne na *backend*, uneseni podaci se preslikavaju u DTO i proces sličan registraciji korisnika se ponavlja. Ako korisnik želi promijeniti svoje podatke, prvo mora pristupiti stranici za promjenu podataka i pritom proslijediti id korisnika. Na toj stranici se aktivira "*useEffect*" koji koristi id korisnika kako bi putem *backenda* dohvatio sve potrebne podatke iz baze podataka (Ispis 4).

```
useEffect(() => {

    getUserById(userId)

      .then((result: any) => setUser(result.data));
```

Ispis 4: Dohvaćanje korisnikovih podatka

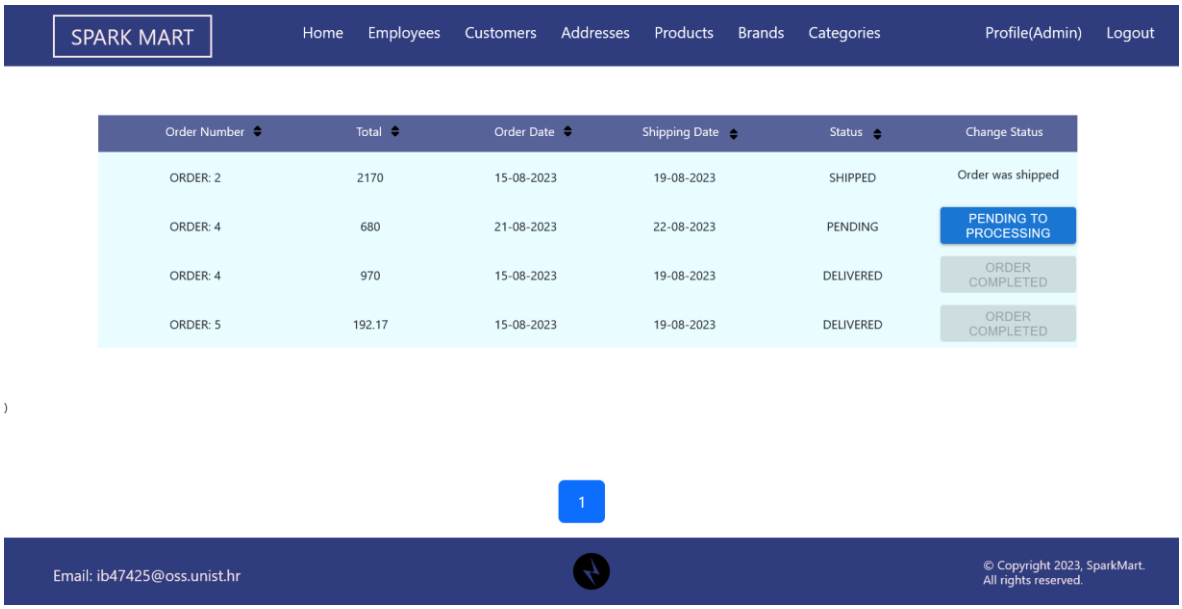
Izgled stranice za ažuriranje podataka je isti kao kôd dodavanja novog korisnika, s jedinom razlikom da su sva polja već popunjena podacima korisnika prilikom učitavanja. Nakon što se završi proces ažuriranja podataka za korisnika, stvara se novi JSON koji sadrži sve unesene podatke. U taj JSON se dodaje i JWT token te se potom prosljeđuje funkciji "*updateUser*" (Ispis 5).

```
return await axios
      .put(API_URL + '/update-user/' + userId, userInformation,
      { headers: { Authorization: `${userToken}` } });
```

Ispis 5: Ažuriranje korisnikovih podatka

Podaci se prosljeđuju u *backend* kako bi se promjene spremile u bazu podataka. Brisanje korisnika se inicira pritiskom na odgovarajuće dugme koje se nalazi u retku tablice korisnika. Nakon pritiska dugmeta, zahtjev se šalje u *backend* zajedno s identifikacijskim brojem korisnika (id) i JWT tokenom. Prije nego što se korisnik izbriše, izvrši se provjera postoji li veza s postojećim narudžbama ili je njegov popis želja povezan s nekim proizvodima. Ako takve veze postoje, korisnik se deaktivira u bazi podataka, ali se ne briše.

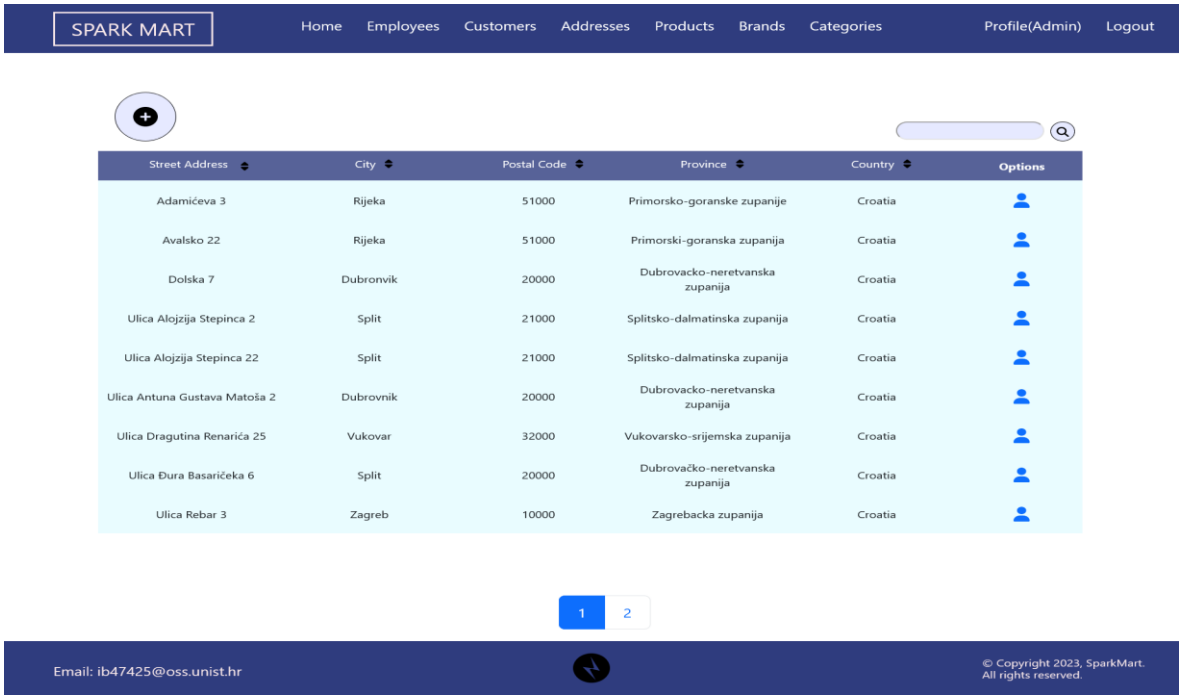
U suprotnom se slučaju, korisnik zajedno s popisom želja trajno briše iz sustava (Ispis 3). Kao što je prethodno navedeno, kupcima je omogućen pregled narudžbi i popisa želja. U okviru popisa narudžbi, moguće je izvršiti promjenu statusa narudžbe, koji može biti "u tijeku", "u obradi" ili "isporučeno" (Slika 11). Kada se narudžba isporuči, kupac je dužan potvrditi primitak svakog pojedinog proizvoda. Nakon što su svi proizvodi potvrđeni kao dostavljeni, narudžba se automatski zaključava i njezino stanje se mijenja na "dostavljeno". U popisu želja kupca prikazani su svi proizvodi koje je kupac dodao na svoj popis želja.



Slika 11: Mijenjanje status narudžbe

4.2.2 Pregled adresa

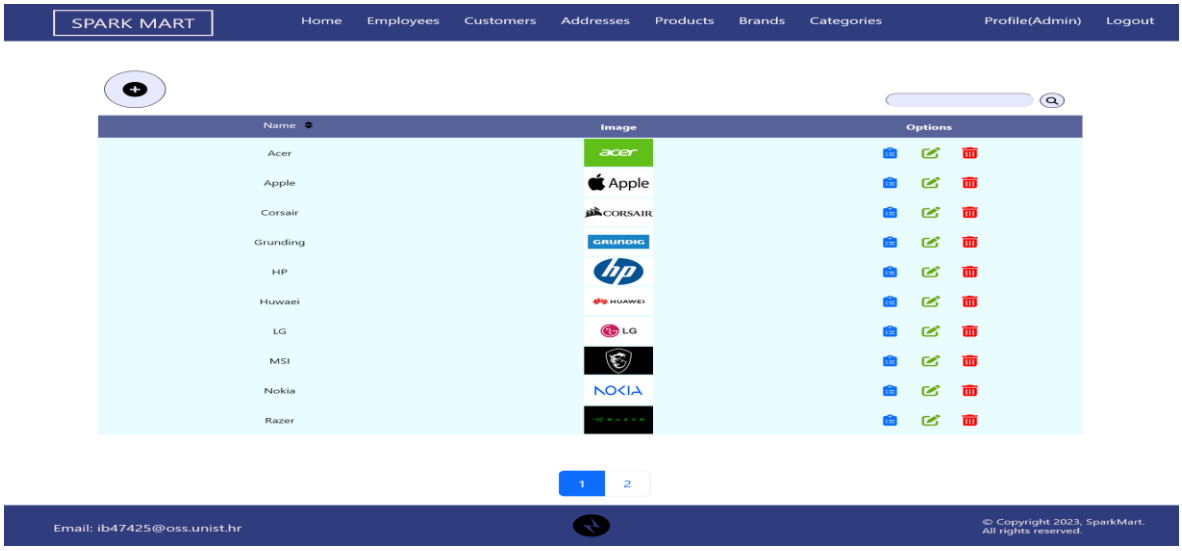
Administracijskoj stranici za adrese mogu pristupiti zaposlenici i administrator. Stranica za adrese omogućava paginaciju, filtriranje, sortiranje i dodavanje nove adrese (Slika 9), te pregled svih korisnika koji žive na određenoj adresi (Slika 12).



Slika 12: Administratorska stranica za adrese

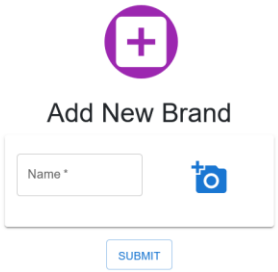
4.2.3 Upravljanje marki

Administracijskoj stranici za marke mogu pristupiti zaposlenici i administrator. Stranica za marke omogućava paginaciju, filtriranje, sortiranje, dodavanje nove marke, mijenjanje postojeće marke, te pregled svih proizvoda koji imaju određenu marku (Slika 13).



Slika 13: Administracijska stranica za marke

Dodavanje novih marki obavlja se putem stranice za dodavanje novih marki. Stranica za dodavanje sadržava samo dva polja u obrascu, a to su polje za naziv marke i polje za sliku (Slika 14). Kada se pošalje obrazac, stvori se novi JSON sa svim potrebnim podacima i prosljeđuje u funkciju za stvaranje novih marki (Ispis 6).



Slika 14: Forma za novu marku

Kada *backend* zaprimi zahtjev s *frontenda*, preslikaju se svi podaci u DTO i prosljeđuju se u servis za marke. U servisu se stvori nova instanca entiteta marke, promjeni se naziv slike tako da ne poremeti spremanje slike, spremi se u neki folder i novi naziv slike se dodijeli instanci i spremi se u bazu podataka. Ako korisnik želi ažurirati marku, onda mora pristupiti stranici za ažuriranje marke.

Obrazac na stranici za ažuriranje marke je identičan onome za dodavanje novih marki (Slika 14). Prije prikaza obrazaca, s prijašnje stranice proslijedi se id marke i pošalje se zahtjev u *backend* kako bi dohvatio sve podatke o marki i prikazao ih u prikladna polja. Kada korisnik pošalje obrazac, stvara se novi JSON i prosljeđuje se u funkciju za ažuriranje marke.

```
export const createNewBrand = async(newBrand: any ) => {
    return await axios.post(API_URL, newBrand, { headers:
{ Authorization: `${userToken}`, 'Content-Type': 'multipart/form-
data' } });
}
```

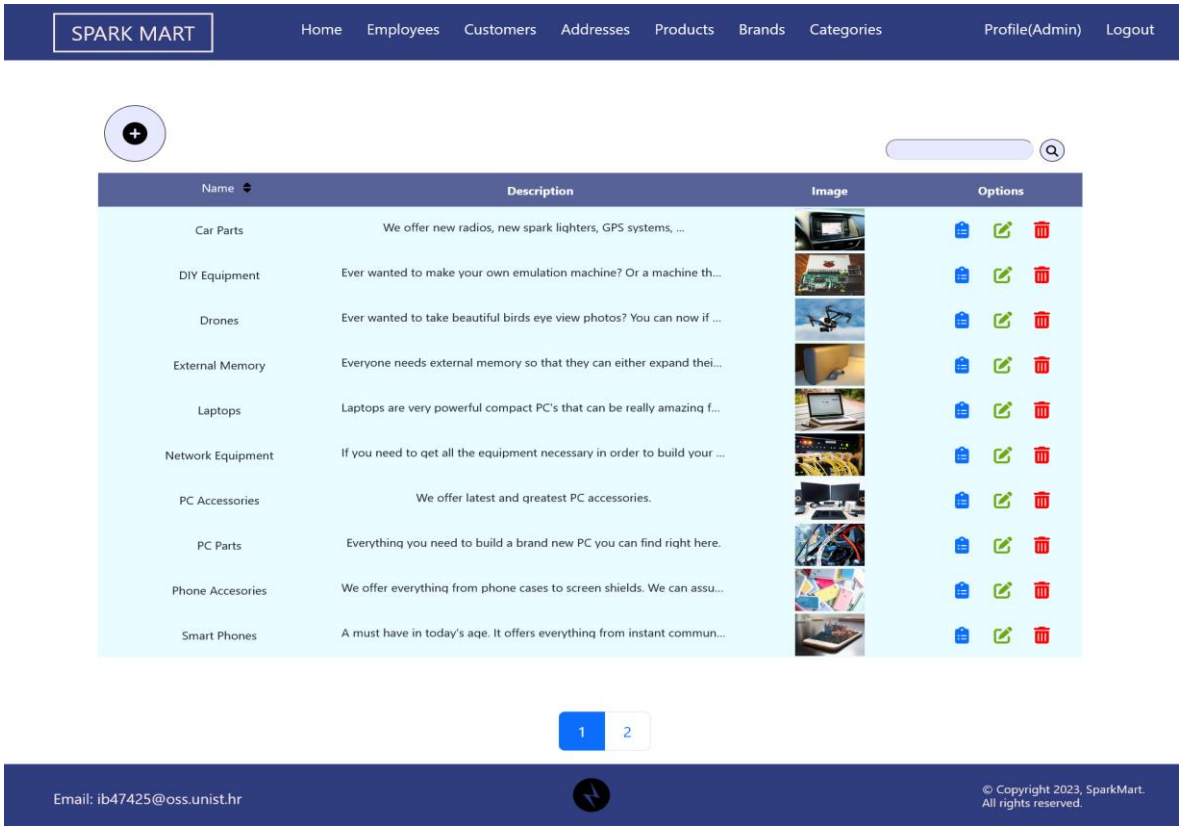
Ispis 6: Dodavanje nove marke

Kada *backend* zaprimi sve zahtjeve, provjerava se je li uključena nova slika za marku, ako jest, tada sprema novu sliku i briše prijašnju. Pomoću id marke dohvati se instanca marke iz baze, potrebna polja se promijene i spremaju se natrag u bazu podataka. Ako korisnik želi izbrisati marku, mora pritisnuti dugme za brisanje marke. Kada *backend* dohvati zahtjev, dobavlja se marka iz baze podataka i provjerava ima li kakvu postojeću vezu s proizvodom. Ako ima, onda se marka i svi proizvodi pod tom markom isključe.

U slučaju da nema, samo se izbriše iz baze podataka. Da bi se pristupilo stranici za prikaz svih proizvoda koji spadaju pod određenu marku, mora se pritisnuti dugme koji će odvesti do te stranice. Stranica sadržava tablicu koja prikazuje sve proizvode pod određenom markom. Ako marka nema ni jedan proizvod, onda se prikazuje tekst koji govori upravo o tome da proizvoda pod tom markom nema.

4.2.4 Upravljanje kategorija

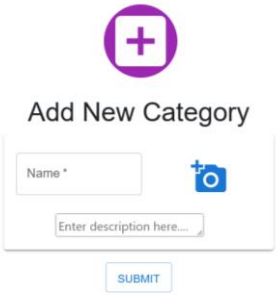
Administracijskoj stranici za kategorije mogu pristupiti zaposlenici i administrator. Stranica za marke omogućava paginaciju, filtriranje, sortiranje, dodavanje nove kategorije, mijenjanje postojeće kategorije, te pregled svih proizvoda koji imaju određenu kategoriju (Slika 15).



Slika 15: Administracijska stranica za kategorije

Dodavanje novih kategorija obavlja se putem stranice za dodavanje kategorija. Stranica za dodavanje sadržava samo tri polja u obrascu, a to su naziv kategorije, opis i slika (Slika 16). Kada se pošalje obrazac, stvori se novi JSON sa svim potrebnim podacima i prosljeđuje u funkciju za stvaranje novih kategorija. Kada *backend* zaprimi zahtjev s *frontenda*, preslikaju se svi podaci u DTO i prosljeđuje se u servis za kategorije.

U servisu se stvori nova instanca entiteta kategorije, promjeni se naziv slike tako da se ne poremeti njezino spremanje, spremi se u neki folder, novi naziv slike se dodijeli instanci te spremi u bazu podataka. Ako korisnik želi ažurirati kategoriju, onda mora pristupiti stranici za ažuriranje kategorija. Obrazac na stranici za ažuriranje kategorija identičan je onom za dodavanje novih kategorija (Slika 16).



Slika 16: Dodavanje nove kategorije

Prije prikaza obrazaca, s prijašnje stranice proslijedi se id kategorije te se pošalje zahtjev u *backend* da dohvati sve podatke o kategoriji i da se prikažu u prikladna polja. Kada korisnik pošalje obrazac, stvara se novi JSON i prosljeđuje se u funkciji za ažuriranje kategorije. Kada *backend* zaprimi sve zahtjeve, slijedi provjera je li uključena nova slika za kategoriju, ako jest onda se nova slika sprema, a prijašnja briše. Pomoću id kategorije dohvati se instanca kategorije iz baze, potrebna polja se mijenjaju i spreme se natrag u bazu podataka.

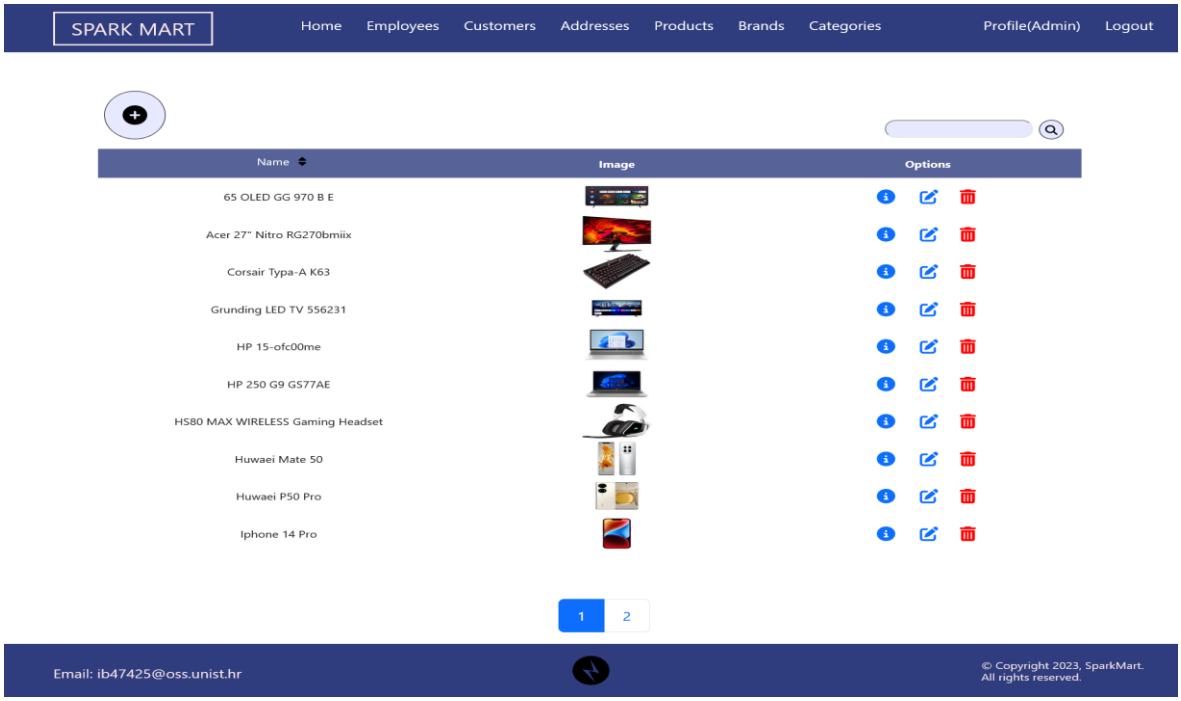
Ako korisnik želi izbrisati kategoriju, onda mora pritisnuti dugme za brisanje kategorije. Kada *backend* dohvati zahtjev, dobavlja kategoriju iz baze podataka i provjerava ima li kakvu postojeću vezu s proizvodom. Ako ima onda se kategorija i svi proizvodi pod tom kategorijom isključe. U slučaju da nema, samo se izbriše iz baze podataka. Da bi se pristupilo stranici za prikaz svih proizvoda koji pripadaju određenoj kategoriji, mora se pritisnuti dugme koji će odvesti do te stranice.

Stranica sadržava tablicu koja prikazuje sve proizvode koje su u toj određenoj kategoriji. U slučaju da kategorija nema ni jedan proizvod, prikazat će se tekst koji kaže da proizvoda pod tom kategorijom nema.

4.2.5 Upravljanje proizvoda

Administracijskoj stranici za proizvode mogu pristupiti zaposlenici i administrator. Stranica za proizvode omogućava paginaciju, filtriranje, sortiranje, dodavanje novih

proizvoda, mijenjanje postojećih proizvoda, brisanje proizvoda, te pregled statistike vezane za proizvod (Slika 17). Dodavanje novih proizvoda obavlja se putem stranice za dodavanje novih proizvoda. Stranica zadržava obrazac koji ima sljedeća polja: naziv proizvoda, opis, kratki opis, cijena, količina, slika marki, kategorije i tehnički podaci (Slika 19).



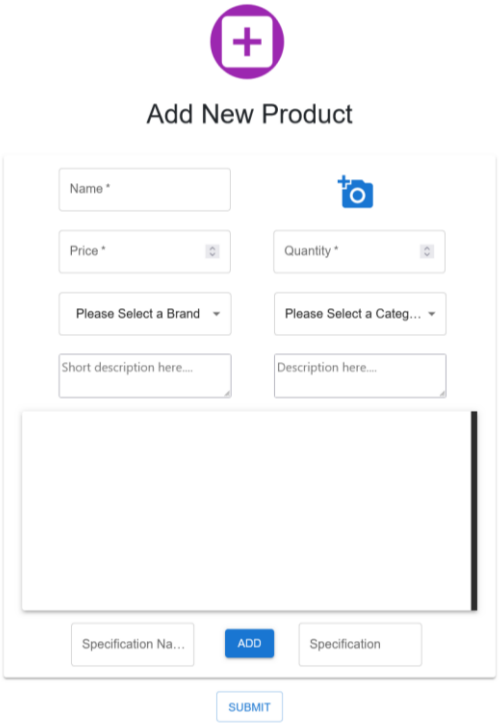
Slika 17: Administratorska stranica za proizvode

Tehnički podaci se spremaju u JSON. Dodavanje novih vrijednost u tehničke podatke obavlja se tako da se prvo upiše naziv tehničkog podataka (npr. veličina ekrana) i vrijednost tehničkog podatka (7.5 *inch*). Kada se podaci unesu, odmah se prikažu u polje za prikaz tehničkih podataka. Moguće je i izbrisati određene tehničke podatke ili ih je moguće mijenjati, tako da se upiše naziv tehničkog podatka i nova vrijednost.

Kada korisnik pošalje obrazac, stvori se novi JSON sa svim potrebnim podacima i proslijedi se u funkciju za spremanje proizvoda koja šalje zahtjev u *backend*. Kada *backend* dobije zahtjev, podaci se preslikavaju u DTO i prosljeđuju u servis za proizvode.

U servisu se normalizira naziv slike proizvoda, sprema se u mapu s novim normaliziranim imenom, stvori se nova instanca entiteta proizvod. U instanci se preslikaju svi podaci o proizvodu s DTO-a i dodjeljuje se novo normalizirani naziv za sliku, a nakon toga se sprema u bazu podataka. Ažuriranje podataka proizvoda izvodi se tako da korisnik pristupi stranici za ažuriranje podataka proizvoda.

U stranici se prosljeđuje id proizvoda koji se ažurira, dohvati se iz baze podataka i prikažu se sve informacije u prikladna polja. Nakon ažuriranja podataka, stvori se novi JSON i proslijedi se u funkciju za ažuriranje podataka, koji će stvoriti zahtjev za *backend*. Kada *backend* zaprimi zahtjev, automatski se preslika u DTO i prosljeđuje se u servis za proizvode.



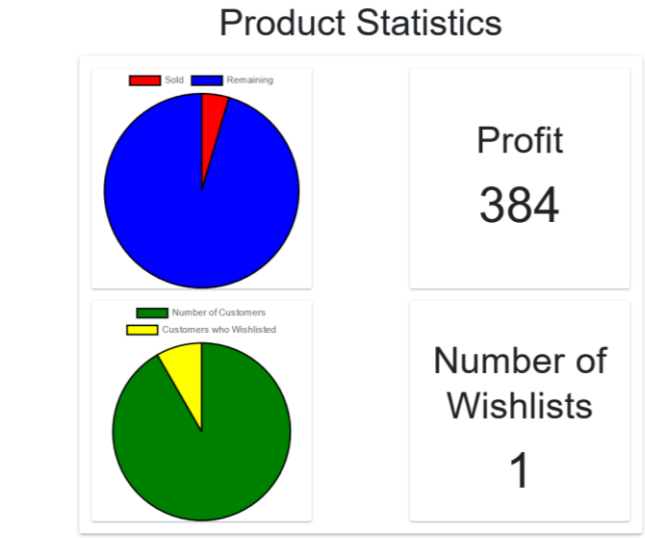
Slika 18: Dodavanje novog proizvoda

Ako se u DTO nalazi slika, onda će se naziv slike normalizirati, spremiti u bazu podataka i izbrisati prijašnja. Pomoću id proizvoda dohvati se instanca proizvoda iz baze, potrebna se polja promijene i spremaju natrag u bazu podataka. Brisanje proizvoda može se izvesti pritiskanjem dugmeta za brisanje proizvoda. Prosljeđuje se id proizvoda u funkciju

za brisanje i stvori se zahtjev za *backend*. Kada *backend* zaprimi zahtjev, id korisnika se prosljeđuje u servis za proizvode.

Pomoću id proizvoda dohvaća se proizvod iz baze podataka. Ako proizvod ima veze u bazi Narudžba-proizvod ili Popis želja-proizvod, onda se proizvod isključi iz baze podataka, inače se proizvod samo izbriše iz baze podataka. Prikaz statistike o proizvodu može se vidjeti na stranici za statistiku.

Proslijedi se id proizvoda u *backend*, gdje se dohvati različita statistika vezana za proizvod (npr. ukupno prodano, ukupna zarada, preostalo u skladištu, itd.) i vrati se natrag u *frontend*. U *frontendu* je prikazana ukupna statistika uz pomoć JavaScript biblioteke koja se zove Chart.js (Slika 18).

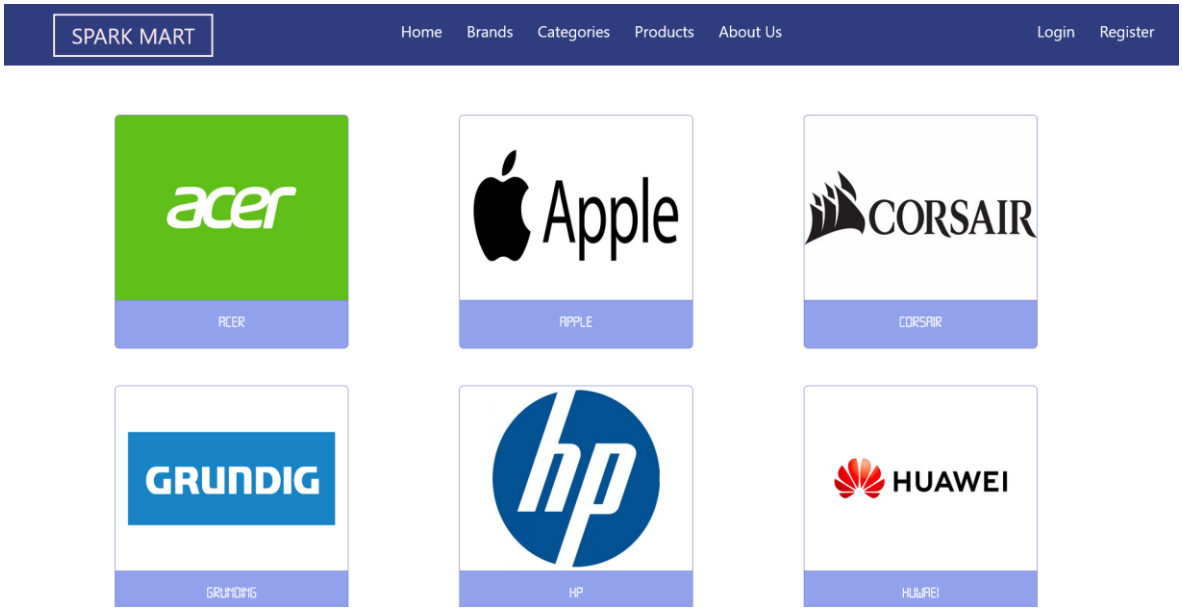


Slika 19: Statistika proizvoda

4.3 Izrada sučelja za kupce

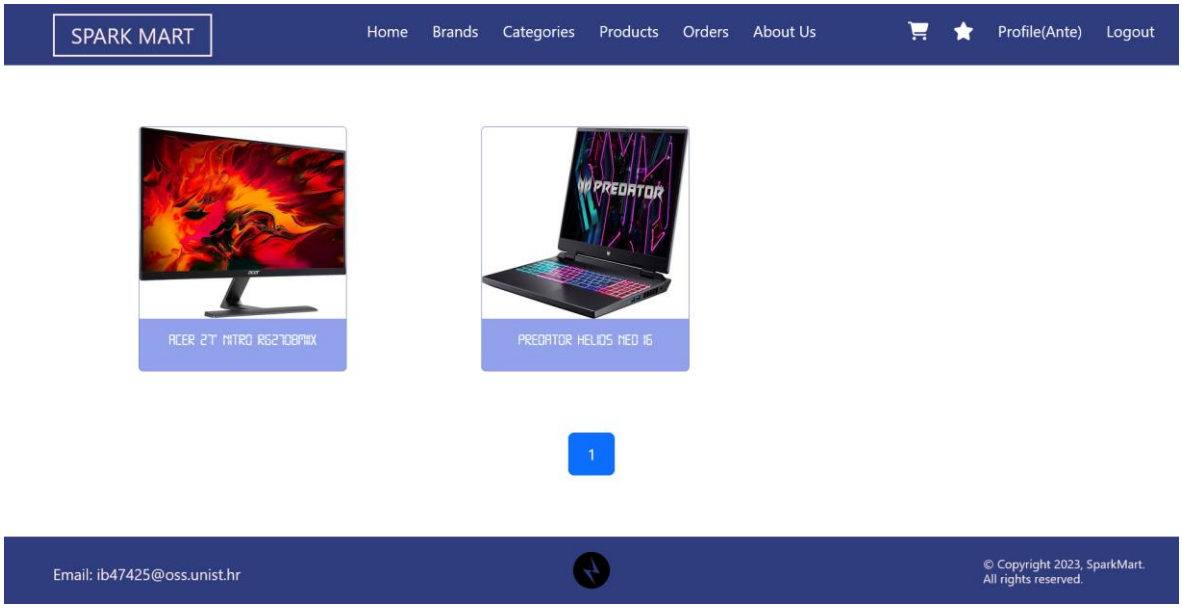
4.3.1 Pregled proizvoda

Kupci mogu pregledavati proizvode na različite načine: preko marke, preko kategorije i pregledom svih proizvoda. Prijavljeni i neprijavljeni kupci imaju mogućnost pristupiti stranicama koje će prikazati sve marke, sve kategorije i sve proizvode (Slika 2). Stranica za marke dohvaća sve marke iz baze podataka i prikazuje ih u obliku kartica raspoređene u redu od tri kartice (Slika 20).



Slika 20: Kartični prikaz marke

Kartica marke sastoji se od slike marke i naziva marke. Naziv marke je poveznica koja, kada se pritisne, odvede korisnika do nove stranice koja prikazuje sve proizvode koji pripadaju toj marki (Slika 21). Stranica za kategorije dohvaća sve kategorije iz baze podataka i prikazuje ih u obliku kartica raspoređenih u redu po tri kartice, isto kao i marke (Slika 21).

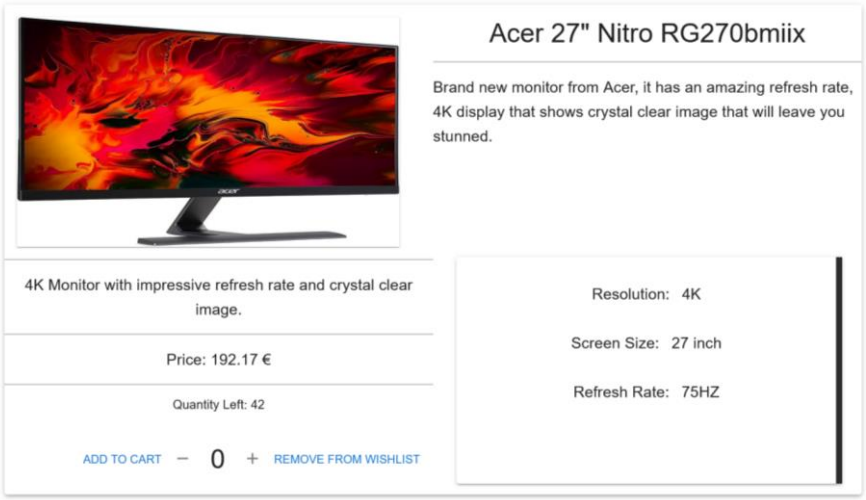


Slika 21: Proizvodi po marki ili kategoriji

Kartica kategorije sastoji se od slike kategorije i imena kategorije. Naziv kategorije je poveznica koja, kada se pritisne, odvede korisnika do nove stranice koja prikazuje sve proizvode koje spadaju u navedenu kategoriju (Slika 21). Stranica za proizvode prikazuje sve trenutačne proizvode u ponudi. Kartica proizvoda sastoji se od imena proizvoda i slike proizvoda. Kada korisnik pritisne naziv proizvoda, onda se otvori nova stranica koja prikazuje informacije o proizvodu.

4.3.2 Dodavanje proizvoda u košaru i popis želja

Stranica proizvoda za kupce prikazuje sve bitne informacije koje bi mogle zanimati kupca pri dodavanju proizvoda u košaricu. Stranica sadržava naziv proizvoda, sliku proizvoda, kratki opis, detaljniji opis proizvoda, sveukupnu količinu, tehničke podatke, dugme za dodavanje na popis želja, dugme za mijenjanje količine, za dodavanje u košaricu (Slika 22). Dugme za dodavanje u košaru, mijenjanje količine u košari i stavljanje na popis želja dostupni su jedino ako je kupac prijavljen. U slučaju da kupac dodaje već prije uneseni proizvod na popis želja dugme će kupcu dati informaciju da je proizvod već dodan na popis želja.



Slika 22: Stranica proizvoda za kupce

Ako kupac želi maknuti proizvod s popisa želja, samo mora pritisnuti dugme koje kaže „*Remove From Wishlist*“ i proizvod će nestati s popisa. Isti postupak vrijedi i pri dodavanju proizvoda na popis želja. Kao što je već prije navedeno, entitet za popis želja i entitet za proizvod imaju više na više kardinalnu vezu.

```
existingWishlistProduct.ifPresentOrElse(wishlistProduct -> {
          user.getWishlist().getProducts().remove(wishlistProduct);
             product.getWishlists().remove(wishlistProduct);
             wishlistProductRepository.delete(wishlistProduct);
         }, () -> {
             var wishlistProduct = WishlistProduct.builder()
                     .wishlist(user.getWishlist())
                     .product(product)
                     .createdAt(LocalDateTime.now())
                     .build();
             user.getWishlist().getProducts().add(wishlistProduct);
             product.getWishlists().add(wishlistProduct);
             wishlistProductRepository.save(wishlistProduct);
});
```

Ispis 7: Dodavanje i brisanje proizvoda iz popis želja

Kada se proizvod dodaje u popis želja, stvori se instanca entiteta popis želja-proizvod i sprema se u bazu, ali u slučaju da podatak već postoji u bazi, ta instanca s istim kupcem i proizvodom briše se iz baze podataka (Ispis 7). Kada kupac želi dodati proizvode u košaru,

mora prvo promijeniti količinu s nule, jer će u protivnom stranica prijaviti grešku. Naime, količina ne smije biti nula. Košara je „*slice*“ iz Redux Toolkita koji sadržava prazan JSON objekt u kojem će se spremati id proizvoda kao ključ i količina proizvoda koju kupac želi kupiti kao vrijednost, te polje koje označava jesu li svi proizvodi učitani.

```
addToCart: (state, { payload }) => {
      const newJson = {
        ...state.cart,
        [payload.productId]: payload.amount,
      };
      state.cart = newJson;
      state.productsLoaded = true;
      cookies.set(payload.userId, JSON.stringify(newJson), {
        expires: new Date(Date.now() + 1_800_000),
        sameSite: "none",
        secure: true,
      });
    }
```

Ispis 8: Dodavanje proizvoda u košari

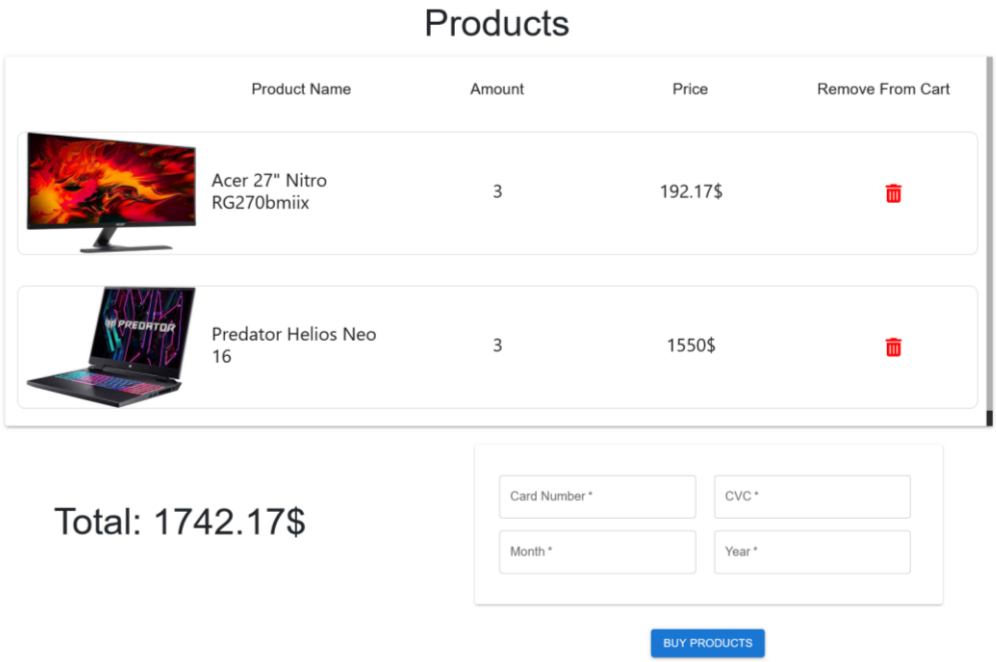
Košara sadržava i funkcije koje joj pomažu ukloniti proizvod, dodati novi (Ispis 8) ili promijeniti količinu za već postojeći, brisati sve proizvode iz košare i dohvatiti proizvod iz kolačića. Kupac koji se odjavi sa svog računa imat će mogućnost zadržati svoje odabire iz košarice, tako da će se odabiri spremiti u kolačić, ali taj kolačić trajat će samo 15 minuta (Ispis 9). Kolačić se sastoji od id kupca koji je ključ u kolačiću i vrijednosti koji je JSON objekt u obliku *stringa*. Ako je kolačić i dalje valjan, nakon prijave kupca, onda se dohvaćaju svi odabiri kupca iz kolačića i spremaju se košaru.

```
getFromCookiesToCart: (state, { payload }) => {
      const allCookies = cookies.getAll();
      if (!state.productsLoaded) {
        if (allCookies.hasOwnProperty(payload.userId)) {
          state.cart = allCookies[payload.userId];
          state.productsLoaded = true;
        }
      }
    }
```

Ispis 9: Dohvaćanje proizvoda iz kolačića

4.3.3 Kupnja proizvoda i plaćanje narudžbe

Nakon dodavanja proizvoda u košaru, kupac može pristupiti stranici gdje može vidjeti sve dodane proizvode u košarici (Slika 23). Košara prikazuje sve dodane proizvode u stupcu, sveukupnu cijenu koju korisnik mora platiti u polju za unos kartice. Proizvode prikazuje u obliku kartice sa slikom proizvoda, prikazuje količinu koju korisnik želi kupiti, cijenu pojedinačnog proizvoda i omogućava brisanje proizvoda iz košare (Ispis 10). Stranica prikazuje sveukupnu cijenu koju korisnik mora platiti prije nego što se stvori narudžba.



Slika 23: Košara kupca

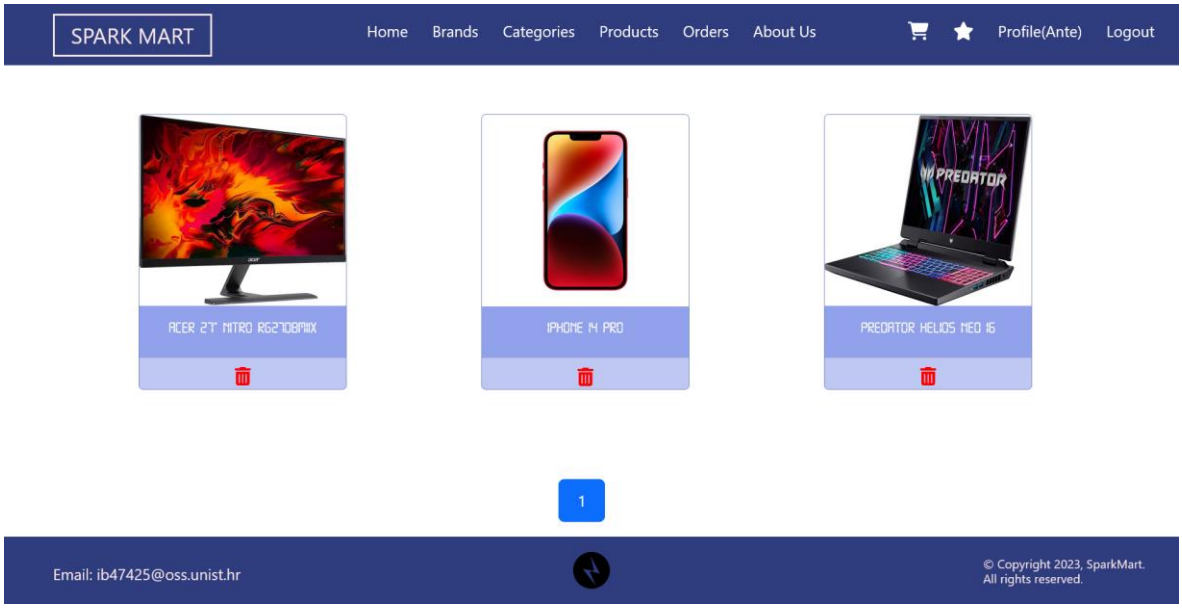
Polje za karticu se sastoji od četiri polja: polje za broj kartice, polje za CVC i polje za mjesec i godinu isteka kartice. U slučaju da kupac upiše krive informacije (nevaljani CVC i broj kartice) ili mu je kartica istekla, kupca se upozorava na grešku odnosno na neispravnu karticu. Kada kupac unese ispravne informacije o kartici, kreira se nova narudžba i sprema se u bazu podataka. Narudžba će se vezati za kupca, a proizvodi i narudžba će se međusobno vezati, jer oni imaju kardinalnu vezu više naprama više i koristi se proizvod-narudžba kao među entitet. Proizvod se uklanja iz popisa želja kada dođe do kupnje proizvoda.

```
removeFromCart: (state, { payload }) => {
      const json = state.cart;
      delete json[payload.productId];
      state.cart = json;
      if (Object.keys(state.cart).length === 0)
state.productsLoaded = false;
      cookies.set(payload.userId, JSON.stringify(json), {
        expires: new Date(Date.now() + 1_800_000),
        sameSite: "none",
        secure: true,
      });
    }
```

Ispis 10: Brisanje proizvoda iz košarice

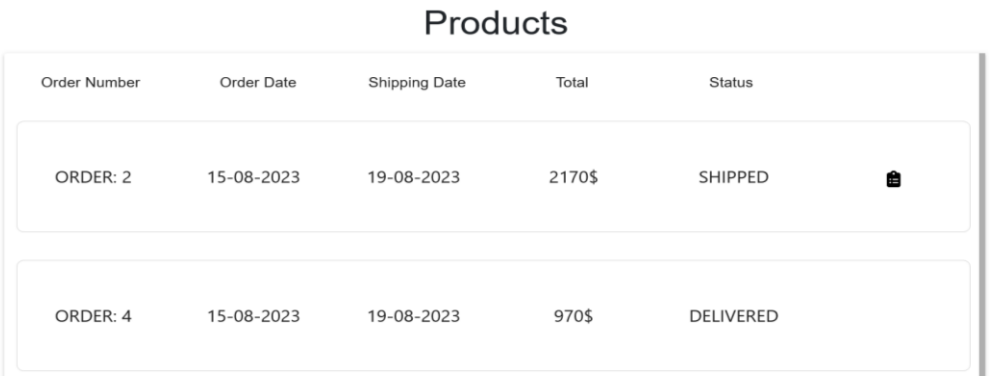
4.3.4 Pregled narudžbi, popis želja i potvrda dostave narudžbe

Kupac može pristupiti stranici za popis želja tako da pritisne dugme koje izgleda kao bijela zvijezda u gornjem desnom kutu pokraj ikone za košaricu. Popis želja sadržava sve proizvode koje je korisnik ranije dodao u popis želja. Proizvodi su prikazani u obliku kartice sa slikom i njihovim nazivom. Ako kupac pritisne naziv proizvoda, to će ga odvesti na stranicu za proizvod (Slika 22). Kupac ima mogućnost maknuti proizvod s popisa želja tako da pritisne ikonu za brisanje koja izgleda kao crvena kanta za smeće (Slika 24).



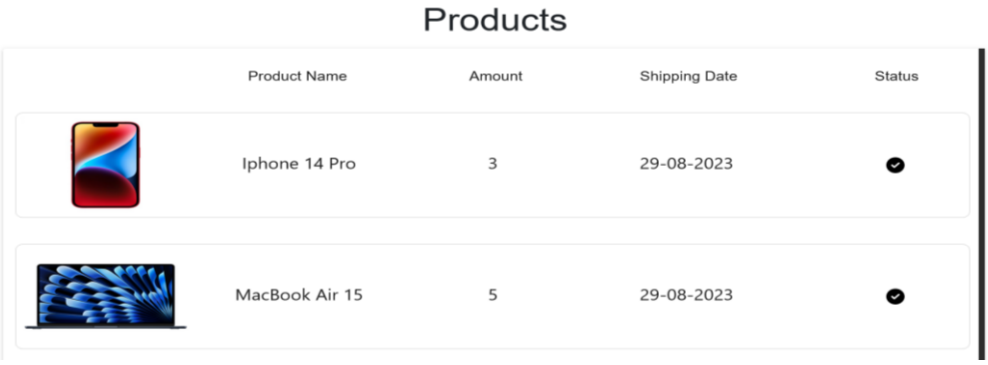
Slika 24: Popis želja proizvoda

Kupac može pristupiti stranici za narudžbe u navigaciji u sredini zaglavlja stranice. Stranica prikazuje sve narudžbe kupca i uz svaku narudžbu prikazuje broj narudžbe, datum narudžbe i isporuke, sveukupnu cijenu, status narudžbe i dugme koje vodi do stranice na kojoj omogućava korisniku potvrdu dostave pojedinačnih proizvoda. Dugme za prikaz svih proizvoda je jedino dostupan ako je status narudžbe isporučeno, inače se dugme neće prikazati (Slika 25).



Slika 25: Stranica za narudžbe od kupca

Kada kupac pristupi stranici za pojedinačne proizvode, dohvaćaju se svi proizvodi koji su vezani za tu narudžbu, ali samo proizvodi čija dostava još nije potvrđena. Stranica prikazuje sve proizvode čija dostava nije potvrđena i uz svaki proizvod prikazuje njegovu sliku, naziv, količinu, oznaku je li kupljeno, datum isporuke i dugme za potvrdu dostave. Kada je potvrđena dostava i posljednjeg proizvoda u narudžbi, onda je status narudžbe završen (Slika 26).



Slika 26: Potvrda dostave proizvoda

5. Zaključak

Razvoj aplikacije završnog rada pruža mogućnost studentima da na kreativan način prezentiraju znanja stečena tijekom studiranja. Omogućuje učenje novih tehnologija, pojmova i tehnika koje će poslužiti u budućim projektima.

Korištenje JWT tokena omogućuje učenje kako se autentifikacija i autorizacija odvijaju na modernom internetu. Sama konfiguracija JWT tokena nije trivijalna, ali njegovo korištenje mnogo doprinosi sigurnosti aplikacije. TypeScript olakšava korištenje JavaScripta i njegov tipizirani sustav je riješio mnogo problema vezanih uz JavaScript kao što su bolja kvaliteta, čitljivost i održavanje kôda. Naime, TypeScript često vraća lažno pozitivne (engl. *false positive*) pogreške koje se moraju ručno isključiti. Redux Toolkit omogućava lakše upravljanje složenim stanjima i dijeljenje podataka između komponenti uz dodavanje dodatnog sloja složenosti projekta. Material Design UI je omogućio lijepi, moderan izgled stranice bez dodatnog pisanja CSS-a i JavaScripta, no za slučaj da se želi unikatno dizajnirati izgled stranice ipak bi trebalo koristiti CSS i JavaScript. React omogućava stvaranje interaktivnih korisničkih sučelja ali dodatne biblioteke će imati utjecaja na veličinu projekta. Java omogućava stvaranje pouzdane i skalabilne *web* aplikacije ali ima nedostatak modernih karakteristika drugih programskih jezika kao na primjer asinkrono programiranje. Spring Boot omogućava brzi razvoj *web* aplikacija uz malo konfiguracije pomoću njegovog razvijenog ekosustava, ali zahtjeva pisanje velike količine *boilerplate* kôda.

U slučaju nadograđivanja aplikacije, trebalo bi se dodatno posvetiti izgledu korisničkog sučelja kako bi bio privlačniji kupcima koji uz njega provode vrijeme u potrazi za željenim proizvodima. Funkcionalnosti s kojima bi se mogla proširiti aplikacija su: mogućnost ocjenjivanja proizvoda, mogućnost ostavljanja recenzija, bolja zaštita od vanjskih napada, algoritam za predlaganje proizvoda kupcima, mogućnost biranja između više opcija konfiguracija proizvoda, prijava korisnika s nekim već postojećim računom (npr. Google) i mijenjanje jezika stranica ovisno o jeziku internetskog preglednika.

LITERATURA

[1] Microsoft, „TypeScript Documentation”, https://www.typescriptlang.org/docs/ (posjećeno 7.8.2023.)

[2] Meta, „React Documentation”, https://react.dev/learn (posjećeno 8.8.2023.)

[3] Dan Abramov, „Getting Started with Redux Toolkit”, https://redux-toolkit.js.org/introduction/getting-started (posjećeno 8.8. 2023.)

[4] Google, „Material UI – Overview”, https://mui.com/material-ui/getting-started/ (posjećeno 8.8 2023.)

[5] Oracle, „Java Documentation”, https://docs.oracle.com/en/java/ (posjećeno 8.8.2023.)

[6] VMware, „Spring Boot Documentation”, https://spring.io/projects/spring-boot#learn (posjećeno 8.8.2023.)

[7] Oracle, „MySQL Documentation”, https://dev.mysql.com/doc/ (posjećeno 8.8.2023.)

[8] Micheal B. Jones, John Bradley, Nat Sakimura, „JSON Web Token(JWT)“, https://datatracker.ietf.org/doc/html/rfc7519 (posjećeno 13.8.2023.)