Izrada mrežne trgovine u Angular programskom okviru

Pisačić, Leon

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:096651

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-20**

Repository / Repozitorij:

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Leon Pisačić

IZRADA MREŽNE TRGOVINE U ANGULAR PROGRAMSKOM OKVIRU

ZAVRŠNI RAD

Čakovec, kolovoz 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Leon Pisačić

CREATION OF AN ONLINE STORE IN THE ANGULAR FRAMEWORK

Završni rad

Mentor:
dr. sc. Sanja Brekalo

Čakovec, kolovoz 2023.

MEĐIMURSKO VELEUČILIŠTE U

ČAKOVCU ODBOR ZA ZAVRŠNI RAD

Čakovec, 5. svibnja 2023.

Države: **Republika Hrvatska**
Predmet: **Izrada Web Sadržaja-izborni**

ZAVRŠNI ZADATAK br. 2022-RAČ-R-36

Pristupnik: **Leon Pisačić (0313025578)**
Studij: Redoviti preddiplomski stručni studij Računarstvo
Smjer: Inženjerstvo računalnih sustava i mreža

Zadatak: **Izrada mrežne trgovine u Angular programskom okviru**

Opis zadataka:

Aplikacija se izrađuje u Angular programskom okviru. Koristi se baza podataka za zapisivanje podataka koji se skupljaju od strane korisnika i administratora. Aplikacija se izrađuje na primjeru mrežnog dućana. Za aplikaciju se predviđa da korisnici, koji su prodavači, unose svu robu koja se prodaje u sustav te tako prikazuju mrežnu robu koju je moguće naručiti *online* putem dostave. Pozadinski dio aplikacije omogućuje izdavanje računa i vođenje skladišta dućana. Administratori, nakon naručivanja robe, izdaju račune za *online* prodaju automatski ili sami izrađuju račune za prodaju u trgovini. Klijenti u aplikaciji mogu pregledavati vlastite narudžbe i ispisivati račune u PDF formatu.

Zadatak uručen pristupniku: 23. siječnja 2023.
Rok za predaju rada: 20. rujna 2023.

Mentor: Predsjednik povjerenstva za završni ispit

dr. sc. Sanja Brekalo, prof. v. š.

SAŽETAK

Mrežna, internetska trgovina ili e-trgovina, oblik je poslovanja u kojem se proizvodi i usluge prodaju putem interneta. To je virtualni prostor u kojem se trgovci i kupci mogu povezati, pregledavati proizvode, kupovati i plaćati bez fizičke prisutnosti u trgovini. Sukladno današnjim potrebama tržišta i utjecajem modernizacije, potražnja za mrežnim trgovinama sve je veća.

Tema završnog rada usmjerena je prema izradi i implementaciji mrežne trgovine. Aplikacija se izrađuje na primjeru mrežnog dućana koji se bavi prodajom tehničke opreme, uključujući i bijelu tehniku.

U radu su opisane korištene tehnologije te najvažniji postupci pri kreiranju mrežne trgovine. Trgovina je implementirana korištenjem različitih tehnologija i biblioteka kao što su HTML, CSS, TypeScript, Bootstrap, Visual Studio Code, MongoDB Atlas, Font Awesome i jsPDF.

Mrežna trgovina dijeli se na vidljivi i pozadinski dio. Vidljivi dio je onaj koji krajnji korisnici vide i s kojima direktno komuniciraju te je on implementiran putem HTML-a, CSS-a i TypeScript programskog jezika zaduženog za funkcionalnost. Pozadinski dio *web* stranice izrađen je korištenjem TypeScripta te MongoDB Atlas baze podataka koja direktno surađuje s vidljivim dijelom koda. Preko API zahtjeva, podacima se manipulira na pozadinskoj strani koda, nakon čega se rezultat vraća nazad prema prednjoj strani koda, gdje se isti podaci prikazuju vizualno krajnjem korisniku ili administratoru.

U radu su opisane tehnologije koje se koriste za izradu mrežne trgovine. Također, prikazani su i isječci dijelova mrežne trgovine. Mrežna trgovina podržava dva tipa korisničkih računa, od kojih je jedan administrativni i jedan korisnički (kupac). Korisnik ima mogućnost pregleda proizvoda, dodavanja proizvoda u košaricu i dovršavanja narudžbe, dok uz to ima i mogućnost pregledavanja vlastitih narudžbi i ispisivanje za svaki račun u PDF formatu. Administratoru je omogućena kompletna manipulacija proizvodima i korisnicima. Administrator ima mogućnost dodavanja, brisanja ili uređivanja postojećih proizvoda, kao i brisanje i prikaz trenutačno registriranih korisnika.

Ključne riječi: *mrežna trgovina, programski okvir, aplikacija, korisničko iskustvo, korisničko sučelje, Angular*

Sadržaj

1. UVOD 8

2. MREŽNE TRGOVINE 9

2.1 Povijest mrežnih trgovina 9

2.2 Mrežne trgovine danas 9

2.3 Mrežne trgovine u budućnosti 10

3. ANGULAR 11

3.1 Povijest Angulara 11

3.2 Razlike između AngularJS-a i Angulara 11

3.3 Angular komponente 12

3.4 Angular servisi 13

3.5 Komunikacija između komponenata 14

3.6 Angular rutiranje 14

3.7 Angular moduli 15

4. PROGRAMIRANJE PREDNJEG DIJELA 16

4.1 HTML 17

4.2 CSS 17

4.3 Bootstrap 18

4.4 TypeScript 20

5. PROGRAMIRANJE POZADINSKOG DIJELA 22

5.1 MongoDB baza podataka 23

5.2 API zahtjevi 23

6. PROCES IZRADE MREŽNE TRGOVINE 25

6.1 Struktura mrežne trgovine 26
6.2 Korisničko sučelje i iskustvo 31
6.3 Izrada i povezivanje baze podataka 36
6.4 Segmenti baze podatka 37
6.5 Komunikacija prednjeg i pozadinskog koda 38
6.6 Pokretanje aplikacije 39
7. ZAKLJUČAK 42
8. POPIS LITERATURE 43
9. PRILOZI 45

1. UVOD

U današnjem digitalnom dobu, *online* trgovine, odnosno mrežne trgovine, postale su neizostavan dio svakodnevice. S obzirom na sveprisutnost interneta, tvrtke i pojedinci sve više koriste moderne načine privlačenja klijenata koji im potencijalno mogu biti potrošači. Posjedovanje mrežne trgovine jedan je od načina širenja poslovanja. Putem mrežnih trgovina, poslodavci šire svoju klijentelu na globalnu razinu, dok je ista mogućnost ograničena samo posjedovanjem fizičkih trgovina. Mrežne trgovine postale su unosne tako da mogućnost kupnje proizvoda preko interneta omogućuju čak i vrlo mala poduzeća koja svjesno razumiju nedovoljan utjecaj na klijente samo putem fizičke trgovine.

Mrežna trgovina kreirana u Angular programskom okviru strukturno je podijeljena na prednji (engl. *front end*) dio, koji označuje dio koda koji je vidljiv krajnjem korisniku, i na pozadinski dio koda (engl. *back end*), odnosno dio koda koji krajnji korisnik ne vidi. Oba dijela su jednako i izrazito važna za rad mrežne trgovine. Mrežna trgovina razlikuje se po mogućnostima sukladno ulogom (administrator ili korisnik – kupac) te je stoga sam prikaz i upravljanje aplikacijom drugačije sukladno tipu korisničkog računa koji se koristi.

2. MREŽNE TRGOVINE

2.1 Povijest mrežnih trgovina

Povijest mrežnih trgovina počinje 1979. godine kada je poduzetnik Michael Aldrich iz Ujedinjenog Kraljevstva omogućio spajanje domaće televizije s korisničkim računalom za obradu transakcija putem telefonske linije [2]. Međutim, to je bio samo početak, odnosno prvo zabilježeno vrijeme kada se tematika mrežnih trgovina pojavila. Razvitkom *World Wide Weba* došlo je do velikog pomaka u upotrebi interneta te je jedno od prvih mrežnih mjesta za digitalnu trgovinu bio Amazon, koji je prvobitno započeo kao mrežna trgovina za prodaju knjiga. Amazon je od prvobitnog proizvoda i po broju narudžbi danas prerastao u najveću mrežnu trgovinu na svijetu.

2.2 Mrežne trgovine danas

Mrežne trgovine se danas ne razlikuju mnogo od njezinih početaka, a tematika je ista - omogućiti korisnicima velik izbor proizvoda, omogućavajući im obavljanje kupnje iz udobnosti doma. Prednosti mrežne trgovine su nebrojene, a među glavnima su:

- širi izbor asortimana koji nije limitiran na fizički prostor trgovine
- ušteda vremena koje bi osoba potrošila na odlazak u fizičku trgovinu
- povoljnije cijene, češće mogućnosti popusta
- kupnja se obavlja iz udobnosti doma
- pregledniji uvid u raspoložive veličine/dimenzije i količine.

No, uz sve pozitivne strane, mrežne trgovine, naravno, imaju i mane od kojih je možda i najveća nemogućnost fizičkog isprobavanja/ispitivanja proizvoda. Kupci ne mogu dobiti pravi osjećaj za kvalitetu, veličinu i izgled koji može varirati od onog sa slike u mrežnoj trgovini. Drugi je nedostatak dostava i njeni vremenski okviri koji mogu zavisiti od mjesta stanovanja i dostavljačke službe. Paketi se mogu čekati i do mjesec dana prije no što dođu u ruke kupca,

što mrežnu trgovinu ne stavlja u prvi izbor ako je riječ o kupnji u kojoj je proizvod korisniku potreban odmah.

2.3 Mrežne trgovine u budućnosti

Točnu budućnost i razvoj mrežnih trgovina teško je predvidjeti, no može se pretpostaviti temeljem tehnologija koje su zadnjih godina dobile na popularnosti. Među takvim tehnologijama valja izdvojiti umjetnu inteligenciju. Neke naznake kako bi se umjetna inteligencija mogla upotrijebiti u mrežnim trgovinama su da se pretpostavlja da će ona osmisliti i prikazati opis idealnog kupca. Personalizacija je još jedna od mogućnosti u kojoj se vidi mjesta za poboljšanje, predviđa se da će korisnici u budućnosti zahtijevati još veću razinu personalizacije, na što su vlasnici mrežnih trgovina već spremni jer koriste analitike podataka kako bi svakom korisniku isporučili potencijalno dobar izbor proizvoda. Umjetna inteligencija također se može koristiti za personalizaciju, a uloga će biti preporuka proizvoda, ciljano oglašavanje i personalizirani marketing putem elektroničke pošte.

Jedna od trenutno najvećih prepreka mrežnih trgovina možda će u skorijoj budućnosti biti riješena te tada i službeno postati bolji oblik trgovine. Najveći problem trenutno je nemogućnost isprobavanja/testiranja željene robe, a taj problem riješit će se putem proširene i virtualne stvarnosti. Navedene tehnologije omogućit će kupcu da virtualno isprobaju odjeću ili vizualiziraju namještaj u vlastitom domu. U narednim godinama također se očekuje veća ekološka svjesnost prodavača. Sve više i više trgovaca okreće se prema ekološki osviještenom načinu poslovanja pa tako vlastite proizvode pakiraju i isporučuju upravo u takvim pakiranjima, napravljenim od razgradivih i recikliranih materijala. Također se promoviraju novi načini pogona koji će rezultirati količinskim smanjenjem nastajanja ugljičnog dioksida [3].

Zaključno svim pretpostavkama, budućnost mrežnih trgovina može ići samo nabolje, kontinuirane promjene novih tehnologija pristižu svaki dan, korisničko iskustvo se poboljšava, a to će rezultirati većim prometom za prodavače, odnosno za vlasnike mrežnih trgovina. Tehnologija u koju se polaže najviše očekivanja je umjetna inteligencija koja će svojim širokim spektrom mogućnosti zasigurno podići trenutačnu razinu mrežnih trgovina na još višu i bolju.

3. ANGULAR

Angular je programski okvir otvorenog koda napisan u TypeScriptu, izvedenici JavaScript programskog jezika. Primarna svrha Angulara razvoj je jednostraničnih aplikacija, a sam programski okvir podržan je i razvijan od Googlea. Angular je objavljen pod licencom MIT-a, a kao okvir donosi mnoge prednosti, među kojima je to da omogućuje stvaranje velikih aplikacija koje su lake za održavanje [4].

3.1 Povijest Angulara

Povijest Angulara počinje 2010. godine kada su ga, tada prvobitno AngularJS, razvili dvoje Googleovih programera, Miško Hevery i Adam Abronas. AngularJS predstavljen je kao okvir otvorenog koda te je već od samog početka bio popularan. Programski okvir razvijen na temelju Apache Cordova (okvir za razvoj mobilnih aplikacija) poticao je programere da kreiraju mobilne aplikacije upravo koristeći Angular programski okvir. Nakon kraja prve verzije te početka iduće druge verzije Angular programskog okvira, naziv istoga promijenili su Googleovi programeri u samo Angular. Od tada pa sve do zadnje raspoložive verzije (16.) ime je ostalo isto [5].

3.2 Razlike između AngularJS-a i Angulara

Najočitija razlika koja se dogodila s prijelaza iz prve na drugu verziju Angular programskog okvira krije se i u imenu. AngularJS prvobitno je bio baziran na programskom jeziku JavaScript (JS), dok je druga i svaka naredna verzija preuzela programski jezik TypeScript, koji je proširena verzija JavaScripta. Sukladno TypeScript kompleksnosti, potrebno je više vremena kako bi se isti programski jezik savladao pa to upravo dovodi Angular do kompleksnosti učenja, što može i ne treba biti nedostatak.

Razlika u direktivama između prvih dviju verzija programskog okvira je ta što je prva verzija koristila paket istih, dok svaka naredna koristi standardne direktive. Na primjer, ako se s AngularJS-a želi koristiti *two way binding*, koristi se 'ng-model', dok se za *one way binding* koristi 'ng-bind'. Ista stvar je provedena u novijim verzijama Angulara, samo preko 'ngModel'

direktive. Za korištenje *two way bindinga* koriste se znakovi '[()]', odnosno '[]' za *one way binding*.

```
<input type="text" [(ngModel)]="username">
```

Kod 1. Primjer korištenja *one way bindinga* u novijim verzijama Angulara

Izvor: Autor

AngularJS nema podršku za mobilne uređaje, dok je s kasnijim verzijama ovog programskog okvira ta mogućnost omogućena.

Performanse su još jedan od faktora koji je poboljšan s obzirom na prvu verziju Angulara. Naime, *two way binding* izazvao je mnogo problema u prvoj verziji programskog okvira i s obzirom na to da struktura istog nije bilo dovoljno dobro posložena, to je rezultiralo usporavanjem rada aplikacije. S verzije na verziju, Angular tim kontinuirano je radio na poboljšanju vlastite strukture, što je, već u početnim verzijama Angulara, rezultiralo poboljšanim performansama i bržim radom aplikacija [6].

3.3 Angular komponente

Komponente su glavni blokovi sadržaja koji se koriste za aplikacije u Angular programskom okviru. Svaki blok se sastoji od:

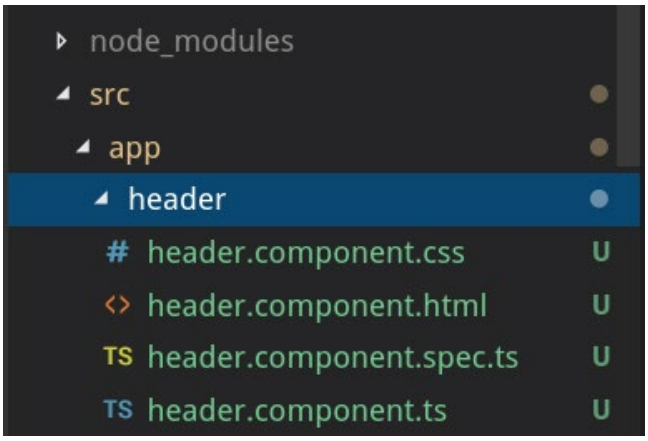
- HTML dokumenta
- CSS dokumenta
- TypeScript dokumenta
- Spec.ts (testni dokument). [7]

Kreiranje komponente, osim onih predefiniranih (app.component) odvija se na sljedeći način:

```
ng generate component ime-komponente
```

Kod 2. Komanda za kreiranje komponente putem naredbenog retka

Izvor: Autor



Slika 1. Angular komponenta s njezinim datotekama

Izvor: https://education.launchcode.org/intro-to-professional-web-dev/chapters/angular-lsn1/components.html (24.5.2023.)

Komponente su sastavni dio Angular programskog okvira te bez njih nije moguć rad aplikacije. Mrežna trgovina, kao i sve druge, sadrže određeni broj komponenti. Komponente mogu biti zasebna cjelina ili se unutar jedne može kreirati i druga. Kreiranje podkomponente obično se izvršava kada je riječ o funkcionalnosti koja ide jedna s drugom, odnosno potrebna je *parent* komponenta kako bi podkomponenta *child* funkcionirala.

Komponente se također mogu kreirati 'ručno', no Angular isporučuje mnogo bržu i bolju verziju koja se implementira putem naredbenog retka. Ako se komponenta kreira „ručno“, korisnik se treba pobrinuti da novokreirani blok sadržaja uključi u predefinirani modul (app.module.ts) kreiran od Angulara. Stoga je svakako bolje kreirati komponentu bržim putem zbog smanjenja mogućnosti pogrešaka i uštede vremena.

3.4 Angular servisi

Servis je mjesto u Angular aplikaciji gdje su smještene sve metode i logika koja se želi koristiti kroz više komponenata. Prednost servisa je ta da se pomoću njih izvršava kompletna

komunikacija između komponenata. Sva logika koja ne bi trebala biti ograničena na lokalnu primjenu piše se u servise jer je način pristupa varijabli ili metodi iz nje vrlo jednostavan.

```
ng generate service ime-servisa
```

Kod 3. Naredba za kreiranje servisa

Izvor: Autor

3.5 Komunikacija između komponenata

Komunikacija između Angular komponenata najčešće se provodi na dva načina.

Prvi način komunikacije je putem ulaza i izlaza. Ovakav način komunikacije odnosi se isključivo na komponente koje su u odnosu roditelj-dijete, odnosno komponente koje unutar sebe sadrže druge podkomponente. Ako je riječ o takvom odnosu, glavna komponenta ima ulogu roditelj, dok sve njegove podkomponente imaju ulogu dijete. Roditeljska komponenta može slati podatke do komponente dijete putem izlaza. Realizacija se obavlja deklariranjem svojstva dekoratora '@Input()'. Komponenta dijete također može komunicirati s roditeljskom komponentom putem izlaza. Realizacija se obavlja putem EventEmittera i dekoratora '@Ouput()'.

Drugi i najčešći način za komunikaciju obavlja se putem servisa. Pozitivna strana komunikacije putem servisa je ta da je sadržaj unutar servisa dostupan svim komponentama pa se tako vrijednosti ili metode jednostavno mogu pozivati iz komponente u komponentu. Način na koji se provodi takva komunikacija je da se unutar komponente koja zahtijeva vrijednost poziva servis preko konstruktora, a nakon toga se istom vrijednosti mogu pozvati varijable ili metode kreirane u servisu.

3.6 Angular rutiranje

Prilikom kreiranja novog Angular projekta putem naredbenog retka, korisniku se postavlja upit želi li automatski dodati Angular datoteku za rutiranje. Datoteka za rutiranje služi kako bi u istoj definirali rute za navigaciju između stranica. Ovo je također neizostavna stavka ako je riječ o valjanoj aplikaciji s više stranica. Svaka komponenta mora imati rutu koju korisnik

proizvoljno odabire, a kao drugi argument postavlja se ime komponente koja se želi koristiti pod istom rutom. Imena komponenta automatski su generirana od programskog okvira i to po dodijeljenom imenu komponente.

```
{ path: "registracija", component: RegistracijaComponent }
```

Kod 4. Primjer deklariranja rute za registracijsku komponentu
Izvor: Autor

3.7 Angular moduli

Aplikacije u Angularu su modularne i Angular ima vlastiti sustav modularnosti koji se naziva 'NgModules'. To su mehanizmi koji grupiraju komponente, direktive i servise koji su povezani na takav način da se kombiniraju s drugim modulima s krajnjim rezultatom stvaranja aplikacije [9]. Implementacija modula već je automatski stvorena u vrijeme kreiranja novog Angular projekta. Moduli su smješteni u datoteci pod imenom 'app.module.ts'. Modul je kreiran pomoću 'NgModule' dekoratora koji omogućuje pretvaranje klase u Angular modul. Unutar modula nalaze se tri predefinirana svojstva, a to su svojstva: 'imports', 'declarations' i 'bootstrap', a česta je i pojava svojstvo 'providers'.

Svojstvo 'imports' služi kako bi se u njega uključili drugi moduli od kojih je vlastiti modul zavisan. To mogu biti moduli kreirani od Angulara ili moduli kreirani ili pozvani od korisnika. Moduli zapisani unutar ovog područja krucijalni su za rad aplikacije s obzirom na to da sa sobom nose određenu vrstu funkcionalnosti koja je potrebna aplikaciji.

'Declarations' služi za deklariranje svih komponenata i direktiva koji se nalaze u aplikaciji. Komponente koje su uključene u ovo svojstvo mogu biti korištene unutar modula ili podijeljene s ostalim modulima koji uvoze isti. Prilikom samog kreiranja komponente ona će automatski biti dodijeljena u ovo svojstvo modula, stoga nema potrebe za ručnim unosom.

Zadnje zadano svojstvo modula je svojstvo 'bootstrap' koje služi za označavanje *root* komponente za primjenjivanje Bootstrapa prilikom pokretanja aplikacije. Tipično, ovo svojstvo sadrži samo 'AppComponent' komponentu u kojoj se pozivaju sve druge komponente aplikacije.

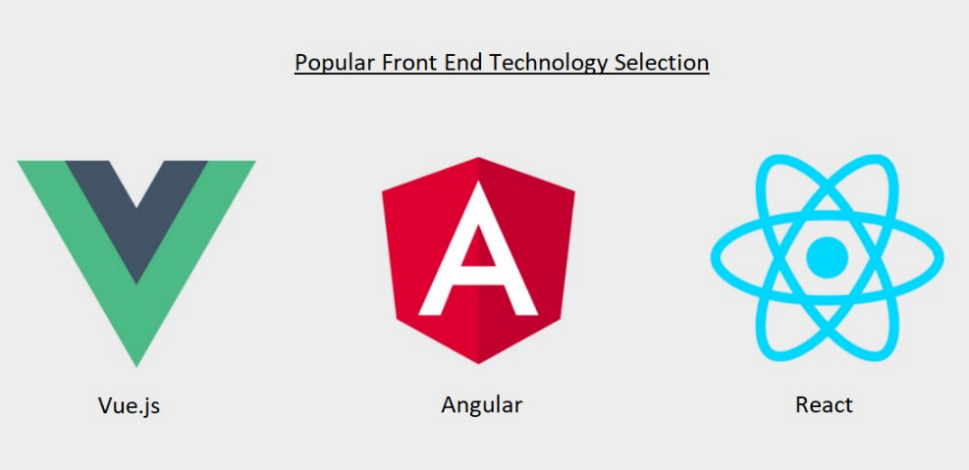
4. PROGRAMIRANJE PREDNJEG DIJELA

Front end development vrsta je programiranja u kojoj se programer fokusira na 'prednju' stranu aplikacije, odnosno sve što korisnik može vidjeti, a u to se ubraja korisničko sučelje i korisničko iskustvo aplikacije. Uloga ovakvog programera uključuje rad na dizajnu, razvoju i implementaciji vizualnih i interaktivnih elemenata koje korisnici izravno vide i s kojima međusobno komuniciraju. Također, ovakva vrsta programera zaslužna je za to da sadržaj koji se kreira bude jednostavan za korištenje, estetski ugodan i funkcionalno učinkovit.

Navedena vrsta programera uključuje pisanje koda u prezentacijskom jeziku HTML, u stilskom jeziku CSS, kao i programskom jeziku JavaScript, odnosno TypeScript.

Front end programeri također mogu koristiti biblioteke ili programske okvire za lakše i bolje pisanje koda. Osim Angular programskog okvira, postoje i popularni su sljedeći programski okviri:

- React.js
- Vue.js.



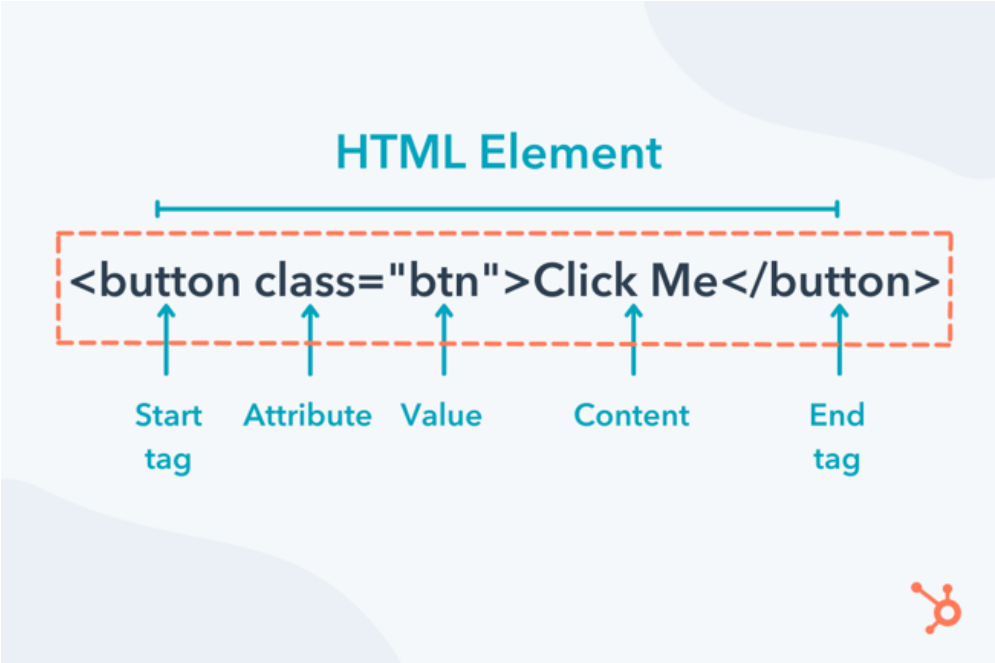
Slika 2. Najpoznatiji mrežni programski okviri

Izvor: https://www.hiremobiledeveloper.com/blog/why-we-use-vue-js-more-than-angular-and-react (24.5.2023.)

4.1 HTML

HTML ili engl. *HyperText Markup Language* prezentacijski je jezik za izradu mrežnih stranica i aplikacija.

HTML služi za stvaranje strukture mrežne aplikacije i koristi se za definiranje elemenata i njihove međusobne veze na mrežnoj stranici. Posebne oznake (elementi) koriste se za označavanje različitih dijelova stranice. Primjeri elemenata su naslovi, odlomci, tablice, poveznice. Način na koji se implementiraju elementi je taj da je početak elementa napisan u znakovima '<element>', u sredini se piše sadržaj koji se želi prikazati, dok je kraj elementa označen za znakovima '</element>'. Potencijalno je moguće dodavanje klasa ili ID-a kako bi se omogućila komunikacija sa stilskim jezicima.



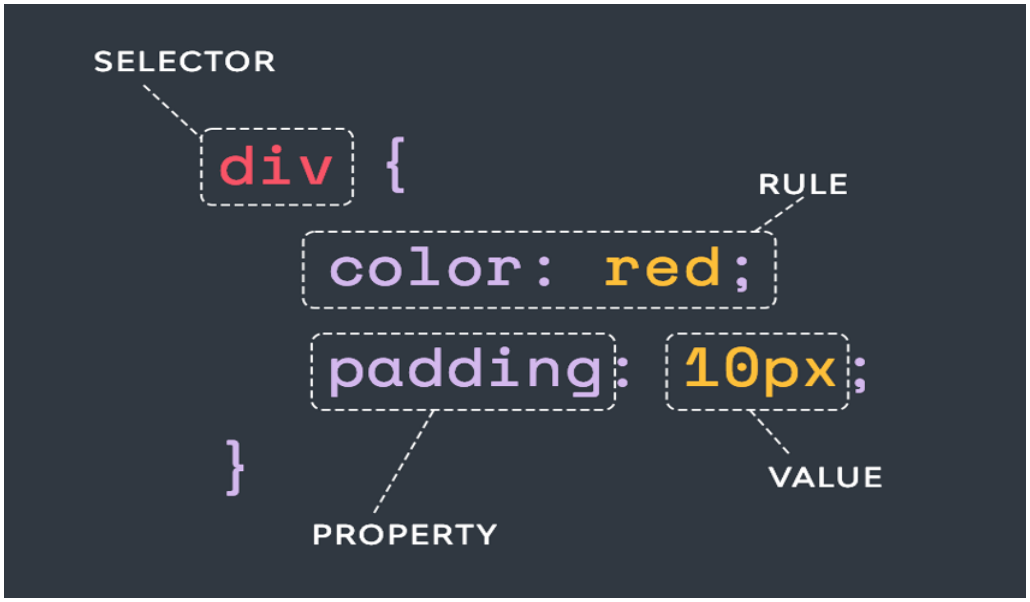
Slika 3. HTML element (gumb) i njegova struktura

Izvor: https://blog.hubspot.com/website/html-elements (24.5.2023.)

Sadržaj mrežne aplikacije nije moguć bez HTML-a s obzirom na to da je sav sadržaj koji se vidi na stranici, uključujući i riječi, slike ili tablice napisan i pozvan preko istog prezentacijskog jezika.

4.2 CSS

CSS ili engl. *Cascading Style Sheets* stilski je jezik koji se koristi za definiranje izgleda i dizajna mrežne stranice. On pruža mogućnost mijenjanja boja, fontova, omogućuje animacije i uvelike pridonosi korisničkom iskustvu i korisničkom sučelju. Većinu vremena služi zajedno s HTML prezentacijskim jezikom s kojim se povezuje putem klasa ili ID selektora. Putem CSS-a obavlja se kompletna manipulacija izgleda i ponašanja mrežne aplikacije, što je također jedan od bitnijih faktora gledano sa strane korisnika. Korisnici uglavnom zahtijevaju da stranica koju posjete bude jednostavna i estetski ugodna, a za ispunjenje tih faktora koristi se ovaj stilski jezik.



Slika 4. Dodjeljivanje CSS vrijednosti HTML div elementu

Izvor: https://webdesign.tutsplus.com/articles/html-css-for-beginners-mega-free-course--cms-93199 (24.5.2023.)

4.3 Bootstrap

Bootstrap je popularni CSS programski okvir koji se koristi za izradu responzivnog dizajna mrežne aplikacije. Bootstrap uvelike olakšava način i prikazivanje mrežnog sadržaja, s obzirom na to da s lakoćom primjenjuju njegove predefinirane klase. Jedna od najvećih prednosti Bootstrapa je *grid* sistem. *Grid* sistem omogućava organizaciju elemenata na mrežnoj stranici u redove i kolone, što olakšava izradu responzivnog dizajna. Danas nije dovoljno napraviti aplikaciju isključivo za jednu dimenziju ekrana jer velika količina

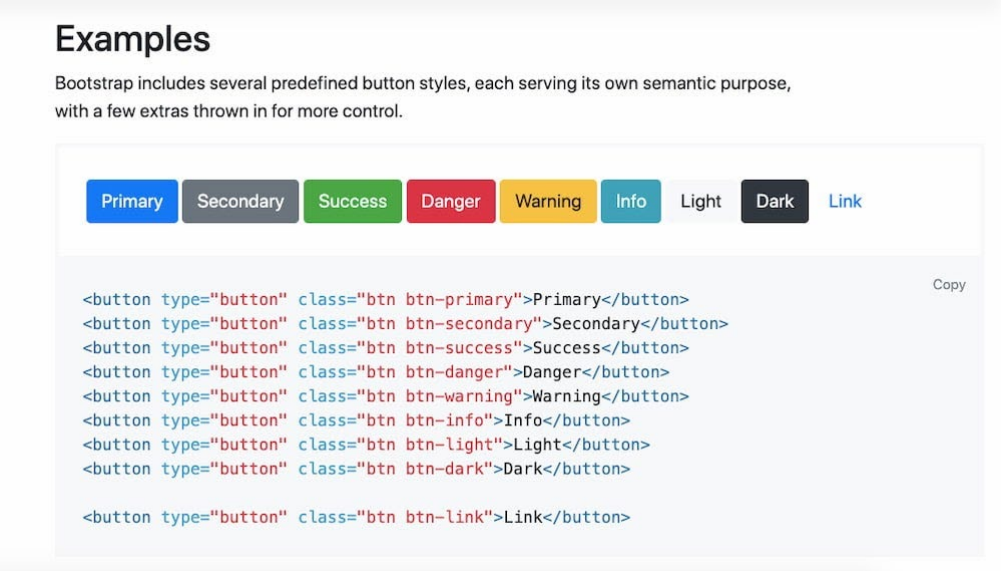
korisnika ima uređaje s raznovrsnim veličinama ekrana. S obzirom na tu činjenicu, ne može se ograničiti na samo na jednu vrstu uređaja pri izradi mrežne stranice te je nužno koristiti i implementirati responzivan dizajn. Responzivan dizajn je dizajn koji se prilagođava svim (ili barem većini) veličinama ekrana, a on se, između ostalog, primjenjuje pozivanjem Bootstrap predefiniranih klasa. Iz tog razloga, korisniku bi na mobilnom uređaju preglednost trebala biti jednaka kao i korisniku na prijenosnom računalu, iako je sam sadržaj drugačije implementiran upravo zbog manje, odnosno veće površine zaslona. Responzivan dizajn trebao bi podržavati standardne dimenzije mobilnih uređaja, tableta, prijenosnih računala i stolnih računala, tj. monitora.

Slika 5. Primjer responzivnog dizajna

Izvor: https://www.bluecorona.com/faq/what-is-a-responsive-website-design (24.5.2023.)

Način kojim se Bootstrap klasa povezuje s kodom je da se Bootstrap poziva unutar HTML prezentacijskog jezika ili putem modela u slučaju korištenja Angular programskog okvira. Nakon toga korisnik ima pristup kompletnoj Bootstrap CSS biblioteki klasa.

'Container', 'btn' i 'dark-mode' samo su neke od Bootstrap klasa koje dokumentu mogu odmah promijeniti vizualnu strukturu sadržaja aplikacije.



Slika 6. Različite Bootstrap klase

Izvor: https://blog.hubspot.com/website/bootstrap-image-classes (24.5.2023.)

4.4 TypeScript

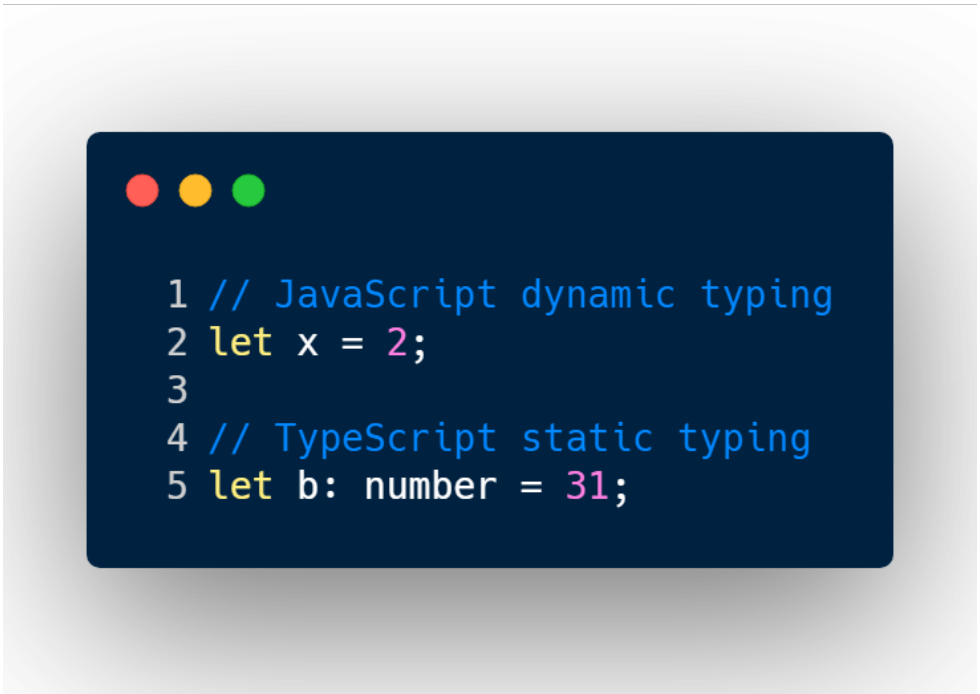
TypeScript je programski jezik otvorenog koda razvijen od Microsofta, a predstavlja nadskup JavaScript programskog jezika. To znači da su svi validni JavaScript kodovi također validni i u TypeScriptu. Iako je TypeScript proširena verzija JavaScripta, za vrijeme izvršavanja programski se kod izvršava u JavaScript programskom jeziku jer je to jedini jezik koji internetski preglednik razumije. TypeScript kompajler prevodi kod u JavaScript kako bi bio kompatibilan s okruženjima (okruženja su takva da je JavaScript jedini programski jezik koji je ugrađen u većinu modernih internetskih preglednika).

TypeScript se u usporedbi s JavaScriptom razlikuje po strogom deklariranju podatkovnog tipa varijable (statička tipizacija) i raznim dodacima koji jeziku olakšavaju razvoj i održavanje velikih i kompleksnih aplikacija [10].

Statička tipizacija je stvar subjektivne preferencije, no ne smije se zaboraviti zašto je ona ipak bolji izbor od dinamičke tipizacije:

- otkrivanje grešaka prije izvršavanja koda
- veća pouzdanost i produktivnost

- bolja dokumentacija i razumijevanje koda.



Slika 7. Dinamično deklariranje varijable u JavaScriptu, odnosno statičko u TypeScriptu

Izvor: https://linguinecode.com/post/what-is-typescript-5-easy-steps-to-add-it-to-react (24.5.2023.)

5. PROGRAMIRANJE POZADINSKOG DIJELA

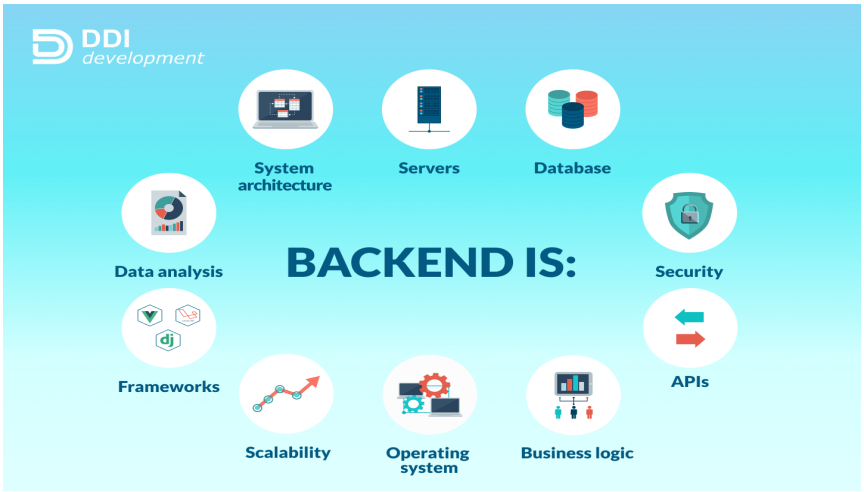
Pozadinsko programiranje (engl. *back end*) odnosi se na razvoj softvera koji se izvršava na serveru ili poslužitelju. Takva vrsta programera fokusira se na izgradnju funkcionalnosti, stvara osnovnu logiku aplikacije i omogućava komunikaciju između korisničkog sučelja i baze podataka.

Najjednostavnije rečeno, pozadinski dio omogućuje da se svi podaci ispravno šalju u preglednik. Za razliku od *front end* programiranja, pozadinsko (engl. *back end*) programiranje fokusira se na stvari koje krajnjem korisniku nisu vidljive, no krucijalno su važne za rad aplikacije jer, bez ijednog od navedena dva načina programiranja, mrežna aplikacija ne bi bila valjana.

Pozadinski način programiranja koristi različite programske jezike i tehnologije kako bi se izradila *server-side* aplikacija, odnosno aplikacija koja se izvršava na serveru ili poslužitelju.

Neki od najpopularnijih jezika za izvršavanje pozadinskog programiranja su:

- JavaScript (pomoću Node.js)
- Java
- C#
- PHP.



Slika 8. Prikaz obaveza (engl. *back end*) programera
Izvor: https://ddi-dev.com/blog/programming/backend-development-key-languages-technologies-features-in-2020 (26.5.2023.)

5.1 MongoDB baza podataka

MongoDB je višeplatformski program baze podataka klasificiran kao NoSQL baza podataka, odnosno baza podataka koja ne koristi SQL jezik kao standardni jezik upita. MongoDB pruža fleksibilne sheme koje se lako skaliraju s velikim količinama podataka i velikim korisničkim opterećenjem. NoSQL baze koriste se gotovo u svakoj industriji. Primjenjuju se kada se koristi pohranjivanje strukturnih i polustrukturnih podataka te kada se posjeduju ogromne količine podataka ili zahtjevi za *scale out* arhitekturu.

Umjesto korištenja tablica i redaka kao u SQL tipu podataka, MongoDB arhitektura sastoji se od zbirki i dokumenata. Dokumenti se sastoje od parova ključ-vrijednost, što je osnovna jedinica podataka MongoDB-a. MongoDB nudi podršku za mnoge programske jezike kao što su C, C++, C#, Go, Java, Python, Ruby i Swift [11].

5.2 API zahtjevi

API (engl. *Application Programming Interface*) je zahtjev koji se šalje putem aplikacijskog programskog sučelja kako bi se ostvarila komunikacija između aplikacije i sustava. API zahtjevi omogućuju klijentskim aplikacijama da pristupe i manipuliraju podacima na udaljenim serverima putem standardnih protokola kao što je HTTP [12].
API zahtjevi najčešće se stvaraju u Node.js tehnologiji, točnije njezinom programskom okviru Express.js. API zahtjevi koriste Node.js kao temeljnu platformu, ali koriste Express.js kako bi se olakšalo definiranje ruta, obrada zahtjeva i druge funkcionalnosti vezane uz razvoj mrežnih aplikacija.

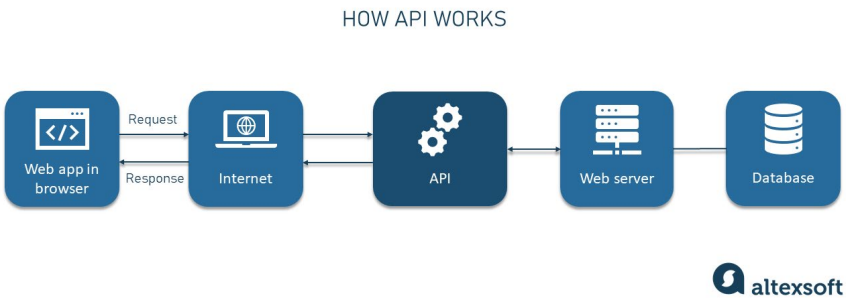
API zahtjevi također mogu sadržavati dodatne parametre i zaglavlja koji specificiraju dodatne informacije ili uvjete zahtjeva. Također, API zahtjevi mogu koristiti različite formate podataka za razmjenu informacija, najčešće su to JSON (engl. *JavaScript Object Notation*) ili XML (engl. *eXtensible Markup Language*). Četiri osnovne vrste HTTP zahtjeva su:

GET – kao što i sam naziv kaže, GET zahtjev je vrsta API zahtjeva koja samo dohvaća i vraća podatke, bez ikakve druge funkcije.

POST – zahtjev koji se koristi za obavljanje radnje, odnosno postavljanje novih podataka. Primjerice, ova vrsta API zahtjeva koristi se prilikom slanja obrasca novokreiranog korisnika. Obrazac tada obično sadrži osnovne podatke o novostvorenom korisniku te njegovu jedinstvenu ID vrijednost.

PUT – ovakva vrsta zahtjeva koristi se za ažuriranje već postojećih podataka ili vrijednosti. PUT zahtjev omogućuje zamjenu cijelog sadržaja ili samo određenih polja.

DELETE – ova vrsta API zahtjeva zadužena je za trajno brisanje određenog resursa, a primjena ovog zahtjeva koristi se pri brisanju proizvoda u administracijskom načinu rada.



Slika 9. Način na koji API zahtjevi rade

Izvor: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation (26.5.2023.)

Ne treba izostaviti da su API zahtjevi vrlo važan aspekt u *back end* programiranju i da bi bez njih komunikacija između aplikacija, platforma i sustava bila izuzetno otežana. Nedostatak API zahtjeva rezultirao bi ograničenom integracijom, prakticirao bi se ručni unos podataka, odnosno razmjenjivanje informacija teklo bi upravo takvim putem. Upravo zbog toga je potrebno koristiti API zahtjeve.

6. PROCES IZRADE MREŽNE TRGOVINE

Mrežna aplikacija sastoji od prednjeg (vidljivog) i pozadinskog dijela mrežne aplikacije.

Prednji dio sastoji se od uobičajenih tehnologija - HTML, CSS i TypeScript programskog jezika. Također, koristi se Angular programski okvir, dok je kod pisan u kodnom uređivaču Visual Studio Code. Struktura stranice, tj. sav sadržaj vidljiv u aplikaciji izrađen je preko HTML-a. Gumbi, cijene proizvoda, slike i općenito sav sadržaj koji u sebi sadrži neki oblik slova ili brojeva kreiran je u navedenom prezentacijskom jeziku. CSS stilski jezik upotrebljava se za stiliziranje cijele aplikacije, obrube za svaki pojedini proizvod na početnoj stranici, animacije izbornika s kategorijama proizvoda, animacije pri mijenjanju reklamnog

sadržaja i svom ostalom estetikom i dizajnom vidljivom u mrežnoj aplikaciji. Uz sirov CSS, koristi se i CSS programski okvir Bootstrap koji svojim predefiniranim klasama pomaže pri stvaranju responzivnog dizajna aplikacije. TypeScript programski jezik služi za stvaranje logike i funkcionalnosti koja se nalazi u aplikaciji. Prijava, odjava, registracija novih korisnika i filtriranje proizvoda po kategorijama samo su neke od funkcija koje su implementirane putem TypeScript programskog jezika.

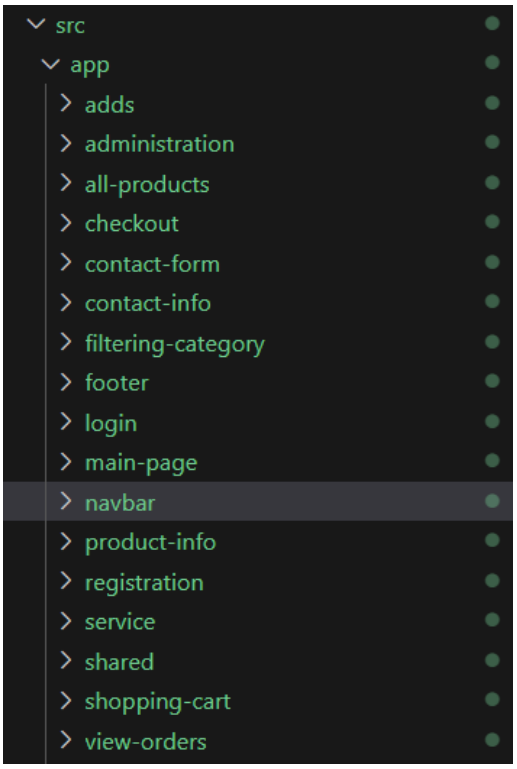
Pozadinski dio aplikacije ostvaruje se također putem pisanja programskog jezika TypeScript i nastavlja prema MongoDB bazi podataka, u kojoj se čuvaju svi podaci potrebni za nesmetan rad aplikacije. Koristi se oblak verzija baze podataka pod točnijim nazivom MongoDB Atlas. Svi proizvodi, registrirani korisnici s njihovim osobnim podacima i povijest narudžbi za pojedinog korisnika podaci su koji se spremaju i automatski mijenjaju ako dođe do kakvih promjena. Razlog korištenja Cloud verzije baze podataka je radi automatske konfiguracije, postavljanja razine sigurnosti, rezervne kopije podataka i radi korisničkog sučelja, što omogućuje bolji i pregledniji uvid u sve raspoložive tablice i podatke unutar baze podataka.

API zahtjevi implementirani su putem Express.js programskog okvira korištenog u kombinaciji s Node.js platformom za izvršavanje koda na serverskoj strani. Značajke koje se koriste prilikom korištenja Express.js programskog okvira su rutiranje, dinamički parametri ruta, Middleware, korištenje modula itd.

6.1 Struktura mrežne trgovine

Struktura mrežne trgovine sastoji se od osnovnih Angular funkcija. Komponente, moduli i datoteka za rutiranje funkcije su koje su implementirane pri izradi mrežne aplikacije.

Komponente su osnova Angular projekta, a mrežna trgovina načinjena je od mnogo komponenti od kojih su sve samostalne, osim jedne (administrativne) koja je u odnosu roditelj-dijete. Ostale komponente su samostalne i ne sadrže svoje podkomponente. Unutar komponente nalaze se datoteke za implementiranje već prije spomenutih tehnologija HTML, CSS i TypeScript (ts). U komponenti se također nalazi testna datoteka (s ekstenzijom .spec.ts), no ona nije neophodna za nesmetan rad mrežne trgovine.



Slika 10. Sve korištene komponente potrebne za rad mrežne trgovine

Izvor: Autor (28.5.2023.)

Datoteka za rutiranje jedna je od važnijih funckija, s obzirom na to da omogućuje preusmjeravanje između stranica, odnosno komponenti. Prelaskom na drugu stranicu mijenja se adresa koja se poziva kao prvi argument, dok se u drugom argumentu poziva komponenta koju korisnik želi pozvati pod istom adresom.

```
const routes: Routes = [ /*routing file, zaduzen za redirekciju izmedu više stranica */

  { path: "opis", component: OpisProizvodaComponent },
  { path: "kosarica", component: KosaricaComponent },
  { path: "checkout", component: CheckoutComponent },
  { path: "kontaktInfo", component: KontaktInfoComponent },
  { path: "pregledNarudzbi", component: PregledNarudzbiComponent },
  { path: "editProizvoda", component: EditProizvodaComponent },
  { path: "administracija", component: AdministracijaComponent },
  { path: "login", component: LoginComponent },
  { path: "registracija", component: RegistracijaComponent },
  { path: "kontaktForma", component: KontaktFormaComponent },
  { path: "glavno", component: MainPageComponent },
  { path: ":id", component: FiltriranjeKategorijaComponent },
  { path: "", component: PocetnaComponent }//glavna stranica
];
```

Slika 11. Način implementacije datoteka za rutiranje (app-routing.module.ts)

Izvor: Autor (28.5.2023.)

Kada se govori o osnovnoj strukturi Angular programskog okvira, gotovo je nemoguće ne spomenuti module. Moduli se koriste za uvoz drugih potrebnih modula, deklariranje komponenti i navođenje komponente za primjenjivanje Bootstrapa prilikom pokretanja aplikacije (to je najčešće komponenta 'AppComponent').

```
@NgModule({
  declarations: [
    AppComponent,
    NavbarComponent,
    MainPageComponent,
    FooterComponent,
    LoginComponent,
    RegistracijaComponent,
    OpisProizvodaComponent,
    ReklameComponent,
    KosaricaComponent,
    CheckoutComponent,
    AdministracijaComponent,
    EditProizvodaComponent,
    PregledNarudzbiComponent,
    KontaktInfoComponent,
    FiltriranjeKategorijaComponent,
    PocetnaComponent,
    KontaktFormaComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    FontAwesomeModule,
    HttpClientModule
  ],
  providers: [
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Slika 12. Glavni modul mrežne trgovine (app.module.ts)

Izvor: Autor (28.5.2023.)

Servisi su zaslužni za pristupanje metodama i varijablama na globalnoj razini. Način na koji se pozivaju i implementiraju je da se unutar TypeScript datoteke u konstruktoru poziva servis, pritom određujući tip reference (javni ili privatni), a nakon toga sav se sadržaj iz servisa nesmetano može pozvati i koristiti unutar komponente.

```
lokalniIndex: number = 0;

constructor(private data: DataService, private router: Router) { /*uključivanje servisa za pristup njegovim sadržajem */
  this.lokalniIndex = this.data.trenutniIndexPorizvoda;
  //dodjeljivanje globalne vrijednosti iz servisa, lokalnoj varijabli (lokalniIndex)
}
```

Slika 13. Upotreba servisa u jednoj od komponenata

Izvor: Autor (28.5.2023.)

'App.component.html' datoteka je koja je automatski kreirana od programskog okvira, a u njoj se nalaze i pozivaju sve HTML oznake drugih komponenata. To je ujedno i glavna HTML komponenta mrežne trgovine jer se u istu upisuje sav HTML sadržaj. Oznaka 'router-outlet' poziva sve ostale komponente koje nisu navedene unutar ili izvan Bootstrap predefiniranih klasa koje služe za automatsko stiliziranje aplikacije.

app.component.html M

zavrsniRad > src > app > app.component.html > ...

Go to component

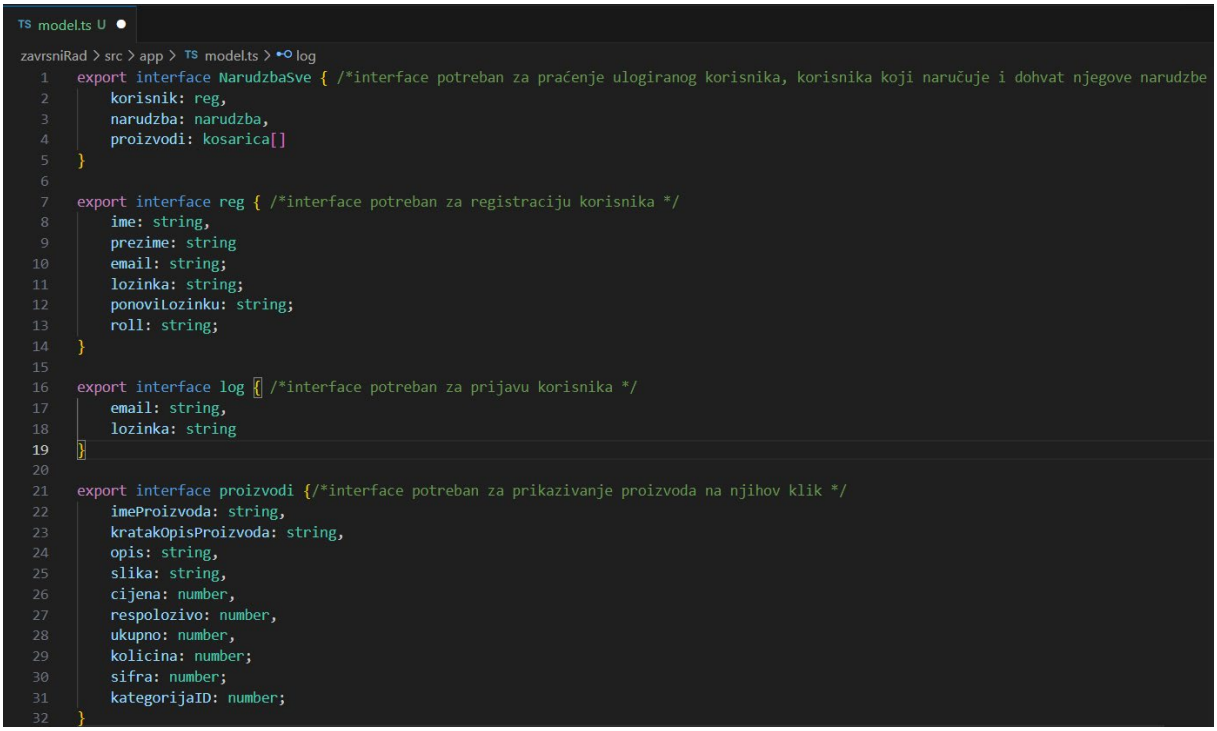
```
1  <app-navbar></app-navbar> <!--pozivanje 'navbar' componente-->
2
3  <div class="wrapper">
4    <div class="content-wrapper" style="background-color: white;">
5      <router-outlet></router-outlet> <!--povezivanje svih drugih komponenti-->
6    </div>
7  </div>
8
9  <app-footer></app-footer> <!--povezivanje 'footer' componente-->
```

Slika 14. Glavna HTML datoteka potrebna za prikazivanje sadržaja cijele aplikacije

Izvor: Autor (28.5.2023.)

Kao zadnju glavnu stavku koja čini mrežnu trgovinu, valja spomenuti datoteku (najčešće data.ts) sučelja (engl. *interface*).

Između ostalih, postoje i sučelja (engl. *interface*) za prijavu, registraciju, podatci o proizvodima, košarica s proizvodima i ostali. Koriste se u cijeloj aplikaciji i zadani su kao podatkovni tipovi.



Slika 15. Datoteka s prikazanim, samo nekih od sučelja potrebnih za rad aplikacije

Izvor: Autor (28.5.2023.)

6.2 Korisničko sučelje i iskustvo

Mrežna trgovina ima dva načina rada. Prvi i osnovni je korisnički, odnosno račun na kojem se proizvodi mogu dodati u košaricu i naručiti. Korisnik također ima mogućnost pregleda vlastitih izvršenih narudžbi te uz svaku narudžbu za istu preuzeti virtualni račun u PDF formatu. Navigacijska traka korisnika sadržava košaricu s indeksom koji označava trenutni broj proizvoda u košarici. Pored košarice, nalazi se tekst sa znakom pozdrava i imenom pod kojim je taj korisnički račun kreiran. Na kraju se nalazi gumb za odjavu koji korisniku omogućuje istu funkciju raspoloživu u bilo kojem trenutku.

Slika 16. Navigacijska traka prijavljenog korisnika

Izvor: Autor (29.5.2023.)

Administrator je osoba koja ima privilegije i odgovornosti za upravljanje i stvaranje promjena unutar mrežne aplikacije. Administrator ima više ovlasti, stoga je za njega prilagođen drugačiji prikaz sukladno njegovim mogućnostima. Administrator nema mogućnost dodavanja proizvoda u košaricu, a samim time niti obavljanja narudžbi. Navigacijska traka administratora umjesto korisničke košarice sadrži postavke u kojima ima daljnju manipulaciju proizvodima, poruke korisnika, pregled svih narudžbi, dok na klik 'Pozdrav' + ime administratora dobiva mogućnost manipuliranja korisnicima.

Slika 17. Navigacijska traka administratora

Izvor: Autor (29.5.2023.)

Početna stranica je po prikazu ista za korisnika ili administratora, no mijenja se korištenje komponenata sukladno ovlastima. Administrator posjeduje sve korisničke komponente, no nije ih u mogućnosti koristiti zbog svoje administrativne uloge. Glavna komponenta koju koristi je 'administration' i podkomponenta 'edit-product' koje pružaju manipulaciju

proizvodima. Ako administrator pokuša dodati proizvod u košaricu ili na bilo koji drugi način izvršiti korisničku radnju, primit će poruku o nemogućnosti izvršavanja iste.

Početna stranica obilježena je novim proizvodima, a proizvode se direktno mogu dodati u košaricu iz glavnog prikaza ili s prikaza o više detalja proizvoda. S lijeve strane je smješten izbornik s popisom svih kategorija i za filtriranje proizvoda prema istima. Na dnu stranice smješteno je podnožje, u kojoj su smještene informacije o lokaciji, kontakt, uvjeti i privatnosti, poveznica na stranicu koja ispisuje detaljnije informacije o tvrtki mrežne trgovine, poveznice na društvene mreže i gumb koji omogućuje slanje upita u obliku forme.

Nakon što korisnik proizvode doda u košaricu, unutar iste ima pregled, povećanje količine i mogućnost brisanja željenih proizvoda. Nakon košarice, slijedi završni dio procesa kupnje u kojem korisnik upisuje osobne podatke, dok s desna ima ponovni pregled proizvoda koje namjerava naručiti. Plaćanje se isključivo obavlja putem pouzeća. Nakon uspješno obavljene kupnje, prijavljeni korisnik dobiva obavijest o istome i, ako želi, može pregledati povijest svojih narudžbi klikom na polje u navigacijskoj traci pod 'Pozdrav' + korisničko ime.

Slika 18. Prikaz povijesti jedne od narudžba korisnika

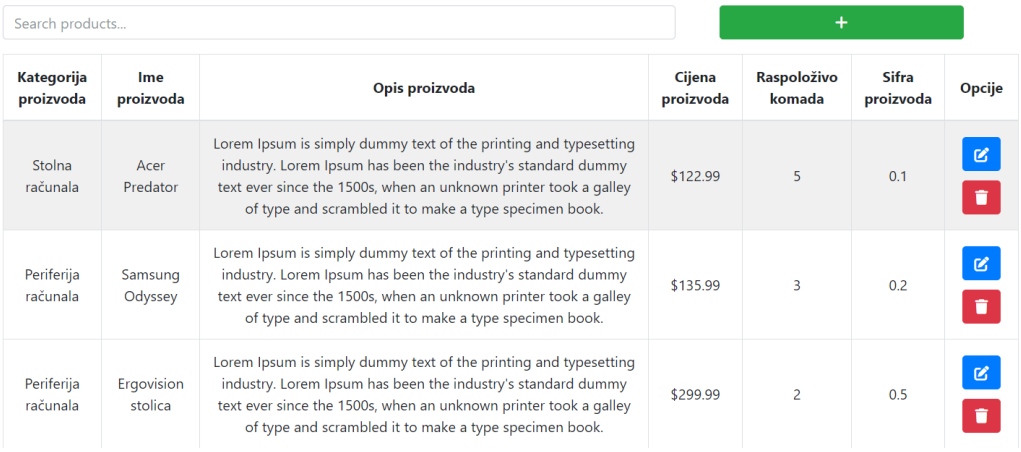
Izvor: Autor (30.5.2023.)

Ako prijavljeni korisnik želi pristupiti povijesti vlastitih narudžbi, a na istom korisničkom računu nije kreirana niti jedna, dobiva poseban prikaz koji ga obavještava o tome. Ista stvar provedena je i za kategorije koje ne posjeduju u sebi niti jedan proizvod.

Slika 19. Prikaz za korisnika koji ne posjeduje povijest narudžbi

Izvor: Autor (30.5.2023.)

Administrator pritiskom na simbol postavka dobiva tablicu s prikazom svih postojećih proizvoda. Pored svakog proizvoda postoji gumb za uređivanje i brisanje proizvoda, a iznad same tablice postoji pretraživač koji radi prema nazivu, kategoriji ili opisu proizvoda. Administrator također ima mogućnost dodavanja novih proizvoda. Nakon potvrde o mijenjanju ili dodavanju novog proizvoda, promjene će biti vidljive odmah unutar administrativne tablice i na samim stranicama proizvoda.



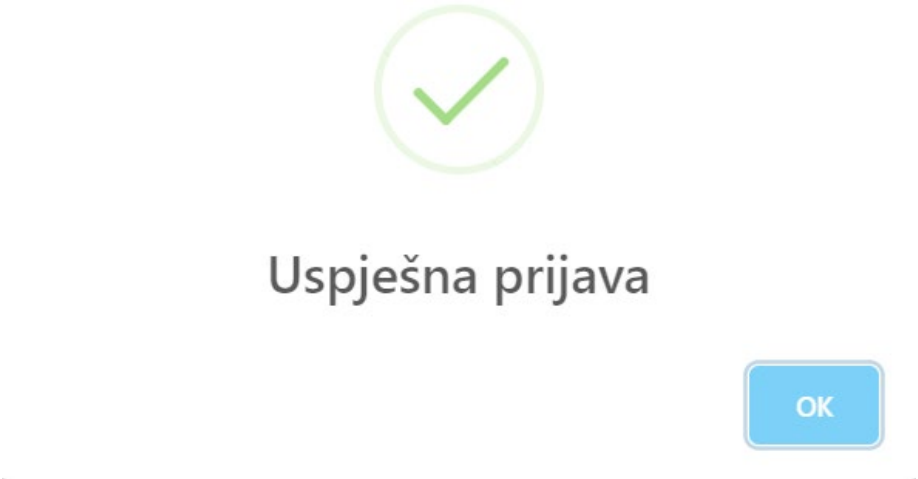
Slika 20. Manipulacija proizvoda u administrativnom načinu rada

Izvor: Autor (30.5.2023.)

Pored manipulacije proizvoda, administrator ima mogućnost pregleda kupaca. Prikaz se obavlja preko tablice u kojoj stoje osnovni podaci kupca kao što su ime i prezime, adresa

elektroničke pošte i lozinka korisničkog računa. Osim pregleda svojih registriranih korisnika, administrator ima mogućnost brisanja pojedinog korisničkog računa, čija je promjena vidljiva istog trenutka nakon brisanja.

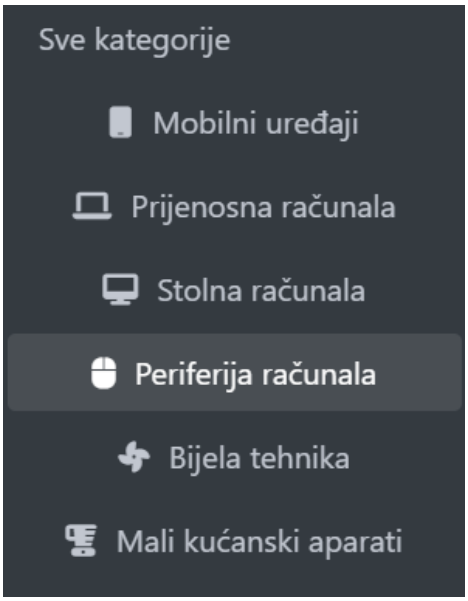
Razina korisničkog iskustva, uz jednostavan i pregledan dizajn, povećala se uz pomoć SweetAlert alata, već kreiranih animacija koje obavještavaju o uspješnosti/neuspješnosti radnje. Točnije, alat se koristi kako bi obavijestio korisnika o uspješnoj prijavi, registraciji, neuspješnom ili uspješnom dodavanju proizvoda u košaricu i slično.



Slika 21. Primjer SweetAlert prikaza o uspješnosti prijave

Izvor: Autor (30.5.2023.)

Također, uz SweetAlert, tijekom cijele aplikacije koristi se biblioteka ikona Font Awesome. Ikone se postavljaju na gumbe, dok postoje i u izborniku kategorija kod kojih prikazuju osnovni proizvod za navedenu kategoriju. Ikone se također koriste u podnožju stranice, točnije kao ikone poznatih društvenih mreža, s obzirom na to da se tako dobiva čišći izgled i dizajn u usporedbi s onim koji bi društvene mreže bile opisivane riječima.



Slika 22. Upotreba Font Awesome ikona u izborniku

Izvor: Autor (30.5.2023.)

6.3 Izrada i povezivanje baze podataka

S obzirom na to da je baza podataka MongoDB Atlas *cloud* verzija baze podataka, povezivanje i izrada jednostavni su procesi. Pozadinski dio koda odvija se u odvojenoj mapi (ZavrsniRad_Backend) od one mape koja prikazuje prednji dio koda. Pozadinska mapa sadržava sve API zahtjeve, s datotekama namijenjenim za spajanje na *cloud* tip baze podataka. Prije svega, potrebno je kreirati korisnički račun na službenoj stranici tvrtke baze podataka. Za povezivanje na bazu, potrebno je kreirati konfiguracijsku datoteku koja će sadržavati varijablu čiji će cilj biti spajanje na bazu uz pomoć specifične adrese dobivene od baze podataka.

database.config.ts

zavrsniRad_Backend > src > configs > database.config.ts > dbConnect

```
1  import { connect, ConnectOptions } from "mongoose";
2
3
4  export const dbConnect = () => { //ostvarivanje konekcije sa mongoDB baze koristeci connection url i opcije
5      connect( , {
6          useNewUrlParser: true,
7          useUnifiedTopology: true
8      } as ConnectOptions).then(
9          () => console.log("Connected to MongoDB"),
10         (error) => console.log(error)
11     )
12 }
```

Slika 23. Konfiguracijska datoteka za spajanje na bazu podataka

Izvor: Autor (31.5.2023.)

Nakon kreirane varijable za spajanje na bazu, kreirane su i odvojene mape za sve module i *routere* kako bi svaki model, odnosno *router*, bio odvojen i okružen isključivo vlastitim API zahtjevima, što ujedno rezultira i preglednijim kodom. Unutar svakog modela nalaze se sheme s vlastitom strukturom i svojstvima s obzirom na tip *interfacea*.

Uobičajena praksa je da HTTP zahtjevi ne pristupaju direktno bazi podataka. Umjesto toga, HTTP zahtjevi koriste se za komunikaciju između klijenta (npr. *web* preglednik, mobilna aplikacija) i poslužitelja. Podaci se šalju pomoću Express.js programskog okvira putem HTTP zahtjeva na specifičan URL, a zatim pohranjuju u bazu podataka. Razlog tomu je u više spektara. Prvi i najvažniji je sigurnost. Otvaranje izravnog pristupa bazi podataka putem HTTP zahtjeva može predstavljati veliki sigurnosni rizik. Ovakav način pristupa podacima omogućava bolju modularnost i fleksibilnost aplikacije.

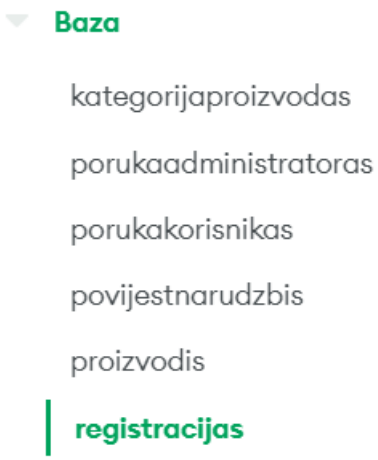
```
const app = express(); // Stvaranje instance aplikacije pomoću Express modula
const bodyParser = require('body-parser'); // Uvoz body-parser modula

app.use(bodyParser.json()); // Upotreba body-parsera za parsiranje ulaznih podataka u JSON formatu
app.use(cors({
    credentials: true,
    origin: ['http://localhost:4200'],
})
); // Upotreba Cors modula za omogućavanje Cross-Origin Resource Sharing (CORS) između poslužitelja i klijenta koji se nalazi na http://


/*kada request dođe do url-a (etc. '/api/registracija'), odgovarajuca rukovoditeljska ruta definiranna unutar
(etc. 'registracijaRouter') biti će izvršena da se nose sa tim zahtjevom */
app.use('/api/registracija', registracijaRouter);
app.use('/api/filtriranjeKategorija', kategorijaRouter);
app.use('/api/proizvodi', proizvodiRouter);
app.use('/api/povijestNarudzbi', povijestNarudzbiRouter);
```

Slika 24. Primjer samo dijela serverske datoteke s deklariranim poveznicama za spremanje podataka prije upisa u bazu podataka

Izvor: Autor (31.5.2023.)

6.4 Segmenti baze podatka

Segmenti baze podataka čine šest tablica: tablica kategorije proizvoda, poruke administratora kupcu, poruke korisnika administratoru, povijest svih narudžbi, popis aktivnih proizvoda i registracijska tablica, tj. tablica koja bilježi sve postojeće registrirane korisnike mrežne trgovine.



Slika 25. Segmenti baze podataka

Izvor: Autor (31.5.2023.)

Podacima se može manipulirati kompletno i direktno iz baze podataka na svim tablicama, dok putem aplikacije isključivo administrator ima mogućnost manipulacije proizvoda i korisnika.

6.5 Komunikacija prednjeg i pozadinskog koda

Komunikacija između prednjeg, korisničkog sučelja i pozadinskog dijela koji sprema, dohvaća i radi promjene u podacima vrlo je važna. Bez te se komunikacije podaci ne bi mogli uopće očitati. Komunikacija između navedenih dvaju tipova kodova ostvaruje se preko HTTP zahtjeva.

Servis je glavna stanica unutar prednjeg koda i iz njega kreću metode koje kod svojeg pozivanja šalju HTTP zahtjev ovisno o metodi slanja zahtjeva (POST, GET, PUT, DELETE). S obzirom na metodu, zahtjevi se mogu slati s ili bez funkcijskih argumenata. Argumenti se najčešće šalju ako se s istim podatkom treba izvršiti radnja, a u tom slučaju koriste se POST, PUT ili DELETE metode slanja. Metoda GET koristi se bez argumenata jer ima jednostavnu ulogu dohvata i vraćanja podataka. Argument koji je uvijek implementiran je poveznica. Na poveznici se rade promjene ili dohvaćaju podaci, s obzirom na to da podacima ne pristupamo izravno kroz bazu podataka.

Unutar mape pozadinskog koda nalaze se svi API zahtjevi vezani za određeni dio funkcionalnosti mrežne trgovine. API zahtjevi implementirani su pomoću Express.js programskog okvira glavne biblioteke Node.js. Svaka datoteka pokriva određenu adresu na kojoj se nalaze podaci zapisani u JSON formatu, a ako se adresa iz servisa poklopi s adresom na kojem neka datoteka posjeduje podatke, API zahtjev radi promjene deklarirane unutar 'asyncHandler' funkcije. Dohvaćeni podaci šalju se kao odgovor unatrag prema servisu, odnosno prednjem dijelu koda gdje se promjene apliciraju i prikazuju putem korisničkog sučelja.

```
//api za dodavanje novih proizvoda (postaviDodanNoviProizvod) u data.service.ts
router.post("/", asyncHandler(
    async (req, res) => {
        const proizvod = req.body;
        proizvod.kolicina = 1;
        const newProizvod = await proizvodiModel.create(proizvod);

        res.status(201).json(newProizvod);
    }
));
```

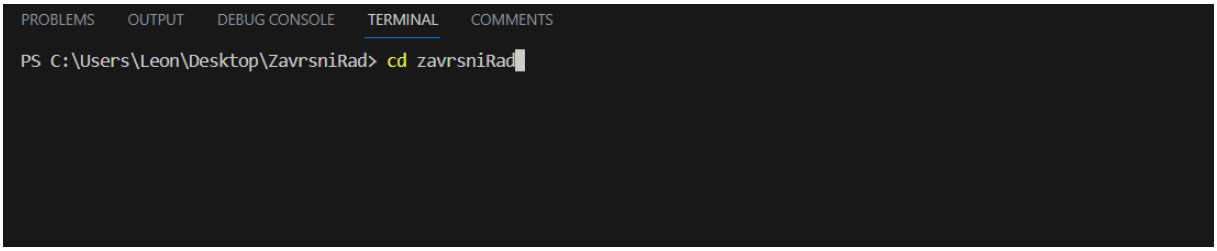
Slika 26. API POST zahtjev koji služi za dodavanje novih proizvoda u mrežnu trgovinu

Izvor: Autor (31.5.2023.)

6.6 Pokretanje aplikacije

S obzirom na svoje lokalno izvršavanje, aplikacija se pokreće i izvršava putem sučelja naredbenog retka. Aplikacija je podijeljena unutar glavne mape na mapu za izvršavanje (engl. *front end*) i *back end* tehnologiju. Za izvršavanje naredbi tijekom cijelog projekta koriste se ugrađeni naredbeni redci unutar uređivača koda Visual Studio Code.

Prilikom prvog pokretanja potrebno je pristupiti mapi u kojoj se nalazi kod za izvršavanje prednjeg dijela koda, a način na koji se isto implementira je 'cd' komanda i ime mape kojoj se želi pristupiti.



Slika 27. Ulazak u mapu putem naredbenog retka

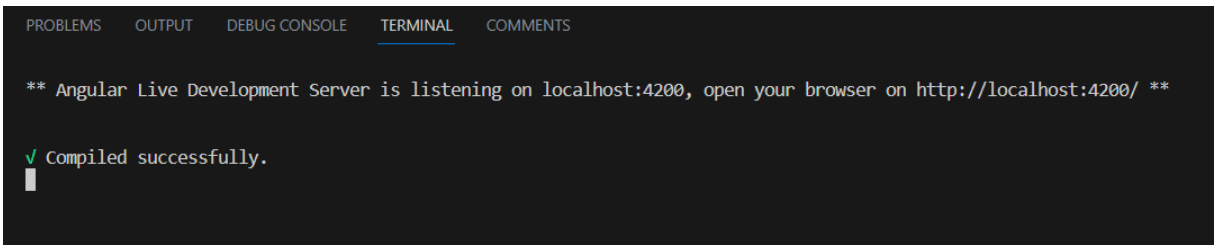
Izvor: Autor (1.6.2023.)

Prilikom pristupa datoteci u kojoj se nalazi sav kod potreban za izvršavanje prednjeg dijela koda, potrebno je upisati specifičnu komandu 'ng serve' koja pokreće prikazivanje prednjeg dijela koda. Navedena komanda specifična je za Angular programski okvir.

Slika 28. Komanda za pokretanje prednjeg dijela koda

Izvor: Autor (1.6.2023.)

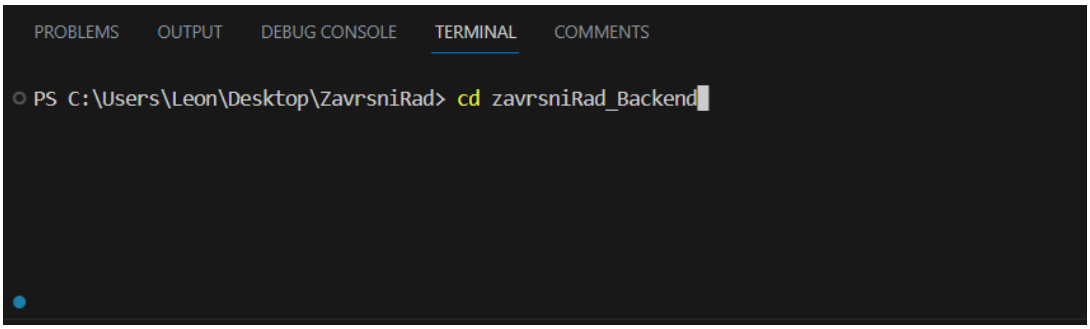
Komanda pokreće lokalni poslužitelj i kompilira cijelu Angular aplikaciju, istovremeno omogućavajući pregledavanje u mrežnom pregledniku. Ako se u kodu ne nalaze nikakve greške koje bi uzrokovale neispravno prikazivanje koda, na kraju dolazi poruka o uspješnom pokretanju i posluživanju prednjeg dijela aplikacije.



Slika 29. Uspješno pokrenuta mrežna aplikacija putem naredbenog retka

Izvor: Autor (1.6.2023.)

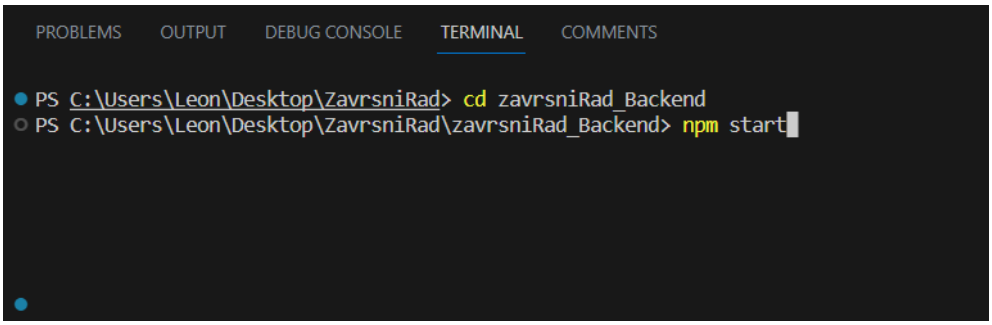
Pokretanje se također mora izvršiti i za pozadinski dio koda jer bez njega prikaz trgovine ne bi bio ispravan zbog toga što se na istome nalaze svi podaci potrebni za manipuliranje izravno s bazom podataka. Kao i kod prednjeg dijela koda, prvobitno je potrebno ući u mapu pozadinskog koda.



Slika 30. Ulazak u pozadinsku mapu koda

Izvor: Autor (1.6.2023.)

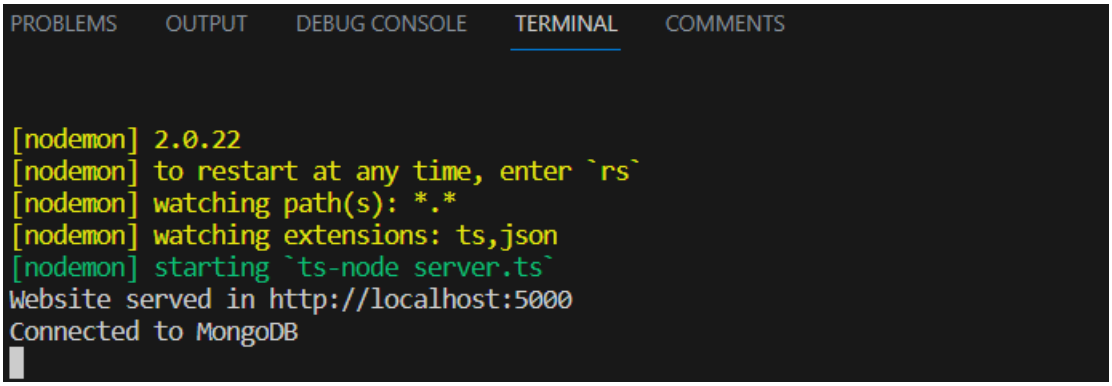
Nakon pristupa pozadinskoj mapi koda, potrebno je upisati jedinstvenu komandu 'npm start' za pokretanje servera Node.js tehnologije.



Slika 31. Komanda za pokretanje Node.js servera

Izvor: Autor (1.6.2023.)

U slučaju uspješnosti, na kraju se dobije obavijest o uspješnom pokretanju Node.js servera, a nakon čega se mrežnoj aplikaciji može pristupiti sa svim odgovarajućim sadržajem.



Slika 32. Obavijest o uspješno pokrenutom Node.js serveru

Izvor: Autor (1.6.2023.)

7. ZAKLJUČAK

Izradom mrežne trgovine nastojala se stvoriti funkcionalna, moderna i jednostavna mrežna aplikacija. Mrežna trgovina bazirana na Angular programskom okviru, pisana u TypeScript programskom jeziku, korištena Express i Node.js tehnologijama i MongoDB Atlas bazom podataka rezultirala je funkcionalnim i ispravnim radom. Mrežna trgovina rađena je na stranu korisnika, odnosno da svi, pa čak i manje tehnički educirani pojedinci svoju kupnju obave bez pomoći druge osobe.

Mrežna aplikacija korisnicima omogućuje mnoštvo sadržaja i opcija, dok odgovornoj osobi, tj. administratoru omogućuje punu kontrolu nad proizvodima i postojećim korisnicima unutar trgovine.

Učenje programskog okvira i pratećih tehnologija, kao i rad na izradi mrežne trgovine, kompleksan je zadatak - potrebno je učiti, proučavati i ispravljati greške kako bi se dobio krajnji rezultat.

Poboljšanja su moguća, među kojima valja spomenuti spajanje na verziju baze podataka koja nije *cloud*, a samim time dolazi i ručno konfiguriranje i održavanje baze podataka, rad na sigurnosti iste, evidencija, redovito stvaranje sigurnosne kopije itd.

Od korisničkih poboljšanja, mogućnost postoji u kartičnom plaćanju koje bi uveliko olakšalo komunikaciju i kupnju između korisnika i vlasnika mrežne trgovine.

8. POPIS LITERATURE

[1] codeinwp. Angular vs React vs Vue: Which Framework to Choose [Online]. 2023.

Dostupno na: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#gref (20.5.2023.)

[2] Miva. The History Of Ecommerce: How Did It All Begin? [Online]. 2023. Dostupno na: https://blog.miva.com/the-history-of-ecommerce-how-did-it-all-begin (21.5.2023.)

[3] Forbes. The Future Of E-Commerce: Trends To Watch In 2023 [Online]. 2023.

Dostupno na: https://www.forbes.com/sites/forbesmarketplace/2023/03/21/the-future-of-e-commerce-trends-to-watch-in-2023/?sh=16c2e0fa631e (22.5.2023.)

[4] Angular. Angular [Online]. 2023. Dostupno na : https://angular.io

[5] mDevelopers. What is the Angular framework? [Online]. 2022.

Dostupno na : https://mdevelopers.com/blog/what-is-the-angular-framework (23.5.2023.)

[6] Stackify. Angular vs AngularJS: Differences Between Angular and AngularJS [Online]. 2023. Dostupno na: https://stackify.com/angular-vs-angularjs-differences-between-angular-and-angularjs (23.5.2023.)

[7] Angular. Angular components overview [Online]. 2023.

Dostupno na: https://angular.io/guide/component-overview (23.5.2023.)

[8] Angular. Introduction to modules [Online]. 2023.

Dostupno na: https://angular.io/guide/architecture-modules (24.5.2023.)

[9] Rangle.io. Angular Traning [Online]. 2023.

Dostupno na: https://angular-training-guide.rangle.io/modules/introduction (24.5.2023.)

[10] Wikipedia. TypeScript [Online]. 2023.

Dostupno na : https://en.wikipedia.org/wiki/TypeScript (24.5.2023.)

[11] TechTarget. MongoDB [Online]. 2023. Dostupno na:

https://www.techtarget.com/searchdatamanagement/definition/MongoDB (26.5.2023.)

[12] Altexsoft. What is API: Definition, Types, Specifications, Documentation [Online]. 2022.

Dostupno na: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/ (26.5.2023.)

9. PRILOZI

Slika 1. Angular komponenta s njezinim datotekama 13
Slika 2. Najpoznatiji mrežni programski okviri 16
Slika 3. HTML element (gumb) i njegova struktura 17
Slika 4. Dodjeljivanje CSS vrijednosti HTML div elementu 18
Slika 5. Primjer responzivnog dizajna 19
Slika 6. Različito djelovanje Bootstrap klasa 20
Slika 7. Dinamično deklariranje varijable u JavaScriptu, odnosno statičko u TypeScriptu 21
Slika 8. Prikaz obaveza *back end* programera 23
Slika 9. Način na koji API zahtjevi rade 24
Slika 10. Sve korištene komponente potrebne za rad mrežne trgovine 27
Slika 11. Način implementacije datoteka za rutiranje (app-routing.module.ts) 28
Slika 12. Glavni modul mrežne trgovine (app.module.ts) 28
Slika 13. Upotreba servisa u jednoj od komponenata 29
Slika 14. Glavna HTML datoteka potrebna za prikazivanje sadržaja cijele aplikacije 29
Slika 15. Datoteka s prikazanim, samo nekih od sučelja potrebnih za rad aplikacije 30
Slika 16. Navigacijska traka prijavljenog korisnika 31
Slika 17. Navigacijska traka administratora 31
Slika 18. Prikaz povijesti jedne od narudžba korisnika 32
Slika 19. Prikaz za korisnika koji ne posjeduje povijest narudžbi 33
Slika 20. Manipulacija proizvoda u administrativnom načinu rada 33
Slika 21. Primjer SweetAlert prikaza o uspješnosti prijave 34
Slika 22. Upotreba Font Awesome ikona u izborniku 35
Slika 23. Konfiguracijska datoteka za spajanje na bazu podataka 36
Slika 24. Primjer samo dijela serverske datoteke s deklariranim poveznicama za spremanje podataka prije upisa u bazu podataka 37
Slika 25. Segmenti baze podataka 37
Slika 26. API POST zahtjev koji služi za dodavanje novih proizvoda u mrežnu trgovinu 38
Slika 27. Ulazak u mapu putem naredbenog retka 39
Slika 28. Komanda za pokretanje prednjeg dijela koda 39
Slika 29. Uspješno pokrenuta mrežna aplikacija putem naredbenog retka 40

Slika 30. Ulazak u pozadinsku mapu koda 40
Slika 31. Komanda za pokretanje Node.js servera 41
Slika 32. Obavijest o uspješno pokrenutom Node.js serveru 41