IZRADA WEB TRGOVINE SPECIJALIZIRANE ZA PRODAJU KNJIGA

Lučin, Ivan

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

IVAN LUČIN

ZAVRŠNI RAD

IZRADA WEB TRGOVINE SPECIJALIZIRANE ZA PRODAJU KNJIGA

Split, ožujak 2023.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Informacijski sustavi

Z A V R Š N I R A D

Kandidat: Ivan Lučin

Naslov rada: Izrada web trgovine specijalizirane za prodaju knjiga

Mentor: mr. sc. Karmen Klarin, viši predavač

Split, ožujak 2023.

SADRŽAJ

Sažetak

Razvoj interneta i računalnih tehnologija promijenili su način na koji komuniciramo, utjecali na kreiranje novih oblika poslovanja preselivši i trgovine na zaslone računala i pametnih telefona. Web aplikacija GreenBook specijalizirana za prodaju knjiga udovoljava zahtjevima online kupaca, a prodaju knjiga čini jednostavnijom, pristupačnijom i konkurentnijom. Aplikacija omogućuje registriranom korisniku prijavljivanje i uređivanje vlastitog korisničkog računa, pretraživanje i naručivanje svih dostupnih naslova, pregledavanje svih svojih narudžbi kao i najprodavanijih naslova te sigurno plaćanje.

Poslužiteljska strana (engl. *backend*) napravljena je u platformi Node.js koja koristi programski jezik Javascript, dok su klijentska strana i korisničko sučelje (engl. *frontend*) napravljeni pomoću Javascript biblioteke (engl. *library*) React, HTML-a (HyperText Markup Language), osnovnog CSS-a (Cascading Style Sheets), JavaScripta te pomoću CSS okvira Bootstrap koji omogućuje jednostavno implementiranje responzivnog dizajna kao i uređivanje mrežne (engl. *web*) stranice. Za pohranu podataka korištena je NoSQL (Not Only SQL) baza podataka MongoDB.

Ključne riječi: Bootrstap, MongoDB, Node.js, React, web aplikacija

Summary

Development of a web shop specialized for selling of books

Development of the Internet and computer technology changed the way we communicate, infuenced the creation of new ways of business by moving the stores on to computer and smartphone screens. Web application GreenBook specialized for selling books meets the needs of online customers and makes selling books easier, more accessible and more competition worthy. Application enables the registered user to log in and edit their profile, to search and place an order of all of the titles available, to view all of their orders as well as the bestsellers and, also, enables secure payment methods.

The server side is made in the platform called Node.js in which the programming language Javascript is used, while the client side and frontend are made using a Javascript library called React, HTML (HyperText Markup Language), basic CSS (Cascading Style Sheets), JavaScript and using the Bootstrap CSS framework that enables simple implementation of responsive design as well as editing of a web page. MongoDB which is a NoSQL dana base, is used for data storage.

Keywords: Bootrstap, MongoDB, Node.js, React, web application

1. Uvod

Živimo u vremenu izrazito brzog razvoja internetske kupnje u kojem broj online kupaca svakodnevno raste. Internetska (engl. *online*) trgovina postala je jedan od primarnih kanala prodaje i kupnje, a u budućnosti je, bez sumnje, jedna od najpropulzivnijih gospodarskih djelatnosti. Brojne su prednosti pokretanja internetske trgovine vidljive na prvi pogled: ne zahtijeva velika ulaganja, ne trebate skup poslovni prostor na dobroj lokaciji, svojom prisutnošću na internetu otvarate veliko tržište za vaše proizvode, ostvarujete izravnu komunikaciju s kupcima i povratnu informaciju, bivate istaknutiji u konkurentnom okruženju. Prednosti ovakvog načina kupnje vidljive su i krajnjem korisniku jer: štedi vrijeme i novac (nešto niže cijene od onih u klasičnim trgovinama), praktičnija je i nudi dulje radno vrijeme.

Cilj je ovoga rada izrada web aplikacije koja će poslužitelju omogućiti pozicioniranje na tržištu bilo da proširuje svoje poslovanje ili pokreće samostalno poslovanje i prodaju knjiga putem interneta s jedne strane. S druge strane, krajnjem korisniku web aplikacije, odnosno kupcu, nudi jednostavnu, dostupnu i praktičnu internetsku trgovinu s ponudom kao i u klasičnoj knjižari, brzo i glatko učitavanje stranica i pregled naslova u web shopu, brzo plaćanje te sigurnost korisnika i posjetitelja trgovine. Web aplikacija sastoji se od dvaju prikaza koji se razlikuju po ovlastima, korisničkog i administratorskog. Administrator ima više ovlasti od samoga korisnika, a neke su od njih: mogućnost pregleda svih kupaca, dodavanje novih naslova ili brisanje postojećih ukoliko aplikacija proširuje ili smanjuje svoj repertoar te pregled i izmjena svih dosadašnjih narudžbi.

Završni rad strukturiran je od pet poglavlja, od kojih se prvo odnosi na uvodni dio u kojem se opisuje razlog pisanja završnoga rada na temu internetske trgovine usmjerene prema tržištu krajnjih korisnika, cilj i struktura rada. Poglavlje „Korištene tehnologije“ definira i opisuje tehnologije korištene u izradi ovoga rada, njihove sintakse i uporabe. Treće poglavlje „Izrada baze podataka“ opisuje detaljnu izradu baze podataka i odnose između entiteta. Četvrto poglavlje „Izrada aplikacije“ prikazuje način pisanja kôda i izgled same internetske aplikacije. Peto je poglavlje zaključak o aplikaciji i cijelom projektu, a sadrži literaturu, popis slika i tablica konzultiranih u izradi ovoga završnog rada.

2. Korištene tehnologije

U ovom poglavlju definirani su osnovni elementi svake od korištenih tehnologija u izradi web aplikacije. JavaScript, tehnologija okosnica je ove web aplikacije na koju se nadograđuju i ostale korištene tehnologije: MongoDB, Express.js, React.js i Node.js (akronim MERN). Slijedi kratak uvod u svaku od korištenih tehnologija.

2.1. JavaScript

Zbog osobina koje ga opisuju, JavaScript programski jezik temeljna je tehnologija na kojoj se zasniva izrada ovoga završnog rada. Prvu inačicu JavaScript programskog jezika razvio je Brendan Eich 1995. godine, a danas je najkorišteniji jezik. Naime, korišten je za izradu 95 % svih internetskih stranica, odnosno aplikacija današnjice [3]. Zbog tako široke primjene stvoren je veliki broj web okvira i biblioteka kako bi se proširilo i olakšalo korištenje jezika. Najpoznatija JavaScript biblioteka React korištena je za izradu korisničkog sučelja te je znatno olakšala stvaranje fluidne i interaktivne web aplikacije. Pozadinsko sučelje aplikacije realizirano je pomoću Node.js platforme koja koristi JavaScript kao interpretacijski jezik za obradu zahtjeva/odgovora na strani poslužitelja. Za upravljanje bazama podataka koristi se NoSQL baza MongoDB, jedna od standardnih baza Node.js-a.

Uz Node.js i React korištene su tehnologije kao što su HTML, CSS i Bootstrap. Ove su se tehnologije koristile isključivo za izradu korisničkog sučelja. HTML tehnologija koristi se kao standardni način za pisanje teksta koji će biti prikazan na web aplikaciji. Dizajn aplikacije, njezin detaljniji i privlačniji izgled postignut je uporabom CSS-a i njegova okvira (engl. *framework*) Bootstrap.

2.2. Node.js

Node.js pokretačko je okruženje otvorenog kôda (engl. *open source*) koje se bazira na potpunosti, svestranosti, sigurnosti, skalabilnosti te mogućnosti jednostavnog održavanja i prijenosa [4].

Slika 1: *Logo node.js-a* (izvor: node.js.org)

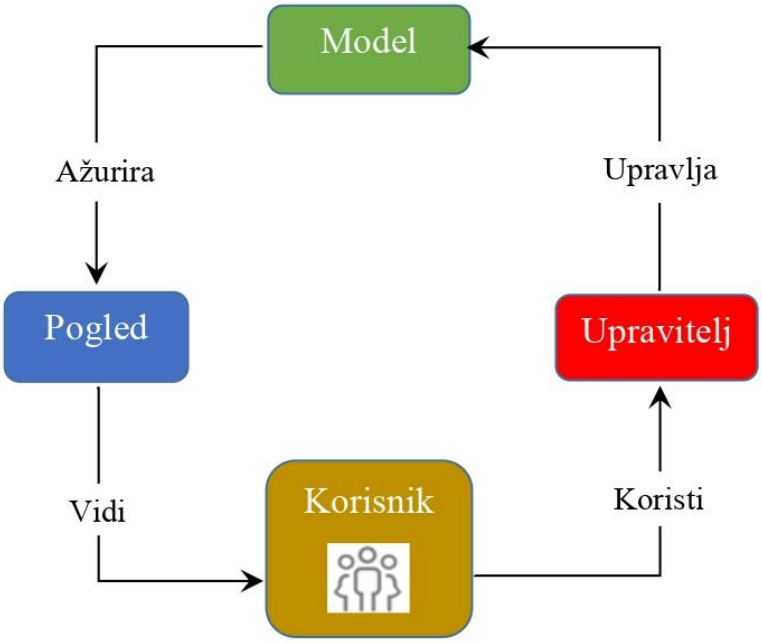
Node.js ima sve što je potrebno za pokretanje JavaScript kôda na strani poslužitelja. Izgrađen je na Google-ovom Chrome V8 JavaScript engine-u, tj. motoru ili pokretaču, odgovornom za interpretiranje i kompajliranje JavaScript kôda u izvorni kôd. V8 samo je ime pokretača koji je originalno namijenjen za korištenje u pretraživaču (engl. *browser*), a prešavši na stranu otvorenog kôda, postao je podloga za Node.js. Prednost Node.js-a kao programskog jezika je to što je neblokirajući (engl. *non-blocking*). Naime, svi događaji definirani u Node.js-u dodaju se u tzv. skup događaja (engl. *event pool*) te se registriraju i pokreću kada je potrebno bez blokiranja daljnjeg izvršavanja kôda. Mogućnost povratne funkcije (engl. *callbacks*), koju događaji mogu imati nakon njihova izvršavanja, pogodna je za izradu pravovremenih, brzih i lakih aplikacija.

Dodatna je prednost Node.js-a mogućnost proširenja funkcionalnosti dodavanjem paketa u okruženje Node.js putem Node Package Manager-a (*npm*), upravljača paketima za Node.js [5]. Paketi, odnosno dodatci okviru Node.js u obliku JavaScript biblioteka, mogu se uključiti u izvedbu bilo koje skripte.

Web okvir Express.js jedan je od najpopularnijih paketa koji uvelike olakšava pisanje kôda na strani poslužitelja uspješno otklanjajući probleme pisanja HTTP zahtjeva, uključujući rukovanja sa zaglavljima (engl. *header*) i podatcima iz tijela (engl. *body*) HTML dokumenata [6]. Express.js također nudi mogućnost organiziranja aplikacije u jednostavniju MVC (*Model View Controller*) arhitekturu (engl. *architecture*) koja razdvaja korisničko sučelje (engl. *User Interface*) aplikacije na tri glavna dijela:

- **Upravitelja** (engl. *Controller)* koji funkcionira kao posrednik između *Model* i *View* komponenti. Prima HTTP zahtjeve od korisnika, dohvaća podatke iz *Modela,* obrađuje ih i prosljeđuje na pripadajuću *View* komponentu kako bi korisnik mogao vidjeti rezultat svojih radnji.

- **Modela** - sučelja koje komunicira s bazom podataka preko njenog API-a (Aplikacijsko Programsko Sučelje). Na zahtjev upravitelja dohvaća podatke koji su pohranjeni na bazi ili stvara nove entitete koji će se pohranit na bazi.
- **Pogleda** (engl. *View)*, prikaza koji korisnik vidi. Najčešće je to kombinacija HTML-a, CSS-a i JavaScripta koji na promjenu modela se ponovo renderira i prikazuje podatke koje je korisnik zatražio [7].



Slika 2*: Način rada MVC aplikacije* (Izvor: vlastita izrada)

2.3. React

React jedna je od najpopularnijih biblioteka za razvoj web aplikacija. Besplatna je i otvorenog kôda te služi isključivo za izradu korisničkih sučelja SPA (engl. *single-page application*) [8]. SPA aplikacija koristi se za web aplikacije bogatog korisničkog sučelja jer omogućavaju podjelu korisničkog sučelja na manje komponente koje se nakon toga povezuje u veće cjeline. Prednosti su SPA aplikacije: jednostavnija implementacija responzivnog dizajna, nije potrebno ponovno učitavati svaku stranicu, pojedini dijelovi stranice koji se ne mijenjaju učitavaju se samo jednom, moguće je potpuno odvojiti poslužiteljsku stranu od klijentske i time olakšati proces izrade aplikacije. Najpoznatiji primjeri SPA aplikacije su Gmail, Facebook te GitHub.

Od prvoga izdanja 2013. godine biblioteka React postala je jednom od najpopularnijih JavaScript biblioteka zbog svoje jednostavnosti i deklarativnosti. BBC, Facebook, Imgur, Instagram, Netflix, PayPal, Reddit, Uber, WhatsApp samo su neka od web mjesta koja koriste React za izradu svojih web stranica [9]. Jedna od velikih prednosti Reacta je mogućnost sastavljanja korisničkog sučelja pomoću osnovnih elemenata koji se nazivaju komponente (eng. components), a predstavljaju male, kompaktne i ponovno iskoristive elemente. Komponenta je zapravo JavaScript klasa ili funkcija koja prima proizvoljan broj parametara koji se nazivaju svojstva (engl. *property*) i vraća React element koji opisuje kako će izgledati određeni dio UI-a.

Najvažniji i najkorisniji koncept koji React uvodi u izgradnji web aplikacija je koncept unutarnjeg stanja komponenata (engl. *state*) kako bi se izbjegla nemogućnost dinamičke promjene podataka koji se prikazuju na korisničkom sučelju bez uništavanja i ponovnog učitavanja tih komponenti. Stanje je promjenjivi (engl. *mutable*) skup podataka unutar objekta *state* neke komponente. Svaka komponenta ima svoj vlastiti skup stanja, a može se izmjenjivati samo unutar komponente jer su ti podatci za nju privatni. Uvođenjem tzv. *Hooksa* verzijom 16.8 Reacta znatno je olakšano upravljanje i praćenje stanja komponenti. Najpoznatiji i najkorišteniji *hook-ovi* su `useState, useEffect, useSelector`.

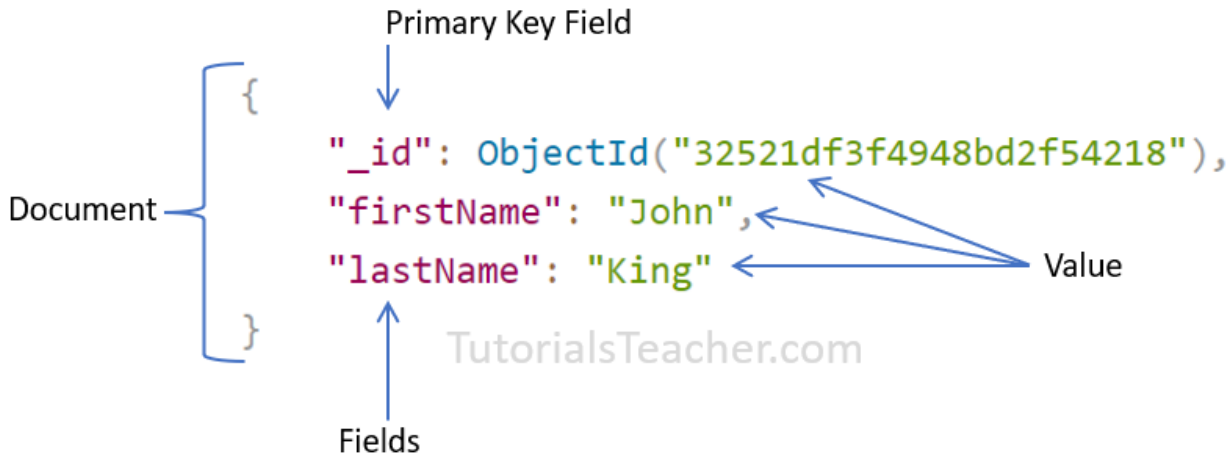
- **UseState** koristi se za definiranje dinamičkih vrijednosti i njegove funkcije. Vrijednosti se mogu mijenjati unutar komponenti pozivajući njihove funkcije i definirajući novu vrijednost unutar funkcije.
- **UseEffect** koristi se za dohvaćanje vrijednosti ili postavljanje DOM elementa nakon faze renderiranja. Inicijalno se poziva pri renderiranju komponente unutar koje se nalazi, ali može primiti niz vrijednosti zbog kojih bi se ponovo pozvao, npr. Promjena stanja varijable.
- **UseSelector** ima pristup globalnom stanju cijelog React projekta te može dohvaćati vrijednosti unutar komponenti.

2.4. MongoDB

MongoDB je nerelacijska ili NoSQL baza podataka kreirana 2007. godine (tvrtka MongoDB Inc). Riječ je o sustavu za pohranjivanje dokumenata otvorenog kôda koji osigurava brzinu i skalabilnost baze podataka zbog čega je MongoDB i stekao svoju popularnost. Visoka

dostupnost, vodoravno skaliranje te rasprostranjenost omogućavaju njeno lako korištenje. Besplatna verzija MongoDB baze podataka dovoljna je za manje ili srednje velike aplikacije i može se stvoriti na službenoj poveznici: https://www.mongodb.com/ . [10]

Glavni strukturni element MongoDB-a su tzv. dokumenti koje MongoDB koristi umjesto tablica ili redaka za pohranu podataka što rezultira bržom izvedbom i skalabilnošću jer jedna *read* operacija može dohvatiti cijeli dokument. Za pohranjivanje dokumenata MongoDB koristi se podatcima pripremljenim u BSON formatu, „Binary JSON“, formatu baziranom na JSON formatu što daje fleksibilnost te je moguće koristiti isti format podataka od klijenta do poslužitelja i na kraju do baze podataka [9]. MongoDB prikladan je za velike količine podataka pri čemu je orijentiran na brzinu i pohranu, pouzdanost i učinkovitost.



Slika 3*: MongoDB element* (Izvor: tutorialsteacher.com)

NoSQL baze podataka pohranjuju podatke na različite načine: stupčasto orijentirani, dokumentno orijentirani, temeljeni na grafovima ili organizirani kao par ključ/vrijednost. To znači da možete stvoriti dokumente bez potrebe da prethodno definirate njihovu strukturu, svaki dokument može imati svoju jedinstvenu strukturu i možete dodavati polja izravno pri radu. NoSQL baze podataka su vodoravno skalabilne, što znači da se obrađuje više informacija dodavanjem više poslužitelja u NoSQL bazu podataka. Osim kreiranja, čitanja, ažuriranja i brisanja podataka, MongoDB je baza podataka koja daje mogućnost: indeksiranja, agregacije, posebne kolekcije i datotečni sustav pri svemu ne utječući na brzinu rada.

2.5. Bootstrap

Bootstrap je skup alata kreiranih u CSS-u i JavaScript-u za razvijanje vidljivog dijela web aplikacija i web stranica, a dostupan je svima jer je besplatan i otvorenog kôda [12]. Izvorno su ga kreirali dizajneri i programeri Twittera nazvavši ga Twitter Blueprint. U početku je služio kao interni vodič za stiliziranje aplikacije i web stranice, ali je već nakon godinu dana u kolovozu 2011. godine. predstavljen široj publici.

Pruža skup komponenata za izradu kvalitetnog grafičkog sučelja te je bitan za prilagodljive web stranice koje se trebaju ispravno prikazivati na svim uređajima uključujući mobilni, tablet, prijenosno računalo i stolno računalo [13]. No potrebno je napomenuti da nije svaka komponenta u Bootstrapu responzivna. Bootstrap 5.2 posljednja je verzija Bootstrapa čiji je logotip moderniji, ali i dalje prepoznatljiv.

3. Izrada baze podataka

Baza podataka u ovom završnom radu napravljena je od četiriju tablica od kojih svaka ima određen broj atributa koji je opisuju.

U poglavlju „Entiteti i kardinalnost“ prikazane su sve tablice s njihovim atributima, ključevima te njihova kardinalnost. Polje s kardinalnošću prima dva broja. Prvi broj određuje minimalni broj različitih vrijednosti koje atribut daje za opis jednog elementa entiteta, dok drugi broj označava maksimalni broj različitih vrijednosti koje jedan atribut može dati za opis jednog elementa entiteta".

3.1. Entiteti i kardinalnost

Tablica 1: *Entitet "User" (Izvor: vlastita izrada)*

User				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	ObjectId	30	DA	(1,1)
Name	String	50		(1,1)
First_name	String	50		(1,1)
Last_name	String	50		(1,1)

Email	Email	50		(1,1)
Phone Number	Long	11		(1,1)
Password	String	20		(1,1)
isAdmin	Boolean	4		(1,1)
createdAt	Date			(0,1)
updatedAt	Date			(0,1)

Entitet `User` predstavlja sve standardne atribute koji mogu opisivati jednog korisnika uz njegov `ID`. Sve ove informacije korisnik će morati unijeti tijekom registracije i imat će mogućnost izmjene istih jednom kada napravi račun, osim `isAdmin` vrijednost koja je postavljena samo za račun administratora. Sva polja u tablici su obvezna jer će se koristiti u slučaju da korisnik odluči naručiti neki od proizvoda. `CreatedAt` je datum koji se pohrani u standardnom vremenskom formatu kada se korisnički račun kreirao, a `updatedAt` je datum i vrijeme kada se račun ažurirao.

Tablica 2: *Entitet "Product" (Izvor: vlastita izrada)*

Product				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	ObjectId	30	DA	(1,1)
Name	String	50		(1,1)
Author	String	50		(1,1)
Image	String	200		(1,1)
ImagePublicId	String	50		(1,1)
Description	String	1000		(1,1)
Category	Category			(1,n)
Rating	int			(0,1)
NumReviews	Int			(0,1)
Price	Int			(1,1)
CountInStock	Int			(0,1)
Reviews	Array			(1,n)
CopiesSold	Int			(0,1)
createdAt	Date			(0,1)

updatedAt	Date			(0,1)

Entitet `Product` predstavlja tablicu u kojoj je detaljno opisan određeni proizvod. U ovom entitetu atribut `CountInStock` predstavlja količinu proizvoda dostupnog za kupnju. Svakom novom narudžbom ova se vrijednost smanjuje, a administrator može ažurirati tu vrijednost u administracijskom sučelju. Također postoji i atribut `CopiesSold` koji definira koliko je primjeraka određenog proizvoda već prodano. `Category` predstavlja niz kategorija. Kategorija je strani ključ prema entitetu `Category` i svaki proizvod može imati više kategorija. `Image` i `ImagePublicId` predstavljaju atribute koji su bitni za pohranu slika na Cloudinary web servisu. Atribut `Image` je URL koji služi za dohvaćanje i prikazivanje slike na web trgovini, dok atribut `ImagePublicId` služi kao jedinstveni identifikator slike na Cloudinary web servisu pomoću kojeg možemo brisati i izmjenjivati već pohranjene slike. `Rating` je srednja vrijednost svih recenzija koje su korisnici ostavili za određeni proizvod. `NumReviews` predstavlja količinu recenzija, a `Reviews` predstavlja niz svih recenzija u kojima se nalazi korisnik koji je napravio recenziju te ocjena i opis. Proizvod također sadrži i vlastiti opis `Description` koji ukratko opisuje proizvod i cijenu `Price`.

Tablica 3: *Entitet "Category" (Izvor: vlastita izrada)*

Category				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	ObjectId	30	DA	(1,1)
Name	String	50		(1,1)
Description	String	200		(1,1)
createdAt	Date			(0,1)
updatedAt	Date			(0,1)

Tablica 4: *Entitet "Order" (Izvor: vlastita izrada)*

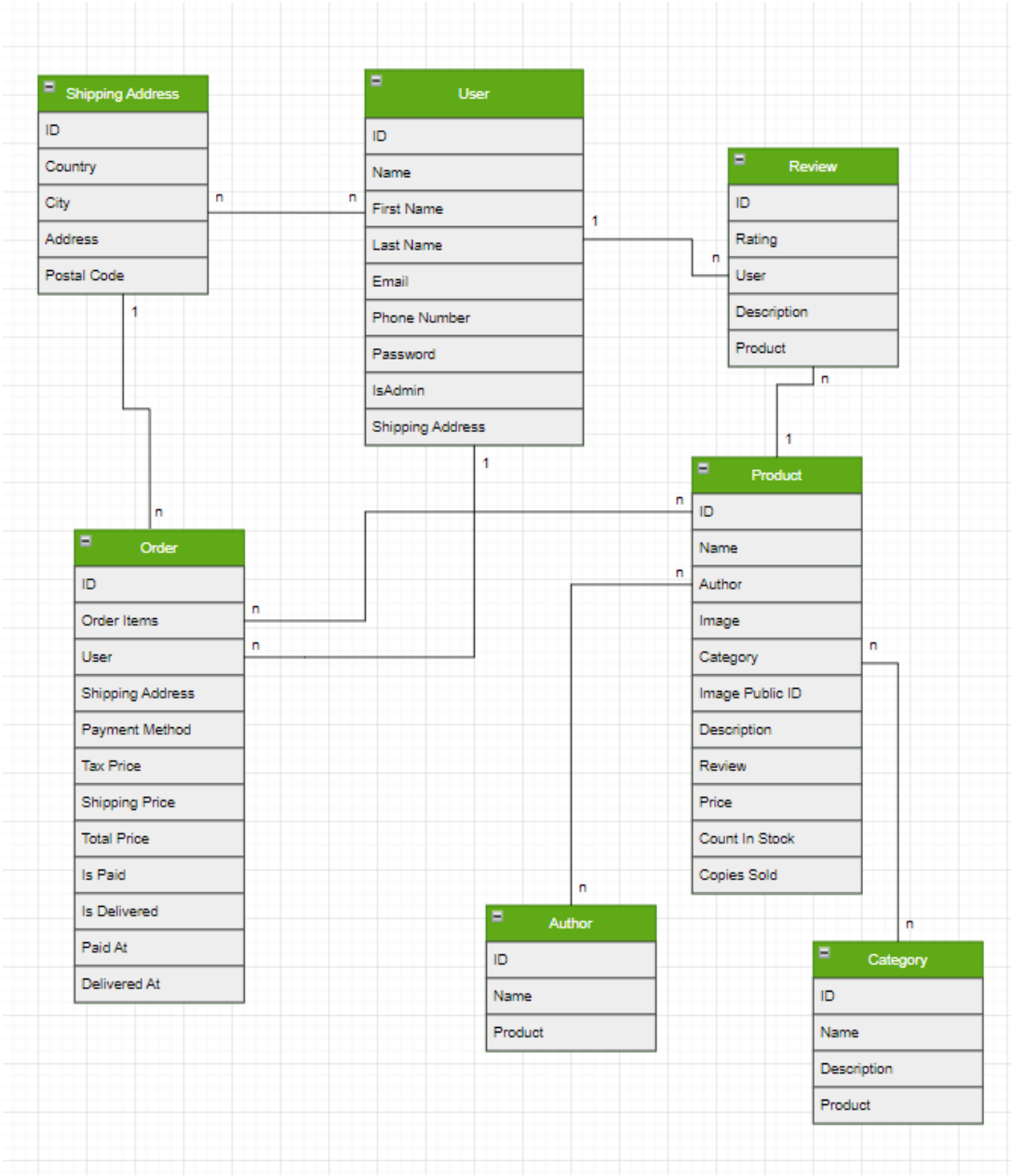
Order				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	ObjectId	30	DA	(1,1)
User	User	50		(1,1)
OrderItems	Array			(1,n)

ShippingAddress	Object			(1,1)
PaymentMethod	String	20		(1,1)
TaxPrice	Double			(1,1)
ShippingPrice	Int			(1,1)
TotalPrice	Double			(1,1)
IsPaid	Boolean			(1,1)
IsDelivered	Boolean			(1,1)
PaidAt	Date			(0,1)
DeliveredAt	Date			(0,1)
CreatedAt	Date			(0,1)
UpdatedAt	Date			(0,1)

Entitet `Order` sadrži dva strana ključa `User` i `Product`. `User` predstavlja korisnika koji je napravio narudžbu i pomoću toga stranog ključa moguće je dohvatiti sve podatke korisnika. `Product` se koristi u atributu `OrderItems` koji je niz svih proizvoda u narudžbi. Atribut `ShippingAdress` je objekt koji sadrži informaciju o mjestu dostave: država, grad, adresa i poštanski broj. `PaymentMethod` može biti Paypal, kreditna kartica ili plaćanje pouzećem. `TotalPrice` uključuje zbroj: cijena svih proizvoda u narudžbi, `TaxPrice` cijene PDV-a i `ShippingPrice` cijene poštarine. Kada korisnik plati narudžbu, `isPaid` postaje `True` i postavlja se vrijeme plaćanja `PaidAt`. Tek kada se narudžba plati, administrator ima mogućnost postavljanja `isDelivered` na `True` i tada se postavlja vrijeme kada je obavljena dostava `DeliveredAt`.

3.2. E-R Dijagram

Logički tok podataka web aplikacije prikazan je pomoću ER dijagrama (*Entity-Relationship*) dijagram koji je vidljiv na slici 4. ER dijagram ilustrira odnose između entiteta baze podataka.



Slika 4: *E-R dijagram logičkog toka podataka*(Izvor: vlastita izrada)

4. Izrada aplikacije

4.1. Izrada korisničkog sučelja

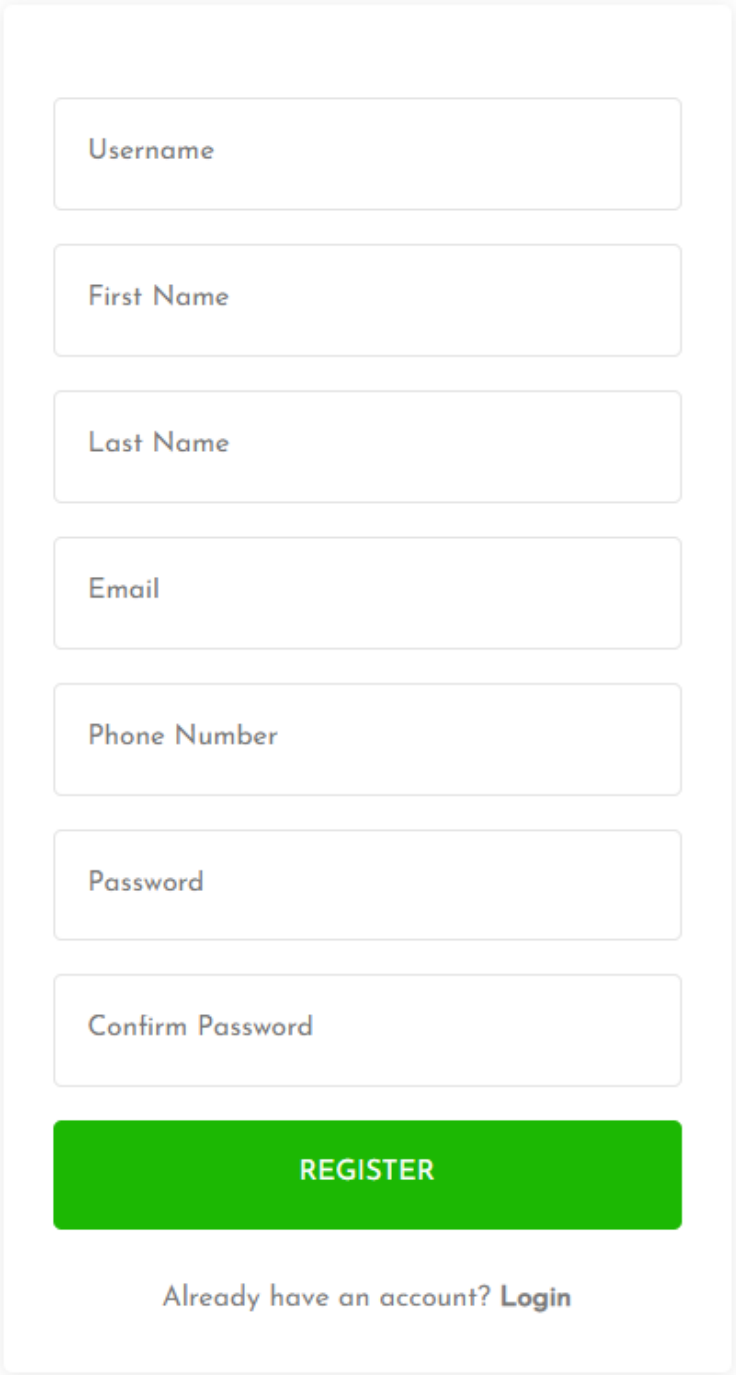
Korisničko sučelje dio je web trgovine koji korisnik vidi. Neregistrirani korisnik ovdje može pregledavati sve dostupne proizvode, filtrirati ih po kategoriji, cijeni, vremenu kada su dodani te dodavati proizvode u košaricu. Da bi korisnik pristupio ostalim funkcionalnostima aplikacije, potrebna je registracija.

4.1.1. Izrada korisničkog računa i prijava

Kada neprijavljeni korisnik prvi put pristupa stranici, navigacijska mu traka nudi pet poveznica: „Logotip“, „Tražilica“, „Registriranje“, „Prijava“ i „Košarica“. Klikom na Logotip korisniku se otvara početna stranica web trgovine s prikazom svih proizvoda. Tražilicom može pretraživati proizvode bez ikakvih restrikcija, dok se idućim dvjema poveznicama prijavljuje na svoj postojeći račun ili izrađuje novi. Zadnja poveznica je Košarica. Korisnik može puniti Košaricu proizvodima, ali mu je potreban korisnički račun kako bi ostvario kupnju. Izgled navigacije nalazi se na slici 5.

Slika 5: *Navigacijska traka neprijavljenog korisnika* (Izvor: vlastita izrada)

U poveznici Registracija korisnik mora ispuniti sve podatke unutar forme kako bi kreirao svoj korisnički račun. Nakon unosa, pritiskom na dugme „Register“ korisnički se podatci šalju poslužitelju pomoću HTTP *POST* zahtjeva na adresu: http://localhost:5000/api/users/ .



Slika 6: *Izgled registracijske forme* (Izvor: vlastita izrada)

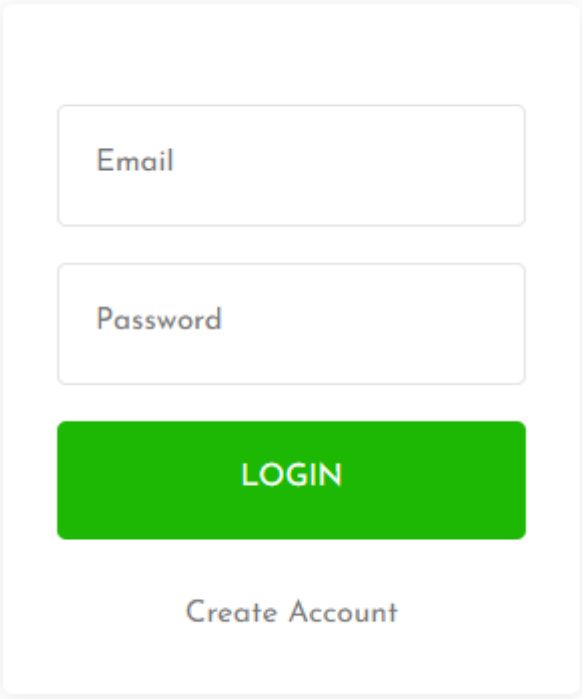
Ispis 1 prikazuje kako poslužitelj prima podatke o novom korisniku i provjerava postoji li već korisnik s tom e-mail adresom. Ako postoji, korisnik dobiva poruku upozorenja da već postoji korisnik s tom e-mail adresom pa novi korisnik mora promijeniti svoj unos. Ukoliko korisnik već ne postoji, stvara se novi korisnik s unesenim podatcima i sprema u bazu podataka. Generira se unikatni dokaz *token* pomoću kojeg se vrši autentifikacija korisnika pri korištenju

aplikacije. Podatci o korisniku uz *token* šalju se natrag na korisničko sučelje gdje se pohranjuju u globalno stanje aplikacije React te se elektroničkim putem šalje korisniku potvrda registracije. Korisnik se preusmjerava na početnu stranicu web trgovine gdje sada ima sva prava prijavljenog korisnika.

```
userRouter.post(
  "/",
  asyncHandler(async (req, res) => {
    const { name, firstName, lastName, email, number, password } = req.body;
    const userExists = await User.findOne({ email });
    if (userExists) {
      res.status(400);
      throw new Error("User already exists");
    }
    const user = await User.create({
      name,
      firstName,
      lastName,
      email,
      number,
      password,
    });
    if (user) {
      res.status(201).json({
        _id: user._id,
        name: user.name,
        email: user.email,
        isAdmin: user.isAdmin,
        token: generateToken(user._id),
      });
      transport.sendMail({
        to: user.email,
        from: "lukinivan326@gmail.com",
        subject: "Signup succeded",
        html: `<h2>Welcome ${user.name} to the shop!</h2>`,
      });
    } else {
      res.status(400);
      throw new Error("Invalid User Data");
    }
  })
);
```

Ispis 1*: Funkcija za registriranje na serveru*

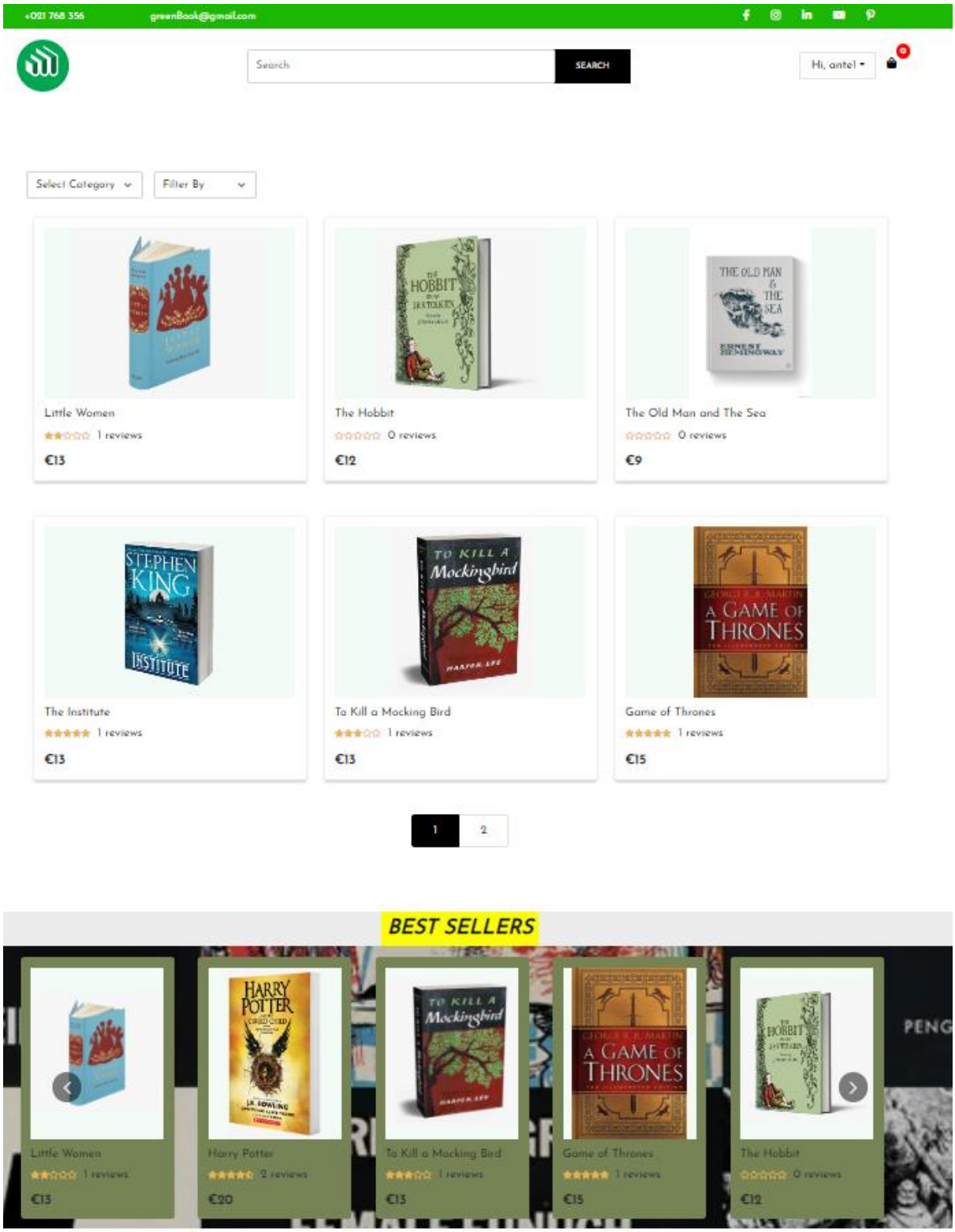
Korisnik se prijavljuje putem forme za prijavljivanje (engl. *login*) koja je vidljiva na slici 7. Prilikom prijave korisnika traži se unos odgovarajućeg e-maila i lozinke te nakon uspješne prijave poslužitelj provjerava postoji li korisnik s tom e-mail adresom u bazi podataka. Ukoliko se korisnik pronađe, lozinka poslana preko forme uspoređuje se s haširanom lozinkom spremljenom za tog korisnika unutar baze podataka. Ako se podudaraju, generira se novi *token* koji se s ostalim podatcima o korisniku šalje natrag na korisničko sučelje. Ondje se pohranjuje u globalno stanje i u lokalnu pohranu web preglednika. Potom se korisnik prebacuje s obične na privatnu rutu (engl. *private route*) unutar React usmjerivača . Svoj pristup korisnik ostvaruje autentikacijom, odnosno provjerom *tokena* iz lokalne pohrane. Nakon uspješne prijave, korisnik je preusmjeren na početnu stranicu gdje sada može vršiti narudžbe, ostavljati recenzije, pregledavati svoje prijašnje narudžbe, mijenjati svoje korisničke podatke itd.



Slika 7: *Forma za prijavljivanje* (Izvor: vlastita izrada)

4.1.2. Početna stranica

Na slici 8 vidljiv je izgled početne stranice za prijavljenog korisnika.



Slika 8: *Početni zaslon prijavljenog korisnika* (Izvor: vlastita izrada)

Korisnik može listati sve naslove knjiga ili ih filtrirati po kategoriji ili drugim opcijama poput cijene, datuma, recenzije itd. Također, postoji i tražilica u koju korisnik može upisati

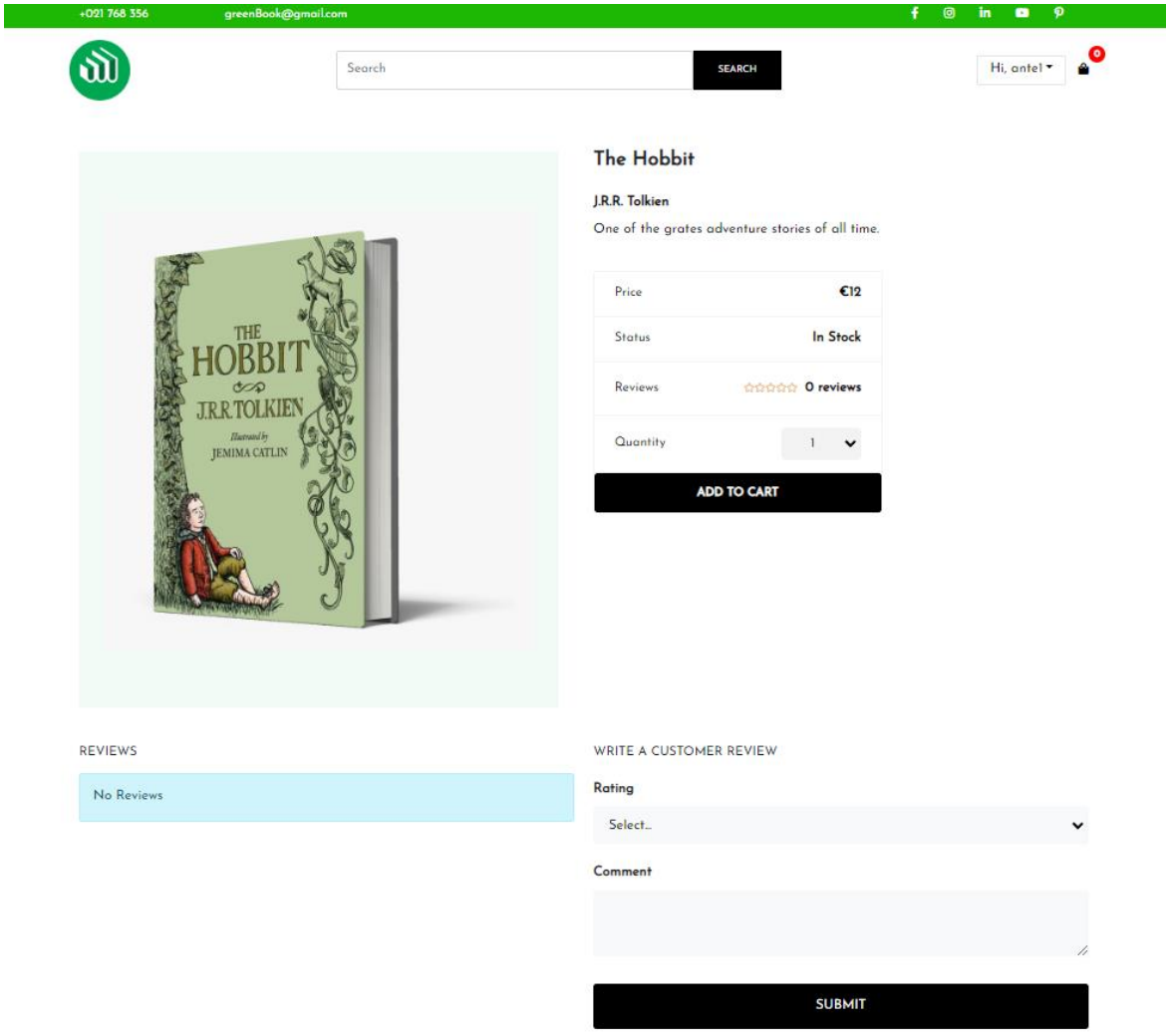
ključnu riječ ili pojam te će poslužitelj vratiti knjige čiji naziv sadrži taj pojam. Ispod svih knjiga nalazi se vrtuljak (engl. *carousel*) najprodavanijih naslova koji se zajedno sa svim knjigama filtriraju po kategoriji. Svaka knjiga ima svoju sliku, naslov, recenziju, broj ostavljenih recenzija i cijenu. Na ispisu 2 vidljiva je funkcija dohvaćanja knjiga. Funkcija definira koliko će knjiga biti prikazano na jednoj stranici te prima parametar stranica koji definira koliko će knjiga funkcija preskočiti prije nego vrati knjige te stranice. Odabir vraćenih knjiga ovisi i o filterima koje je korisnik aktivirao pomoću traži (engl. *find*) i sortiraj (engl. *sort*) funkcije. Ovisno o odabranoj kategoriji, filteru i upisanom pojmu u tražilici filtriraju se naslovi knjiga. Odabrane knjige, izabranu stranicu i broj stranica poslužitelj vraća na korisničko sučelje.

```
productRouter.get(
  "/",
  asyncHandler(async (req, res) => {
    var sort = { _id: -1 };
    if (req.query.sortBy === "price-asc") {
      sort = { price: 1 };
    } else if (req.query.sortBy === "price-dsc") {
      sort = { price: -1 };
    } else if (req.query.sortBy === "newest") {
      sort = { createdAt: 1 };
    } else if (req.query.sortBy === "rating-asc") {
      sort = { rating: 1 };
    } else if (req.query.sortBy === "rating-dsc") {
      sort = { rating: -1 };
    }
    const pageSize = 6;
    const page = Number(req.query.pageNumber) || 1;
    const category = req.query.category !== " " && req.query.category !== "all"
        ? {
            "category.value": req.query.category,
          }
        : {};
    const keyword = req.query.keyword
      ? {
          name: {
            $regex: req.query.keyword,
            $options: "i",
          },
        }
      : {};
    const count = await Product.find(category).countDocuments({ ...keyword });
    const products = await Product.find(category)
      .find({ ...keyword })
      .limit(pageSize)
      .skip(pageSize * (page - 1))
      .sort(sort);
    res.json({ products, page, pages: Math.ceil(count / pageSize) });
  })
);
```

Ispis 2: *Funkcija za dohvaćanje knjiga* (Izvor: vlastita izrada)

4.1.3. Pregled proizvoda

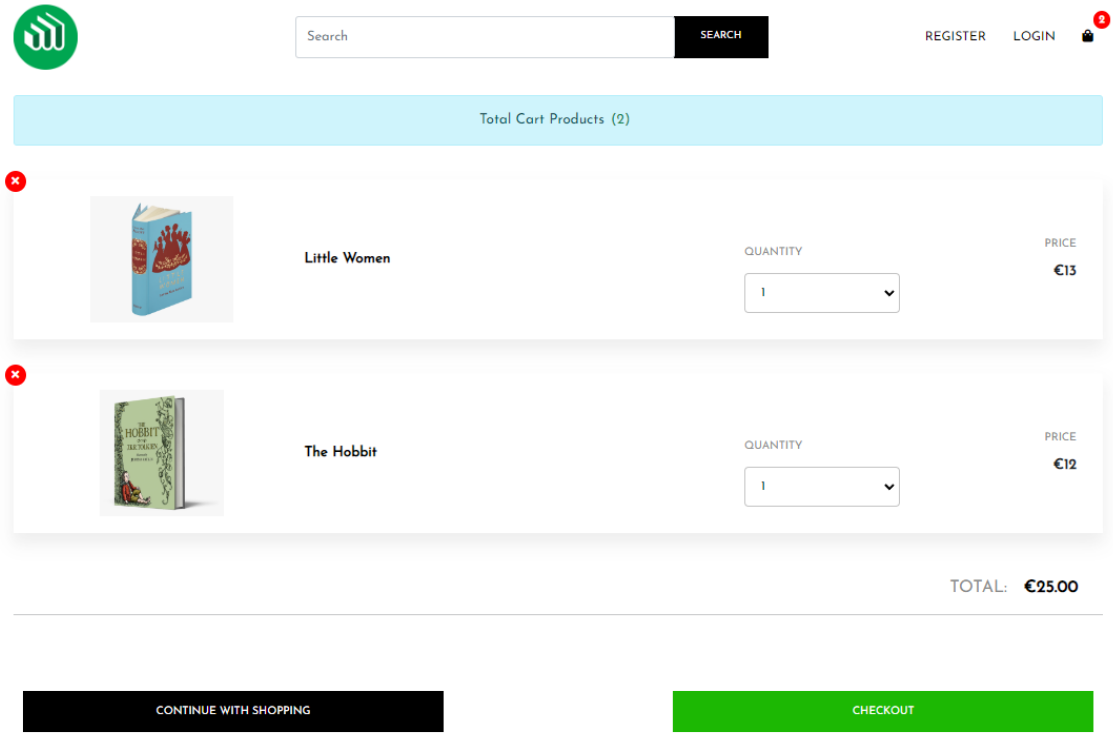
Početna stranica omogućuje pretraživanje knjiga, a klik na knjigu s početne stranice vodi do stranice koja će detaljnije prikazati odabranu knjigu. Na slici 9 vidljiv je izgled stranice. Desno od slike knjige ispisane su sve korisne informacije o knjizi: Naslov, Ime autora, Opis, Cijena, Status dostupnosti, Recenzije i dostupna količina. Klikom na dugme „Add to cart“ odabrana knjiga i količina dodaju se u košaricu. Korisnici također imaju opciju ostavljanja recenzija kako bi izrazili svoje mišljenje o knjizi kratko je komentirajući i ocjenjujući od 1 do 5. Sve ocjene prethodnih korisnika, odnosno recenzije, nalaze se ispod slike knjige te se računa srednja vrijednost svih recenzija kao ocjena knjige. Ukoliko knjiga više nije dostupna, status joj je nedostupan (engl. *unavailable)* te polje količina i dugme za dodavanje u košaricu nestaju.



Slika 9: *Stranica pregleda knjiga* (Izvor: vlastita izrada)

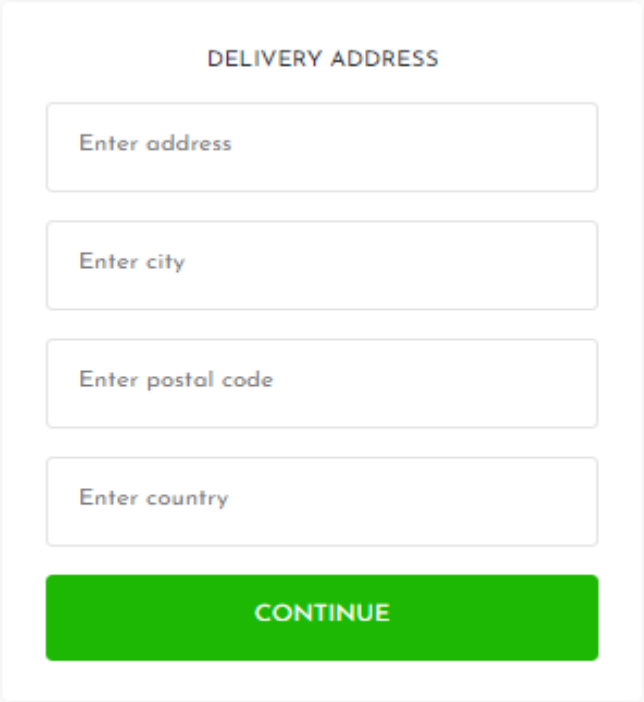
4.1.4. Košarica

Klikom na „Add to cart„ ili ikonu u gornjem desnom kutu, korisnika se preusmjerava na njegovu košaricu gdje su prikazane sve dodane knjige. Primjer izlistanih knjiga vidljiv je na slici 10. Pri vrhu je ispisan ukupan broj proizvoda, „Total Cart Products“, unutar košarice, a ispod sve knjige izlistane po redu dodavanja. Unutar kartice proizvoda vidljiva je slika, naslov, količina i cijena određene knjige. Količina se može mijenjati unutar košarice. Klikom na crveni dugme knjiga se uklanja iz košarice. Pri dnu stranice vidljiv je izračun vrijednosti svih knjiga unutar košarice zajedno s porezom i troškovima dostave. Troškovi dostave postaju 0 eura ako ukupna cijena proizvoda prijeđe 50 eura.



Slika 10: *Košarica s knjigama* (Izvor: vlastita izrada)

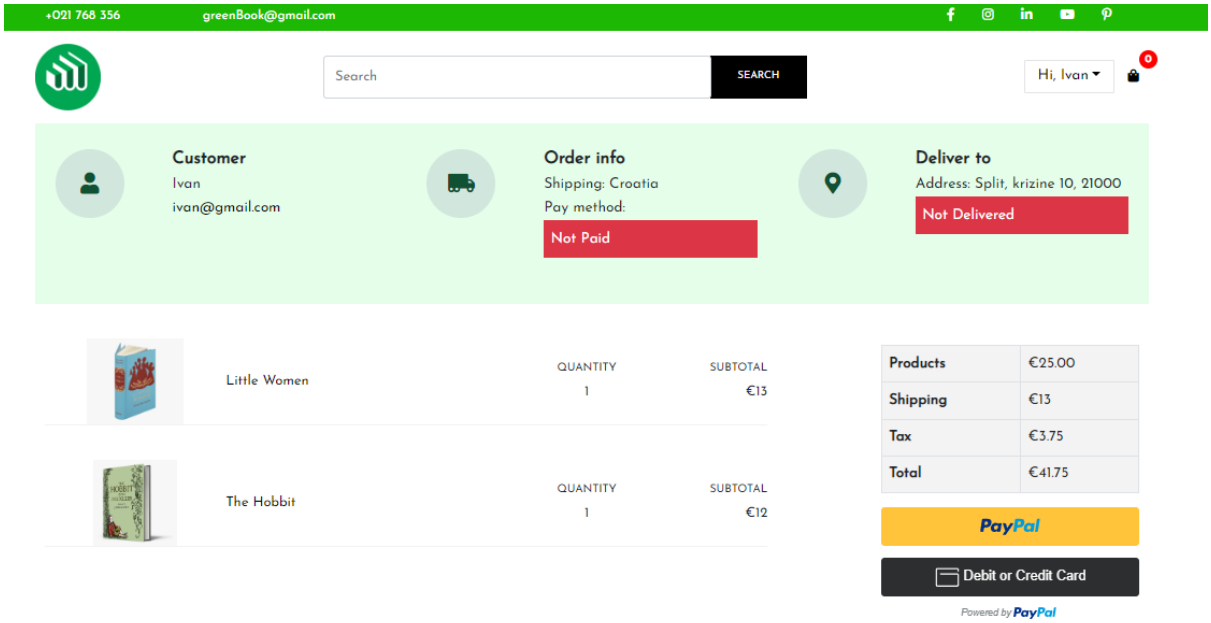
Na samome dnu stranice nalaze se dva dugmeta. Klikom na dugme „Continue with shopping“ korisnika se vraća na početnu stranicu gdje može nastaviti dodavati knjige u košaricu, a klikom na dugme „Checkout“ korisnik se dalje preusmjerava na stranicu za unos adrese prikazanu na slici 11. Adresa svakog korisnika pohranjuje se u lokalnu pohranu web preglednika tako da formu nije potrebno unositi svaki put. Klik na dugme „Continue“ preusmjerava korisnika na stranicu za plaćanje.



Slika 11: *Forma za unos adrese* (Izvor: vlastita izrada)

4.1.5. Plaćanje narudžbe

Unosom adrese korisnik prelazi na formu za odabir načina plaćanja gdje postoje dva izbora: PayPal/Kreditna kartica i plaćanje pouzećem. Nakon odabira stvara se narudžba i korisnik se preusmjerava na stranicu za pregled narudžbe vidljive na slici 12. Na vrhu stranice ispisani su podatci o kupcu, načinu plaćanja i adresi dostave. Odabirom plaćanje pouzećem narudžba je prihvaćena. Odabirom PayPal/Kreditna kartica korisnik mora dovršiti plaćanje klikom na jedan od dva dugmeta. Ako korisnik ne dovrši plaćanje u roku od jednog sata, narudžba se briše.



Slika 12: *Pregled narudžbe* (Izvor: vlastita izrada)

Navedene vrste plaćanja implementirane su koristeći PayPal API i `PayPalButton` komponente iz `react-paypal-button-v2` biblioteke. Implementacija je vidljiva na ispisu broj 3.

```
useEffect(() => {
    const addPayPalScript = async () => {
      const { data: clientId } = await axios.get("/api/config/paypal");
      const script = document.createElement("script");
      script.type = "text/javascript";
      script.src = `https://www.paypal.com/sdk/js?client-id=${clientId}`;
      script.async = true;
      script.onload = () => {
        setSdkReady(true);
      };
      document.body.appendChild(script);
    };
    if (!order || successPay) {
dispatch({ type: ORDER_PAY_RESET });
      dispatch(getOrderDetails(orderId));
    } else if (!order.isPaid) {
      if (!window.paypal) {
        addPayPalScript();
      } else {
        setSdkReady(true);
      }
    }
  }, [dispatch, orderId, successPay, order]);

  const successPaymentHandler = (paymentResult) => {
    dispatch(payOrder(orderId, paymentResult));
  };
```

Ispis 3: *Implementacija PayPal skripte* (Izvor: vlastita izrada)

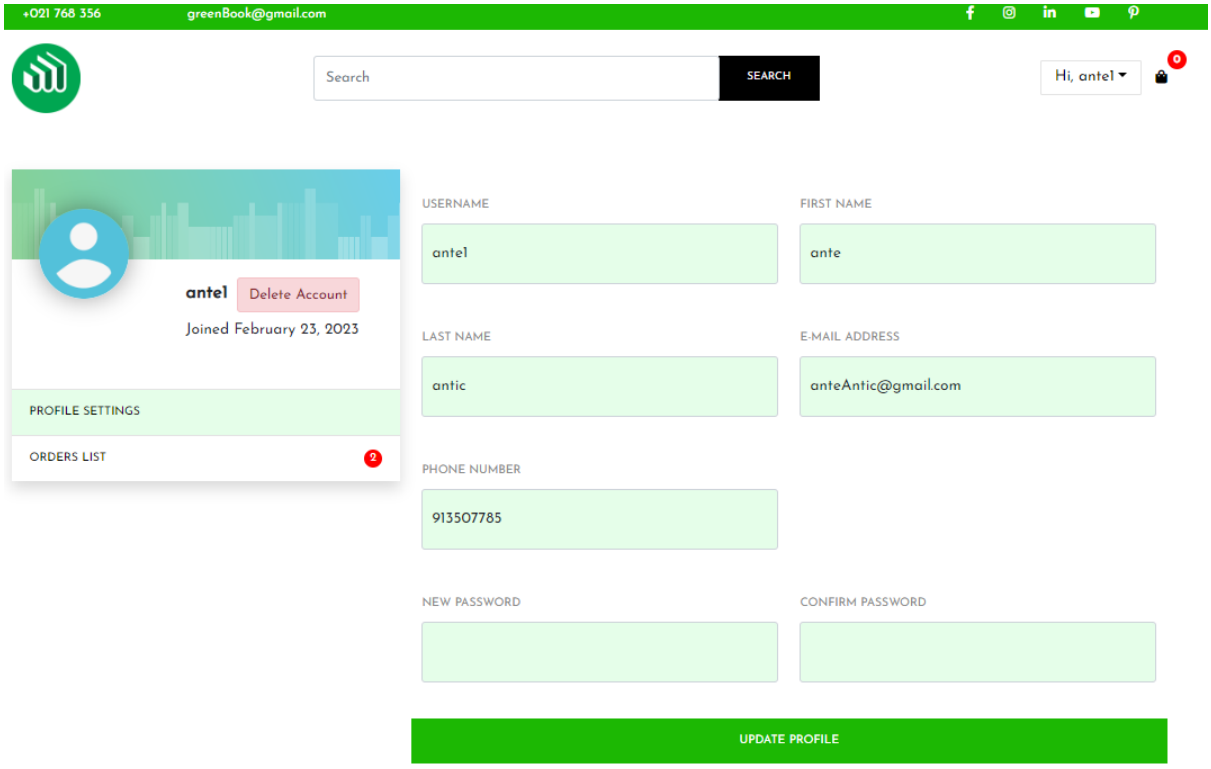
Potrebno je stvoriti skriptu čiji izvor dohvaća PayPalov SDK (engl. *Software Development Kit).* Unutar URL-a potrebno je dodati `query` parametar, jedinstveni identifikator PayPal Developer računa na koji će biti položena sredstva nakon plaćanja narudžbe. Skriptu dodajemo unutar tijela (engl. *body*) dokumenta i postavljamo `SdkReady` zastavu na *True* kako bi renderirali dugmeta na korisničko sučelje i omogućili korisniku da izvrši plaćanje. Kada korisnik unese podatke i potvrdi plaćanje, poziva se funkcija `successPaymentHandler` koja prima rezultate plaćanja. Pomoću `dispatch hooka` poziva se `payOrder` akcija kojoj se prosljeđuju poseban identifikator narudžbe i rezultati plaćanja. Akcija oblikuje HTTP *PUT* zahtjev i prosljeđuje rezultate plaćanja poslužitelju koji

ažurira narudžba sukladno rezultatima. Sve podatke narudžbe poslužitelj potom šalje elektroničkom poštom korisniku. Slika 13 prikazuje primjer e-mail potvrde o primitku narudžbe.

Slika 13: *Email potvrde plaćanja* (Izvor: vlastita izrada)

4.1.6. Korisnički profil i pregled narudžbi

Klikom na svoje ime na navigacijskoj traci spušta se *drop-down* gdje klikom na prvi izbor korisnika se preusmjerava na njegov profil. Izgled profila vidljiv je na slici 14. Prikazani su svi podatci o korisničkom računu. Podatci se mogu mijenjati, a klikom na dugme „Update profile“ novi se podatci pohranjuju u bazu podataka. Korisnik može pregledati svoje prijašnje narudžbe klikom na „Order List“ kao što je prikazano na slici 15, te vidjeti potpune detalje narudžbe klikom na njezin ID.



Slika 14: *Korisnički profil* (Izvor: vlastita izrada)

Slika 15: *Pregled svih narudžbi* (Izvor: vlastita izrada)

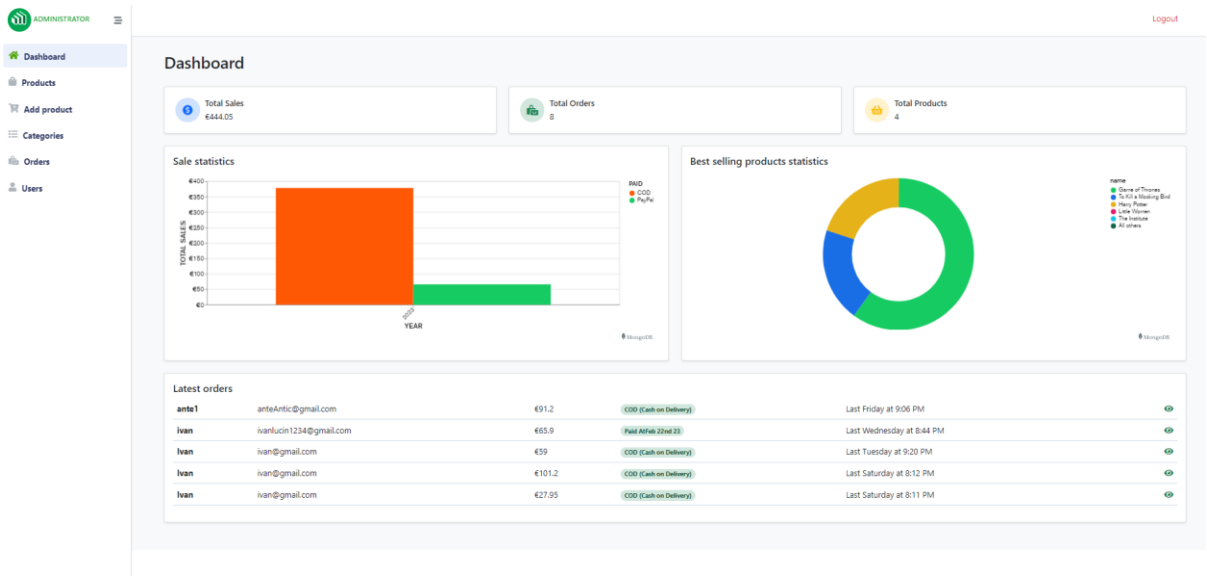
Ukoliko korisnik odluči izbrisati svoj račun, to može ostvarit klikom na dugme „Delete Account“. Otvara se skočni prozor za potvrdu brisanja te se nakon potvrde korisnika račun sa svim podatcima trajno briše iz baze podataka.

4.2. Izrada administracijskog sučelja

Administracijsko sučelje poseban je dio web aplikacije namijenjen administratoru kako bi imao pregled svih podataka poslovanja. Pomoću tog sučelja administrator može: brisati i uređivati postojeće te dodavati nove knjige, pregledavati i brisati narudžbe i označavati ih kao poslane kada se narudžba isporuči korisniku. Također, administrator ima mogućnost pregleda svih korisnika te pregled statistike narudžbi i najprodavanijih proizvoda

4.2.1. Početna stranica

Nakon prijave administrator se preusmjerava na početnu stranicu, `Dashboard`. Na početnoj stranici administracijskog sučelja prikazana je statistika trgovine, grafovi i posljednje narudžbe što je prikazano na slici 16.



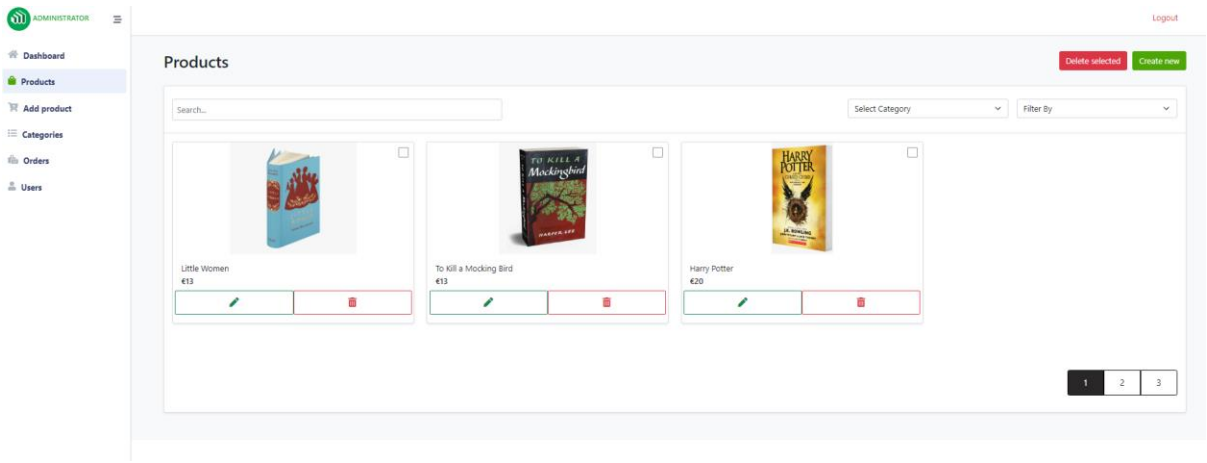
Slika 16: *Početna stranica administracijskog sučelja* (Izvor: vlastita izrada)

Na samom vrhu stranice nalaze se tri stupca statistike podataka. Prvi stupac ukazuje na ukupnu zaradu, drugi na ukupan broj narudžbi i treći na ukupan broj knjiga. Ispod statistike nalaze se dva grafa. Prvi graf pokazuje statistiku narudžbi na kojem X os predstavlja godinu, a Y os prikazuje ukupnu zaradu ostvarenih narudžbi. Prelazeći mišem preko stupca prikazuje se mali skočni prozor s detaljima. Drugi graf prikazuje najprodavanije naslove. Grafovi su generirani na MongoDB korisničkom sučelju i prikazani na administracijskom sučelju preko `iFrame-a` HTML dokumenta koji omogućuje prikaz podataka s vanjske web stranice.

Podatci na grafovima ažuriraju se svakih sat vremena. Pri dnu stranice nalazi se lista posljednjih pet narudžbi. Prikazani su podatci poput: korisničko ime kupca, e-mail adresa, ukupna cijena, podatci o plaćanju, vrijeme naručivanja i dugme za prijelaz na stranicu o pojedinosti narudžbe.

4.2.2. Pregled proizvoda

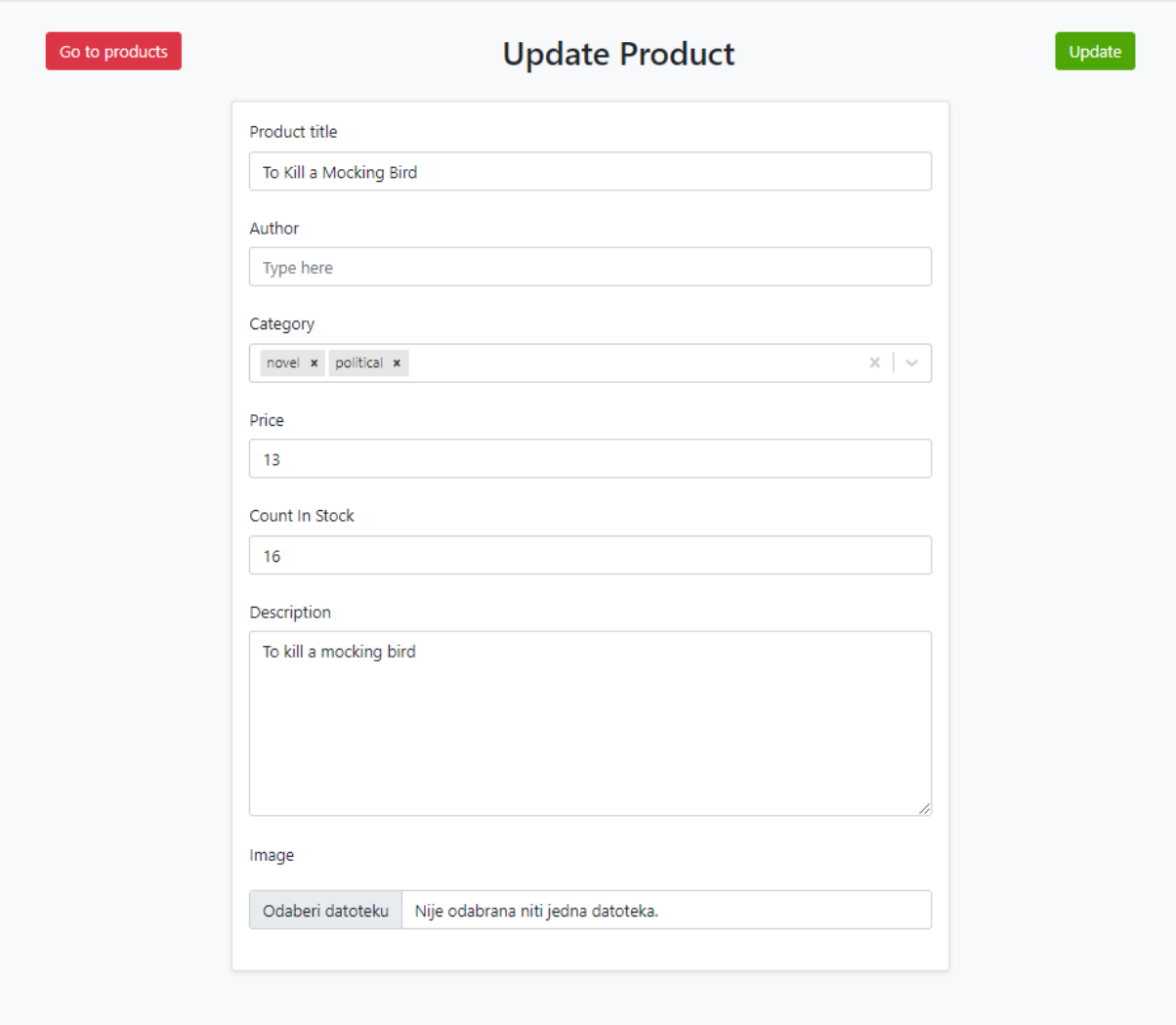
Sljedeća je stranica pregled proizvoda, `Products`, koja omogućuje administratoru uvid u sve proizvode. Primjer stranice vidljiv je na slici 17. Kao i korisnik na korisničkom sučelju, administrator na vlastitom sučelju ima mogućnost filtriranja proizvoda po kategoriji, cijeni, vremenu kada su dodani te pretraživati pomoću tražilice. Knjige su poredane prema dostupnoj količini, od najmanje prema najvećoj. Ako je količina jednaka nuli, proizvod dobiva crveni okvir kako bi administrator lakše uočio njegovu nedostupnost. Administrator ima opciju brisati proizvode pojedinačno ili odabrati više proizvoda i izbrisati ih.



Slika 17: *Pregled svih proizvoda* (Izvor: vlastita izrada)

Može i uređivati proizvode klikom na dugme „Edit“ koji će ga preusmjeriti na formu za uređivanje proizvoda vidljivu na slici 18. Trenutni podatci proizvoda uneseni su unutar svojih polja forme. Promjenom podataka i klikom na dugme „Update“ novi podatci o proizvodu šalju se poslužitelju. Na ispisu 4 vidljivo je kako poslužitelj prima nove podatke o proizvodu i ažurira podatke proizvoda na poslužitelju. Pomoću jedinstvenog identifikatora dohvaća se proizvod iz baze. Ukoliko proizvod ne postoji, na administracijskom sučelju pojavi se *error* 404, a potom novi *error* s porukom *Product not found*. Ako proizvod postoji, administrator provjerava je li nova slika unesena u formu te s web servisa Cloudinary briše

postojeću sliku i zamjenjuje ju novom, ukoliko je potrebno. Provjeravaju se i ostali podatci o proizvodu, pohranjuju se promjene ako postoje ili se zadržava stara vrijednost. Konačno se proizvod sprema u bazu podataka, a odgovor se šalje administracijskom sučelju kako bi administrator bio obaviješten o uspješnim promjenama.



Slika 18: *Forma za uređivanje proizvoda* (Izvor: vlastita izrada)

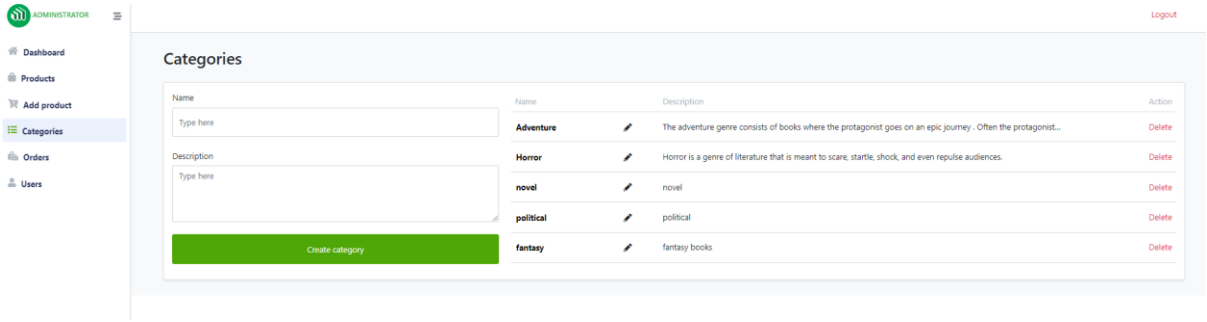
```
productRouter.put(
  "/:id",
  protect,
  admin,
  asyncHandler(async (req, res) => {
    const { name, author, category, price, description, image, countInStock } =
      req.body;
    const file = req.files ? req.files.file : null;
    const product = await Product.findById(req.params.id);
    if (product) {
      if (file) {
        await cloudinary.uploader.destroy(product.imagePublicId);
        var imageFile = await cloudinary.uploader.upload(
          file.tempFilePath,
          options,
          (err, res) => {
            console.log("error", err);
            console.log("res", res);
          }
        );
      }
      product.name = name || product.name;
      product.author = author || product.author;
      product.category = JSON.parse(category) || product.category;
      product.price = price || product.price;
      product.description = description || product.description;
      product.image = imageFile ? imageFile.secure_url : product.image;
      product.imagePublicId = imageFile
        ? imageFile.public_id
        : product.imagePublicId;
      product.countInStock = countInStock || product.countInStock;

      const updateProduct = await product.save();
      res.json(updateProduct);
    } else {
      res.status(404);
      throw new Error("Product not found");
    }
  })
);
```

Ispis 4: *Funkcija za editiranje proizvoda*

Na administracijskom sučelju nalazi se i dugme za stvaranje novih proizvoda čija je forma ista kao i forma za uređivanje proizvoda.

4.2.3. Upravljanje kategorijama

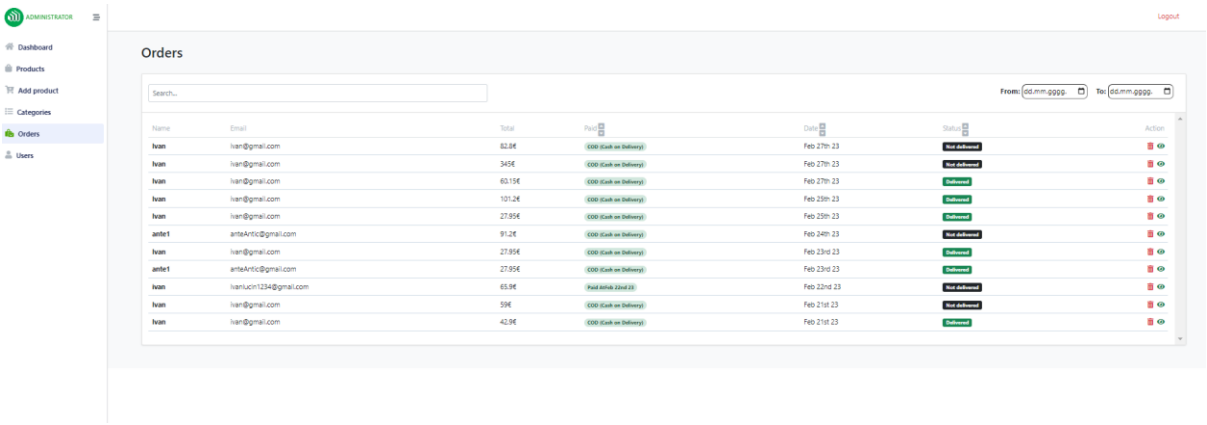
Administracijsko sučelje ima i posebnu stranicu za pregled i upravljanje svim kategorijama. Na slici 19 vidljiv je izgled stranice. Na lijevoj strani nalazi se forma za stvaranje novih kategorija. Unosom imena i kratkog opisa od najviše 200 karaktera i pritiskom na dugme „Create category“ stvara se nova kategorija. Sada administrator ima mogućnost dodavanja novih proizvoda, spremanja u novu kategoriju ili uređivanja postojećih proizvoda. S desne strane nalazi se popis svih kategorija. Administrator može uređivati nazive i opise kategorija klikom na dugme olovke ili brisati kategorije. Brisanjem određene kategorije brišu se i proizvodi koji pod atributom kategorija imaju samo navedenu kategoriju.



Slika 19: *Stranica za upravljanje kategorijama* (Izvor: vlastita izrada)

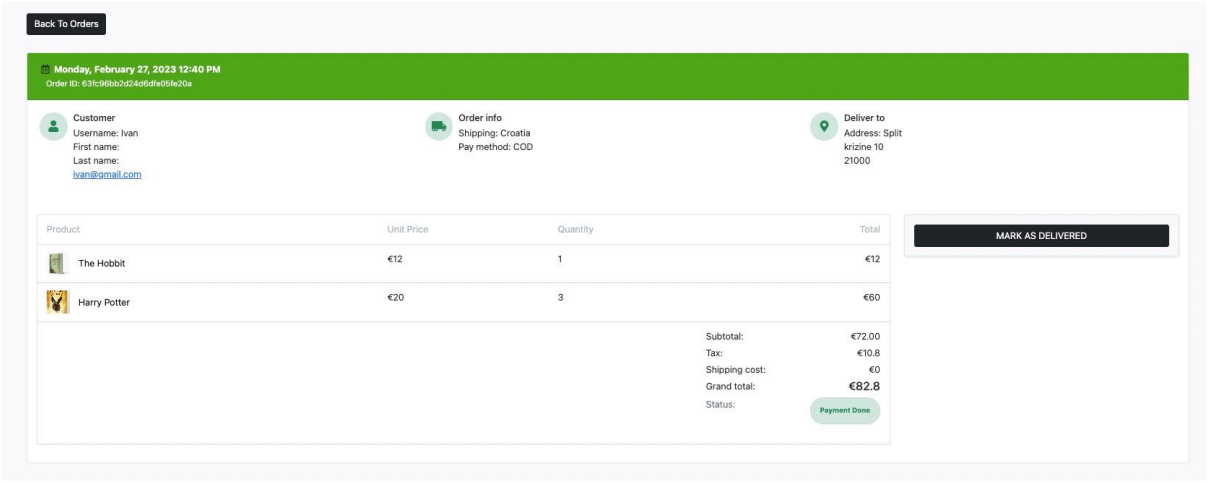
4.2.4. Upravljanje narudžbama

Stranica za upravljanje narudžbama omogućuje administratoru pregled svih narudžbi i upravljanje istima što je vidljivo na slici 20. Svaka narudžba prikazuje podatke o imenu i e-mail adresi kupca, ukupnoj cijeni, informaciji o plaćanju, datumu kada je narudžba stvorena, statusu narudžbe te dugme za brisanje i pregled detalja narudžbe.



Slika 20: *Stranica za upravljanje narudžbama* (Izvor: vlastita izrada)

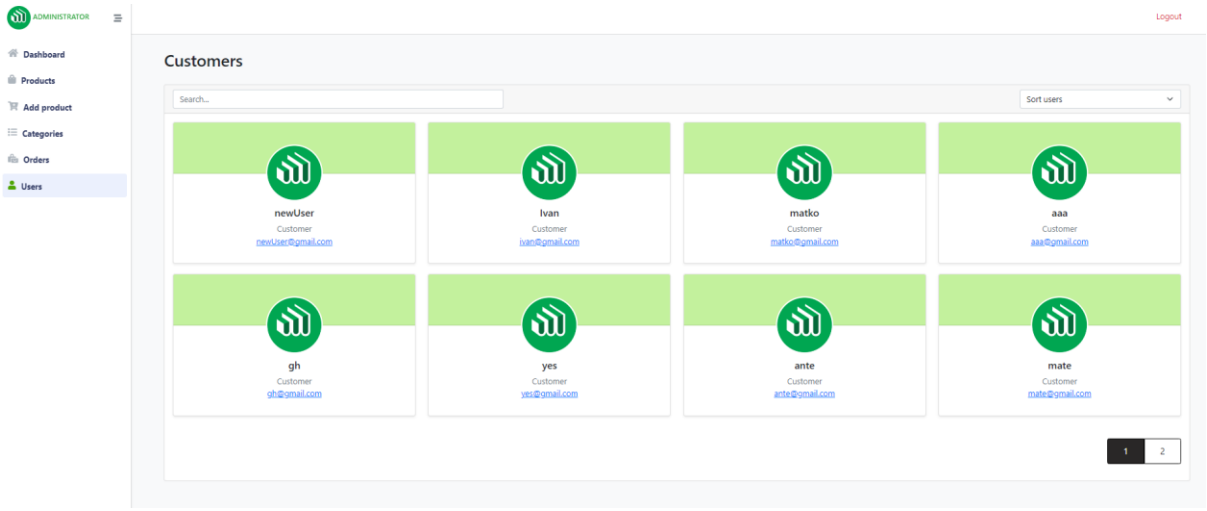
Klikom na dugme za pregled detalja administrator se preusmjerava na stranicu gdje može vidjeti sve detalje o narudžbi što je vidljivo na slici 21. Kada je narudžba isporučena korisniku, označava se kao poslana klikom na dugme „Mark as delivered“. Podatci o narudžbi ažuriraju se na poslužitelju i pohranjuju u bazu podataka.



Slika 21: *Detalji o narudžbi* (Izvor: vlastita izrada)

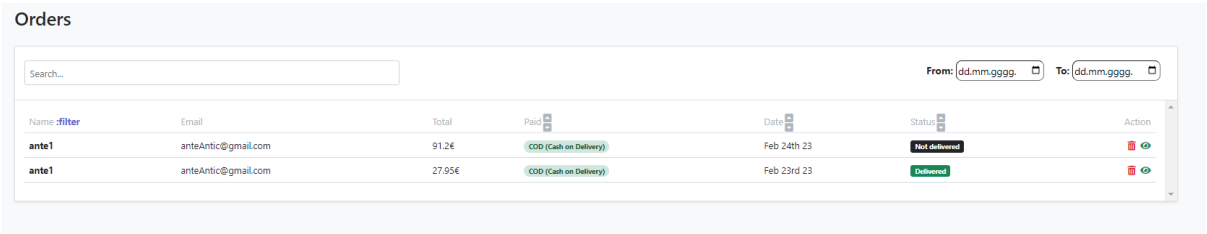
4.2.5. Pregled korisnika

Posljednja stranica administracijskog sučelja, vidljiva na slici 22, pregled je korisnika. Administrator može pregledavati sve korisničke račune s određenim podatcima o korisniku, a to su korisničko ime i e-mail adresa.



Slika 22: *Pregled korisnika* (Izvor: vlastita izrada)

Administrator se klikom na određenog korisnika preusmjerava na stranicu za pregled narudžbi gdje su izlistane samo narudžbe odabranog korisnika. Kako bi administrator lakše raspoznao da je riječ o narudžbama samo odabranog korisnika, pokraj stupca `Name` nalazi se plavim podebljanim slovima *:filter* (slika 23).



Slika 23: *Pregled narudžbi odabranog korisnika* (Izvor: vlastita izrada)

Filtriranje po korisniku postiže se parametrom *query* za traženje unutar URL-a. Funkcija za dohvaćanje narudžbi vidljiva je na ispisu 5.

```
orderRouter.get(
  "/all",
  protect,
  admin,
  asyncHandler(async (req, res) => {
    const allOrders = await Order.find({}).sort({ _id: -1 });
    deleteOrders(allOrders);

    const userID = req.query ? req.query.user : null;
    const userFilter = userID
      ? { "user.userId": mongoose.Types.ObjectId(userID) }
      : {};
    const orders = await Order.find({ isPaid: true })
      .find(userFilter)
      .sort({ _id: -1 })
      .populate("user", "id name email");
    res.json(orders);
  })
);
```

Ispis 5: *Funkcija za dohvaćanje svih ili korisnikovih narudžbi* (Izvor: vlastita izrada)

Unutar funkcije inicijalno se dohvaćaju sve narudžbe i prosljeđuju u funkciju `deleteOrders` kojom se provjerava jesu li narudžbe plaćene kao i vrijeme proteklo od trenutka kreiranja narudžbe. Ukoliko su narudžbe neplaćene te je prošlo duže od sat vremena od njihova kreiranja, narudžbe se brišu. Zatim se provjerava postoje li *query* parametri unutar HTTP zahtjeva. Ako postoje, pohranjuju se unutar varijable `userID`. Stvara se objekt `userFilter` koji u slučaju postojanja `userID-a` ima vrijednost, a prazan je ako ne postoji. Konačno se dohvaćaju narudžbe koje je potrebno vratiti administracijskom sučelju. Filtriraju se po atributu `isPaid` kako bi se vratile samo plaćene narudžbe te ako je proslijeđena vrijednost `userFilteru` po izabranom korisniku.

5. Zaključak

Zadatak završnoga rada bio je kreirati dinamičnu web trgovinu specijaliziranu za prodaju knjiga. Web trgovina GreenBook vrlo je intuitivna, jednostavna i praktična za korištenje bilo da je riječ o administratoru ili korisniku. Aplikacija je izrađena po pravilima i preporukama dobroga web dizajna, a sastoji se od administracijskog i korisničkog sučelja.
Korisnik ima mogućnost prijavljivanja i uređivanja vlastitog korisničkog računa, brzog i jednostavnog pretraživanja dostupnih knjiga i korisnih informacija o knjizi, pretraživanja ocjena drugih korisnika, mogućnost naručivanja i pregledavanja najprodavanijih naslova te sigurno plaćanje. Administrator ima potpuni pregled nad svim podatcima vezanim za web trgovinu poput mogućnosti pregleda svih korisnika, dodavanja novih naslova ili brisanja postojećih ukoliko web trgovina proširuje ili smanjuje svoj repertoar. Također, prati statistiku prodaje i popularnosti proizvoda, pregled narudžbi i izmjenu svih dosadašnjih narudžbi.

U radu su pobliže opisane tehnologije korištene za izradu i izgled web aplikacije te sama implementacija aplikacije. Ova se aplikacija odlikuje svojom jednostavnošću, responzivnim dizajnom i fleksibilnošću zahvaljujući MERN tehnologijama (**M**ongoDB, **E**xpress.js, **R**eact, **N**ode.js). Poslužiteljska strana (engl. *backend*) izrađena je u platformi Node.js te programskom jeziku JavaScript. Biblioteka React znatno je pojednostavila stvaranje korisničkog sučelja (engl. *frontend*) ove internetske trgovine. MongoDB kao baza podataka omogućila je kreiranje dinamičnih entiteta jednostavnih za stvaranje, upravljanje i brisanje zahvaljujući Expressu. HTML tehnologija poslužila je za pisanje teksta prikazanog na web aplikaciji, a dizajn i privlačniji izgled aplikacije postignut je uporabom CSS-a i Bootstrapa.

Web aplikacije neprestano se razvijaju s napretkom tehnologije. Postoji mnogo prostora za napredak i nadogradnju web aplikacije GreenBook: od bolje strukture kôda do novih funkcionalnosti za administratora i korisnika, poput ostvarivanja dodatnih bodova pri kupnji knjiga, popusti na proizvodima, nagrađivanja vjernosti korisnika, prikazivanja obavijesti na sučelju itd. Izrađivanje web trgovine GreenBook upoznao sam i suočio se s problemima razvoja *Full-stack* web aplikacije. Potrebno je znanje kao dobra podloga za rad te mnogo uloženog vremena u fazu planiranja kako bi se susreli sa što manjim brojem problema tijekom procesa razvoja.

Literatura

[1] MDN, „How Popular Is JavaScript on 2019?“, (11.05.2019). [Na internetu]. Dostupno: https://medium.com/javascript-scene/how-popular-is-javascript-in-2019-823712f7c4b1 .[pristupano 08.02.2023.].

[2] Priyesh Patel, „What exactly is Node.js?“. 2018. [Na internetu]. Dostupno: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/ . [pristupano 08.02.2023.].

[3] W3Schools, „What is npm?“, (bez dat.). [Na internetu]. Dostupno: https://www.w3schools.com/whatis/whatis_npm.asp . [pristupano 10.02.2023.].

[4] TutorialsTeacher, „Express.js“, (bez dat.). [Na internetu]. Dostupno: https://www.tutorialsteacher.com/nodejs/expressjs. [pristupano 10.02.2023.].

[5] Codecademy, „MVC: Model, View, Controller“, (bez dat.). [Na internetu]. Dostupno: https://www.codecademy.com/articles/mvc [pristupano 10.02.2023.].

[6] Mdn Web Docs, „SPA (Single-page application)?“, [Na internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Glossary/SPA . [pristupano 10.02.2023.].

[7] Alesia Sirotka, „What is React?“, (18.02.2022.). [Na internetu]. Dostupno: https://flatlogic.com/blog/what-is-react/ . [pristupano 11.02.2023.].

[8] MongoDB, (bez dat.). [Na internetu]. Dostupno: https://www.mongodb.com/ . [pristupano 11.02.2023.].

[9] Guru99, „SQL vs NoSQL: What's the difference?“, (bez dat.). [Na internetu]. Dostupno: https://www.guru99.com/sql-vs-nosql.html . [pristupano 11.02.2023.].

[10] Johnny Youssef, Pros and cons of using bootstrap, (04.05.2021.). [Na internetu]. Dostupno: https://www.bemea.com/pros-and-cons-of-using-bootstrap/. [pristupano 12.02.2023.].

[11] Emina Mustabašić, Šta je responzivni web dizajn i zašto je bitan?, (17.09.2020.). [Na internetu]. Dostupno: https://grm.digital/bs/Blog/sta-je-responzivni-web-dizajn. [pristupano 12.02.2023.].