Web aplikacija za evidenciju putnih naloga

Dražetić, Danijel

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB APLIKACIJA ZA EVIDENCIJU PUTNIH NALOGA

Diplomski rad

Danijel Dražetić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 14.09.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Danijel Dražetić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1112R, 13.10.2020.
OIB studenta:	49031832249
Mentor:	Izv. prof. dr. sc. Ivica Lukić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	Izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 2:	Izv. prof. dr. sc. Mirko Köhler
Naslov diplomskog rada:	Web aplikacija za evidenciju putnih naloga
Znanstvena grana diplomskog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	U aplikaciji oomogućiti registraciju i prijavu djelatnika, a djelatnici mogu imati više različitih uloga. Djelatnici podnose zahtjev za putovanje u određeno mjesto, početak putovanja, trajanje i slično. Odabiru projekt s kojeg se financira putovanje. Projekt ima voditelja koji treba odobriti nalog, tražiti izmjene ili odbiti. Kad voditelj odobri direktor tvrtke također prihvaća, korigira ili odbija nalog. Kad se djelatnik vrati s putovanja podnosi izvješće. Ako je išao van države unosi granice i računaju se dnevnice. Kad popuni sve šalje računovodstvu na isplatu. Rezervirano za: Danijel Dražetić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	14.09.2022.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.* Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2022.

Ime i prezime studenta:	Danijel Dražetić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1112R, 13.10.2020.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za evidenciju putnih naloga**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivica Lukić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Tema ovog diplomskog rada je izrada web aplikacije za evidenciju putnih naloga unutar nekog poduzeća. Cilj web aplikacije je olakšavanje izrade i provjere putnih naloga digitalnim putem. Aplikacija ima mogućnost prijave za četiri različite uloge u firmi. Uloge su admin, direktor, voditelj jedinice i radnik. Admin ima uvid na sve registrirane korisnike u bazi podataka te ima dopuštenje za izmjenu uloga ostalih korisnika. Direktoru je omogućen uvid u sve putne naloge koji se nalaze unutar baze podataka te ih može uređivati, odbiti i prihvatiti. Voditelju jedinice je omogućen uvid u sve putne naloge koji pripadaju njegovoj ustrojbenoj jedinici. Voditelji mogu poništiti ili prihvatiti nalog. Radnicima je omogućeno kreiranje i uređivanje putnih naloga. Radnicima u korisničkom sučelju su prikazani svi putni nalozi koje su oni kreirali.

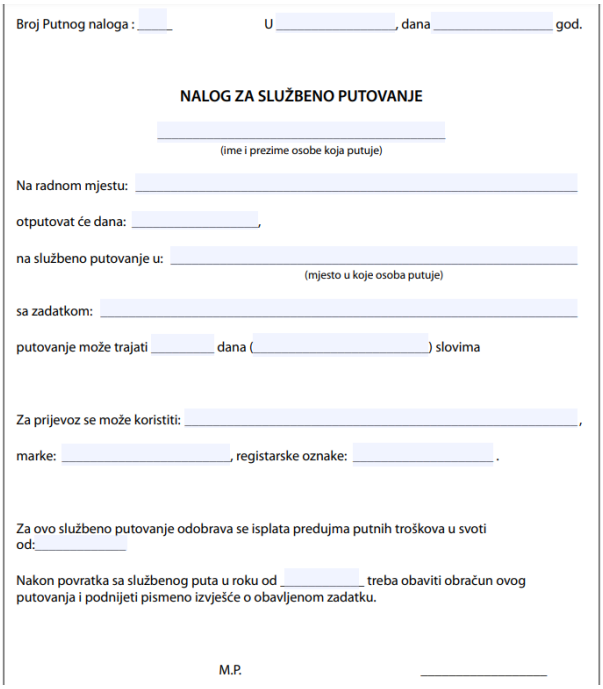
U drugom poglavlju nalaze se opće informacije o putnim nalozima te pregled područja teme. U trećem poglavlju detaljnije su objašnjene korištene tehnologije i alati. Za izradu korisničkog sučelja korištena je JavaScript biblioteka React.js, dok je za backend korišten Node js. U četvrtom poglavlju je opisan postupak izrade web aplikacije koji se dijeli na tri dijela, a to su baza podataka, backend i frontend. Unutar baze podataka se spremaju svi korisnici i svi izrađeni putni nalozi. Putni nalozi koji su odbijeni od strane voditelja ili direktora ostaju u bazi podataka. Korisniku je omogućen uvid u odbijene putne naloge u arhivi. Backend sadrži osnovnu računalnu logiku web aplikacije dok frontend pruža grafičko sučelje koje služi za interakciju sa korisnicima.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izrada web aplikacije za evidenciju putnih naloga unutar poduzeća. Omogućena je prijava i registracija korisnika te je korisniku omogućen unos, izmjena i prikaz putnih naloga. Zadatak je izvršen u softverskom okruženju Visual Studio Code-u.

2. PREGLED PODRUČJA TEME

Putne naloge i obračune putnih naloga je moguće izrađivati na tiskanom primjerku koji se može vidjeti na slikama broj 2.1 i 2.2. Primjerak je preuzet sa stranice knjigovodstvenog servisa Rk mreža usluga d.o.o [10].



Broj Putnog naloga : _____ U _____, dana _____ god.

NALOG ZA SLUŽBENO PUTOVANJE

_____
(ime i prezime osobe koja putuje)

Na radnom mjestu: _____

otputovat će dana: _____,

na službeno putovanje u: _____
(mjesto u koje osoba putuje)

sa zadatkom: _____

putovanje može trajati _____ dana (_____) slovima

Za prijevoz se može koristiti: _____,

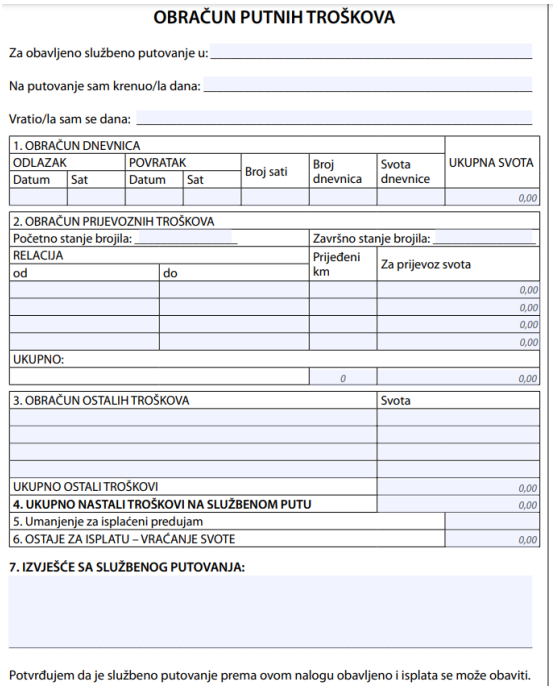
marke: _____, registarske oznake: _____ .

Za ovo službeno putovanje odobrava se isplata predujma putnih troškova u svoti od:_____

Nakon povratka sa službenog puta u roku od _____ treba obaviti obračun ovog putovanja i podnijeti pismeno izvješće o obavljenom zadatku.

M.P. _____

Slika 2.1. Putni nalog



OBRAČUN PUTNIH TROŠKOVA

Za obavljeno službeno putovanje u: _____

Na putovanje sam krenuo/la dana: _____

Vratio/la sam se dana: _____

1. OBRAČUN DNEVNICA							
ODLAZAK		POVRATAK		Broj sati	Broj dnevnica	Svota dnevnice	UKUPNA SVOTA
Datum	Sat	Datum	Sat				
							0,00

2. OBRAČUN PRIJEVOZNIH TROŠKOVA			
Početno stanje brojila: _____		Završno stanje brojila: _____	
RELACIJA		Prijeđeni km	Za prijevoz svota
od	do		
			0,00
			0,00
			0,00
			0,00
UKUPNO:			
		0	0,00

3. OBRAČUN OSTALIH TROŠKOVA	Svota
UKUPNO OSTALI TROŠKOVI	0,00
4. UKUPNO NASTALI TROŠKOVI NA SLUŽBENOM PUTU	0,00
5. Umanjenje za isplaćeni predujam	
6. OSTAJE ZA ISPLATU – VRAĆANJE SVOTE	0,00

7. IZVJEŠĆE SA SLUŽBENOG PUTOVANJA:

Potvrđujem da je službeno putovanje prema ovom nalogu obavljeno i isplata se može obaviti.

Slika 2.2. Obračun putnog naloga

GOOMA je web aplikacija koja po mjeri klijenta koji putne troškove i putne naloge čini jednostavnijima. Kao integrirani geolokacijski sustav, GOOMA u stvarnom vremenu prikuplja i obrađuje podatke o putnim troškovima [6].

KONTO je web sučelje u kojem se zaposlenici mogu nalaze na više lokacija i imati veći broj zaposlenika. Svaki zaposlenik može za sebe zatražiti odobrenje putnog naloga. implementirana je potpuna automatizacija obračuna dnevnica [14].

RELAGO je web aplikacija koja nudi usluge poput: automatskog obračuna putnog naloga, unos više odredišta i prijevoznih sredstava, unos akontacija, terenske dodatke [12].

2.1. Putni nalog

Dnevnica je naknada nastalih troškova prijevoza, prehrane i pića tijekom službenog putovanja u zemlji i u inozemstvu. Mjesto na koje se zaposlenik upućuje na službeno putovanje mora biti udaljeno od mjesta rada ili mjesta prebivališta / boravišta zaposlenika minimalno 30 km. Putni nalog je dokument kojim se dokazuje da je zaposlenik upućen na službeno putovanje i predstavlja osnovu za obračun i knjiženje troškova nastalih tijekom službenog putovanja, a samim time i refundaciju istih zaposleniku [8].

Putni nalog se sastoji od dva dijela: putnog naloga i obračuna putnog naloga. Putni nalog se ispunjava prije putovanja od strane korisnika te se potvrđuje od strane voditelja i direktora. Neki od podataka koje sadrži su: osobni podatci putnika, datum i odredište putovanja, svrha i vrijeme trajanja putovanja. Obračun putnog naloga sastavlja po povratku putnika s putovanja, a sadrži sljedeće podatke: datum i odredište putovanja, osobne podatke putnika, dan početka i završetka putovanja, ostale troškove [9].

3. KORIŠTENE TEHNOLOGIJE I ALATI

U suvremenom vremenu izrada web aplikacija zahtjeva kompleksno znanje koje sažima razne vrste alata poput:

- Visual Studio Code,
- DBeaver
- HTML
- Sass (CSS)
- Javascript
- Bootstrap
- React.js
- Node.js
- MySQL

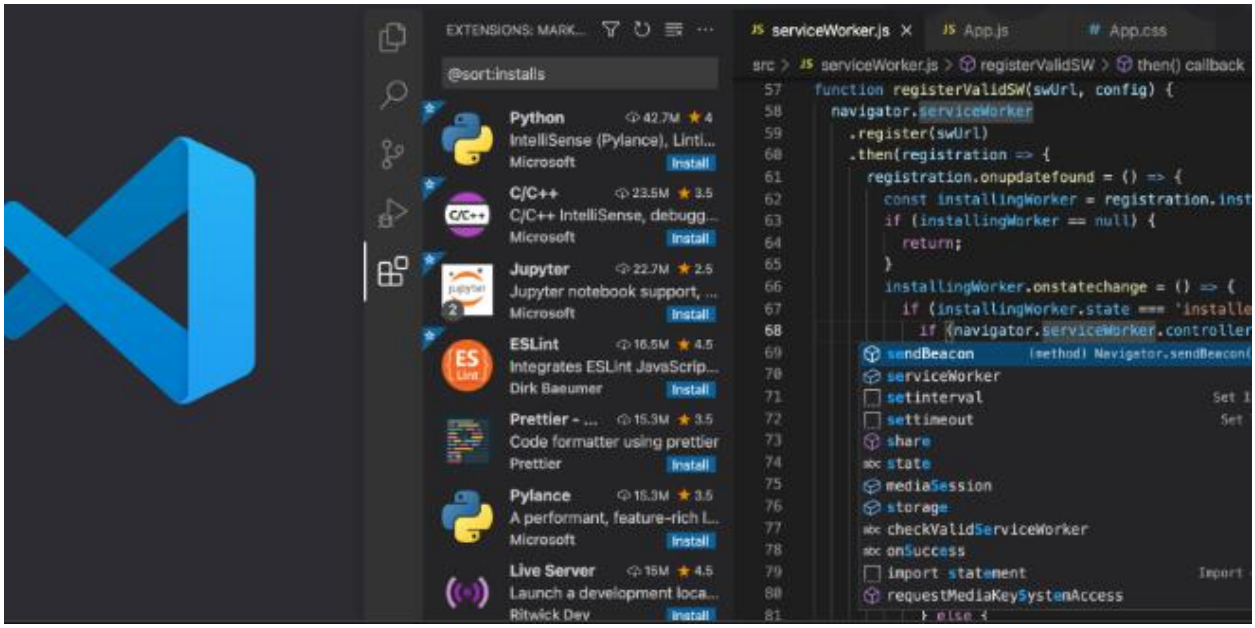
3.1. Visual Studio Code

Visual Studio Code se može opisati kao uređivač koda koji sadrži podršku razvojnih operacija kao što su: [16]

1. Otklanjanje pogrešaka
2. Pokretanje zadataka
3. Kontrole verzija

Cilj Visual Studio Code-a je pružiti alate koje developer treba za brzi ciklus izrade koda, otklanjanja pogrešaka te prepuštanja složenije tijekove rada IDE-ima s punim značajkama, kao što je Visual Studio IDE. Visual Studio Code je dostupan za Windows, macOS i Linux. Dolazi sa ugrađenom podrškom za JavaScript, TypeScript i Node.js. Osim cijele ideje da bude lagan i da se brzo pokreće, VS Code ima IntelliSense dovršavanje koda za varijable, metode i uvezene module; grafičko otklanjanje pogrešaka; linting, uređivanje s više kursora, savjeti za parametre i druge moćne značajke za uređivanje. Velik dio toga prilagođen je tehnologiji Visual Studio. Pravi VS kod izgrađen je pomoću Electron shell-a, Node.js-a, TypeScript-a i protokola jezičnog poslužitelja te se ažurira na mjesečnoj bazi. Proširenja se ažuriraju onoliko često koliko je potrebno. Bogatstvo podrške razlikuje se od različitih programskih jezika i njihovih proširenja, u rasponu od

jednostavnog isticanja sintakse i podudaranja zagrada do otklanjanja pogrešaka i refaktoriranja [16].



Slika 1.1. Visual Studio Code.

3.2. DBeaver

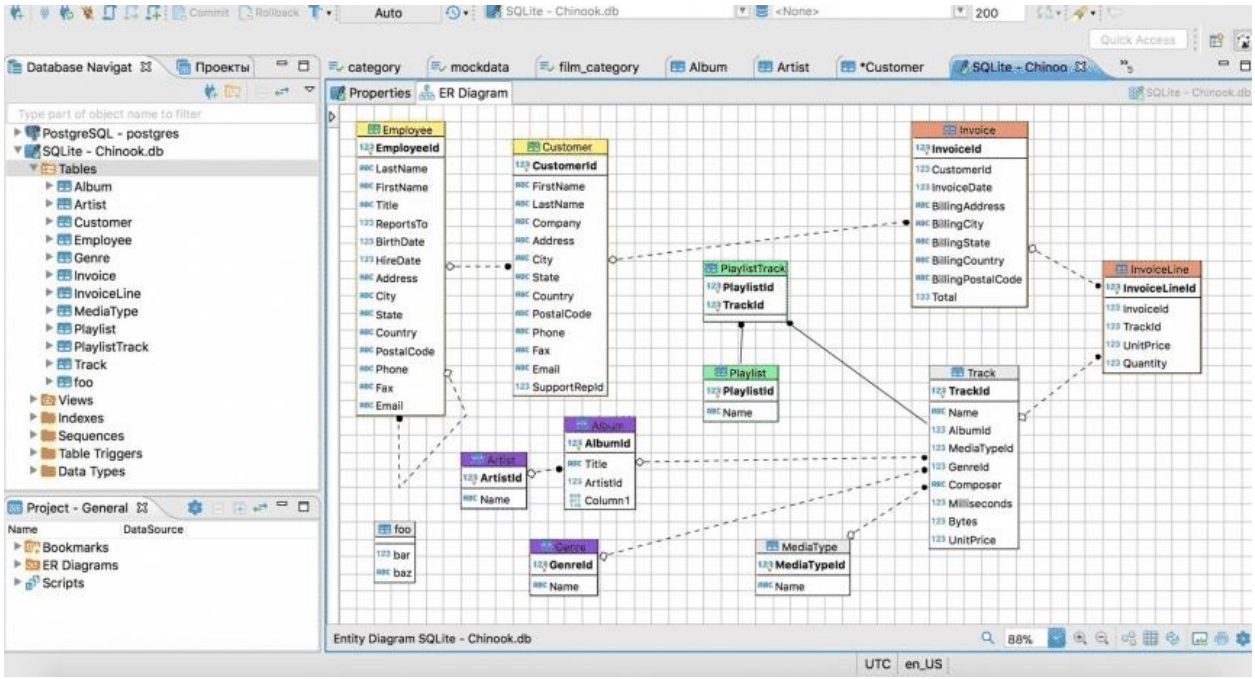
DBeaver predstavlja suvremen alat koji služi za upravljanje bazama podataka te omogućava manipulaciju podacima jednaku kao u proračunskim tablicama. Također ima mogućnost stvaranja analitičkih izvješća koji se temelje na zapisima iz nekoliko izvora te se podaci također mogu izvoziti u odgovarajućim formatima. U profesionalnom korištenju, DBeaver pruža mogućnosti administrativnih te razvojnih značajki koje konvertiraju sate developmenta u nekoliko klikova. Također podržava više od 70 baza podataka [2].

DBevaer datira iz 2010. godine, kada je postao produkt hobija. Početna zamisao bila je besplatan softver otvorenog koda s praktičnim korisničkim sučeljem. Prvo službeno izdanje uslijedilo je godinu dana kasnije na Freecode-u. Godinu dana kasnije, 2012. godine izlazi verzija dodataka za Eclipse. 2015. godine DBeaver zajednica u potpunosti se preselila na GitHub.

DBeaver je danas alat s više platformi te je podržavan od strane Windowsa, Linuxa, MaxOSa. Također je dostupan na nekoliko jezika odnosno na kineskom, engleskom, njemačkom te talijanskom jeziku [4].

Uključuje značajke poput: [3]

1. Uređivač podataka koji sadrže velik broj značajki
2. Upravljanje SQL skriptama
3. SSL podrška
4. Backup podataka
5. Testiranje baze podataka
6. Pretpregled
7. Uređivanje i sl.



Slika 3.2. Prikaz sučelja DBeaver-a

3.3. HTML

HTML (engl. *HyperText Markup Language*) se može opisati kao programski jezik koji se služi za dokumente koji su dizajnirani za prikazivanje na webu. Često se koristi u kombinaciji s CSS-om (engl. *Cascading Style Sheets*) te jezikom *JavaScript.* HTML datira iz 1993. godine kada je prvi puta prezentirana verzije HTML jezika. Prva verzija ovakvog jezika je za to vrijeme bila napredna no i dalje ograničena na način da nije postojala mogućnost dodavanja slika u dokumente. Veći napredak ostvario se 1995. godine kada se objavila treća verzija koja se unaprijedila s mogućnošću definiranja tablica.[11] Kroz godine ovaj programski jezik bilježio je razvoj. HTML dokument

ima svoju strukturu koja se sastoji od HTML elemenata odnosno svaki html element građen je od jednog para HTML oznaka. Kao što je već navedeno, HTML jezik može se kombinirati sa JavaScriptom te CSS-om, pri tome JavaScript odražava se na ponašanje te sam sadržaj web-stranica dok se CSS država na raspored sadržaja unutar stranice te na izgled odnosno dizajn. Prema CSS standardu postoje tri vrste stilskih obrazaca odnosno [1]:

1. Linijski – obilježja kroz kompletan HTML dokument
2. Vezani – pohrana obilježja u zasebnoj datoteci koja se povezuje s HTML dokumentom s <link>
3. Građeni – obilježja se ugrađuju u <style> na samom početku dokumenta

3.4. Sass (CSS)

CSS (engl. *Cascading Style Sheets*) opisuje se kao stilski jezik čija je namjena opis dokumenta koji je napisan u programskom jeziku kao što je HTML. CSS se također može opisati kao temelj WWW-a. CSS je kreiran kako bi se omogućio vizualan sadržaj te izgled web stranice. Pri tome se smatraju boje i fontovi odnosno sam dizajn. Sass (engl. *Syntactically Awesome Style Sheets*) predstavlja pretprocesorski skriptni jezik koji se tumači u CSS-u. Sass je građen od dvije sintakse odnosno: [5]

1. Izvorna sintaksa – ona koristi uvlaku prilikom odvajanja bloka
2. SCSS – odnosno engl. *Sassy CSS* ima oblikovanje kao kod CSS-a – prilikom označavanja blokova koriste se zagrade dok se za odvajanje koriste točka zarez.

Sass ima nekoliko značajki odnosno: [5]

1. Definiranje varijabli
2. Podržava: brojeve, nizove, boje
3. Argumenti
4. Pozivanje elemenata pomoću naziva tog elementa ili njegovih IDA-a

3.5. Javascript

JavaScript se može opisati kao izrazito jednostavan programski jezik koji je namijenjen radu u razvoju stranica putem HTML-a. JavaScript koriste preglednici poput *Google Chrome-a, Mozilla*

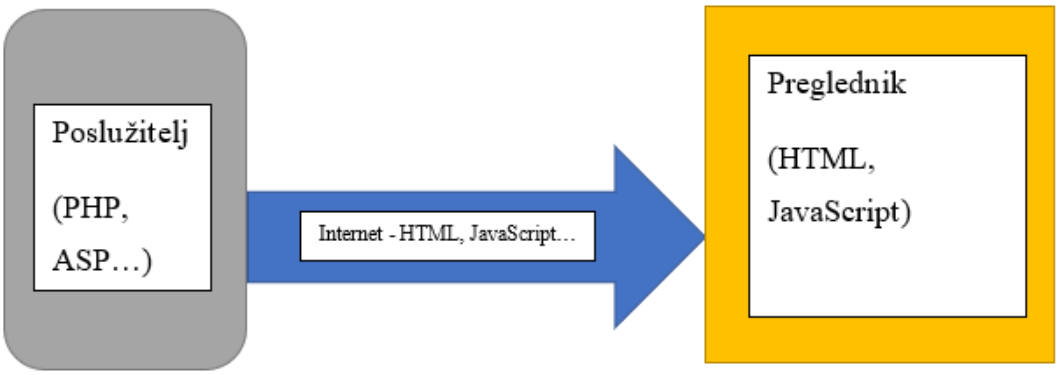
Firefox-a, Opera i sl. JavaScript datira iz 1995. godine od strane *Netscape-a*. [19] JavaScript predstavlja programski jezik koji se može opisati kao komponenta temelja WWW-a. Danas skoro 98% web-sjedišta koristi JavaScript odnosno svi glavi pretraživači imaju namjenski mehanizam JavaScripta koji služi za izvršavanje koda kod korisničkih uređaja. JavaScript predstavlja programski jezik visoke razine zbog svojih karakteristika:

- Dinamičko tipkanje,
- Prvoklasne funkcije,
- Podržava stilove kodiranja koji su potakni događajima,
- Sučelje za razvoj aplikacija,
- Objektno orijentirano programiranje

JavaScript se definira kao skriptni jezik odnosno sastoji se od 3 koraka: [18]

1. Pisanje/popravljanje skripte
2. Pokretanje interpretera
3. Prilikom korigiranja ponavlja se prvi korak.

U nastavku na slici 3.3. prikazan je tok podataka od poslužitelja do preglednika i uloga JavaScripta pri tom procesu.



Slika 3.3. Tok podataka od poslužitelja do preglednika.

3.6. Bootstrap

Bootstrap predstavlja CSS okvir otvorenog koda koji se bazira na respozivnom frontend *developmentu*. On je upotpunosti besplatan te se prvenstveno koristi za mobilne uređaje. On sadrži: [7]

- HTML,
- CSS,
- Opcionalne dizajn predloške,
- Obrasce,
- Gumbe,
- Navigaciju te još razne komponente sučelja.

Bootstrap se može opisati i kao JavaScript, CSS te HTML biblioteka koja za cilj ima pojednostavljivanje razvoja web-stranica. Bootstrap pridodaje osnovne stilske definicije za elemente što rezultira jednakom izgledu tablica i elemenata u web preglednicima.

Bootstrap također dolazi sa određenim JavaScript komponentama koje ne zahtijevaju druge biblioteke kao što je na primjer *jQuery*. Te komponente pružaju dodatne elemente korisničkog sučelja kao što su: [7]

- dijaloški okviri,
- opisi alata,
- trake napretka,
- navigacijski padajući izbornici,
- karuseli.

Svaka Bootstrap komponenta sastoji se od HTML-ove strukture, CSS-ovih deklaracija i u nekim slučajevima popratnog JavaScript koda. Oni također proširuju funkcionalnost nekih postojećih elemenata sučelja, uključujući na primjer funkciju automatskog dovršavanja za polja za unos. Najnaglašenije komponente Bootstrapa su komponente izgleda jer imaju svoj utjecaj kroz cijelu web stranicu. Osnovna komponenta izgleda naziva se *container*, budući da je svaki drugi element na stranici smješten u njega. Programeri mogu birati između spremnika fiksne širine i spremnika fluidne širine. Dok potonji uvijek ispunjava širinu web stranice, prvi koristi jednu od pet unaprijed definiranih fiksnih širina, ovisno o veličini zaslona koji prikazuje stranicu:

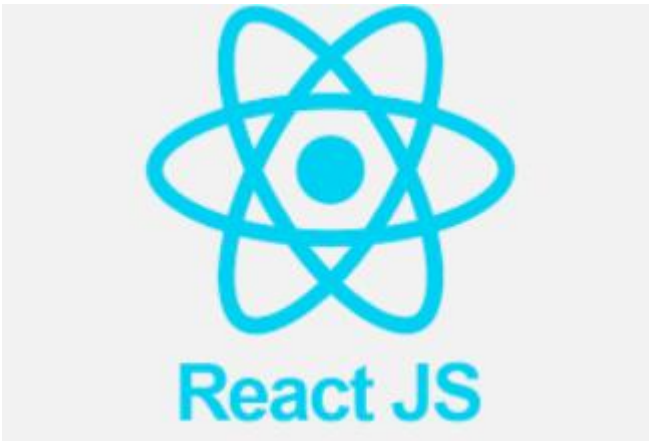
- Manje od 576 piksela

- 576–768 piksela
- 768–992 piksela
- 992–1200 piksela
- Više od 1200 piksela

Nakon što je *container* postavljen, druge komponente Bootstrap-a implementiraju CSS Flexbox raspored kroz definiranje redaka i stupaca. Prethodno kompajlirana verzija Bootstrapa dostupna je u obliku jedne CSS datoteke i tri JavaScript datoteke koje se mogu odmah dodati bilo kojem projektu. Međutim, sirovi oblik Bootstrapa omogućuje razvojnim programerima provedbu daljnje prilagodbe i optimizacije veličine. Ovaj neobrađeni oblik je modularan, što znači da programer može ukloniti nepotrebne komponente, primijeniti temu i modificirati neprevedene Sass datoteke [7].

3.7. React.js

React koji se može prepoznati i kao React.js te ReactJS opisuje se kao besplatna frontend biblioteka otvorenog koga koja služi za izgradnju korisničkih sučelja koja se temelje na UI komponentama. React.js podržava Meta odnosno nekadašnji Facebook te zasebni programeri te kompanije. React.js se koristi kao baza za mobilne i poslužiteljske aplikacije. Suština React.js-a jest da se bavi upravljanjem stanja te renderiranjem stanja. Prilikom stvaranja React.js aplikacija potrebno je koristiti dodatne biblioteke koje će usmjeravati te određene funkcionalnosti od strane klijenta [15].



Slika 3.4. React.js logo

3.8. Node.js

Node.js se može predstaviti kao platforma otvorenog tipa te backend okruženje JavaScript-a koje radi na temelju *JavaScript Engine-a*. Node.js izvršava JavaScript kod izvan web-preglednika te je dizajniran za razvoj mrežnih aplikacija. Node.js pruža mogućnost korištenja JavaScript-a prilikom pisanja alata naredbenog retka te za pokretanje skripti na strani poslužitelja. Također na strani poslužitelja koristi za izrađivanje dinamičkog sadržaja nakon čega se web-stranica šalje na korisnikov preglednik. Node.js se temelji na strukturi koja je vođena događajima te sadrži velik broj ulazno izlaznih operacija. Također služi i za aplikacije u stvarnom vremenu. Node.js distribuiranim razvojnim projektom prethodno je upravljala Zaklada Node.js, a sada se spojila sa Zakladom JS kako bi formirala Zakladu *OpenJS,* što je omogućeno programom *Collaborative Projects* zaklade *Linux.* [13]

3.9. MySQL

MySQL definira se kao sustav s kojim se upravljaju relacijske baze podataka otvorenog koda. S vremenom se pokazao kao jedan od boljih načina skladištenja te pretrage iznimno velikih količina podataka. U kombinaciji s PHP-om, čini najpopularniji sustav za upravljanje bazama podataka. MySQL predstavlja program koji svojim ponašanjem podsjeća na server s više-userskim funkcijama. Svaki korisnik ima svoje određene ovlasti prilikom rada dok na bazi može biti nekoliko korisnika, svaki sa svojim ograničenjima. Cilj ovog načina jest smanjiti postotak pogrešaka. MySQL opisuje se kao program na koji se pristupa preko mreže uz pomoć korisničkih podataka. Na jednom serveru se može nalaziti više baza podataka a unutar jednog projekta na bazi se mogu nalaziti podaci iz nekoliko baza. Prava koja stječe korisnik mogu se razlikovati: [17]

- Kreiranje novih baza podataka,
- Pristup trenutnim bazama,
- Uređivanje postojećih baza i slično.

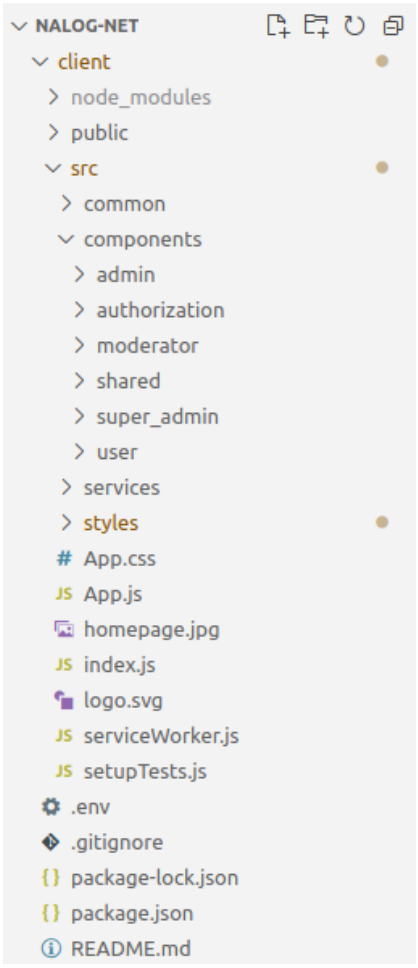
Važno je napomenuti kako MySQL kao program sadrži mape: [17]

- Bin – server i programi
- Lib – biblioteka funkcija
- Scripts – napisane skripte koje obavljaju određene poslove
- Share – mape s porukama o pogreškama
- Include – header

4. IZRADA APLIKACIJE

U ovom poglavlju detaljno je objašnjena izrada web aplikacije počevši sa izradom baze podataka u koju se spremaju svi podatci koje aplikacija koristi. Korisnicima je omogućena registracija i prijava u aplikaciju. Admin dodjeljuje registriranim korisnicima uloge unutar poduzeća. Moguće uloge su radnik, voditelj jedinice i direktor. U zadnja dva poglavlja objasnjen je tijek putnog naloga od radnika do direktora.

Programski kod je podijeljen na dvije cjeline klijent i server. Klijent predstavlja korisničko sučelje i prevladava HTML, Sass i Javascript, tj. Biblioteka React.js. Server predstavlja poslužiteljski dio aplikacije koji je zadužen za interakciju s bazom podataka. Pisan je u Node js-u. Strukture klijenta i servera se nalaze na slikama 4.1 i 4.2.



Slika 4.1. Struktura klijenta

Slika 4.2. Struktura servera

4.1. Izrada baze podataka

Za pohranu podataka koji se obrađuju unutar aplikacije je potrebna baza podataka. Za prikaz baze podataka korišteno je korisničko sučelje DBeaver. Za rad sa bazom podataka potrebno je pokrenuti MySQL server na lokalnom računalu. Unutar DBeavera se kreira baza podataka sa proizvoljnim imenom. Podatci koji su potrebni za spajanje na bazu podataka su ime poslužitelja localhost jer je baza lokalna, ime i lozinka korisnika te naziv baze podataka. Podatci su dostupni u datoteci db.config te su prikazani na slici 4.3. Na slici 4.4 prikazano je dohvaćanje podataka sa slike 4.3 te spajanje aplikacije na bazu podataka koristeći Node.js biblioteku sequelize. Sequelize pojednostavljuje izradu i upravljanje tablica u bazi podataka. Sequelize se može koristiti sa drugim tipovima baza (SQLite, PostgreSQl, Microsoft SQL, itd.) a sintaksa ostaje ista sto korisnicima dosta olakšava uporabu te ne moraju učiti razne sintakse za upravljanje bazama podataka.

server > app > config > JS db.config.js > ...

```
module.exports = {
  HOST: "localhost",
  USER: "user",
  PASSWORD: "password",
  DB: "testdb2",
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
};
```

Slika 4.3. Podatci za spajanje na bazu podataka

server > app > models > JS index.js > ...

```
const config = require("../config/db.config.js");
const Sequelize = require("sequelize");
const sequelize = new Sequelize(config.DB, config.USER, config.PASSWORD, {
  host: config.HOST,
  dialect: config.dialect,
  operatorsAliases: false,
  dialectOptions: {
    multipleStatements: true,
  },
  pool: {
    max: config.pool.max,
    min: config.pool.min,
    acquire: config.pool.acquire,
    idle: config.pool.idle,
  },
});
```

Slika 4.4. Spajanje na bazu podataka

U bazi podataka su stvorene sljedeće tablice: korisnici, uloge, putni nalozi, države sa dnevnicama te tablica koja povezuje korisnika sa ulogom preko identifikacijskog kljuca. Tablice se stvaraju tako da se pomoću sequelize-a definiraju modeli koji u sebi sadržavaju podatke određenog tipa koji predstavljaju stupce u tablici. Primjer izrade modela za korisnika koji sadrži podatke: korisničko ime, mail, lozinku i ime nalazi se na slici 4.5.

server > app > models > JS user.model.js > ...

```
1   module.exports = (sequelize, Sequelize) => {
2     const User = sequelize.define("users", {
3       username: {
4         type: Sequelize.STRING,
5       },
6       email: {
7         type: Sequelize.STRING,
8       },
9       password: {
10        type: Sequelize.STRING,
11      },
12      name: {
13        type: Sequelize.STRING,
14      },
15    });
16
17    return User;
18  };
```

Slika 4.5. Model korisnika

Izrađeni modeli se dohvaćaju iz datoteka i spremaju u konstantu db koja predstavlja sve modele. Na slici 4.6. prikazano je spremanje modela korisnika u konstantu db kao primjer.

server > app > models > JS index.js > ...

```
21  const db = {};
22  db.Sequelize = Sequelize;
23  db.sequelize = sequelize;
24  db.user = require("../models/user.model.js")(sequelize, Sequelize);
```

Slika 4.6. Dodavanje modela

Nadalje, potrebno je još konstantu db povezati s bazom podataka, to se radi na način da se pozove funkcija db.sequelize.sync(); koja služi za sinkronizaciju baze podataka s modelima unutar konstante db. Funkcija će kreirati tablice ako one ne postoje. Ostale tablice su izrađene po uzoru na tablicu korisnika.

4.2. Dodavanje korisnika

JSON Web Token (JWT) se koristi za autentifikaciju i autorizaciju korisnika. Autentifikacija predstavlja proces utvrđivanja identiteta korisnika, a autorizacija definiranje nivoa privilegije. Token se sprema na klijentskoj strani u local storage. JWT funkcijama verify() i sign() je potreban tajni ključ za dekodiranje i kodiranje tokena. Tajni ključ je definiran u datoteci auth.config.js.

4.2.1. Registracija korisnika

Uspješnost registracije novog korisnika se provjerava u među sloju pomoću dvije funkcije. Prva funkcija provjerava postoji li već u bazi podataka korisnik sa istim korisničkim imenom ili mailom dok druga provjerava postoji li uloga korisnika u zahtjevu za izradu računa. Funkcija za provjeru korisničkog imena i lozinke se koristi sa ugrađenom funkcijom Sequelizea findOne(). FindOne() funkcija provjera postoji li korisnik u bazi podataka na osnovu zadanih uvjeta. Na slici 4.7. nalazi se funkcija koja provjerava postoji li korisničko ime već u bazi podataka. Na identičan način se provjerava i za email.

server > app > middleware > JS verifySignUp.js > ...

```
9       where: {
10        username: req.body.username,
11      },
12    }).then((user) => {
13      if (user) {
14        res.status(400).send({
15          message: "Failed! Username is already in use!",
16        });
17        return;
18      }
```

Slika 4.7. Funkcija za provjeru korisničkog imena u bazi podataka

Registracija korisnika se odvija unutar kontrolera auth.controller.js u funkciji signup gdje se korisnik kreira pomoću sequelize funkcije create(). Funkciji se predaju uneseni podatci korisničko ime, email, kriptirana lozinka pomoću funkcije bcrypt.hashSync i id ključ uloge korisnika. Korisniku se postavlja najniža razina uloge user pod brojem 1. Signup() funkcija se nalazi na slici 4.8.

server > app > controllers > JS auth.controller.js > ...

```
11  exports.signup = (req, res) => {
12    // Save User to Database
13    User.create({
14      username: req.body.username,
15      email: req.body.email,
16      password: bcrypt.hashSync(req.body.password, 8),
17      name: req.body.name,
18    })
19      .then((user) => {
20        if (req.body.roles) {
21          Role.findAll({
22            where: {
23              name: {
24                [Op.or]: req.body.roles,
25              },
26            },
27          }).then((roles) => {
28            user.setRoles(roles).then(() => {
29              res.send({ message: "User registered successfully!" });
30            });
31          });
32        } else {
33          // user role = 1
34          user.setRoles([1]).then(() => {
35            res.send({ message: "User registered successfully!" });
36          });
37        }
38      })
39      .catch((err) => {
40        res.status(500).send({ message: err.message });
41      });
42  };
```

Slika 4.8. Singup funkcija

Korisničko sučelje za registraciju se nalazi u komponenti Register.js unutar React routera pod putanjom "/register". React router uvjetno prikazuje određene komponente za prikaz, ovisno u putanji koja se koristi u URL-u. Prilikom učitavanja stranice prvo se provjerava postoji li trenutno prijavljeni korisnik, ako postoji stranica se automatski preusmjerava na stranicu profila korisnika, isto preusmjeravanje se koristi i na stranici za prijavu korisnika. Za automatsko preusmjeravanje zadužena je React router komponenta Redirect. Postoji li prijavljen korisnik se provjerava na način da se vidi postoji li token u local storage-u. Uvjetovanje za preusmjeravanje nalazi se na slici 4.9.

client > src > components > authorization > JS register.js > Register

```
106  const user = AuthService.getCurrentUser();
107  if (user) {
108    return <Redirect to={"/profile"} />;
109  }
```

Slika 4.9. Funkcija za automatsko preusmjeravanje

Na stranici se nalazi forma za registraciju te je korisniku omogućen unos imena i prezimena, korisničkog imena, maila i lozinke. Svaki unos se sastoji od React komponente Input koja omogućava validaciju unosa podataka. Atributi koji se predaju unutar Inputa su: tip podatka (npr. Tekst, broj, mail), ime, klasa, vrijednost koja se pamti u stanju, funkcije koja se poziva svaki puta nakon promjene unosa da ažurira vrijednost u stanju te funkcije koje služe za validaciju. Primjer komponente input za ime i prezime nalazi se na sljedećoj slici.

client > src > components > authorization > JS register.js > Register

```
123                 <Input
124                   type="text"
125                   className="loginInput"
126                   name="name"
127                   placeholder="Ime i prezime"
128                   value={name}
129                   onChange={onChangeName}
130                   validations={[required, vname]}
131                 />
```

Slika 4.10. Komponenta Input

Primjer korištenja stanja i spremanja nove vrijednosti unosa u stanje se nalazi na slici 4.11.

client > src > components > authorization > JS register.js > R

```
71  const [name, setName] = useState("");
72  const onChangeName = (e) => {
73    setName(e.target.value);
74  };
```

Slika 4.11. useState() stanje u Reactu

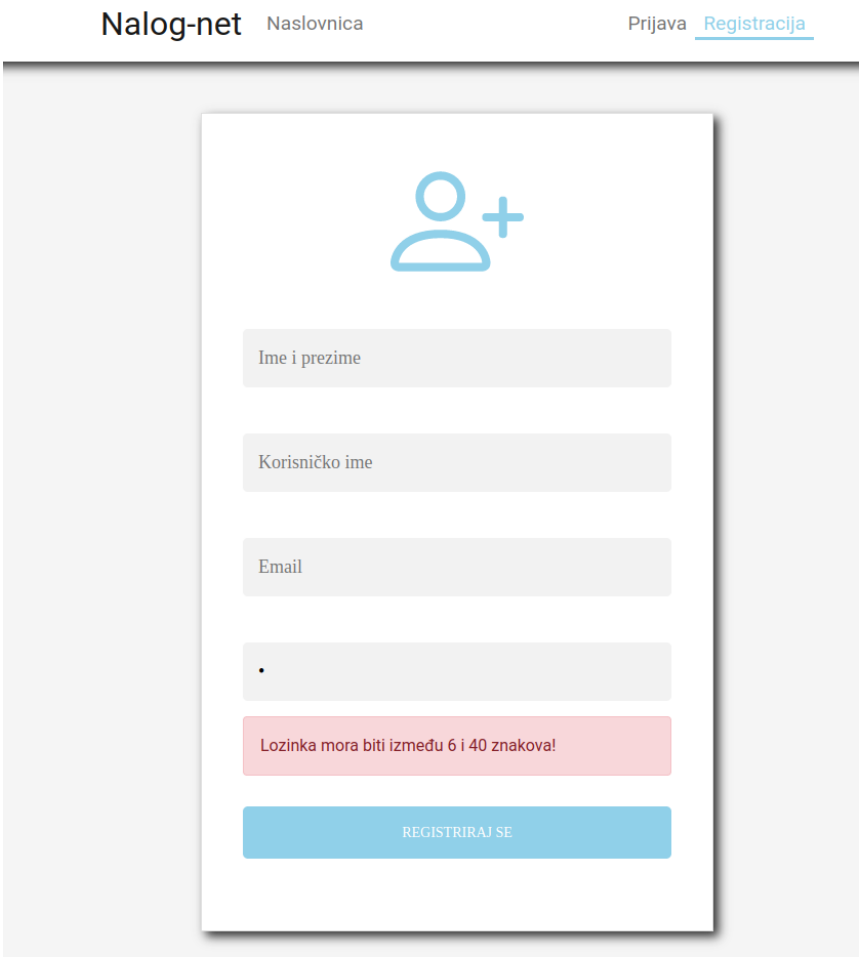
Funkcija koja izvršava validaciju korisničkog imena se nalazi na slici 4.12. te provjerava duljinu unosa koja mora biti između 3 i 20.

client > src > components > authorization > JS register.js > vname

```
31  const vusername = (value) => {
32    if (value.length < 3 || value.length > 20) {
33      return (
34        <div className="alert alert-danger" role="alert">
35          Korisničko ime mora biti između 3 i 20 znakova!
36        </div>
37      );
38    }
39  };
```

Slika 4.12. Validacija unosa

U slučaju neispravnog unosa podataka u trenutku uklanjanja fokusa na ekranu će se ispod unosa pojaviti poruka koja ukazuje na neispravan unos. Izgled stranice za registraciju se nalazi na slici 4.13. i demonstrira neispravan unos lozinke.



Slika 4.13. Izgled registracije

4.2.2. Prijava korisnika

Prijava korisnika na serverskoj strani se odvija u istom dokumentu auth.controller.js kao i registracija samo pod nazivom funkcije signin(). Unutar funkcije se prvo provjerava postoji i uopće uneseno korisničko ime u bazi podataka te ako postoji onda se provjerava točnost unesene lozinke sa spremljenom kriptiranom lozinkom iz baze podataka preko bcrypt.compareSync(). Ako je lozinka točna kreira se JWT token od korisnikovog id-ja i tajnog ključa te se dohvaća uloga korisnika iz baze podataka. Ako je sve uspješno izvršeno kao odziv statusa 200 vraćaju se svi podatci o korisniku, token i ulogu korisnika.

Na klijentskoj strani nakon potvrde forme poziva se preko axiosa POST zahtjev sa korisničkim imenom i lozinkom. U slučaju uspješne prijave. Svi podatci o korisniku iz odziva spremaju se u local storage. Axios služi za pojednostavljeno stvaranje HTTP zahtjeva prema serverskom poslužitelju. Prikaz korištenja axiosa te spremanje korisnika u local storage se nalazi na slici 4.14.

client > src > services > auth.service.js > AuthService

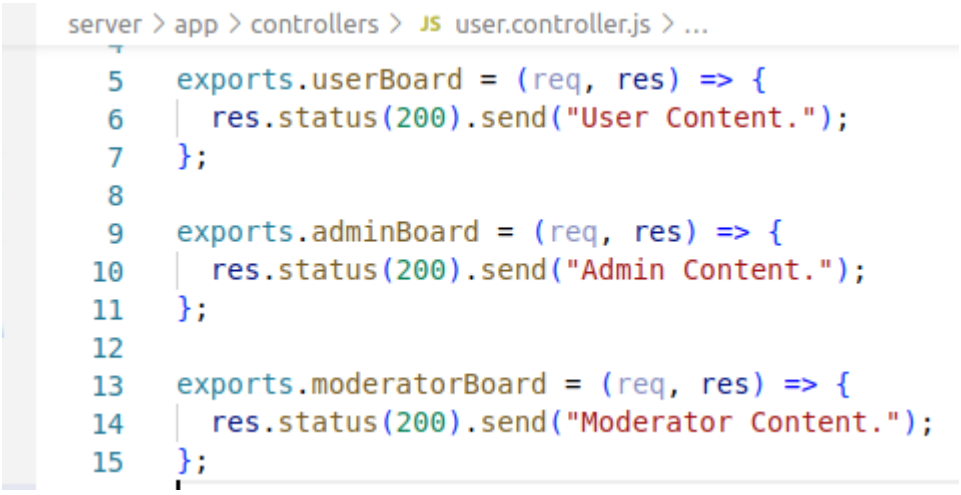
```
1  import axios from "axios";
2  const API_URL = "http://localhost:8080/api/auth/";
3  class AuthService {
4    login(username, password) {
5      return axios
6        .post(API_URL + "signin", {
7          username,
8          password,
9        })
10       .then((response) => {
11         if (response.data.accessToken) {
12           localStorage.setItem("user", JSON.stringify(response.data));
13         }
14
15         return response.data;
16       });
17   }
```

Slika 4.14. Korištenje axiosa

Nakon uspješne prijave korisnika aplikacija preusmjerava na njegov profil. Na profilu se prikazuju osnovni podatci o korisniku koji se nalaze na slici 4.15.

Slika 4.15. Izgled profila

Sadržaj navigacijske trake ovisi o tome postoji li prijavljen korisnik. Ako korisnik nije prijavljen u navigacijskoj traci se nalaze veze na naslovnicu, prijavu i registraciju. Ako je korisnik prijavljen u navigacijskoj traci se nalaze veze na navigaciju, profil, odjavu te sučelje vezano uz ulogu korisnika. Odjava se vrši tako da se korisnik ukloni iz local storage-a. Nakon prijave korisnika na serverskoj strani kreirane su putanje "/api/test/(user, mod ili admin)" gdje user, mod ili admin predstavljaju razinu uloge. Na putanjama se provjerava ispravnost tokena te ima li korisnik pristup sadržaju te ako je ispravno vraća se odziv statusa 200 sa sadržajem na slici 4.16.



Slika 4.16. Sadržaj odaziva

Na klijentskoj strani postoje 3 stranice koje predstavljaju sučelja za različite uloge. Svaka stranica preko axiosa šalje zahtjev na serversku putanju "api/test/(uloga)" te provjerava pomoću statusa odziva ima li korisnik pristup stranici, u slučaju nedozvoljenog pristupa korisnika aplikacija preusmjerava na početnu stranicu. Dodatan način sigurnosti je još uvjetno prikazivanje sadržaja tako što se tekst odziva provjerava. Npr. Ako se tekst odziva za admina ne poklapa sa "Admin Content." neće se ništa prikazivati na ekranu.

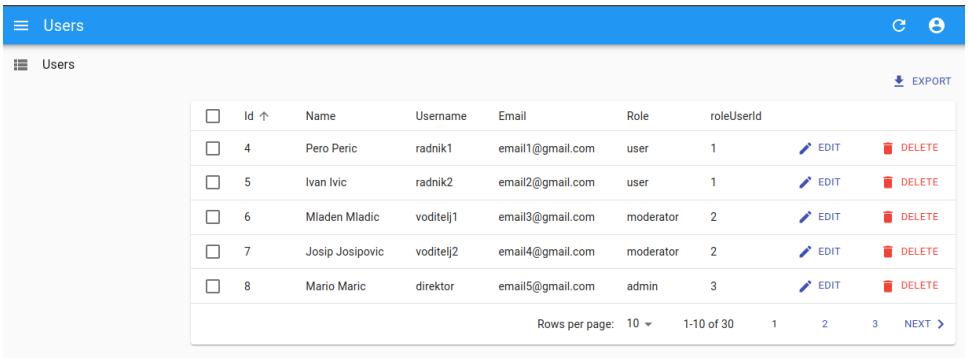
4.2.3. Dodavanje uloga korisnicima

Dodavanje uloga korisnika se dodaje na klijentskoj putanji "/superadmin" gdje je za autentifikaciju potrebno unijeti admin i kao korisničko ime i lozinku. Za uređivanje uloga se koristi react admin. React admin je biblioteka koja omogućava korisniku prikaz, uređivanje te brisanje podataka, u ovom slučaju korisnika. Učitana komponenta Admin iz biblioteke react-admin prima argumente dataProvider koji predstavlja URL serverskog poslužitelja te authProvider koji služi za autentifikaciju admina prilikom prijavljivanja. Unutar admina se slažu komponente Resource koje predstavljaju podatke za obradu, u ovom slučaju postoji samo jedna komponenta Resource koja prima argumente name, list i edit. Name predstavlja putanju gdje se nalaze podatci o svim korisnicima na serverskoj strani. List predstavlja komponentu za prikaz svih korisnika te edit predstavlja komponentu za uređivanje korisnika. Pozivanje komponente admin se nalazi na slici 4.17.

```
client > src > components > super_admin > JS SuperAdmin.js > ...
27      <Admin
28        dataProvider={simpleRestProvider("http://localhost:8080")}
29        authProvider={authProvider}
30      >
31        <Resource name="users" list={UserList} edit={UserEdit} />
32      </Admin>
```

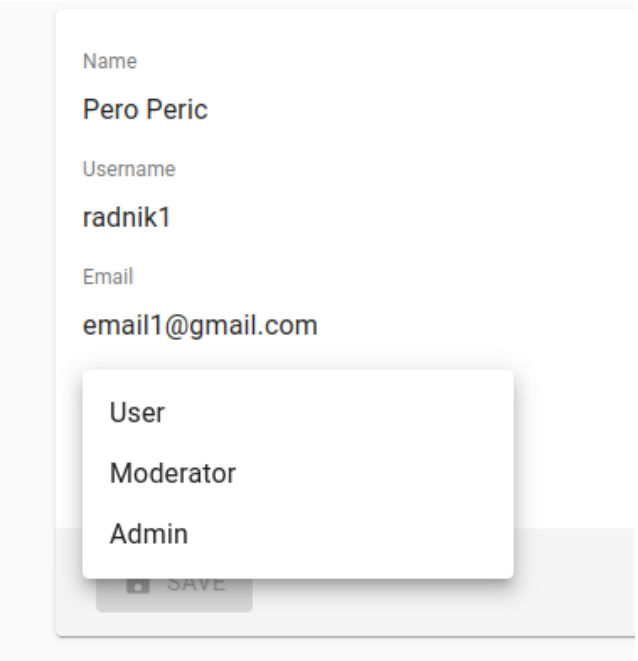
Slika 4.17. Komponenta react admin

Prikaz korisnika nalazi se na slici 4.18. Adminu je omogućen i izvoz korisnika u csv formatu.



Slika 4.18. React admin sučelje

Na slici 4.19. je demonstriran padajući izbornik pomoću kojeg admin mijenja uloge korisniku.



Slika 4.19. Uređivanje uloge korisnika

4.3. Tijek izrade putnog naloga

Putni nalog kreiraju korisnici koji imaju razinu uloge *user*. Nalog se sastoji od naslova, opisa, imena osobe koja je popunila putni nalog, ime voditelja jedinice, zemlju i mjesto putovanja, dnevnicu, putne troškove, ostale troškove, ukupne troškove, datum početka, dodatno i status. Pomoću statusa se određuje lokacija putnog naloga i daju se ovlasti određenim osobama vezanim uz nalog. Mogući statusi se mogu vidjeti na slici 4.20. a detaljnije su objašnjeni u nastavku:

1) Putni nalog je kod djelatnika na uređivanju.
2) Putni nalog je kod voditelja jedinice gdje je njemu omogućeno prihvatiti, odbiti ili zatražiti izmjene naloga.
3) Putni nalog je kod direktora koji uz sve ovlasti moderatora može sam uređivati putni nalog.
4) - Nema.
5) Putni nalog je odbijen te se nalazi u arhivi.
6) Putni nalog je prihvaćen od strane direktora.
7) Djelatnik je popunio troškove izvješća.
8) Putni nalog je isplaćen djelatniku te se prosljeđuje u arhivu.

```
const selectOptions = {
  1: "Kod djelatnika",
  2: "Kod voditelja jedinice",
  3: "Kod direktora",
  5: "Arhiva - odbijeni nalozi",
  6: "Odobreni nalozi",
  7: "Čekanje naplate",
  8: "Arhiva - plaćeni nalozi",
};
```

Slika 4.20. Statusi putnog naloga

4.3.1. Izrada putnog naloga

Na poslužiteljskom dijelu su napisani kontroleri i rutiranje za obradu naloga putem REST protokola. REST predstavlja oblik komunikacije između klijenta i servera korištenjem mrežnih resursa pomoću HTTP protokola. Četiri najpoznatije metode su POST, PUT, GET, DELETE. Podatci se šalju u JSON formatu.

Sučelje za djelatnika se sastoji od 4 komponente, komponenta za kreiranje novih naloga, komponenta za prikaz naloga, komponenta za uređivanje naloga te komponenta za popunjavanje troškova izvješća.

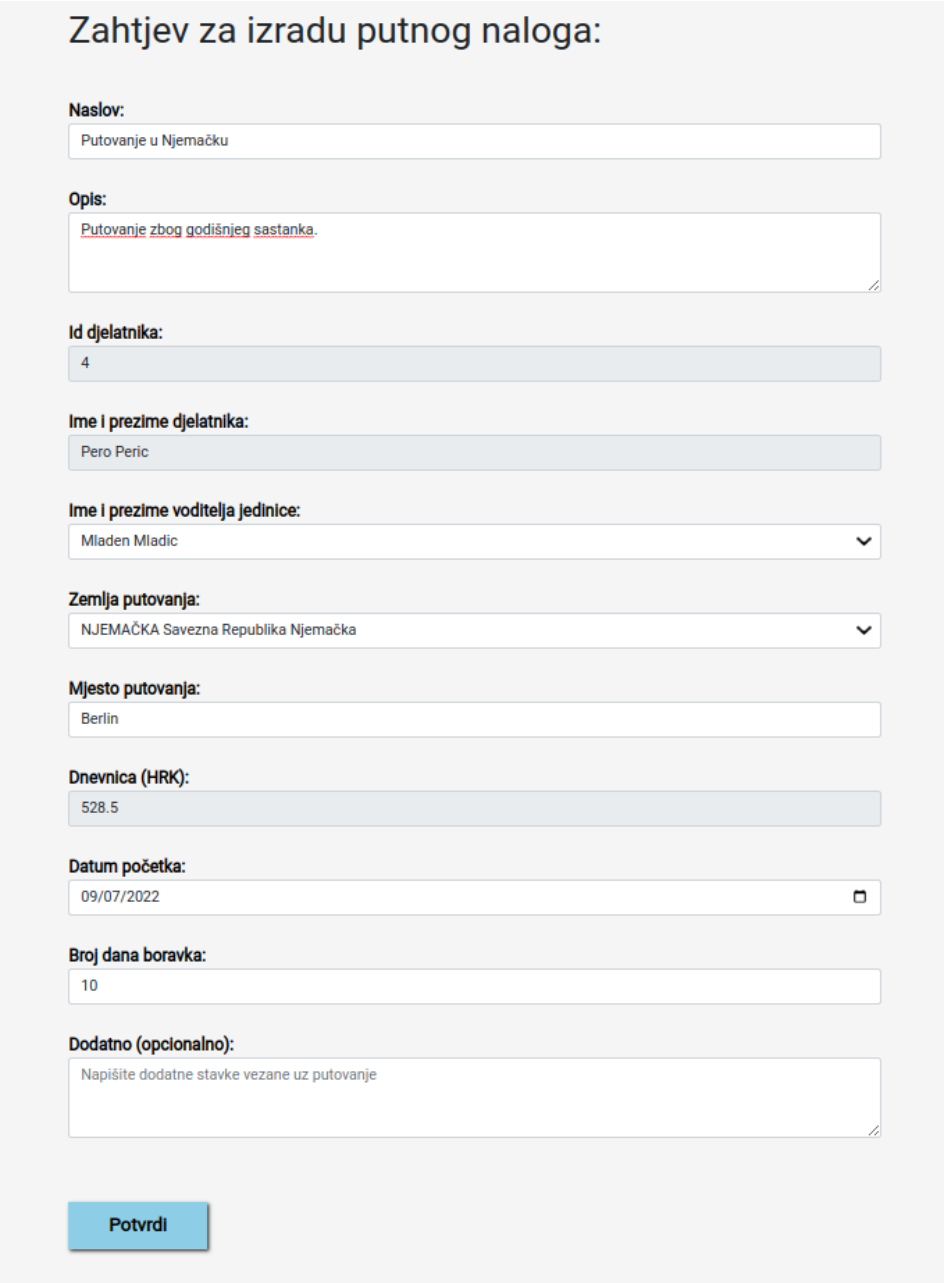
Nakon što se djelatnik logira ponuđena mu je izrada putnog naloga. Forma za izradu putnog naloga je sastavljena po istoj logici kao i forma za registriranje. Na isti način su korišteni Inputi, validacije polja unosa i spremanje unosa u stanja koristeći useState(). Dodatno su korištene dvije React-validation komponente Select koje prikazuju padajući izbornih svih voditelja jedinica unutar djelatnosti i moguće zemlje putovanja koje su povezane sa iznosom dnevnica. Na slici 4.21. je prikazan kod za selekcijsko polje te funkcija .map() pomoću koje se je popunilo polje zemljama. Map() funkcija distribuira (mapira) funkciju po elementima liste. Unutar stanja countries su dohvaćeni podatci o državama i dnevnicama putem axiosa i GET metode.

client > src > components > user > JS order.js > Order

```
289   <Select
290       className="form-control dropdown-toggle"
291       placeholder="odaberite zemlju putovanja"
292       validations={[required]}
293       value={countryDestination || ""}
294       onChange={onChangecountryDestination}
295   >
296       <option></option>
297       {countries.map((eachCountry, i) => (
298         <option key={i}>{eachCountry.naziv}</option>
299       ))}
300   </Select>
```

Slika 4.21. Komponenta select

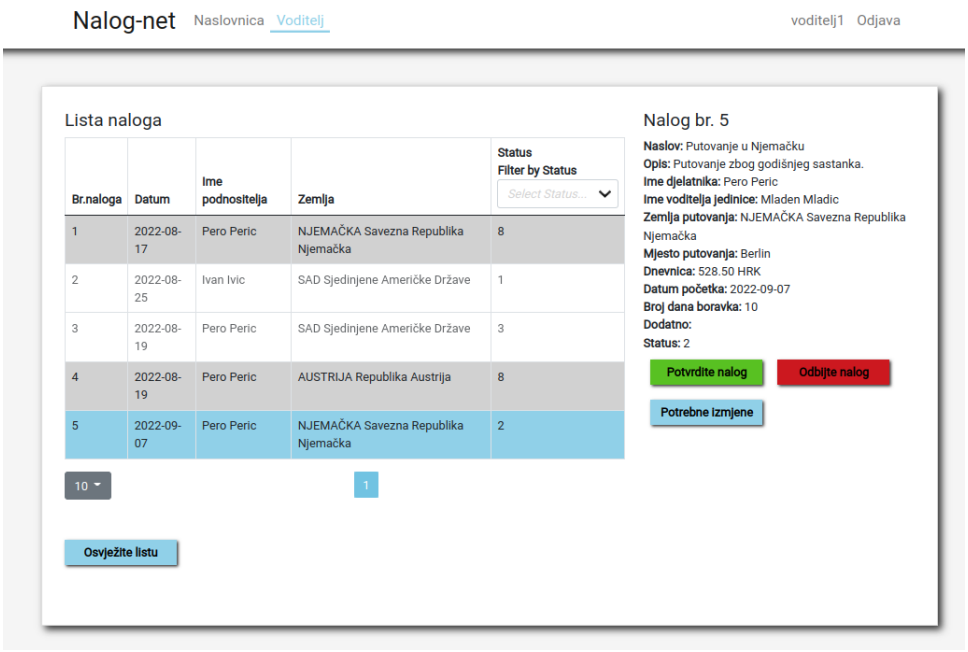
Izgled forme za izradu putnog naloga se nalazi na slici 4.22. Uspješno izrađeni nalog se prosljeđuje kod izabranog voditelja jedinice. Nalog je vidljiv jedino djelatniku, voditelju jedinice koji se nalazi na nalogu i direktoru.



Slika 4.22. Forma za izradu putnog naloga.

4.3.2. Putni nalog kod voditelja jedinice

Voditelj jedinice u svom sučelju ima samo jednu komponentu a to je komponenta za prikaz putnih naloga na kojima je on izabran. Prikaz sučelja se nalazi na slici 4.23.



Slika 4.23. Prikaz putnih naloga

Za prikaz naloga korištena je komponenta BootstrapTable sa pomoćnim komponentama paginationFactory i filterFactory. Komponente su dio react-bootstrap-table-next biblioteke. FilterFactory se koristi za filtriranje naloga po statusu kroz padajući izbornik. Padajući izborni je omogućen tako da se pri inicijaliziranju stupca kod definiranja imena polja u objektu i oznake za prikaz definira i filter sa opcijama. Definiranje stupca sa filterom se nalazi na slici 4.24.

```
{
  dataField: "roleEditId",
  text: "Status",
  filter: selectFilter({
    options: selectOptions,
  }),
```

Slika 4.24. Definiranje stupca u tablici.

Korisnicima je omogućen detaljan pregled naloga nakon što se klikne na njega u tablici. Voditelju jedinice se prikazuju tri gumba za obradu naloga ako status ima vrijednost dva, u suprotnom će na mjestu predviđenom za gumbe biti natpis da se putni nalog ne može uređivati. Pritiskom na neki

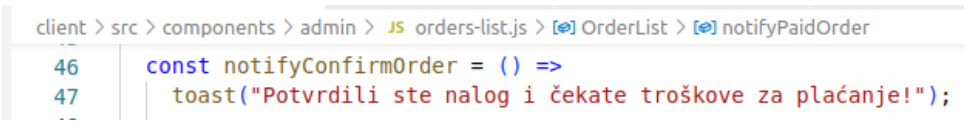
od tri gumba poziva se axios funkcija sa put metodom za ažuriranje naloga u kojoj se ažurira proslijeđeni parametar budućeg statusa te se tako zapravo i kontrolira pozicija naloga.

U slučaju da voditelj zatraži izmjene od djelatnika na nalogu se mijenja status u jedan te se djelatniku omogućuje uređivanje naloga. Nakon što djelatnik uredi nalog ponovno se vraća status u broj dva. Preostale dvije opcije su pretvaranje statusa u broj tri ili pet odnosno potvrditi nalog te ga tako poslati kod direktora ili odbiti nalog i tako ga proslijediti u arhivu.

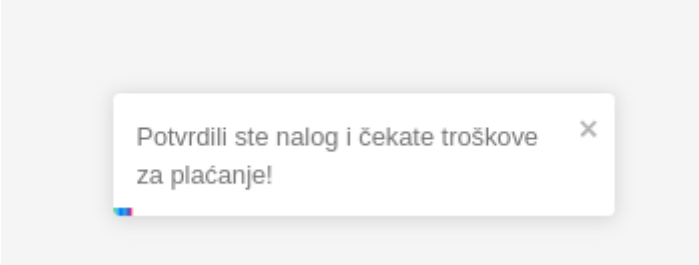
4.3.3. Putni nalog kod direktora

Direktor ima uvid u sve putne naloge unutar cijelog poduzeća. Nabrojane opcije voditelja jedinice vrijede i za direktora. Dodatno direktor može uređivati naloge klikom na četvrti gumb te se otvara forma identična onoj za kreiranje naloga. Potvrđeni nalog od strane direktora dobiva status broj šest te se prosljeđuje nazad do djelatnika.

Prilikom izvođenja bilo koje akcije nad nalogom korisnicima se izbacuje toast poruka na dnu ekrana koja opisuje odrađenu akciju. Poruke su omogućene bibliotekom react-toastify. Za implementaciju poruka potrebno je postaviti komponentu ToastContainer u roditeljsku komponentu App da obavije sve komponente. U komponenti se postave atributi koji su generirani na službenoj stranici react-toastify-a. Prikazivanje poruke se izvodi na način da se pozove funkcija toast koja prima string kao parametar. Primjer funkcije koja poziva poruku kada direktor potvrdi nalog se nalazi na slici 4.25, dok na slici 4.26. se nalazi izgled toast poruke.



Slika 4.25. Pozivanje toast poruke



Slika 4.26. Izgled toast poruke

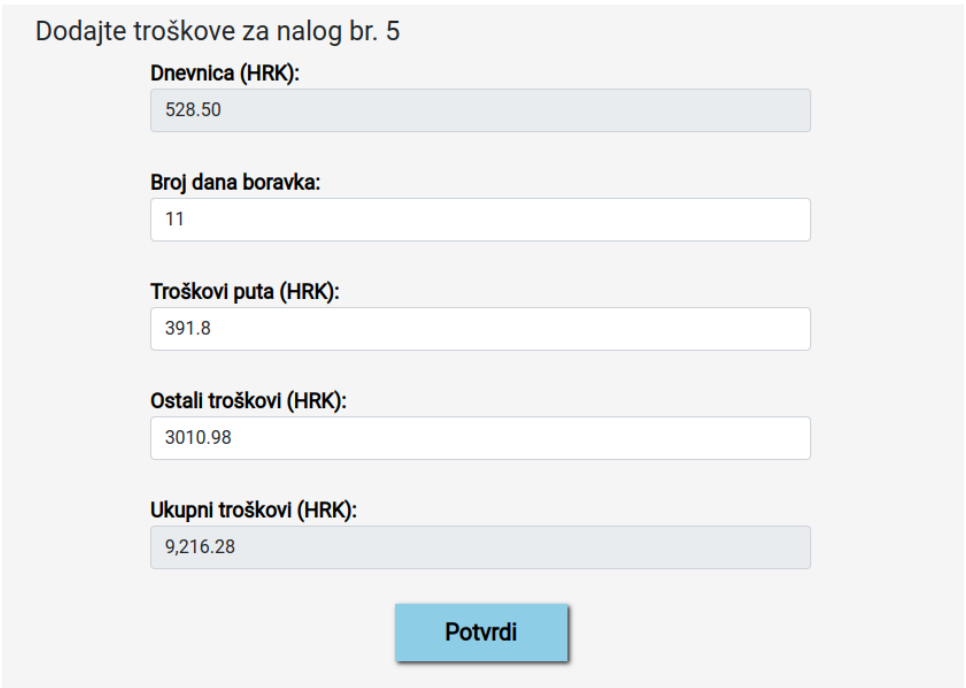
4.3.4. Popunjavanje i potvrda troškova

Nakon što status putnog naloga dobije vrijednost šest, djelatniku je omogućeno da otvori izvještaj putnog naloga u kojem su prikazani dnevnica, broj dana, troškovi puta, ostali i ukupni troškovi. Dnevnica je nepromjenjiva za izabranu državu stoga ju korisnik ne može izmjenjivati. Nepromjenjivi input se postavlja tako da mu se doda atribut readOnly. Ukupni troškovi se računaju na način da se zbroje ostali troškovi, putni troškovi i umnožak broja dana sa dnevnicom. Sve cijene su izražene u kunama. Funkcija s kojim se zbrajaju svi troškovi se nalazi na slici 4.27., dok izgled popunjene forme za izvještaj se nalazi na slici 4.28.

client > src > components > user > JS edit-costs.js > EditCosts > notifyEditCosts

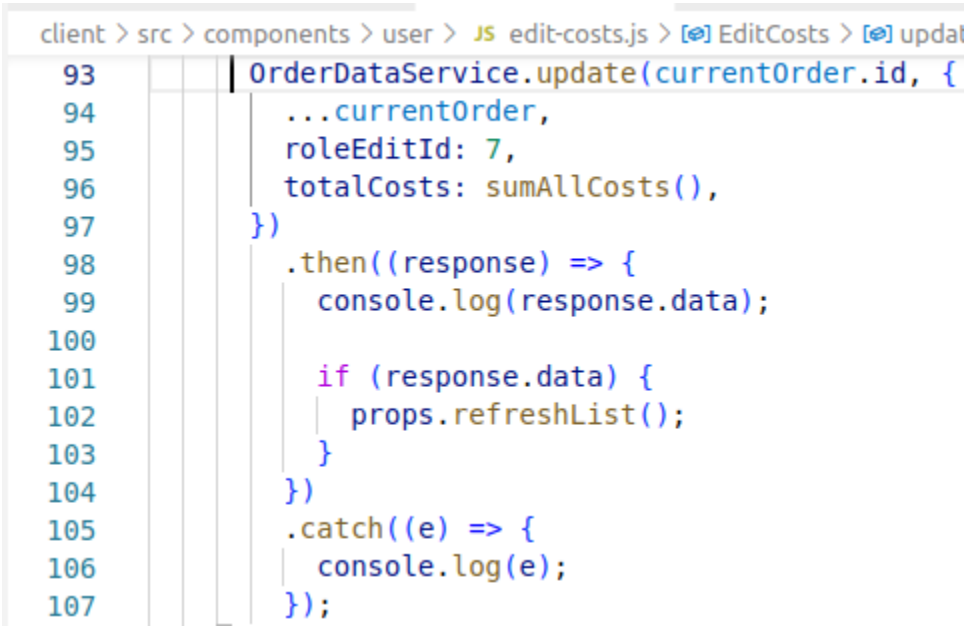
```
63   const sumAllCosts = () => {
64     let sum =
65       parseFloat(currentOrder.travelCosts || 0) +
66       parseFloat(currentOrder.otherCosts || 0) +
67       parseFloat(currentOrder.numberOfDays * currentOrder.salary);
68     return sum;
69   };
```

Slika 4.27. Funkcija za zbrajanje troškova



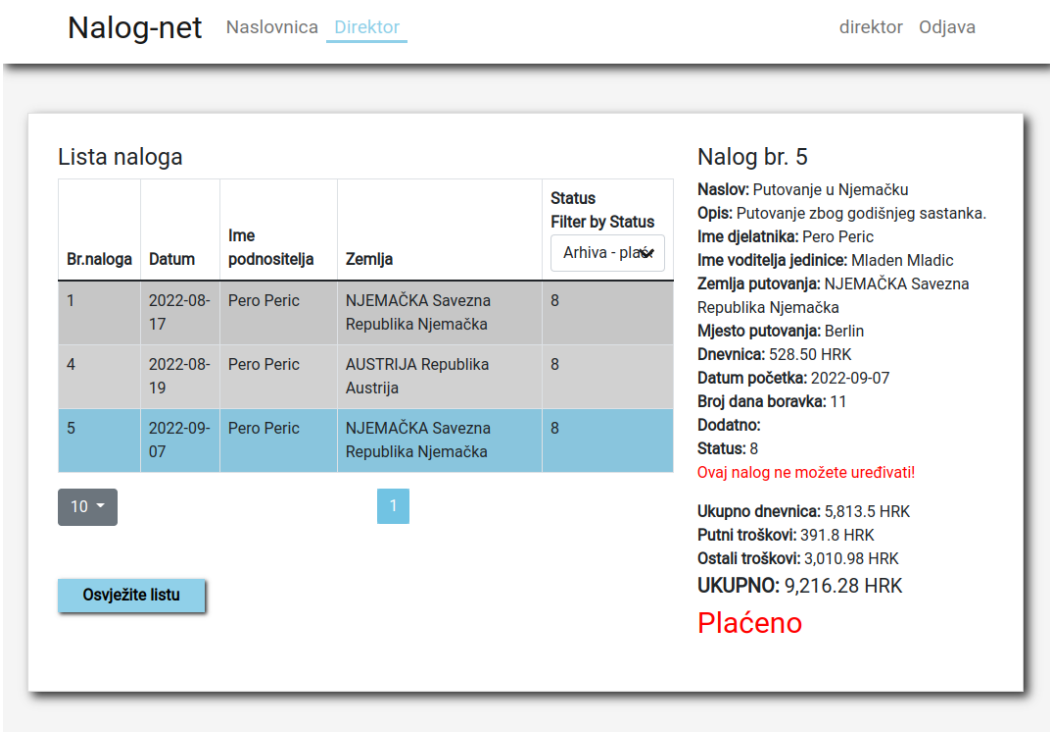
Slika 4.28. Prikaz forme za popunjavanje izvješća

Ažuriranje naloga u bazi podataka se izvršava tako da se pozove PUT metoda koja prima kao parametre ID naloga i podatke naloga. ...currentOrder predstavlja spread operator koji služi za raspršivanje objekta na elemente. Broj dana, ostali troškovi i troškovi puta se automatski prilikom unosa spremaju u objekt currentOrder pa ih nije bilo potrebno posebno postavljati. Dodatno se postavlja roleEditId (status) na broj sedam i za ukupne troškove poziva se funkcija sa slike 4.27. Funkcija update() sa slike 4.29. poziva axios PUT metodu za ažuriranje naloga.



Slika 4.29. Funkcija za ažuriranje naloga

Nakon što je djelatnik uspješno ispunio izvještaj za troškove na direktoru je ostalo da isplati novce djelatniku te u sučelju kada klikne na nalog potvrdi plaćanje. Na slici 4.30. su prikazani plaćeni nalozi koji se nalaze u arhivi te su oni obojani u tamniju sivu boju. Na nalogu desno se može vidjeti da se prikazuju troškovi te je crvenom bojom označeno da je nalog plaćen.



Slika 4.30. Arhivirani plaćeni nalozi

5. ZAKLJUČAK

U ovom diplomskom radu je prikazano kako izraditi i razviti web aplikaciju za evidenciju putnih naloga unutar nekog poduzeća. Aplikacija je jednostavna za korištenje te se može prilagoditi za različite potrebe. Moguće je prenamijeniti aplikaciju da radi sa različitim obrascima unutar poduzeća. Aplikaciju je moguće nadograditi tako da podržava više poduzeća istovremeno pod istom domenom. Učinkovitije je imati sve putne naloge poduzeća na jednoj internetskoj stranici nego ih imati na stotine papira. Lakše je skladištiti naloge u bazu podataka nego spremati papirnate verzije naloga. Web aplikacija je prvenstveno namijenjena za djelatnike unutar poduzeća koji koriste putne naloge.

U diplomskom radu opisane su zadaće frontend i backend dijela. Za izradu aplikacije korištena su softverska okruženja DBeaver i Visual Studio Code. Aplikacija je izrađena od HTML-a, React.js-a, Node.js-a, Sass-a i MySQL-a.

LITERATURA

[1] Castro, E. (2006). HTML, XHTML, and CSS: Visual QuickStart Guide. Pearson Education.

[2] DBeaver CE. https://apps.microsoft.com/store/detail/DBeaver-ce/9PNKDR50694P?hl=en-us&gl=us (pristupljeno 04.09.2022.)

[3] DBeaver Community, https://DBeaver.io/ (pristupljeno 01.09.2022.)

[4] Github – DBeaver. https://github.com/DBeaver/DBeaver (pristupljeno 03.09.2022.)

[5] Goodman, D. (2002). Dynamic HTML: The definitive reference: A comprehensive resource for HTML, CSS, DOM & JavaScript. " O'Reilly Media, Inc.".

[6] Gooma, Sustav za management putnih troškova, GOOMA | sustav za praćenje putnih troškova | Jednostavniji putni nalog (pristupljeno 13.07.2022.)

[7] What is Bootstrap: A Beginner's Guide, *ALEXANDRE OUELLETTE*, 27. prosinca 2021. https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/ (pristupljeno 10.09.2021)

[8] "Kako obračunati dnevnice i putne troškove?", *Aestus Group*, 19. listopada 2021. https://aestus.hr/kako-obracunati-dnevnice-i-putne-troskove/ (pristupljeno 13.07.2022.)

[9] "Kako, kako i tko koristi putni nalog", *Savjetnica udruga*, https://savjetnica-udruga.com/kako-kako-i-tko-koristi-putni-nalog/ (pristupljeno 13.07.2022.)

[10] Knjigovodstveni servis Rk mreža usluga d.o.o., Rk mreža usluga d.o.o. – Knjigovodstveni servis kakav želite! (rkmrezausluga.hr) (pristupljeno 13.07.2022.)

[11] Musciano, C., & Kennedy, B. (2002). HTML & XHTML: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.".

[12] "Najbolji putni nalozi. Bez premca." *Relago*, https://www.relago.hr/Apps/Putni (pristupljeno 13.07.2022.)

[13] Node.js. https://nodejs.org/en/about/ (pristupljeno 10.09.2022.)

[14] "Putni nalozi web", *Konto*, Putni nalozi, program za putne naloge | Konto d.o.o.Konto d.o.o. Požega (pristupljeno 13.07.2022.)

[15] React. https://reactjs.org (pristupljeno 10.09.2022.)

[16] Visual Studio Code, https://code.visualstudio.com (pristupljeno 01.09.2022.)

[17] Why MySQL?. https://www.mysql.com/why-mysql/ (pristupljeno 10.09.2022.)

[18] Wilton, P. (2004). Beginning JavaScript. John Wiley & Sons.

[19] Wirfs-Brock, A., & Eich, B. (2020). JavaScript: the first 20 years. Proceedings of the ACM on Programming Languages, 4(HOPL), 1-189.

SAŽETAK

Ovaj diplomski rad sastoji se od teorijskog dijela u kojemu je razrađena teorijska podloga koja je kasnije potrebna za razumijevanje praktičnog djela rada. Najprije su objašnjeni putni nalozi te njihova primjena. Nadalje su izložene korištene tehnologije te alati. Korištene tehnologije te alate predstavljaju Visual Studio Code, DBeaver, HTML, CSS te ostali. Svakoj tehnologiji te alatu predstavljena je uloga te svrha rada koja će u konačnici omogućiti razumijevanje rada web aplikacije koja je napravljena kao diplomski zadatak u praktičnom dijelu ovog diplomskog rada. U ovom diplomskom radu detaljno je opisan proces izrade web aplikacije za evidenciju putnih naloga. Opisan je način rada aplikacije i korištenje aplikacije. Korisniku je omogućen rad s bazom podataka te mu je omogućeno izrađivanje novih podataka te izmjenjivanje starih podataka. Za backend dio su korišteni Node.js i baza podataka MySQL dok je za frontend dio korišten React.js. Rezultat rada je izrađena potpuno funkcionalna aplikacija za evidenciju putnih naloga.

Ključne riječi: baza podataka, MySQL, Node js, putni nalog, React.js, web aplikacija

ABSTRACT

Title: **Web application for recording travel orders**

This graduation thesis consists of a theoretical part in which the theoretical background is elaborated, which is later necessary for understanding the practical part of the work. First, travel orders and their application were explained. Furthermore, the used technologies and tools are presented. Technologies and tools used are Visual Studio Code, DBeaver, HTML, CSS and others. The role and purpose of the work is presented to each technology and tool, which will ultimately enable the understanding of the work of the web application, which was created as a graduation assignment in the practical part of this graduation thesis. This thesis describes in detail the process of creating a web application for recording travel orders. The mode of operation of the application and the use of the application are described. The user is enabled to work with the database and is enabled to create new data and change old data. Node.js and the MySQL database were used for the backend part, while React.js was used for the frontend part. The result of the work is a fully functional application for recording travel orders.

Keywords: database, MySQL, Node.js, travel order, React.js, Web application