Izrada internet aplikacije za evidenciju posuđivanja

Doko, Petar

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

Sveučilišni studij

IZRADA INTERNET APLIKACIJE ZA EVIDENCIJU POSUĐIVANJA

Diplomski rad

Petar Doko

Osijek, 2019.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 22.09.2019.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Petar Doko
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 838 R, 27.09.2018.
OIB studenta:	49946515349
Mentor:	Doc.dr.sc. Ivan Aleksi
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Tomislav Matić
Član Povjerenstva:	Izv.prof.dr.sc. Tomislav Keser
Naslov diplomskog rada:	Izrada internet aplikacije za evidenciju posuđivanja
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom diplomskom radu potrebno je izraditi internet aplikaciju pomoću React programskog okruženja. Potrebno je pratiti koji su artikli posuđeni, a koji vraćeni. Stranica treba imati sadržaje koji uključuju slike, slanje e-mailova, popis posuđenih artikala itd.
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 mentora:	22.09.2019.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

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

IZJAVA O ORIGINALNOSTI RADA

Osijek, 30.09.2019.

Ime i prezime studenta:	Petar Doko
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 838 R, 27.09.2018.
Ephorus podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Izrada internet aplikacije za evidenciju posuđivanja**

izrađen pod vodstvom mentora Doc.dr.sc. Ivan Aleksi

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

Cilj ovoga diplomskog rada je izrada internet aplikacije kojom se evidentira posudba stvari. Za izradu internet aplikacije koristi se React JavaScript biblioteka koja služi za kreiranje korisničkog sučelja. Napravljena je aplikacija u kojoj imamo pregled o tome koje su osobe posudile stvari, koji su njihovi osobni podatci, vremenski period posudbe, sliku posuđene stvari te je li ta osoba vratila predmet ili nije. Tijekom izrade aplikacije korištena su znanja stečena tijekom obrazovanja na Fakultetu elektrotehnike računarstva i informacijskih tehnologija u Osijeku. Osim znanja stečenih na fakultetu, korišten je i online tečaj kako bi se ova aplikacija mogla što bolje izraditi u potpunosti. U daljnjem dijelu opisuju se korištene tehnologije, struktura aplikacije i kako se ona koristi.

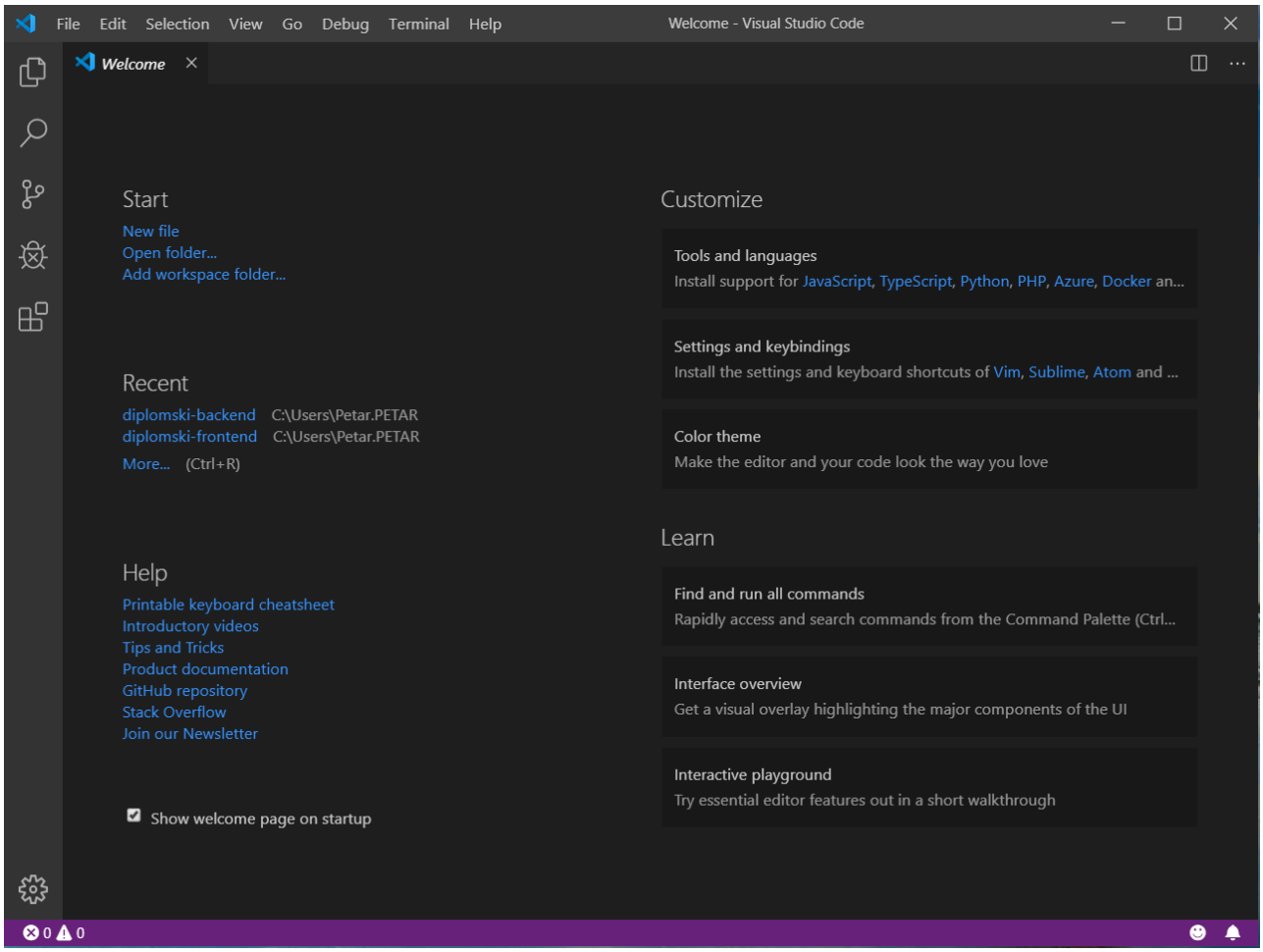
1.1 Zadatak diplomskog rada

Zadatak diplomskog rada je izrada internet aplikacije za evidenciju posuđivanja. Ideja aplikacije je da se vlasniku stvari omogući nadzor nad posuđenim stvarima. Ukoliko neka osoba želi posuditi stvar/i, unose se podatci o toj osobi (ime, prezime, OIB, broj mobitela), vremensko razdoblje posudbe te slika posuđenog artikla. Ukoliko osoba nije vratila posuđenu stvar do krajnjeg roka posudbe, šalje se e-mail kao podsjetnik da osoba koja je posudila nešto nije tu stvar vratila na vrijeme.

2. OPIS KORIŠTENIH TEHNOLOGIJA

2.1 Visual Studio Code

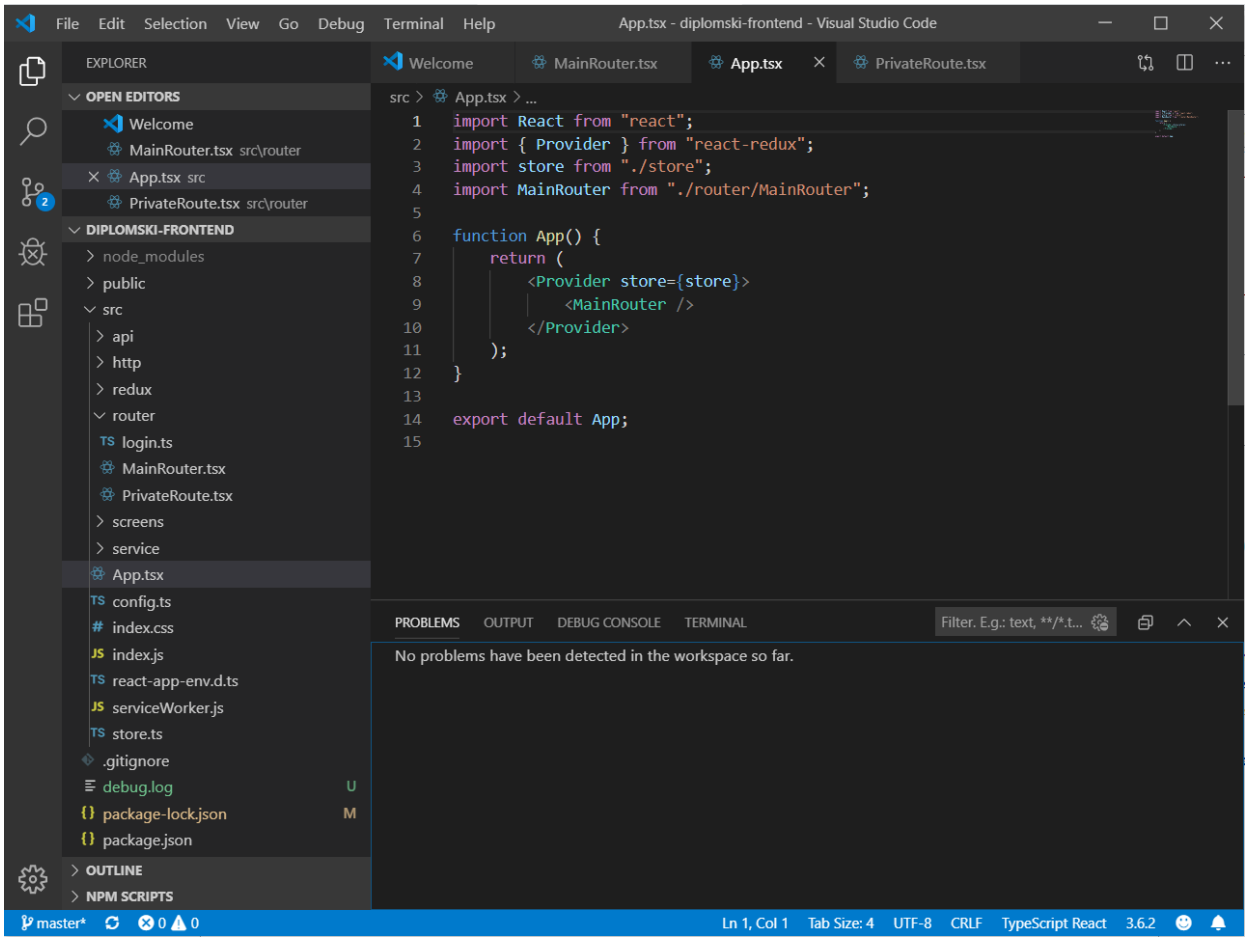
Visual Studio Code (Slika 2.1) je besplatni tekst editor razvijen od strane Microsofta za Windows, Linux i macOS operacijske sustave. Uključuje podršku za ispravak pogrešaka u kodu, posjeduje alate za verzioniranje koda (Git control), označavanje sintakse, inteligentno nadopunjavanje te preuređivanje koda. Visual Studio Code je prilagodljiv, može se promijeniti izgled teme, prečace tipki na tipkovnici, postavke te instalacija određenih proširanja koje pružaju dodatnu funkcionalnost. Visual Studio Code je brz i lagan, fleksibilan je u raznim domenama poput Java, JavaScript, Go, Node.js pa čak i C++. [1]



Slika 2.1. Izgled Visual Studio Code početne stranice

Programski kod Visual Studio Code-a temelji se na Electronu, frameworku koji se koristi za pokretanje Node.js aplikacija na desktop računalima. Prema provedenim anketama u 2019. godini, Visual Studio Code je proglašen najpopularnijim razvojnim alatom među programerima. Na slici

2.2 možemo vidjeti izgled Visual Studio Code okruženja.



Slika 2.2. Izgled Visual Studio Code okruženja

2.2 React

React je JavaScript biblioteka za izradu korisničkih sučelja napravljen od strane Facebooka. React se može koristiti kao osnova u razvoju web stranica ili mobilnih aplikacija. Optimalan je za dohvaćanje podataka koji se brzo mijenjaju i te promjene je potrebno uočiti i zabilježiti.

```
ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

Slika 2.3 Izgled jednostavnog "Hello, world" React programa

Primjer (Slika 2.3) predstavlja ispis naslova „Hello, world!“ na stranici. Render metoda se poziva svaki put kada se ažurira stanje komponente i to se odražava na korisničkom sučelju. React se sastoji od entiteta zvane komponente. Komponente se mogu renderirati u određeni element u

DOM-u (Document Object Model) pomoću biblioteke ReactDOM. Prilikom renderiranja komponente mogu im se proslijediti određena svojstva poznatija kao props (skraćeno od properties). Postoje dvije vrste komponenti, a to su funkcionalne komponente i komponente bazirane na klasama. Funkcionalne komponente su deklarirane sa funkcijom i kao rezultat vraćaju JavaScript XML (skraćeno JSX) zapis. Komponente bazirane na klasama su još poznate kao „stateful“ komponente, jer njihovo stanje može posjedovati određene vrijednosti u komponenti i te vrijednosti se mogu proslijediti drugim komponentama preko svojstava (props). Osim komponenti još jedna značajka je uporaba virtual DOM-a. Prilikom renderiranja, umjesto da se renderira cijela stranica, renderiraju se samo određeni dijelovi stranice tako što React razdvaja određene strukture stranice i sprema ih u razdvojenu memoriju. Prilikom određene promjene na stranici, detektira se mjesto promjene koje je spremljeno u određenoj memoriji. Na taj način se ne mora cijela stranica renderirati odjednom, već njeni pojedini dijelovi na kojima su promjene detektirane. To ujedno i ubrzava rad na stranici. [2]

2.3 TypeScript

TypeScript je open-source jezik za programiranje razvijen i održavan od strane Microsofta. To je strogo sintaktički superset JavaScript-a, dodaje neobavezne statičke tipove podataka na jezik. TypeScript uvelike olakšava izrade velikih aplikacija jer definiranjem statičkih tipova podataka uvelike se dobije na čitljivosti i lakšem razumijevanju koda. Budući da je TypeScript superset JavaScripta, svi JavaScript programi su također valjani TypeScript programi. Neki od osnovnih tipova podataka u TypeScriptu su: string, number, undefined, null, boolean, itd. Također programer može definirati svoje tipove podataka tj. sučelje objekta (Slika 2.4), postoji mogućnost definiranja polja koji se sastoji od različitih tipova podataka tzv. Tuple. [3]

Slika 2.4 Izgled sučelja koji se sastoji od dvije varijable različitog tipa

Tipove podataka također potrebno je definirati prilikom definicije funkcije (Slika 2.5), potrebno je odrediti točno koje tipove podataka funkcija prima te koji tip podatka će funkcija vratiti.

```
export function getNewsById(id: number, data: News[]): News | undefined {
    return data.find(item => item.id === id);
}
```

Slika 2.5 Primjer tipova podataka u TypeScriptu

Slika 2.5. prikazuje primjer funkcije koja prima dva tipa podatka (broj i polje objekata koji su definirani sučeljem News). U JavaScriptu postoji integrirana metoda na polju podataka koja pronalazi određene podatake s određenim uvjetom te ukoliko je podatak pronađen vraća vrijednost podatka, u suprotnom vraća vrijednost undefined. Tako da će definirana funkcija vratiti ili tip podatka News ili undefined. TypeScript koristi kompajler koji prevodi napisani kod u JavaScript te samo jednom na početku prije pokretanja programa provjerava jesu li tipovi podataka dobro definirani. U slučaju da tipovi podataka nisu dobro definirani javlja grešku te onemogućava pokretanje programa.

2.4 Node.js

Node.js je open-source, međuplatformno JavaScript okruženje koje izvršava kod izvan preglednika . Node.js omogućava programerima da koriste JavaScript na strani poslužitelja. Kada se skripte sa kodom pokreću na strani poslužitelja, one stvaraju i omogućavaju izradu dinamičnog sadržaja web stranica prije nego se stranica učita u web pregledniku korisnika. Omogućava stvaranje web servera i alate za umrežavanje pomoću JavaScripta i zbirke modula koji upravljaju raznim funkcionalnostima. Podržan je na Linux, macOS i Windows operacijskim sustavima. [4]

2.5 MongoDB

MongoDB je međuplatformna baza podataka koja je orijentirana na dokumente i omogućava podršku za korištenje različitih tipova podataka. Spada u NoSQL baze jer umjesto tablica i redaka koje se koriste u relacijskim bazama podataka koristi kolekcije i dokumente. MongoDB dokumenti su slicni JSON datotekama, ali se razlikuju od njih po tome što koriste Binary JSON (BSON) - varijanta koja koristi više tipova podataka. Polja u dokumentima srodna su stupcima u relacijskoj bazi, a vrijednosti koje se mogu u njih pohraniti mogu biti različite vrste podataka uključujući ostale dokumente, nizove i nizove dokumenata. [5]

3. RAZVOJ VLASTITOG RJEŠENJA

Proces izrade rješenja počinje strukturiranjem poslužitelja, odnosno podjelom logike koja se odvija na poslužitelju i one koja se odvija na klijentu. Nakon podjele slijedi osmišljavanje komunikacije između klijenta i poslužitelja te implementacija samog rješenja. Web aplikacije je dizajnirana te je definirana paleta boja koja će se koristiti.

3.1 Poslužitelj

3.1.1 Pristupne točke poslužitelja

Prilikom osmišljavanja strukture poslužitelja potrebno je misliti i na način komunikacije s klijentom, odnosno definiranje krajnjih točaka za komunikaciju.
Komunikacija je podijeljena u tri cjeline:

Slika 3.1. Servisi za komunikaciju s klijentom

3.1.2 Servis za autentifikaciju korisnika

Servis za autentifikaciju korisnika se sastoji od samo jedne točke za komunikaciju koja kao argument prima e-mail adresu i lozinku korisnika. Ako su korisnički podaci ispravni, korisnik kao odgovor dobiva token koji se kasnije koristi za pristupanje ostalim komunikacijskim točkama poslužitelja.

```
router.post("/", (req, res) => {
    const { email, password } = req.body;

    // Simple validation
    if (!email || !password) {
        return res.status(400).json({ msg: "Please enter all fields" });
    }

    // Check for existing user
    User.findOne({ email }).then(user => {
        if (!user) return res.status(400).json({ msg: "User Does not exist" });

        // Validate password
        bcrypt.compare(password, user.password).then(isMatch => {
            if (!isMatch)
                return res.status(400).json({ msg: "Invalid credentials" });
```

Slika 3.2 Komunikacijska točka za autentifikaciju korisnika

3.1.3 Servis za upravljanje korisnicima

Koristi se za dodavanje novih i brisanje starih korisnika. Da bi korisnik komunicirao s ovom komunikacijskom točkom, mora posjedovati tajni ključ koji se provjerava prilikom svakog zahtjeva na server. Tajni ključ šalje se putem zaglavlja zahtjeva koji mora sadržavati polje "X-Auth-Token" te u njemu treba biti pohranjena vrijednost tokena.

```
function checkToken(req, res, next) {
    const token = req.header("x-auth-token");

    if (token === process.env.SECRET_TOKEN) {
        next();
    } else {
        res.status(400).json({ msg: "cannot access" });
    }
}

router.post("/", checkToken, (req, res) => {
    const { name, email, password } = req.body;
    if (!name || !email || !password) {
        return res.status(400).json({ msg: "Please enter all fields" });
    }

    User.findOne({ email }).then(user => {
        if (user) {
            return res.status(400).json({ msg: "User already exists" });
        }
        const newUser = new User({
            name,
            email,
            password
        });
```

Slika 3.3 Komunikacijska točka za kreiranje novog korisnika

3.1.4 Servis za upravljanje posudbama

Servis za upravljanje posudbama implementira svu potrebnu logiku za ovu aplikaciju, sastoji se ukupno od 5 komunikacijski točaka. Nekim od komunikacijskih točaka pristupa se putem POST ("/return", "/") metode dok ostale koriste GET ("/allLoans", "/loansByName", "/notReturned") metodu. Putanja servisa :

1. "/" - služi za dodavanje nove posudbe. Svaka posudba u tijelu zahtjeva očekuje: ime posuditelja, prezime posuditelja, oib posuditelja, email posuditelja, broj mobitela posuditelja, slika posuđene stvari (kodiranu u obliku base64), datum posudbe (u obliku YYYY-MM-DD) te datum do kojeg se posudba treba vratiti.

2. "/return" - kao argument unutar tijela zahtjeva prima samo id posudbe te mijenja status posudbe u "vraćeno" te pridružuje trenutni datum kao datum vraćanja posudbe.

3. "/notReturned" - služi za dohvaćanje svih trenutno nevraćenih posudbi

4. "/allLoans" - služi za dohvaćanje svih vraćenih i nevraćenih posudbi

4. “/loansByName” - služi dohvaćanje svih vraćenih i nevraćenih posudbi s obzirom na predane parametre.

3.1.5 Posrednik za autentifikaciju na poslužitelju

Komunikacijske točke unutar servisa za upravljanje posudbama zaštičene su na način da se uz svaki upućen zahtjev prema bilo kojoj od točaka mora prosljediti vrijednost tokena. Klijentska aplikacija dobiva token prilikom prijavljivanja u aplikaciju te se on prosljeđuje u zaglavlju svakog zahtjeva. Ukoliko token postoji te je on valjan, zahtjev se prosljeđuje sljedećem sloju.

middlewares > JS auth.js > auth

```
1   const config = require("config");
2   const jwt = require("jsonwebtoken");
3
4   function auth(req, res, next) {
5       const token = req.header("x-auth-token");
6
7       // Check for token
8       if (!token)
9           return res.status(401).json({ msg: "No token, authorizaton denied" });
10
11      try {
12          // Verify token
13          const decoded = jwt.verify(token, config.get("jwtSecret"));
14          // Add user from payload
15          req.user = decoded;
16          next();
17      } catch (e) {
18          res.status(400).json({ msg: "Token is not valid" });
19      }
20  }
21
22  module.exports = auth;
23
```

Slika 3.4 Posrednik za autentifikaciju korisnika putem tokena

3.1.6 “Cronjob” i Mail servis

“Cronjob” simbolizira funkciju koja će se izvršiti na strani poslužitelja svakih “x” dok mail servis služi za obavještavanje korisnika aplikacije.

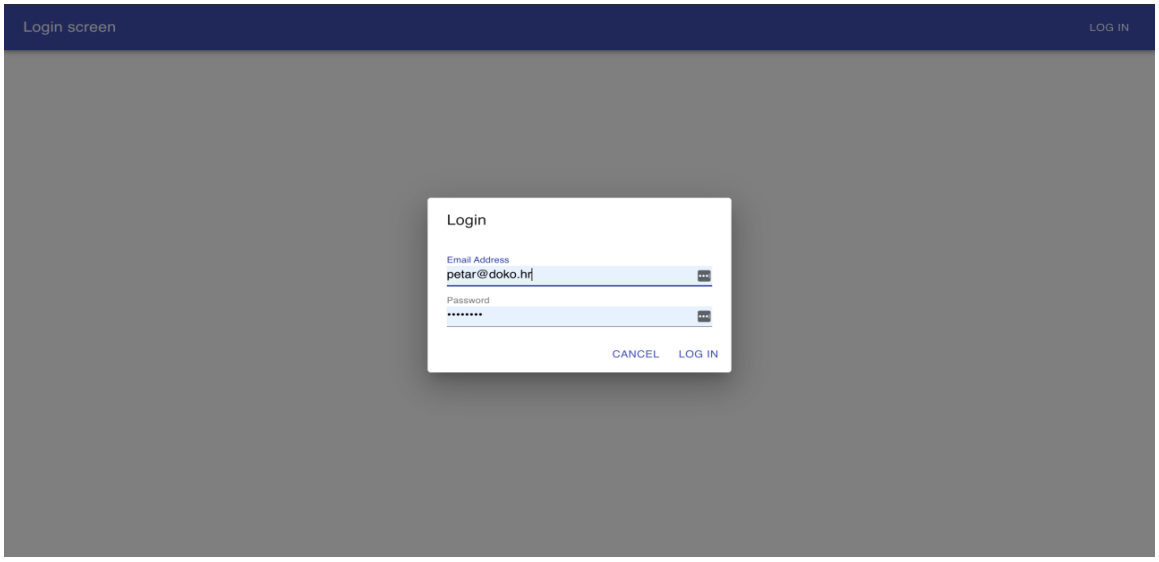
```
cron.schedule("0 0 12 * * *", async () => {
    const loans = await Loan.find();
    loans.forEach(item => {
        if (moment(item.to_date).isBefore(moment())) {
            const data = {
                from: "Loans <me@samples.mailgun.org>",
                to: item.lender_email,
                subject:
                    item.lender_first_name +
                    " " +
                    item.lender_last_name +
                    " kasni s vraćanjem opreme",
                text:
                    "Posuđeno: " +
                    moment(item.from_date).format("YYYY-MM-DD") +
                    "\nTrebalo je biti vraćeno do: " +
                    moment(item.to_date).format("YYYY-MM-DD")
            };
            mg.messages().send(data, async function(error, body) {});
        }
    });
});
```

Slika 3.5. Cronjob za provjeravanje posudbi

Funkcija za provjeravanje posudbi (Slika 3.5) će biti izvršena svaki dan u 12:00 te će mail-om obavjestiti da kasni s vraćanjem pošiljke te da ju treba što prije vratiti.

3.2 Klijentska aplikacija

Klijentska aplikacija izrađena je u React-u i namjenjena je za internet preglednike. Omogućuje korisniku funkcionalnosti kao što je Prijava/Odjava, Unos nove posudbe, Pregled svih posudbi, Pregled nevraćenih posudbi, Filtriranje posudbi po imenu posuditelja, Evidentiranje vraćanja posudbe. Nakon pokretanja aplikacije korisnik se prvo treba prijaviti u nju koristeći e-mail i lozinku (Slika 3.6).



Slika 3.6. Prikaz početne stranice za prijavu korisnika

Nakon unosa e-maila i lozinke korisnik je preusmjeren na glavni zaslon aplikacije gdje može izabrati jednu od tri ponuđene opcije(Slika 3.7).

Slika 3.7. Prikaz glavnog zaslona aplikacije

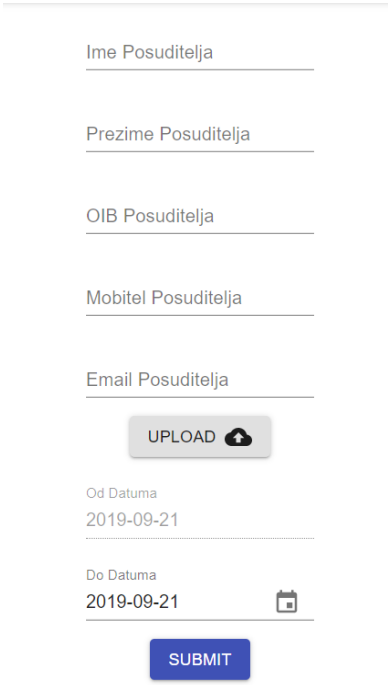
Navigiranje između zaslona odvija se uz pomoc eksterne biblioteke "react-router-dom" koja primjenjuje jednostavan princip za mjenjanje zaslona (pogleda) korisnika. Potrebno je pozvati „BrowserRouter" (Slika 3.8.) komponentu te unutar nje smjestiti sve putanje kojima korisnik može pristupiti.

```
function MainRouter() {
    return (
        <BrowserRouter>
            <Route exact path="/" component={LoginScreen} />
            <PrivateRoute exact path="/home" component={HomeScreen} />
            <PrivateRoute excat path="/newLoan" component={NewLoan} />
            <PrivateRoute excat path="/allLoans" component={AllLoans} />
            <PrivateRoute
                path="/loansByName/:firstName/:lastName"
                component={LoansByName}
            />
            <PrivateRoute path="/notReturned" component={NotReturned} />
            <PrivateRoute
                path="/picture/:firstName/:lastName"
                component={Picture}
            />
        </BrowserRouter>
    );
}
```

Slika 3.8 Router komponenta

Nakon što je omogućeno navigiranje između zaslona pritiskom na gumb Unos nove posudbe

prebacujemo se na zaslon gdje se mogu popunjavati podatci (Slika 3.9).



Slika 3.9 Obrazac za popunjavanje nove posudbe

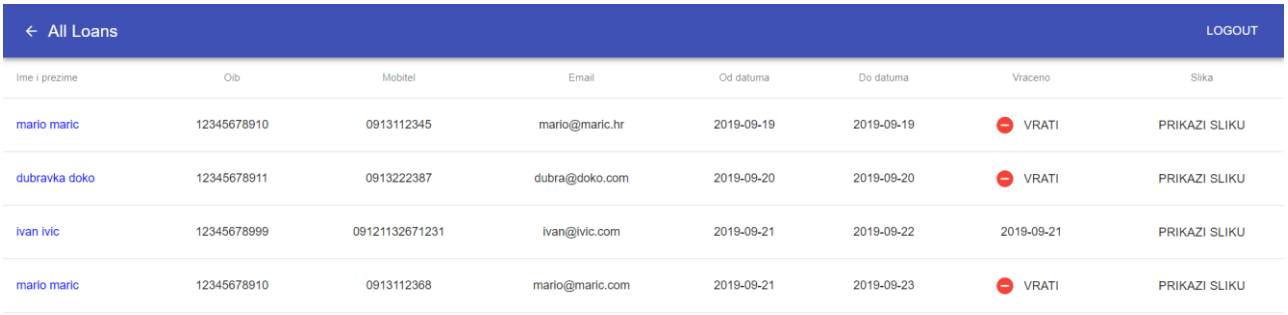
Podatci koji se mogu unijeti su ime, prezime, OIB, broj mobitela, e-mail, slika predmeta, te razdoblje posudbe. Nakon popunjavanja podatci se spremaju u bazu podataka pritiskom na gumb Submit.

```
router.post("/", (req, res) => {
    const {
        lender_first_name,
        lender_last_name,
        lender_oib,
        lender_email,
        lender_phone,
        picture,
        from_date,
        return_date,
        to_date
    } = req.body;
    const newLoan = new Loan({
        lender_first_name: lender_first_name,
        lender_last_name: lender_last_name,
        lender_oib: lender_oib,
        lender_email: lender_email,
        lender_phone: lender_phone,
        picture: picture,
        from_date: moment(from_date).toDate(),
        to_date: moment(to_date).toDate()
    });
```

Slika 3.10 Upisivanje podataka u bazu

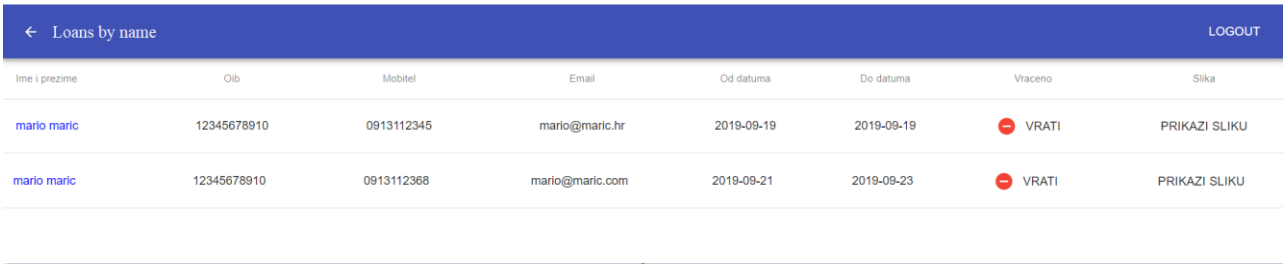
Nakon što su uneseni podatci u bazu (Slika 3.10), klikom na gumb Pregled svih posudbi može se

pregledati popis svih posudbi (Slika 3.11). U toj bazi nalaze se sve vraćene i nevraćene posudbe. Klikom na gumb VRATI vraća se posudba i na to mjesto dolazi pripadajući datum kada je predmet vraćen.



Slika 3.11 Sve posudbe

Klikom na ime i prezime osobe izlistavaju se sve posudbe osobe sa tim imenom (Slika 3.12).

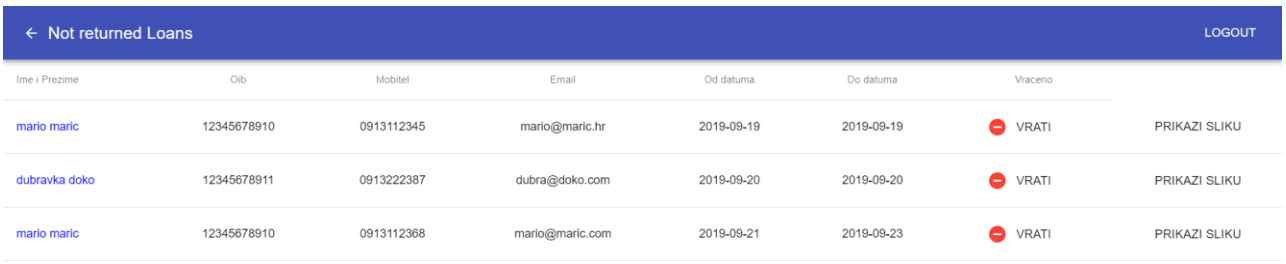


Slika 3.12 Sve posudbe osobe mario maric

Klikom na PRIKAZI SLIKU prikazuje se slika posuđene stvari (Slika 3.13).

Slika 3.13 Slika posuđene stvari

Klikom na gumb Nevraćene posudbe prikazuju se sve posudbe koje nisu vraćene (Slika 3.14).



Slika 3.14 Nevraćene posudbe

Iz aplikacije se možemo odjaviti pritiskom na LOGOUT u gornjem desnom kutu aplikacije (Slika 3.14) i vraćamo se na početnu stranicu za prijavu.

5. ZAKLJUČAK

Nakon uspješnog završetka diplomskog rada, napravljena je internet aplikacija za evidenciju posuđivanja stvari. Aplikacija je namijenjena za korištenje vlasniku stvari koji pomoću nje može voditi računa o posuđenim stvarima. Aplikacija sadrži sve funkcionalnosti kako bi ona mogla raditi ispravno, odnosno omogućava vlasniku da bez ikakvih problema vodi računa o posuđenim stvarima. Pri izradi ove internet aplikacije korištene su neke od najnovijih tehnologija koje su omogućile lakši i jednostavniji razvoj aplikacije. Nakon isprobavanja rada aplikacije sve funkcionalnosti rade nesmetano i smatra se da je ovaj diplomski rad uspješno izvršen.

I. **Literatura**

[1] Wikipedia, Visual Studio Code (IDE) https://en.wikipedia.org/wiki/Visual_Studio_Code (stranica posjećena: 9. rujna 2019)

[2] Wikipedia, React(web framework) https://en.wikipedia.org/wiki/React_(web_framework) (stranica posjećena: 9. rujna 2019)

[3] Wikipedia, Typescript https://en.wikipedia.org/wiki/TypeScript (stranica posjećena 9. rujna 2019)

[4] Wikipedia, Node.js https://en.wikipedia.org/wiki/Node.js (stranica posjećena 10. rujna 2019)

[5] SearchData Management, MongoDB https://searchdatamanagement.techtarget.com/definition/MongoDB (stranica posjećena 10. rujna 2019)

II. **Sažetak**

Naslov: Internet aplikacija za evidenciju posuđivanja

Koristeći neke od najmodernijih tehnologija za razvoj internet aplikacija izrađena je aplikacija kojom se vodi evidencija o posuđivanju stvari. Vlasnik stvari na jednostavan način vodi računa o tome koje su stvari posuđene i tko je te stvari posudio. Može se definirati rok za vraćanje stvari i ukoliko dođe do isteka roka, vlasniku se na e-mail šalje podsjetnik o tome da netko kasni sa vraćanjem posuđene stvari. Nakon testiranja aplikacije i otklanjanja pogrešaka možemo utvrditi da je internet aplikacija spremna za korištenje.

Ključne riječi: React, JavaScript, TypeScript, MongoDB, Node.js

III. **Abstract**

Title: Internet application for borrowing records

Using some of the most advanced technologies for developing internet applications, an application was created to keep records of borrowed things. The owner of the items(things) keeps track of borrowed items and who borrowed them in a simple way. A deadline for returning the item can be defined and if the deadline expires, a reminder e-mail is sent to the owner that someone is being late with returning the borrowed item. After testing the application and debugging, we can determine that the internet application is ready to use.

Keywords: React, JavaScript, TypeScript, MongoDB, Node.js

IV. **Životopis**

Petar Doko rođen je 12.2.1995. godine u Đakovu. Živi na adresi Biskupa Ćirila Kosa 3 u Đakovu. Nakon završetka osnovne škole „Vladimir Nazor“ 2009. godine u Đakovu, upisuje Opću gimnaziju u Đakovu te maturira 2013.g. Iste godine upisuje preddiplomski sveučilišni studij na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, smjer računarstvo. 2016. godine završava preddiplomski sveučilišni studij i upisuje diplomski studij smjer DRD – Informacijske i podatkovne znanosti. Tijekom druge godine diplomskog studija odrađuje praksu na Institutu RT-RK u Osijeku. Nakon završetka studija planira pronaći posao u struci.

_____

Petar Doko