Aplikacija za savjetovanje brucoša

Mandić, Laura

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

Sveučilišni studij

APLIKACIJA ZA SAVJETOVANJE BRUCOŠA

Završni rad

Laura Mandić

Osijek, 2021.

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

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 24.08.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Laura Mandić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4239, 26.07.2018.
OIB studenta:	09023698257
Mentor:	Izv. prof. dr. sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za savjetovanje brucoša
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog 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:	24.08.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

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, 08.09.2021.

Ime i prezime studenta:	Laura Mandić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4239, 26.07.2018.
Turnitin podudaranje [%]:	10%

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za savjetovanje brucoša**

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:

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Laura Mandić_____, OIB: __09023698257_____, student/ica Fakulteta elektrotehnike,
računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Aplikacija za savjetovanje brucoša**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 08.09.2021._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Upis studija na fakultetu jedan je od važnijih događaja u životu svakog studenta. Samim time, brucoši osjećaju nelagodu jer ne znaju što ih očekuje, a možda ni ne poznaju nekoga tko bi ih mogao uputiti u studentski život. Neki od njih koriste društvene mreže kako bi došli u kontakt s kolegama brucošima ili starijim studentima.

U ovom završnom radu bit će opisana struktura i rad aplikacije namijenjene upravo toj kategoriji studenata. Aplikacija omogućava povezivanje brucoša Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek (FERIT) sa studentima savjetnicima koji će ih uvesti u svijet studenata. Prilikom prijave za povezivanje i brucoši i studenti savjetnici popunjavaju upitnik o interesima, prethodnom školovanju, upisanom studiju i slično. Svaki korisnik može pregledati vlastiti profil koji obuhvaća informacije koje su dane pri popunjavanju upitnika te postojeća povezivanja sa savjetnicima ili brucošima. Savjetnicima su prikazani i zahtjevi za savjetovanjem, a brucošima je prikazan izbornik dostupnih savjetnika. Aplikacija također pruža mogućnost pregleda često postavljenih pitanja, čime se olakšava pristup odgovorima na najčešće upite brucoša. Uz to, moguće je i postaviti pitanje svim studentima prijavljenim na sustav, čime je ono vidljivo svima, kao i pripadajući odgovori koje mogu postaviti svi studenti koji koriste aplikaciju.

Drugo poglavlje obuhvaća pregled postojećih rješenja i njihovih mogućnosti. U trećem poglavlju opisuju se korištene tehnologije, njihova svrha i karakteristike. Četvrto poglavlje razmatra teorijsku podlogu vezanu uz pojedinu ključnu tehnologiju. U petom poglavlju navedeni su najvažniji postupci u razvoju aplikacije. U šestom poglavlju demonstriran je rad aplikacije. Posljednje poglavlje odnosi se na zaključak ovog završnog rada.

1.1. Zadatak završnog rada

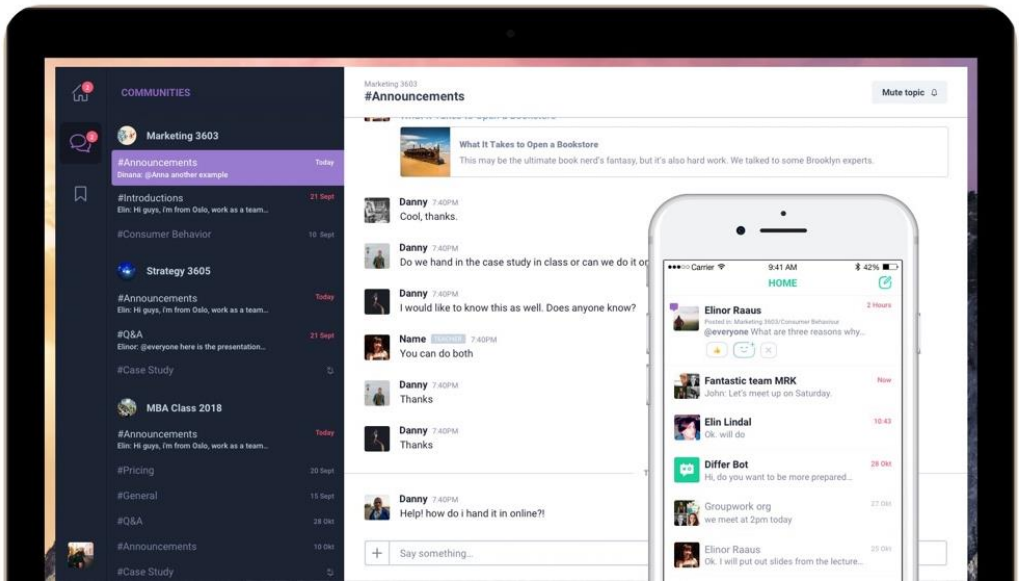
Zadatak završnog rada je izraditi web aplikaciju koristeći React biblioteku i JavaScript programski jezik, a svi podaci o korisnicima spremat će se u MongoDB bazu podataka. Prilikom prijave u sustav aplikacija treba zahtijevati ispravnu adresu elektroničke pošte studenata FERIT-a te zaporku po izboru. Aplikacija će pri dogovorenom savjetovanju omogućiti vidljivost kontakt podataka brucoša i savjetnika.

2. PREGLED SLIČNIH RJEŠENJA

Na tržištu danas postoje mnoge aplikacije čija je svrha povezivanje studenata i lakše snalaženje na fakultetu. Ovo poglavlje razmatra pet takvih aplikacija, njihove funkcionalnosti i mogućnost korištenja na FERIT-u.

2.1. Differ

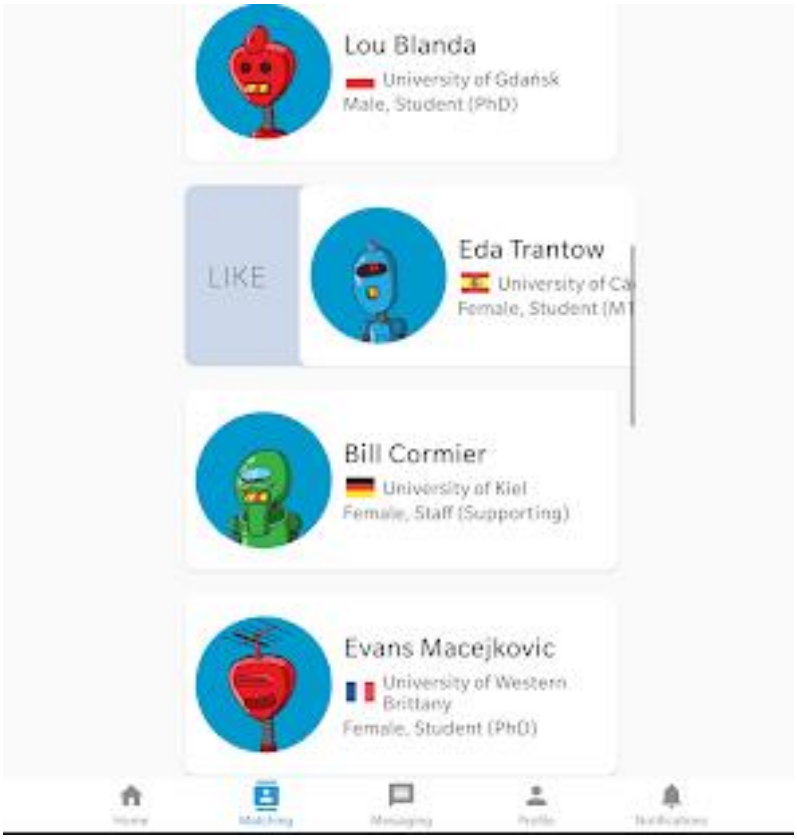
Aplikacija 'Differ' koristi se na 11 različitih sveučilišta u svijetu za povezivanje studenata i unaprjeđenje studentskog iskustva. Nalik je aplikaciji Slack, omogućuje chat na različitim kanalima (slika 2.1.). Dostupan je i razgovor s dva *chatbot*-a. Kreirana je radi lakše integracije brucoša i komunikacije studenata i nastavnika. Aplikacija također omogućava uspostavljanje često postavljanih pitanja te anketiranja studenata o zadovoljstvu provedbom nastave. Postoji mobilna verzija aplikacije za Android i iOS te desktop verzija za Windows i Mac operacijske sustave. Međutim, još uvijek nije dostupna svim studentima, pa tako ni studentima FERIT-a.



Slika 2.1. Prikaz korisničkog sučelja aplikacije 'Differ', [1]

2.2. SEA-EU Around

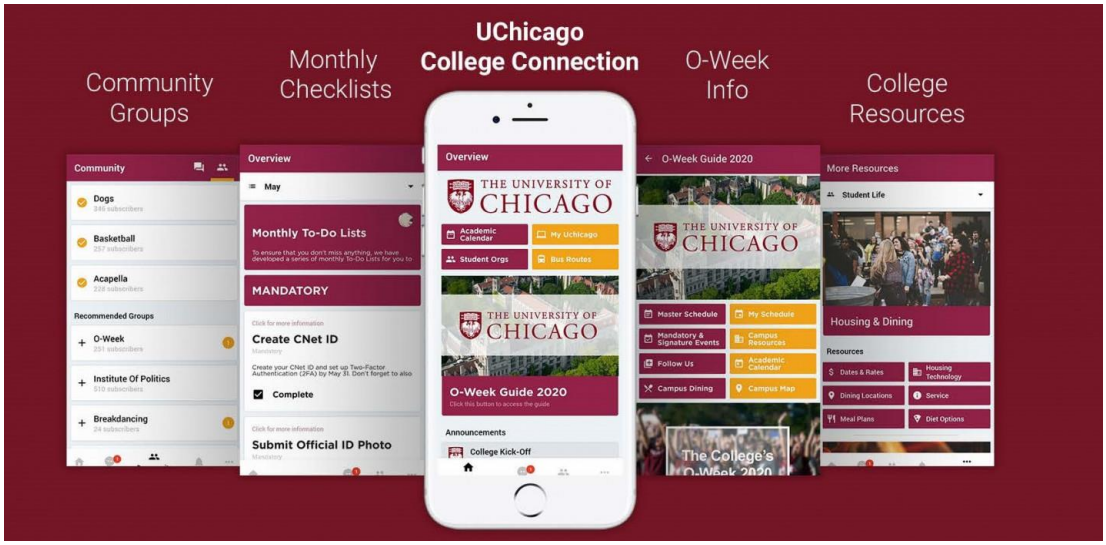
Aplikacija 'SEA-EU Around' dostupna je na šest priobalnih sveučilišta u Europi, uključujući Sveučilište u Splitu. Kreirana je s ciljem povezivanja studenata, ostvarivanjem suradnji među nastavnicima i osobljem te lakšom mobilnošću. Moguće je razgovarati sa svim kolegama iz SEA-EU alijanse, raditi na poboljšanju jezičnih sposobnosti te učiti o drugim kulturama (slika 2.2.). Dostupna je mobilna verzija aplikacije za Android i iOS uređaje. Napravljena je samo za studente priobalnih sveučilišta što ju čini nedostupnom studentima FERIT-a.



Slika 2.2. Prikaz korisničkog sučelja aplikacije 'SEA-EU Around', [2]

2.3. UChicago College Connection

'UChicago College Connection' mobilna je aplikacija dostupna Android i iOS korisnicima. Namijenjena je za korištenje isključivo studentima Sveučilišta u Chicagu, a nudi stvaranje profila, pridruživanje zajednicama s određenim interesima i razmjenjivanje poruka. Uz to, pruža pristup informacijama o događajima na Sveučilištu, karti s važnim lokacijama, prehrani na kampusu i drugome (slika 2.3.).



Slika 2.3. Prikaz korisničkog sučelja aplikacije 'UChicago College Connection', [3]

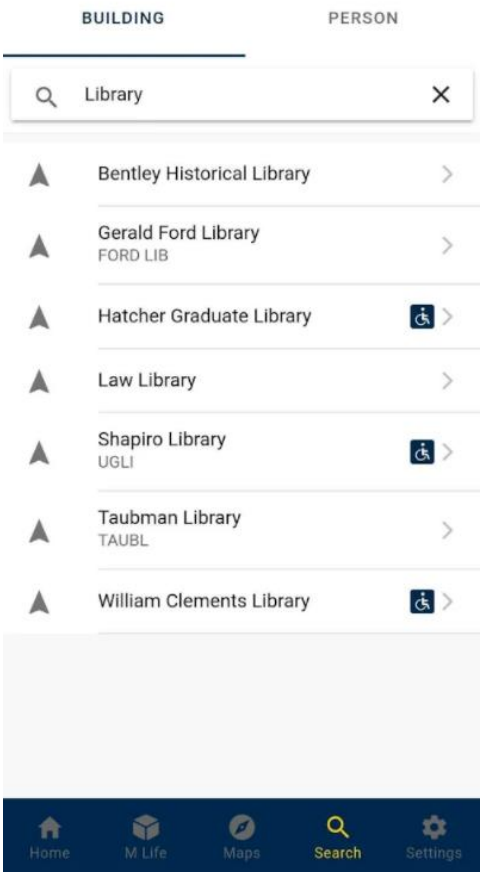
2.4. EduHub

Aplikacija 'EduHub' prvenstveno je namijenjena pripremanju studenata Medicinskog fakulteta za ispite te maturanata za prijemne ispite. Moguće je pokrenuti simulaciju ispita, vježbanje na svim pitanjima iz baze podataka te sudjelovati u raspravama na forumu gdje studenti mogu razmjenjivati iskustva o ispitima, kolegijima, profesorima i drugim temama (slika 2.4.). Dostupna je u mobilnoj verziji za Android i iOS korisnike. Korisnik može dodavati nova pitanja u bazu i gledati statistiku svojih prijašnjih rješavanja simulacije ispita. Aplikacija nije namijenjena studentima FERIT-a.

Slika 2.4. Prikaz korisničkog sučelja aplikacije 'EduHub', [4]

2.5. University of Michigan

Aplikacija 'University of Michigan' omogućava studentima razmjenu poruka s drugim studentima, postavljanje podsjetnika, pronalaženje lokacija na kampusu te akademskih događaja (slika 2.5.). Dostupna je isključivo u mobilnoj verziji za Android i iOS korisnike. Budući da je razvijena specijalno za studente Sveučilišta u Michiganu, razumljivo, nije od koristi studentima FERIT-a.



Slika 2.5. Prikaz korisničkog sučelja aplikacije 'University of Michigan', [5]

3. KORIŠTENE TEHNOLOGIJE

Tehnologije potrebne za razvoj aplikacije navedene su u nastavku poglavlja. Svaka od njih korištena je pri implementaciji funkcionalnosti, skladištenju podataka ili oblikovanju izgleda web aplikacije.

3.1. JavaScript

JavaScript je interpreterski programski jezik, najpoznatiji kao jezik za kreiranje web stranica, ali koristi se i kod mnogih okruženja koja nisu vezana uz preglednike. JavaScript baziran je na prototipima, dinamičan i podržava objektno orijentiran, imperativan i funkcijski stil programiranja. Pokreće se na klijentskoj strani što omogućava dizajniranje stranice koja se ponaša na određen način pojavom određenog događaja. Osnovna sintaksa slična je Java i C++ programskim jezicima [6]. Standardi za JavaScript su ECMAScript Specifikacija jezika (ECMA-262) i ECMAScript Specifikacija API-ja za internacionalizaciju (ECMA-402) [7].

3.2. React

React je JavaScript biblioteka čija je svrha izgradnja korisničkog sučelja. Omogućava stvaranje kompleksnih korisničkih sučelja sačinjenih od manjih, izoliranih dijelova koda, poznatih kao „komponente“ [8]. Kod unutar aplikacija uobičajeno prati JavaScript XML (JSX) sintaksu, ekstenziju sintakse jezika JavaScript. Prednost JSX-a je u tome što nije potrebno odvajati jezik za označavanje i logiku u različite datoteke. Dužnosti su razdvojene pomoću slabo povezanih (engl. l*oosely coupled*) komponenti koje sadrže i jezik za označavanje i logiku [9]. React komponente prihvaćaju ulazne vrijednosti nazvane svojstva (engl. *props*) i vraćaju elemente koji predstavljaju ono što će se prikazati u pregledniku. Komponente mogu biti funkcijske ili klasne [10]. Za razliku od DOM (*Document Object Model*) elemenata preglednika, elementi su jednostavni objekti i ne zahtijevaju puno resursa. React DOM čini izmjene u DOM-u preglednika samo kada je to u potpunosti neophodno.

3.3. Node.js

Node.js je asinkrono JavaScript izvršno okruženje za stvaranje mrežnih aplikacija. Node.js aplikacija pokreće se u jednom procesu bez stvaranja novih niti za svaki zahtjev. Kada Node.js obavlja ulazno-izlaznu operaciju poput pristupa bazi podataka, umjesto blokiranja niti i čekanja, operacija se nastavlja kada odgovor dođe [11]. Node –ov upravljač paketima (engl. *Node Package Manager, npm*) uslužni je program naredbenog retka za interakciju s online repozitorijem za

objavu Node.js projekata otvorenog koda (engl. *open-source projects*) te pomaže u instalaciji paketa, upravljanju verzijama i upravljanju ovisnostima [12].

3.4. MongoDB

MongoDB je nerelacijska baza podataka koja za pohranu i obradu podataka koristi kolekcije umjesto tablica i dokumente umjesto redaka tablice [13]. Dokumenti su oblikovani u binarnom JSON (*JavaScript Object Notation*) formatu, kao skup ključ-vrijednost parova. Vrijednosti u dokumentima mogu biti različitih tipova što omogućava fleksibilnost te je moguće kao vrijednosti pohraniti nizove i ugniježđene objekte. Upiti se pišu u JSON obliku što ih čini jednostavnim za slaganje. Format MongoDB-a manje je restriktivan i ima veće performanse naspram uobičajenih odabira SQL baza podataka, što je ključno kada je potrebna velika brzina i dostupnost [14]. MongoDB radi tzv. rascjepkavanje (engl. *sharding*) koje je metoda distribucije jednog skupa podataka u više baza podataka, a one time mogu biti spremljene na više poslužitelja. Rascjepkavanje je horizontalno skaliranje, dodaju se novi čvorovi za raspodjelu opterećenja te se time omogućuje rješavanje više zahtjeva nego što bi to mogao samo jedan stroj. Za razliku od horizontalnog, vertikalno skaliranje radilo bi se povećanjem snage jednog stroja kroz povećanje kapaciteta za pohranu podataka, većeg RAM-a (*Random Access Memory*) ili korištenjem snažnijeg procesora [15]. Trajnost podataka omogućena je replikacijom podataka na fizički odvojenim serverima. Zahvaljujući replikaciji, aplikacija može ostati online i u slučaju pada servera te je moguće povratiti podatke koji su se nalazili na spornom serveru [16].

3.5. CSS

CSS je stilski jezik korišten za definiranje pravila prikaza web stranica. CSS opisuje kako elementi trebaju izgledati na ekranu. CSS je standardiziran među preglednicima W3C specifikacijama. Nastao je kako bi se odvojili sadržaj i izgled dokumenta. To olakšava i potencijalne izmjene. Nadalje, omogućava definiranje različitih stilova za različite veličine ekrana. Ovaj jezik omogućava opisivanje stila kako HTML dokumenata, tako i XML, XHTML, SVG i XUL dokumenata [17].

3.6. Bootstrap

Bootstrap je besplatan CSS razvojni okvir (engl. *framework*) usmjeren prema responzivnom web dizajnu. Sadržava CSS i JavaScript predloške za oblikovanje. Prvotno su ga kreirali dizajner i programer iz Twittera te je bio poznat kao Twitter Blueprint, a sada ga održava manji tim programera na GitHub platformi [18].

3.7. React Bootstrap

React Bootstrap je JavaScript razvojni okvir identičan Bootstrap-u, a razvijen upravo za React. Definira komponente koje su napisane iz temelja i zato nije potrebno koristiti nikakvu vanjsku ovisnost. Komponente, poput gumba, navigacijske trake i obrazaca, mogu primati svojstva, kao i svaka druga React komponenta [19].

4. TEORIJSKA PODLOGA POTREBNA ZA RAZVOJ APLIKACIJE

U ovom poglavlju razmotreni su važni koncepti karakteristični za programski jezik JavaScript ili biblioteku React. Svaki od njih primijenjen je unutar aplikacije.

4.1. JavaScript

Nadogradnjom ECMAScript standarda na verziju 6 2015. godine nastupile su velike promjene. Većina JavaScript sintakse lako je razumljiva nekome tko ne poznaje JavaScript jezik, ali poznaje neki drugi uvriježeni jezik kao što je C++, C#, Python ili Java. ES6 standard uveo je promjene u sintaksu koje nisu intuitivne. Ovo potpoglavlje obradit će najvažnije promjene uvedene kroz ES6 te jednu uvedenu kroz ES8.

4.1.1. Varijable

ES6 uvodi *let* i *const* ključne riječi za definiciju varijable kao zamjenu za *var* ključnu riječ. Ključna riječ *const* definira konstantnu referencu na vrijednost. Sama varijabla je nepromjenjiva, ali njen sadržaj nije. Tako se mogu mijenjati elementi konstantnog polja ili svojstva konstantnog objekta. Varijable koje se definiraju pomoću ključne riječi *const* moraju se odmah i inicijalizirati. Ključna riječ *let* označava promjenjivu varijablu i nad njome nema ograničenja nalik prethodno spomenutima [20].

4.1.2. Literalni predlošci

Literalni predlošci (engl. *template literal*) izrazi su označeni *backtick* navodnicima (` `) i omogućavaju ugrađivanje varijabli u *string* te jednostavnije pisanje u više redova. Vrijednosti varijabli označavaju se znakom dolara i naziv se obuhvaća vitičastim zagradama [21]. Slika 4.1. prikazuje primjere.

```
console.log(`Vrijednost varijable expression: ${expression}.`);
console.log(`string text line 1
string text line 2`);
```

Slika 4.1. Literalni predlošci

4.1.3. Streličaste funkcije

Streličaste (engl. *arrow*) funkcije pojednostavljuju pisanje funkcije pa tako ne zahtijevaju vitičaste zagrade, ključne riječi *return* i *function*. Moraju se definirati prije nego se koriste. Ključna riječ

return i zagrade mogu se izostaviti samo ako je funkcija jedan izraz [20]. Primjer funkcije prikazuje slika 4.2.

```
const greetUser = name => `Hi ${name}`;
```

Slika 4.2. Streličasta funkcija

4.1.4. Destrukturiranje

Destrukturiranje čini dodjeljivanje vrijednosti iz polja ili objekta u varijable jednostavnijim. Kod objekata potrebno je samo navesti nazive varijabli unutar vitičastih zagrada. Nazivi moraju odgovarati svojstvima objekta. Kod polja, nazivi varijabli pišu se unutar uglatih zagrada [22]. Slika 4.3. prikazuje primjer destrukturiranja objekta, a slika 4.4. primjer destrukturiranja polja.

```
const user = {
  id: 73,
  name: "Alan",

};
const { id, name } = user;
```

Slika 4.3. Destrukturiranje objekta

```
const foo = ['one', 'two'];
const [first, second] = foo;
```

Slika 4.4. Destrukturiranje polja

4.1.5. Parametar ostatka i operator širenja

Parametar ostatka (engl. *rest parameter*) i operator širenja (engl. *spread operator*) imaju istu sintaksu, koja je trotočje (…). Parametar ostatka omogućuje funkciji tretiranje neodređenog broja argumenata kao polje [20]. Njegovo korištenje prikazuje slika 4.5. Operator širenja rasprostire sadržaj polja, a od ES9 može se koristiti i za sadržaj objekta [23]. Primjer uporabe operatora širenja prikazuje slika 4.6.

```
function sum(...numbers) {
  let sum = 0;
  for (let num of numbers) sum += num;
  return sum;
}
let x = sum(4, -11, 25, 29, 107);
```

Slika 4.5. Parametar ostatka

```
let array1 = [1, 2, 3];
let array2 = [4, 5];
let connectedArray = [...array1, ...array2]; //[1, 2, 3, 4, 5]
```

Slika 4.6. Operator širenja

4.1.6. Obećanja

Obećanje (engl. *Promise*) je objekt koji zamjenjuje vrijednost koja će se s vremenom postaviti. Ta vrijednost nije poznata pri stvaranju obećanja. Ona omogućuju asinkronim metodama vraćanje vrijednosti kao sinkrone metode. Umjesto vraćanja konačne vrijednosti odmah, vraćaju obećanje o vraćanju konačne vrijednosti u skoroj budućnosti. Obećanja mogu biti u 3 stanja: neriješeno, ispunjeno i odbijeno. Svako obećanje na početku ima neriješenu vrijednost [24]. Slika 4.7. prikazuje primjer obećanja koje se ispunjava nakon 1 sekunde.

```
let promise = new Promise(function(resolve, reject) {
    setTimeout(() => resolve("Done"), 1000);
  });
```

Slika 4.7. Obećanje

4.1.7. Asinkrone funkcije

Asinkrone (engl. *async*) funkcije omogućuju korištenje *await* ključne riječi unutar njih. Uvedene su u ES8 standardu. Omogućuju asinkrono ponašanje temeljeno na obećanjima te se pritom piše jednostavnije, čistije i bez lanaca obećanja. *Await* izraz pauzira izvođenje dok se obećanje ne vrati kao ispunjeno ili odbijeno. Riješena vrijednost obećanja tretira se kao vraćena vrijednost *await* izraza. Ovaj pristup omogućava omatanje koda *try/catch* blokovima [25]. Primjer korištenja prikazan je slikom 4.8.

```
async function foo() {
 let promise = new Promise((resolve, reject) => {
   setTimeout(() => resolve("Done"), 1000)
 });
 let result = await promise;  //"Done"
```

Slika 4.8. Asinkrone funkcije

4.2. React

4.2.1. Komponente

Komponente omogućavaju razdvajanje korisničkog sučelja na nezavisne dijelove koji se mogu višestruko koristiti. Konceptualno nalik su funkcijama jer primaju ulazna svojstva (engl. *props*), a vraćaju elemente koji će se prikazati na ekranu korisnika. Komponente se dijele na funkcijske i klasne komponente. U prošlosti, funkcijske komponente koristile su se za prikazivanje sadržaja kreiranog korištenjem JSX sintakse. Klasne komponente su uz prikazivanje sadržaja imale mogućnost korištenja metoda životnog ciklusa (engl. *lifecycle methods*) za pokretanje koda u određenom dijelu životnog ciklusa komponente. Također, samo klasne komponente mogle su imati sustav stanja za ažuriranje prikazanog sadržaja [10]. To se izmijenilo 2018. godine predstavljanjem verzije 16.8. i *Hook* funkcija. Primjeri jednostavne klasne i funkcijske komponente prikazani su na slikama 4.9. i 4.10.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello World!</h1>;
  }
}
```

Slika 4.9. Klasna komponenta

```
function Welcome() {
  return <h1>Hello World!</h1>;
}
```

Slika 4.10. Funkcijska komponenta

4.2.2. *Hook* funkcije

Razvojni programeri React-a uvidjeli su da klase stvaraju mnoge probleme. Počevši od težeg učenja React-a zbog specifičnosti JavaScript klasa, kompleksnosti koda u klasnim metodama životnog ciklusa do teže optimizacije. Shvatili su da funkcijske komponente predstavljaju budućnost, no one nisu mogle sve što klasne komponente mogu. To je dovelo do stvaranja *Hook* funkcija. Njihova uporaba omogućava ponovno korištenje logike pamćenja stanja (engl. *stateful logic*) među komponentama bez mijenjanja hijerarhije. Moguće je i osmisliti vlastitu *Hook* funkciju te ju koristiti u različitim komponentama. Spomenute funkcije mogu se koristiti isključivo u funkcijskim komponentama i time su funkcijskim komponentama omogućile sve najvažnije karakteristike klasnih komponenti, uključujući pokretanje koda u određenom trenutku životnog

ciklusa komponente te sustav stanja za ažuriranje sadržaja [26]. Tu su ključne *useState()* i *useEffect()* funkcije, gdje je *useState()* omogućila korištenje stanja, a *useEffect()* omogućila izvršavanje koda „nuspojava“ uz prikazivanje stranice, poput dohvaćanja podataka [27].

4.2.3. Svojstva komponenti

Svojstva (engl. *props*) ulazni su podaci spremljeni u objektu kojeg komponente primaju [10]. Roditeljske komponente prosljeđuju svojstva komponentama „djeci“ na način prikazan na slici 4.11., gdje je naziv svojstva *name*.

```
function App() {
  return (
    <div>
      <User name="Sara" />
      <User name="Dino" />
      <User name="Iva" />
    </div>
  );
}
```

Slika 4.11. Prosljeđivanje svojstva

Primjer korištenja tog svojstva u klasnoj komponenti prikazan je slikom 4.12., a u funkcijskoj slikom 4.13.

```
class User extends React.Component {
  render() {
    return (
      <div>
        <h1>Korisnički podaci</h1>
        <h2>Ime: {this.props.name}</h2>
      </div>
    );
  }
}
```

Slika 4.12. Korištenje svojstva u klasnoj komponenti

```
function User(props) {
  return(
    <div>
      <h1>Korisnički podaci</h1>
      <h2>Ime: {this.props.name}</h2>
    </div>
  );
}
```

Slika 4.13. Korištenje svojstva u funkcijskoj komponenti

4.2.4. Rukovanje događajima

Rukovanje događajima (engl. *event handling*) odlučujuće je kod kreiranja responzivnih web aplikacija koje odgovaraju na svaki potez korisnika. Kod JSX sintakse ne prosljeđuje se *string* kao kod HTML-a nego funkcija, a slika 4.14. prikazuje jedan primjer pozivanja funkcije *doSomething()* na klik gumba.

```
<button onClick={doSomething}>
  Do Something!
</button>
```

Slika 4.14. Primjer rukovanja događajem

Također, nije moguće proslijediti *false* za sprječavanje zadanog ponašanja nego se u funkciji mora pozvati *event.preventDefault()* gdje je event sintetički događaj [28].

4.2.5. Uvjetno prikazivanje

Uvjetno prikazivanje (engl. *conditional rendering*) predstavlja proces dostavljanja komponenti s obzirom na ispunjenje određenih uvjeta. Kada komponenta koristi uvjetno prikazivanje, izgled komponente se mijenja ovisno o stanju koje se provjerava. Uvjetno prikazivanje može se ostvariti pomoću *if-else* izraza, ternarnog operatora, operatora &&, *switch* izraza te drugim metodama [29]. Primjer uvjetnog prikazivanja pomoću ternarnog operatora prikazuje slika 4.15. gdje funkcija vraća *ErrorMessage* komponentu ako je došlo do pogreške, odnosno ako je varijabla *error* postavljena na *true*, a u suprotnom vraća *SuccessMessage* komponentu.

```
return <div>{ error ? <ErrorMessage /> : <SuccessMessage /> }</div>;
```

Slika 4.15. Primjer uvjetnog prikazivanja korištenjem ternarnog operatora

5. REALIZACIJA ZADATKA

Cilj je izraditi web aplikaciju u kojoj će se implementirati registracija, prijava i odjava korisnika, unos osobnih podataka kroz obrazac, forum za rasprave o temama vezanima uz fakultet, gdje je moguće kreirati temu i dodavati komentare. Potrebno je implementirati i prikaz često postavljenih pitanja te prikaz profila korisnika (savjetnika ili brucoša) uz mogućnost brisanja profila.

5.1. Specifikacija zahtjeva i mogućih scenarija

Ovo potpoglavlje razlaže sve zahtjeve koje aplikacija mora ispuniti te scenarije koji se mogu odviti pri ispunjavaju pojedinog zahtjeva.

Zahtjevi koji su postavljeni obuhvaćaju:

1. Mogućnost registracije
2. Mogućnost prijave u sustav
3. Popunjavanje obrasca o školovanju, stanovanju i interesima
4. Izmjena osobnih podataka u bazi
5. Pregled profila koji sadržava osobne podatke korisnika
6. Izlistanje svih dodijeljenih savjetovanja
7. Mogućnost uklanjanja dodijeljenih savjetnika/brucoša
8. Izlistanje dostupnih savjetnika ili zahtjeva brucoša
9. Mogućnost slanja zahtjeva pojedinom savjetniku, odnosno prihvaćanja ili odbijanja zahtjeva brucoša
10. Mogućnost brisanja profila
11. Pregled tema i komentara foruma
12. Dodavanje tema u forum
13. Dodavanje komentara na temu
14. Pregled često postavljenih pitanja
15. Odjava iz sustava.

Zahtjev broj 1 obuhvaća unos adrese elektroničke pošte studenta FERIT-a, zaporke od minimalno 6 znakova te izbor uloge savjetnika ili brucoša. Pri tome su mogući scenariji:

a) Korisnik nije popunio sva polja te se prikazuje poruka da je svako polje neophodno te se nije moguće registrirati u sustav.

b) Korisnik je popunio sva polja, no domena adrese elektroničke pošte nije student.ferit.hr. Pojavljuje se poruka o neispravnoj adresi elektroničke pošte te se nije moguće registrirati u sustav.
c) Korisnik je popunio sva polja, no zaporka ima manje od 6 znakova. Pojavljuje se poruka da je zaporka prekratka te se nije moguće registrirati u sustav.
d) Korisnik je popunio sva polja u skladu sa zahtjevima i klikom gumba korisnik se registrira.

Zahtjev broj 2 obuhvaća prijavu postojećeg korisnika u sustav. Mogući scenariji su:

a) Adresa elektroničke pošte korisnika ne postoji u bazi podataka. Ispisuje se poruka o neispravnom unosu te se onemogućuje prijava.
b) Adresa elektroničke pošte korisnika postoji u bazi podataka, ali je korisnik označen kao obrisan. Ispisuje se poruka o neispravnom unosu te se onemogućuje prijava.
c) Adresa elektroničke pošte korisnika postoji u bazi podataka, korisnik nije izbrisao svoj profil, ali se *hash* unesene zaporke ne podudara s *hash* vrijednosti u bazi podataka. Ispisuje se poruka o neispravnom unosu te se onemogućuje prijava.
d) Unesena adresa elektroničke pošte i *hash* zaporke nalaze se u bazi podataka te je s korisničkim računom sve u redu. Prijava je omogućena.

Pri popunjavanju obrasca iz zahtjeva 3, potrebno je popuniti polja o školovanju, mjestu stanovanja i studiju koja su obvezna te druga neobavezna polja. Moguća u dva scenarija:

a) Korisnik je propustio popuniti jedno od obveznih polja te se ispisuje poruka o obveznosti popunjavanja i obrazac se ne podnosi.
b) Korisnik je popunio sva obvezna polja i obrazac se podnosi.

Pri izmjeni podataka o korisniku iz zahtjeva 4 moguće je izmijeniti sva polja iz zahtjeva 3 pri čemu se ne smije isprazniti jedno od obveznih polja te su scenariji identični onima uz zahtjev 3.

Pregled profila iz zahtjeva 5 obuhvaća pregled svih podataka koji su o korisniku spremljeni u bazi, izuzev *hash* vrijednosti zaporke.

Prema zahtjevu 6 izlistava se popis prihvaćenih brucoša savjetnicima, a dodijeljenih savjetnika brucošu.

Zahtjev 7 odnosi se na mogućnost i brucoša i savjetnika da uklone savjetovanje kako sebi, tako i dodijeljenom studentu.

Prema zahtjevu 8, svim brucošima ispisat će se savjetnici koji nemaju popunjenu kvotu od 3 dodijeljena brucoša te im profil nije obrisan. Svim savjetnicima ispisat će se brucoši koji su im poslali zahtjev za savjetovanjem, a nemaju dodijeljenog savjetnika.

Svaki brucoš, prema zahtjevu 9 može poslati zahtjeve neograničenom broju savjetnika, ako ga već nema dodijeljenog, a savjetnik može prihvaćati i odbijati svaki dobiveni zahtjev, dokle god im nije popunjena kvota.

Zahtjev broj 10 omogućava označavanje profila obrisanim, pri čemu više neće biti vidljiv ostalim korisnicima.

Prema zahtjevu 11 svaki prijavljeni korisnik može pregledavati teme i komentare u forumu te vidjeti kada su dodani i tko ih je dodao.

Pri dodavanju tema u forum iz zahtjeva 12, neophodno je popuniti polje o naslovu i opisu teme. Mogući su sljedeći scenariji:

a) Korisnik nije popunio polje naslova i/ili opisa. Prikazuje se prozor s porukom o obveznom popunjavanju polja i tema se ne dodaje.
b) Korisnik je popunio oba polja, ali tema s istim naslovom već postoji. Prikazuje se poruka da tema već postoji i tema se ne dodaje.
c) Korisnik je ispravno popunio sva polja pri čemu tema ne postoji u bazi podataka. Tema se dodaje.

U skladu sa zahtjevom 13, omogućuje se dodavanje komentara na temu, pri čemu se uz komentar dodaju podaci o vremenu objave te imenu i ulozi autora.

Prema zahtjevu 14, aplikacija sadrži često postavljena pitanja o fakultetu i samoj aplikaciji kako bi se izbjeglo ponavljanje pitanja u forumu.

Zahtjev 15 odnosi se na mogućnost odjave iz sustava klikom gumba u svakom trenutku, čime se sesija prekida i korisnik se preusmjerava na obrazac za prijavu.

5.2. Stvaranje React aplikacije

Početna verzija aplikacije kreirana je pokretanjem *npx* naredbe prikazane na slici 5.1. pri čemu „savjetovanje“ predstavlja naziv aplikacije. Alat *npx* se instalira s *npm* uslužnim programom te služi za pokretanje Node.js paketa. Pri tome, *npx* privremeno preuzima paket s interneta te ga

izvršava [30]. Paket *create-react-app* generira jednostavnu React aplikaciju te odrađuje instalaciju i konfiguraciju webpack i Babel alata umjesto programera [31].

```
npx create-react-app savjetovanje
```

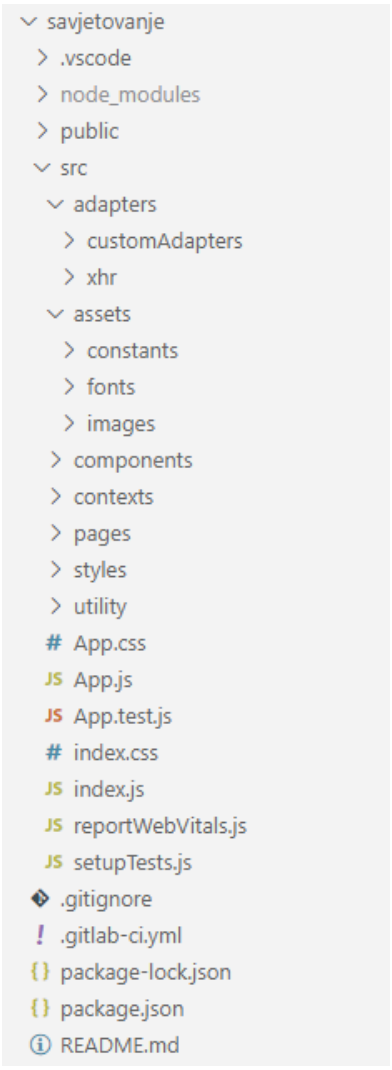
Slika 5.1. Kreiranje polazne React aplikacije

Aplikacija koju *create-react-app* kreira ima strukturu datoteka prikazanu na slici 5.2. Mnoge datoteke nisu potrebne zbog izmjene izvornog koda te ih je potrebno izbrisati ili izmijeniti.

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    ├── reportWebVital.js
    └── setupTests.js
```

Slika 5.2. Početna struktura mapa i datoteka

Nakon svih potrebnih izmjena, arhitektura projekta poprima oblik prikazan na slici 5.3. Direktorij *src* sadržava mape *adapters*, *assets*, *components*, *contexts*, *pages*, *styles*, *utility*. *Adapters* sadržava mapu *xhr* u kojoj je JavaScript kod zadužen za osnovno spajanje na Mongo Realm, a *customAdapters* sve konkretne funkcije koje se povezuju na pojedinačnu kolekciju u bazi podataka i rade samo jednu operaciju nad njome. Direktorij *contexts* sadržava React kontekste koji omogućuju dostupnost podataka kroz cijelu aplikaciju bez slanja podataka kroz cijelo stablo komponenti. Direktorij *pages* sadržava stranice koje su samo jednom iskorištene, a sastoje se od više komponenti. Direktorij *styles* sadržava prilagođene CSS datoteke za oblikovanje pojedinih stranica. Direktorij *utility* sadržava funkcije koje odrađuju pomoćni posao koji nije usko vezan uz konkretne komponente.



Slika 5.3. Prikaz konačne arhitekture aplikacije

5.3. Spajanje na bazu podataka

Spajanje na Mongo bazu podataka odrađuje se preko Mongo Realm aplikacije koja autorizira korisnika i omogućava upite. Ona pruža API s kojim se komunicira slanjem HTTP zahtjeva za koji je zadužen Axios [32]. Potrebno je obaviti prijavu na Realm aplikaciju koja je prikazana na slici 5.4., a budući da se radi o asinkronoj operaciji, funkcija vraća JavaScript Promise. Axios šalje GET zahtjev prema poslužitelju na lokaciju za prijavu te poslužitelj odgovara s pristupnim tokenom.

```
export function mongoLogin(realmApp) {
  return new Promise((resolve, reject) => {
    axios
      .get(
        `https://eu-central-
1.aws.realm.mongodb.com/api/client/v2.0/app/${realmApp}/auth/providers/ano
n-user/login`,
        {},
        { headers: { "Content-Type": "application/json" } }
      )
      .then((res) => {
        resolve(res);
      });
  });
}
```

Slika 5.4. Funkcija za prijavu

Nakon prijave, moguće je slati upite prema GraphQL API-ju. Funkcija koja obavlja slanje upita prikazana je na slici 5.5. Axios šalje POST zahtjev prema GraphQL API-ju koji sadržava pristupni token dobiven mongoLogin() funkcijom i upit.

```
export function mongoQuery(access_token, realmApp, query, source) {
  return new Promise((resolve, reject) => {
    var endpoint = `https://eu-central-
1.aws.realm.mongodb.com/api/client/v2.0/app/${realmApp}/graphql`;
    var token = "Bearer " + access_token;
    axios
      .post(
        endpoint,
        { query: query },
        {
          headers: {
            "Content-Type": "application/json",
            Authorization: token,
          },
          cancelToken: source.token,
        }
      )
      .then((res) => {
        resolve(res);
      })
      .catch((rejectReason) => console.log(rejectReason));
  });
}
```

Slika 5.5. Funkcija za slanje upita

Jedino što je preostalo je napisati GraphQL upit, primjer jednog nalazi se u funkciji za dohvaćanje tema foruma prikazanoj na slici 5.6.

```
export async function fetchForumTopics(token) {
  const query = `query {forumTopics { _id, comments {content date author{n
ame, role}} dateCreated, description title } }`;
  let queryResult = await mongoQuery(token, realmApp, query, sourceLogin);
  if (queryResult.data.data !== null) {
    return queryResult.data.data.forumTopics;
  } else return false;
}
```

Slika 5.6. Funkcija za slanje upita

U deklaraciji funkcije se koristi ključna riječ *async* kako bi se mogla koristiti ključna riječ *await* kojom se čeka odgovor od poslužitelja, odnosno izvršavanje se pauzira dok se obećanje koje vraća *mongoQuery()* ne postavi kao riješeno. U funkciji se definira upit koji se prosljeđuje *mongoQuery()* funkciji s pristupnim tokenom.

5.4. Upravljanje rutama

Za upravljanje rutama koristi se *ReactRouter* i uvjetno prikazivanje u korijenskoj komponenti App, što slika 5.7. i prikazuje.

```
if (id) {
    return (
        <Router basename={Routes.BASENAME}>
          <Route path="/" component={CustomNavbar} />
          <Route exact path={Routes.HOME} component={Cards} />
          <Route exact path={Routes.ADVISOR_FORM} component={AdvisorForm} />
          <Route exact path={Routes.PROFILE} component={ProfilePage} />
          <Route exact path={Routes.FAQ} component={FAQ} />
          <Route exact path={Routes.FORUM} component={Forum} />
          <Route exact path={Routes.FORUM_TOPIC} component={ForumTopic} />
        </Router>
    );
  } else {
    return (
      <Router basename={Routes.BASENAME}>
        <Route exact path={Routes.HOME} component={LoginPage} />
        <Route exact path={Routes.REGISTER} component={RegisterPage}/>
      </Router>
    );
  }
```

Slika 5.7. Upravljanje rutama

Prvo se provjerava je li postavljen *id*, odnosno je li korisnik prijavljen. Ako je, onda se za početnu stranicu učitava komponenta *Cards* koja predstavlja svojevrstan izbornik te je moguće učitavanje svih ostalih komponenti dostupnih prijavljenim korisnicima. Uz pojedinu komponentu, pristupanjem svakoj ruti koja sadržava „/“ učitava se i navigacijska traka. Kako bi se to omogućilo, izostavlja se ključna riječ *exact*. Kada bi ju ta *Route* komponenta sadržavala, navigacijska traka učitavala bi se samo na početnoj stranici. Suprotan pristup korišten je za prikazivanje početne stranice. Kada se ne bi koristila ključa riječ *exact*, pristupanjem svakoj ruti učitavala bi se i komponenta *Cards*. Ako korisnik nije prijavljen, učitava se *LoginPage* komponenta te je omogućeno prikazivanje *RegisterPage* komponente. Ako neprijavljeni korisnik pokuša pristupiti komponentama kojima ne bi trebao imati pristup, one se neće ni učitati bez obzira kako ruta glasila.

5.5. Registracija i prijava korisnika

Stranice za registraciju i prijavu slično izgledaju te obje koriste komponente obrazaca. Korištene su *Form*, *FormControl*, *FormGroup* i *FormLabel* komponente iz react-bootstrap paketa. Polja koja se popunjavaju kontroliraju se korištenjem *useState() Hook* funkcije, kojoj se može predati početno stanje te ona vraća stanje i funkciju koja ga ažurira. Prilikom svake promjene polja za unos, poziva se funkcija za ažuriranje stanja kojoj se predaje vrijednost *target* elementa, odnosno reference na objekt koji je okinuo događaj. Na slici 5.8. prikazan je primjer ažuriranja stanja.

```
<FormControl
 value={email}
 placeholder="ime.prezime@student.ferit.hr"
 onChange={(e) => setEmail(e.target.value)}
/>
```

Slika 5.8. Komponenta obrasca s onChange svojstvom

Prilikom registracije, kada se obrazac podnosi, poziva se *handleSubmit()* funkcija koja poziva *validateForm()* te ako obrazac prođe validaciju ubacuje korisnika u bazu podataka. Validacija provjerava sve uvjete prethodno definirane u specifikaciji zahtjeva, poput ispravne adrese elektroničke pošte i duljine zaporke. Prije ubacivanja korisnika, provjerava se i postoji li već račun s tom adresom elektroničke pošte. Ako postoji, registracija se ne odrađuje. Kao što slika 5.9. prikazuje, nad zaporkom odrađuje se soljenje (engl. *salting*) te se izračunava *hash* vrijednost uz pomoć *bcrypt* paketa i ona se umeće u bazu podataka. Ulaznoj vrijednosti će se *hash* vrijednost izračunavati u 256 (2^8) rundi, a što je broj rundi veći, vrijednost zaporke je sigurnija te ju je teže otkriti [33].

```
let hash = bcrypt.hashSync(password, bcrypt.genSaltSync(8));
```

Slika 5.9. *Hash* zaporke

Prilikom prijave, provjerava se je li neko od polja prazno te ako nije, iz baze se dohvaćaju korisnički podaci koji odgovaraju toj adresi elektroničke pošte. Ako postoji korisnik s važećim računom, *hash* vrijednost iz baze podataka uspoređuje se s zaporkom koju je korisnik unio pomoću *bcrypt* paketa. Nakon toga, korisnik je autoriziran.

5.6. Izmjena korisničkih podataka

Korisnik može mijenjati podatke koje je unio upitnikom, a izmjenom se smatra i prvi unos. Prvo je potrebno povući postojeće podatke iz baze te ih prikazati unutar upitnika. Taj dio odrađuje se pomoću *useEffect() Hook* funkcije, koja pokreće svoj prvi argument, funkciju efekta, prilikom svakog ponovnog prikazivanja komponente i to samo ako je došlo do promjene vrijednosti u polju ovisnosti (engl. *dependency array*). Funkcija *useEffect()* izvršava nuspojave u funkcijskim komponentama, poput dohvaćanja podataka iz baze. Nuspojave se pokreću nakon što je komponenta završila učitavanje te se time ne blokira samo učitavanje. Polje ovisnosti je drugi, izborni argument pri pozivu *useEffect()* funkcije. U njemu se trebaju nalaziti sve varijable korištene unutar efekta, ako su definirane izvan njega. To prikazuje i efekt na slici 5.10. koji se izvodi prilikom učitavanja upitnika. Unutar *useEffect() Hook* funkcije definirana je asinkrona *fetchData()* funkcija koja obavlja dohvaćanje tokena i podataka o korisniku. Ako je korisnik pronađen, a trebao bi biti jer je prijavljen, podaci iz baze spremaju se u stanje. Ako se dogodi da je korisnik jednak *null*, došlo je do pogreške prilikom spajanja na bazu. Ako nekog podatka nema u bazi, što se uvijek dogodi prilikom prvog popunjavanja, njegova vrijednost se postavi na zadanu, što je, ovisno o vrsti podatka, prazan *string*, prazno polje ili *false* vrijednost. U polju ovisnosti nalazi se jedna varijabla koja je definirana izvan efekta i jedna funkcija, jer JavaScript tretira funkcije kao objekte prve klase, odnosno isto kao i varijable.

```
useEffect(() => {
  async function fetchData() {
    let token = await getAccessToken("users");
    let user = await fetchUserData(loggedUserId, token);
    if (user) {
      setState({
        _id: loggedUserId,
        residence: user.residence || "",
        hobbies: user.hobbies || [],
        study: user.study || "",
        education: user.education || "",
        dorm: user.dorm || false,
        knowledge: user.knowledge || false,
      });
    } else throw new Error("Failed to connect to database.");
  }

  try {
    fetchData();
  } catch (error) {
    console.log(error.message);
  }
}, [loggedUserId, getAccessToken]);
```

Slika 5.10. Korištenje *useEffect()* Hook funkcije

Nakon postavljanja početnih vrijednosti stanja, korisnik ih može izmijeniti, no svako polje, izuzev *checkbox* polja, mora biti popunjeno prilikom podnošenja obrasca. Izmjena polja obavlja se na način prikazan na slici 5.11. Prvo se provjerava je li tip polja koje je okinulo događaj *checkbox*. Ako je, ono nema *value* svojstvo nego *checked* koje pokazuje je li ono označeno te se ta vrijednost sprema u stanje.

```
const handleChange = (event) => {
    const value =
      event.target.type === "checkbox"
        ? event.target.checked
        : event.target.value;
    setState({
      ...state,
      [event.target.name]: value,
    });
  };
```

Slika 5.11. Rukovanje promjenom

Hobiji, koji su polje, izmjenjuju se na nešto drugačiji način, zbog svog tipa podatka. Način je prikazan na slici 5.12. Prvo se hobiji izdvajaju iz stanja te se provjerava uključuje li popis dodanih hobija već tu vrijednost. Ako ne uključuje, korisnik ju dodaje na popis. U suprotnom, uklanja ju s popisa. Uklanjanje se odrađuje pronalaskom indeksa te vrijednosti te se nad poljem pozove *splice()* koji uklanja jedan element počevši od elementa na indeksu iz prvog argumenta. Zatim se hobiji u stanju postavljaju na vrijednost izmijenjenog polja hobija.

```
const handleHobbiesChange = (event) => {
    const hobby = event.target.value.toLowerCase();
    const selectedCheckboxes = [...state.hobbies];
    const isChecked = selectedCheckboxes.includes(hobby);
    if (!isChecked) {
      selectedCheckboxes.push(hobby);
    } else {
      selectedCheckboxes.splice(selectedCheckboxes.indexOf(hobby),1);
    }
    setState({
      ...state,
      hobbies: selectedCheckboxes,
    });
  };
```

Slika 5.12. Rukovanje promjenom polja

5.7. Uklanjanje korisnika

Uklanjanje korisnika odrađuje se kao logičko ili meko brisanje. Dokument u kolekciji označava se obrisanim, iako podaci ostaju u bazi podataka i nisu zauvijek izgubljeni. Tako, ako se iz nekog razloga želi doći do podataka ili vratiti korisnički račun, to nije nemoguće. S druge strane, taj postupak djelomično otežava kodiranje jer je uvijek unutar koda ili samog upita potrebno provjeriti je li dokument označen kao obrisan. Funkcija koja obavlja brisanje prikazana je na slici 5.13.

```
  async function setUserDeleted() {
    let token = await getAccessToken("users");
    let result = await updateDeletedUser(loggedUserId, token);
    if (result) {
      logOut();
    } else setQueryError(true);
  }
```

Slika 5.13. Uklanjanje korisnika

Unutar nje se poziva pomoćna funkcija koja izmjenjuje dokument u kolekciji. Slika 5.14. prikazuje posljedično poništavanje dodijeljenih savjetovanja i postavljanje zastavice *deletedAccount*.

```
export async function updateDeletedUser(loggedUserId, token) {
  const queryLoggedUser = `mutation{updateOneUser(query: {_id:"${loggedUserId
}"},set:{ advisements:{link: []}, deletedAccount:true }){deletedAccount}}`;
  let queryResult = await mongoQuery(
    token,
    sRealmApp,
    queryLoggedUser,
    sourceLogin
  );
  if (queryResult.data.data.updateOneUser !== null) {
    return true;
  } else {
    return false;
  }
}
```

Slika 5.14. Postavljanje zastavice obrisanog korisnika

Ako izmjena uspije, korisnika se odjavljuje iz sustava. Također, kada se dohvaćaju savjetovanja korisnika, među kojima je netko s obrisanim računom, to savjetovanje uklanja se iz polja. Budući da MongoDB nema mogućnost CASCADE DELETE kao što imaju SQL baze, logičkim brisanjem se ukida trošak izmjene svakog pojedinačnog dokumenta koji sadržava referencu na obrisanog korisnika.

5.8. Slanje zahtjeva

Mogućnost slanja zahtjeva imaju brucoši koji dodaju željene savjetnike s obzirom na informacije koje imaju o njima. Prije samog slanja zahtjeva, efekt komponente dohvaća dostupne savjetnike.

```
  if (
        (user.role === "brucos" && nondeletedAdvisements.length === 0) ||
        (user.role === "savjetnik" && nondeletedAdvisements.length < 3)
      ) {
        const availableStudents = await fetchAvailableStudents(user, token);
        if (availableStudents) {
          setAvailablePeople(() => [...availableStudents]);
        } else setQueryError(true);
      }
```

Slika 5.15. Dohvaćanje dostupnih studenata

Pri tome, prema slici 5.15., brucoši ne smiju imati dodijeljenog savjetnika. Kod dohvaćanja dostupnih savjetnika ne dohvaćaju se studenti kojima je zahtjev poslan, a nisu ga odbili ni prihvatili te se provjerava ima li kandidat za dostupnog savjetnika obrisan profil ili popunjenu kvotu savjetovanja.

Nakon prikazivanja dostupnih savjetnika, brucoš može svakome od njih poslati zahtjev. Ako klikne gumb za dodavanje, poziva se funkcija sa slike 5.16. uz identifikator savjetnika kao argument. On se dodaje u polje dodanih savjetnika te se izmjena radi i u bazi podataka. Zatim se on uklanja s popisa dostupnih studenata.

```
async function sendRequest(advisorId) {
    state.sentAdvisementRequests.push(advisorId);
    let token = await getAccessToken("users");
    let result = await updateSentRequests(
      loggedUserId,
      state.sentAdvisementRequests,
      token
    );
    if (result) {
      setAvailablePeople(
        availablePeople.filter((item) => item._id !== advisorId)
      );
    } else setQueryError(true);
  }
```

Slika 5.16. Slanje zahtjeva

5.9. Prihvaćanje i odbijanje zahtjeva

Mogućnost prihvaćanja ili odbijanja zahtjeva ima savjetnik. Prvo se dohvaćaju studenti koji su poslali zahtjev prijavljenom savjetniku te se njihovi podaci postavljaju na mjesto izbora ako nemaju savjetnika.

Ako savjetnik prihvati zahtjev, brucošu se zahtjev miče iz poslanih zahtjeva te mu se savjetnik dodaje na popis savjetovanja. Također se brucoš dodaje savjetniku na popis savjetovanja, što prikazuje slika 5.17. Zatim se podaci unose u bazu i radi se dva upita, jedan za svakog studenta. Na kraju se student uklanja s popisa dostupnih brucoša.

Ako savjetnik odbije zahtjev, jedini događaj je uklanjanje zahtjeva iz popisa poslanih zahtjeva kod brucoša, kako je i prikazano na slici 5.18. Zatim se izmijenjeni popis poslanih zahtjeva unosi u bazu podataka. I u ovom slučaju student se uklanja s popisa dostupnih brucoša koji su poslali zahtjev savjetniku.

```
async function acceptRequest(freshmanUser) {
  const filteredRequests = freshmanUser.sentAdvisementRequests.filter(
    (id) => id !== loggedUserId
  );
  freshmanUser = {
    ...freshmanUser,
    sentAdvisementRequests: filteredRequests,
  };
  freshmanUser.advisements.push(state);
  state.advisements.push(freshmanUser);
  let token = await getAccessToken("users");
  let result = await updateAfterRequestAcceptance(
    filteredRequests,
    freshmanUser,
    state,
    token
  );
  if (result) {
    setAvailablePeople(
      availablePeople.filter((item) => item._id !== freshmanUser._id)
    );
  } else setQueryError(true);
}
```

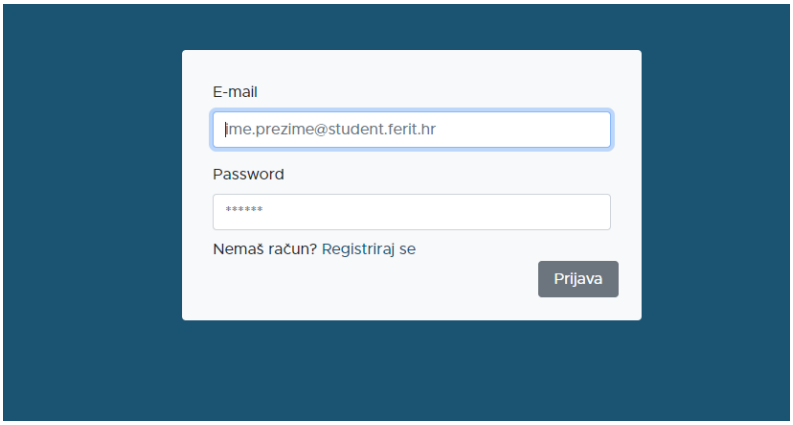
Slika 5.17. Prihvaćanje zahtjeva

```
async function declineRequest(freshmanUser) {
  const filteredRequests = freshmanUser.sentAdvisementRequests.filter(
    (id) => id !== loggedUserId
  );
  freshmanUser = {
    ...freshmanUser,
    sentAdvisementRequests: filteredRequests,
  };
  let token = await getAccessToken("users");
  let result = await updateDeclinedRequest(
    filteredRequests,
    freshmanUser._id,
    token
  );
  if (result) {
    setAvailablePeople(
      availablePeople.filter((item) => item._id !== freshmanUser._id)
    );
  } else setQueryError(true);
}
```

Slika 5.18. Odbijanje zahtjeva

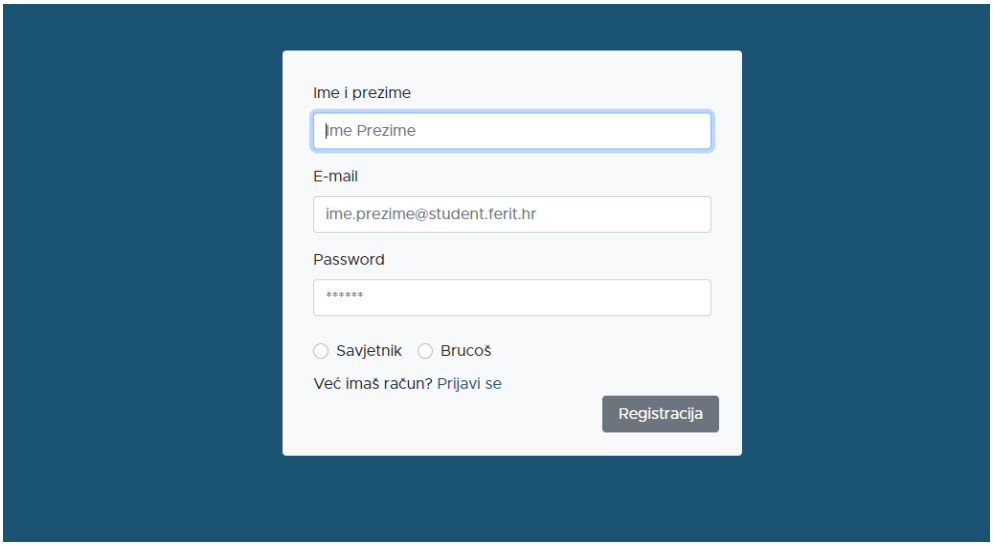
6. IZGLED APLIKACIJE

Početna stranica kada korisnik nije prijavljen je stranica za prijavu u sustav i prikazana je na slici 6.1. Indicira oblik adrese elektroničke pošte s kojim se korisnici prijavljuju te minimalnu duljinu zaporke. Sadržava i poveznicu na stranicu za registraciju te gumb za prijavu.



Slika 6.1. Stranica prijave

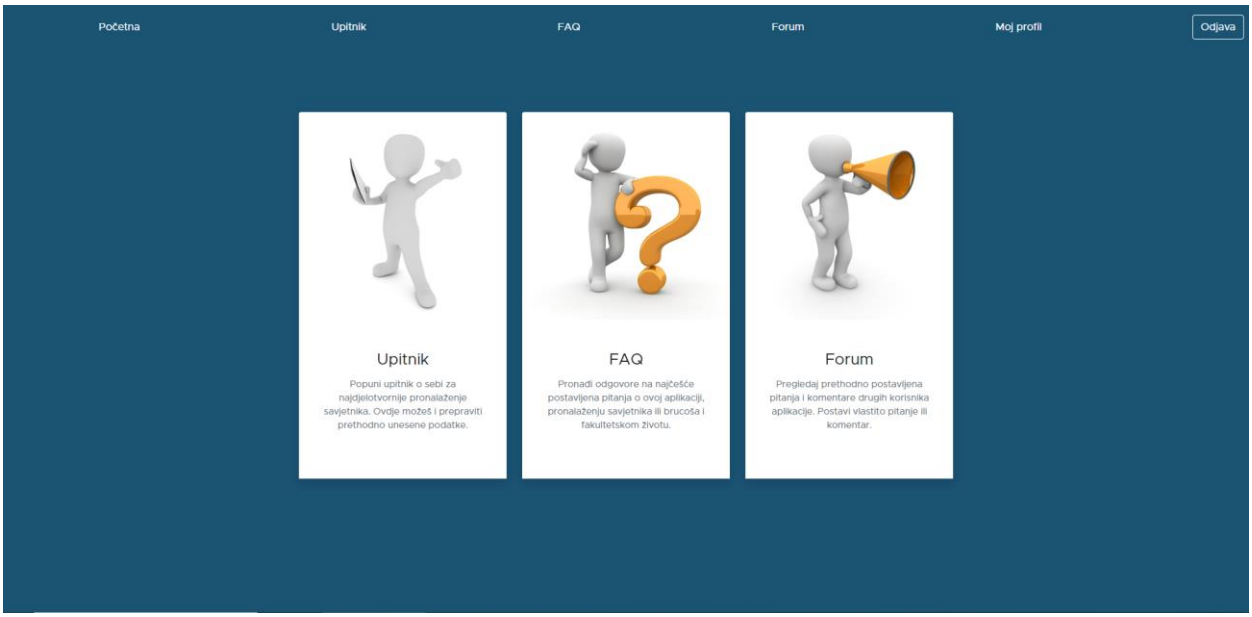
Klikom na „Registriraj se“ korisnik se preusmjerava na stranicu za registraciju u sustav prikazanu na slici 6.2. te se od njega zahtijeva unošenje imena i prezimena, adrese elektroničke pošte te uloge brucoša ili savjetnika. Ako se neki od podataka izostavi, na ekranu se pojavi poruka koja o tome obavještava korisnika. Uz to, stranica omogućava preusmjeravanje na stranicu za prijavu.



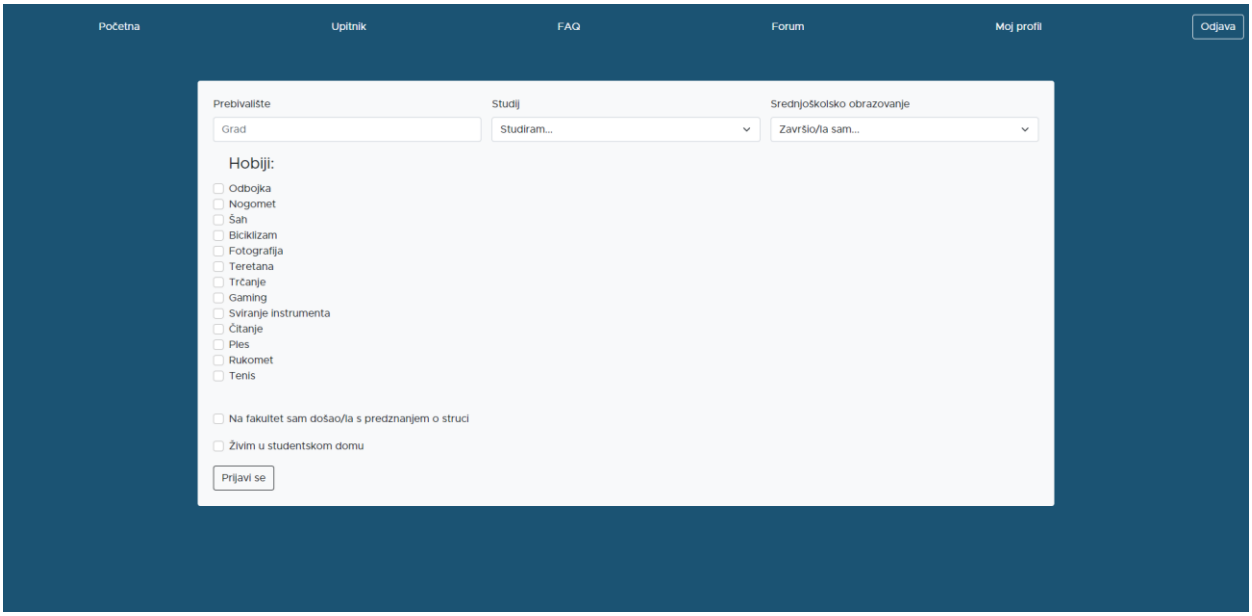
Slika 6.2. Stranica registracije

Prethodno spomenute stranice moguće je otvoriti samo ako korisnik nije prijavljen u sustav.

Nakon prijave korisniku se pojavljuje izbornik i navigacijska traka (slika 6.3.). Izbornik omogućuje preusmjeravanje na upitnik, često postavljena pitanja (engl. *FAQ*) te forum. Uz to, navigacijska traka omogućava i odlazak na profil korisnika te odjavu.



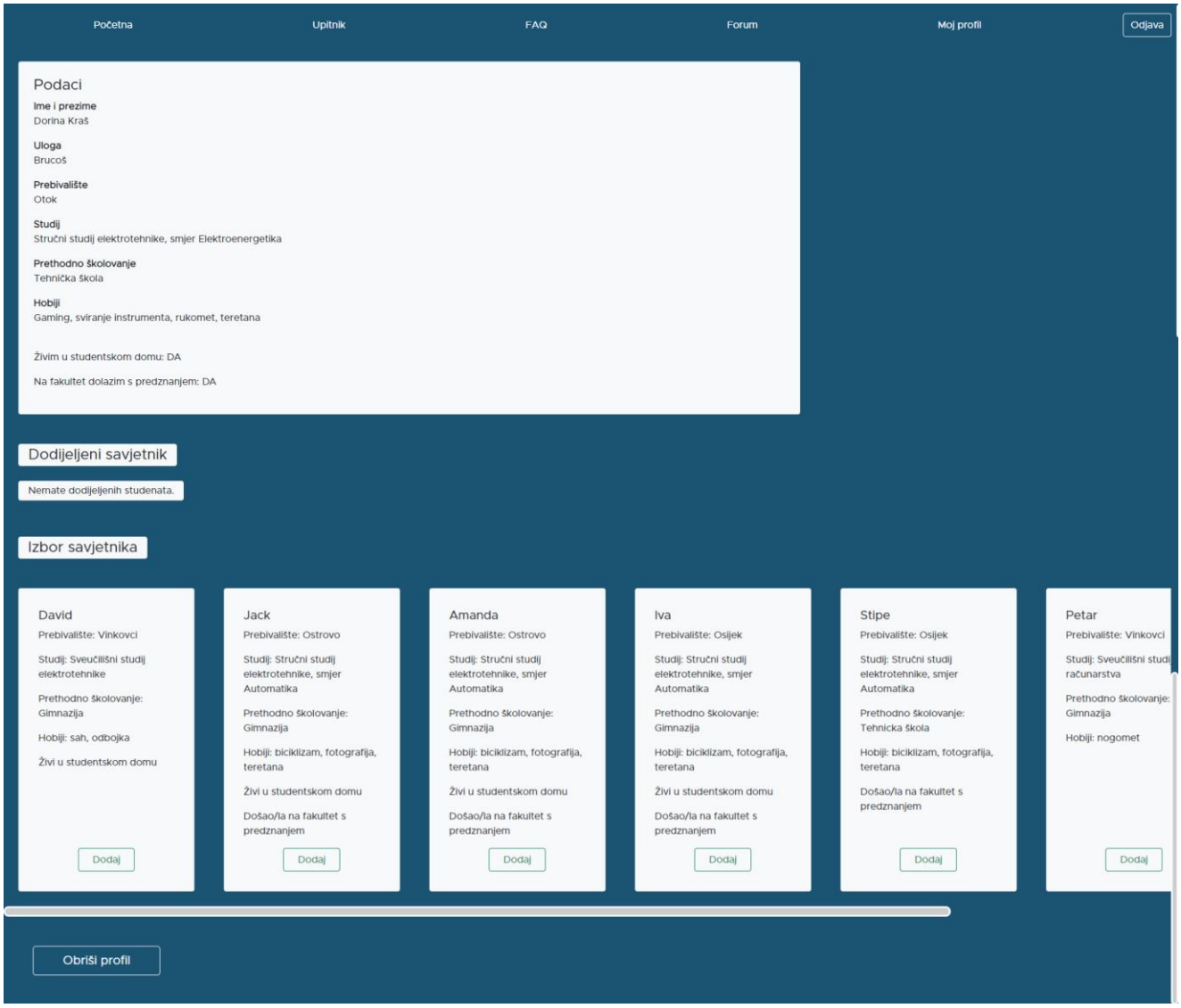
Slika 6.3. Izbornik



Slika 6.4. Upitnik

Stranica Upitnik učitava obrazac koji sadržava polja koja je obvezno ispuniti, poput prebivališta, obrazovanja i studija (slika 6.4.). Ostale podatke nije obvezno unijeti. Klikom na „Prijavi se“ podaci se unose u bazu, a korisnik se preusmjerava na početnu stranicu. Ako unošenje podataka u bazu nije uspjelo, ispisuje se poruka o tome.

Slika 6.5. prikazuje stranicu „Moj profil“. Na stranici se nalaze podaci o korisniku uneseni tijekom registracije i slanja obrasca. Stranica prikazuje i dodijeljene studente te dostupne studente, ako takvi postoje. S obzirom je li prijavljeni student savjetnik ili brucoš mijenjaju se oznake i gumbovi na stranici.

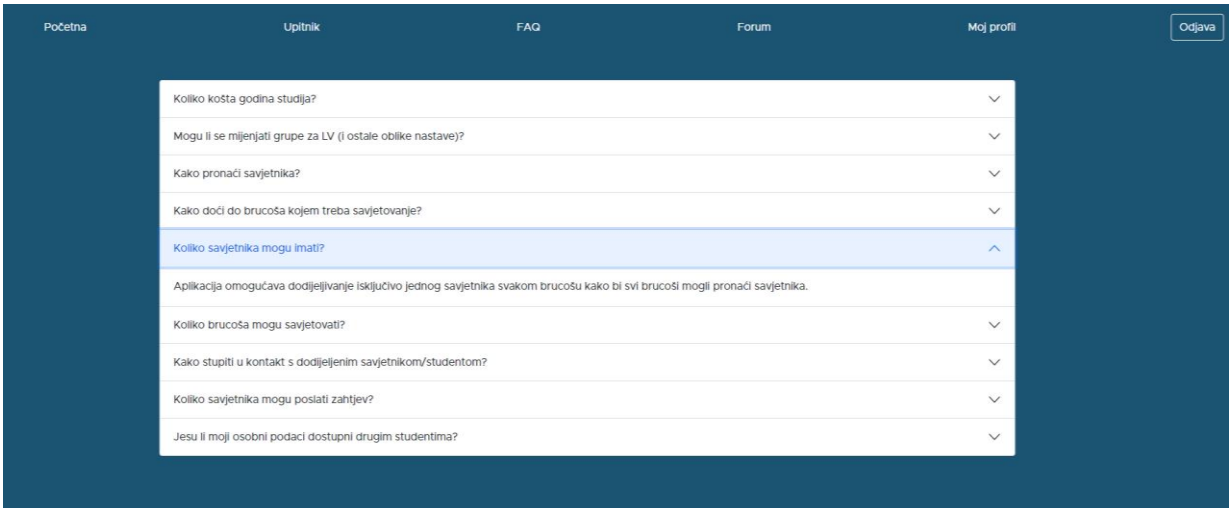


Slika 6.5. Stranica korisničkog profila

Savjetnici kod kartica za izbor brucoša imaju gumbove „Prihvati“ i „Odbij“, dok brucoši imaju samo gumb „Dodaj“. Kartice dodijeljenih studenata imaju gumb „Ukloni“, neovisno o ulozi

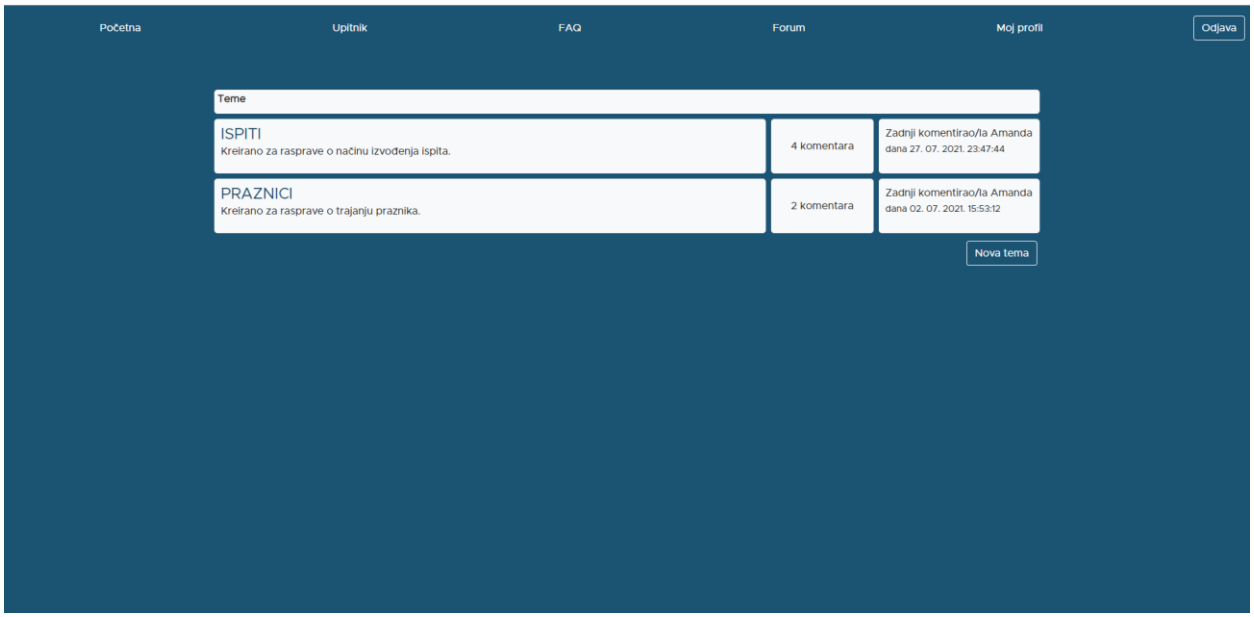
studenta, a same kartice sadržavaju ime i prezime te adresu elektroničke pošte. Ako dodijeljenih studenata nema, ispisuje se adekvatna poruka o tome, što se događa i kada nema dostupnih savjetnika ili kada savjetnik nema primljenih zahtjeva od brucoša. Na stranici se nalazi i gumb za brisanje profila.

Slika 6.6. prikazuje stranicu često postavljenih pitanja. Na njoj se nalaze pitanja iz baze podataka, a odgovori se prikazuju klikom na pitanje. Ponovnim klikom, odgovor se sakriva.



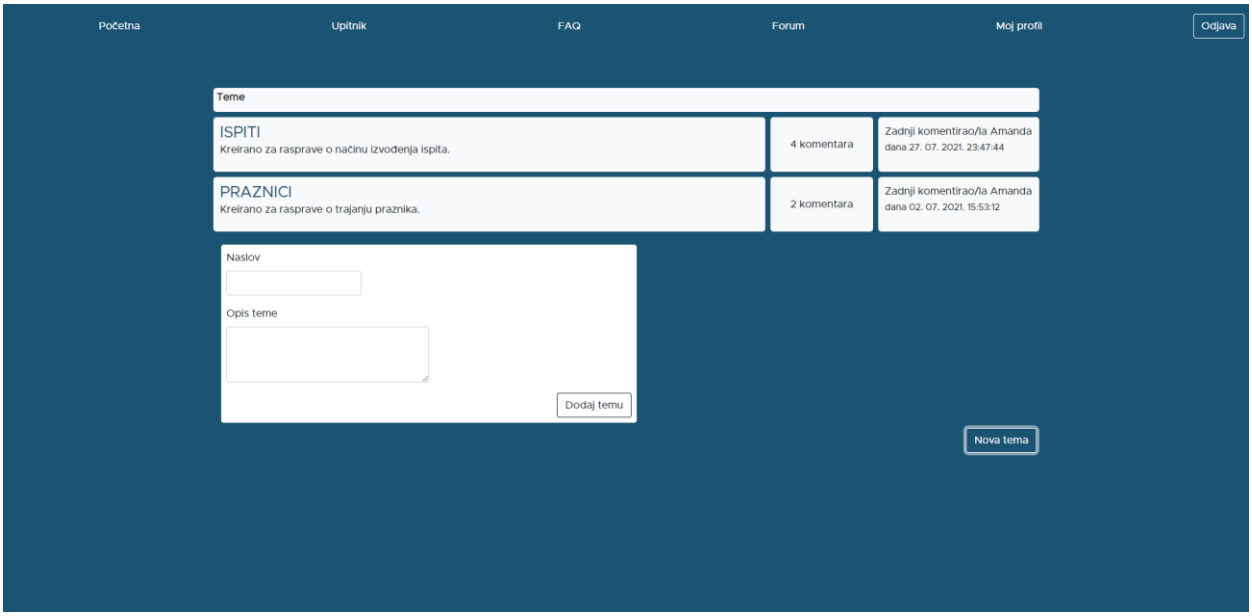
Slika 6.6. Često postavljena pitanja

Slika 6.7. prikazuje popis tema na forumu. Sadržava i količinu komentara te tko je i kada ostavio zadnji komentar. Klikom na naslov otvara se tema s komentarima. Klikom na gumb „Nova tema“ prikazuje se obrazac za unos teme i opisa prikazan na slici 6.8.



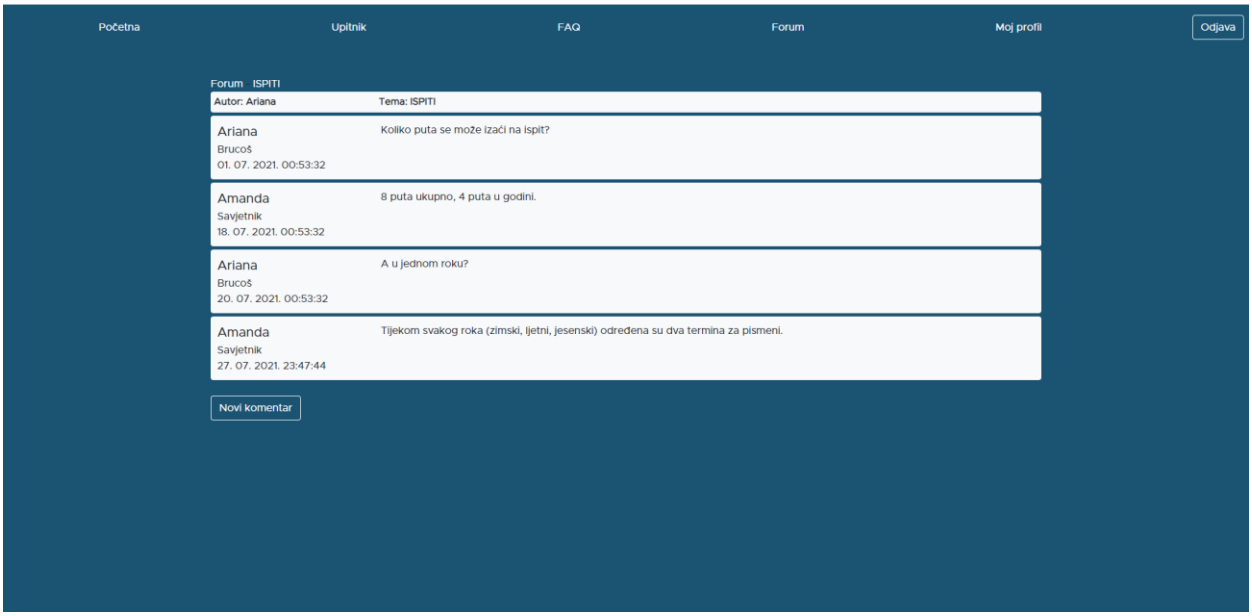
Slika 6.7. Forum

Ako tema i opis nisu prazni te tema već ne postoji, tema se može dodati.



Slika 6.8. Dodavanje nove teme

Kada se otvori tema, prikazuje se autor teme te komentari uz autora, njegovu ulogu i datum objave (slika 6.9.). Uz to, svakom korisniku omogućeno je i dodavanje novog komentara.



Slika 6.9. Tema foruma

7. ZAKLJUČAK

Zadatak završnog rada bio je razviti web aplikaciju koja omogućava povezivanje i savjetovanje studenata koristeći JavaScript programski jezik, React biblioteku i MongoDB bazu podataka. Aplikacija bi koristila brucošima Fakulteta elektrotehnike, računarstva i informacijskih tehnologija u Osijeku uz pomoć starijih studenata u ulozi savjetnika.

Na početku rada spomenute su korištene tehnologije te je napravljen osvrt na postojeća rješenja. Postojeća rješenja nisu dostupna studentima Fakulteta te se ukazuje potreba za ovakvom aplikacijom. Zatim je obrađena teorijska podloga vezana uz JavaScript jezik i biblioteku React te su predstavljeni njihovi glavni koncepti i nove mogućnosti, od kojih je većina korištena u aplikaciji. Rad zatim opisuje ključne postupke pri realizaciji zadatka, uključujući spajanje na bazu podataka, upravljanje rutama, registraciju i prijavu korisnika, izmjenu podataka te manipulacije zahtjevima. Svi ti postupci neophodni su za ispravan rad aplikacije te je time ostvaren zadatak rada. Korisnici mogu tražiti savjetnike ili biti na raspolaganju kao savjetnici, postavljati teme, pitanja i komentare u forumu ili potražiti odgovore na svoja pitanja u često postavljenim pitanjima. Svim korisnicima u svakom trenutku moguće je i odustati od savjetovanja ili pak obrisati korisnički račun.

Aplikacija se može proširiti tako da korisnici manipuliraju osobnim podacima koji će se prikazivati dodijeljenom savjetniku/brucošu, tako što bi sakrili prikazivanje nekog od njih. S druge strane, mogli bi uvrstiti u prikazane podatke i broj telefona ili dodatnu adresu elektroničke pošte. Dodatan korak bio bi postavljanje nove aplikacije, povezane s postojećom, na stranu poslužitelja, koja bi slala elektroničku poštu svakom brucošu kada mu se dodijeli savjetnik s njegovim podacima. Može se dodati i mogućnost promjene zaporke ili uloge studenta. Dodatno proširenje bilo bi i omogućiti slanje privatnih poruka u stvarnom vremenu unutar same aplikacije.

LITERATURA

[1] „Differ - Helping you connect with other new students“, *Differ Chat*. [Online]. Dostupno na: http://www.differ.chat. [Pristupljeno: 2-7-2021]

[2] „SEA-EU Around, Aplikacije na Google Playu“. [Online]. Dostupno na: https://play.google.com/store/apps/details?id=com.sea_eu.around. [Pristupljeno: 2-7-2021]

[3] „Orientation Week 2021 | The College | The University of Chicago | The University of Chicago“. [Online]. Dostupno na: https://college.uchicago.edu/student-life/orientation-2021. [Pristupljeno: 2-7-2021]

[4] „EduHub: MEF - maturanti i studenti, Aplikacije na Google Playu“. [Online]. Dostupno na: https://play.google.com/store/apps/details?id=com.dominikbat.mef_baze&hl=hr. [Pristupljeno: 2-7-2021]

[5] „University of Michigan - Apps on Google Play“. [Online]. Dostupno na: https://play.google.com/store/apps/details?id=edu.umich.michigan&hl=en_US. [Pristupljeno: 2-7-2021]

[6] „About JavaScript - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript. [Pristupljeno: 26-6-2021]

[7] „JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Pristupljeno: 26-6-2021]

[8] „React – A JavaScript library for building user interfaces“. [Online]. Dostupno na: https://reactjs.org/. [Pristupljeno: 26-6-2021]

[9] „Introducing JSX – React“. [Online]. Dostupno na: https://reactjs.org/docs/introducing-jsx.html. [Pristupljeno: 26-6-2021]

[10] „Components and Props – React“. [Online]. Dostupno na: https://reactjs.org/docs/components-and-props.html. [Pristupljeno: 26-6-2021]

[11] „Introduction to Node.js“, *Node.js*. [Online]. Dostupno na: https://nodejs.dev/learn. [Pristupljeno: 26-6-2021]

[12] „What is npm?“, *Node.js*. [Online]. Dostupno na: https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/. [Pristupljeno: 26-6-2021]

[13] Ivica Lukić, „Baze podataka - MongoDB“. [Online]. Dostupno na: https://moodle.srce.hr/2020-2021/pluginfile.php/4468290/mod_resource/content/4/BP%2015%20Mongdb.pdf. [Pristupljeno: 1-7-2021]

[14] „What is MongoDB?“, *IBM*, 29-ožu-2021. [Online]. Dostupno na: https://www.ibm.com/cloud/learn/mongodb. [Pristupljeno: 1-7-2021]

[15] „Database Sharding: Concepts & Examples“, *MongoDB*. [Online]. Dostupno na: https://www.mongodb.com/features/database-sharding-explained. [Pristupljeno: 1-7-2021]

[16] „What is replication in MongoDB?“, *MongoDB*. [Online]. Dostupno na: https://www.mongodb.com/basics/replication. [Pristupljeno: 1-7-2021]

[17] I. Podnar Žarko, K. Pripužić, I. Lovrek, M. Kušek, *Raspodijeljeni sustavi*. Zagreb: Fakultet elektrotehnike i računarstva, 2016 [Online]. Dostupno na: https://www.fer.unizg.hr/_download/repository/Rassus-2016_udzbenik_v_1_3.pdf

[18] M. O. contributors Jacob Thornton, and Bootstrap, „About“. [Online]. Dostupno na: https://getbootstrap.com/docs/4.1/about/overview/. [Pristupljeno: 8-7-2021]

[19] „React-Bootstrap“. [Online]. Dostupno na: https://react-bootstrap.github.io/. [Pristupljeno: 8-7-2021]

[20] „JavaScript ES6“. [Online]. Dostupno na: https://www.w3schools.com/js/js_es6.asp. [Pristupljeno: 8-7-2021]

[21] „Template literals (Template strings) - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function. [Pristupljeno: 8-7-2021]

[22] „Destructuring assignment - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function. [Pristupljeno: 8-7-2021]

[23] „Spread syntax (...) - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax. [Pristupljeno: 8-7-2021]

[24] „Promise - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax. [Pristupljeno: 8-7-2021]

[25] „async function - JavaScript | MDN“. [Online]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function. [Pristupljeno: 9-7-2021]

[26] „Introducing Hooks – React“. [Online]. Dostupno na: https://reactjs.org/docs/hooks-intro.html. [Pristupljeno: 10-7-2021]

[27] „Using the Effect Hook – React“. [Online]. Dostupno na: https://reactjs.org/docs/hooks-effect.html. [Pristupljeno: 10-7-2021]

[28] „Handling Events – React“. [Online]. Dostupno na: https://reactjs.org/docs/handling-events.html. [Pristupljeno: 10-7-2021]

[29] „Conditional Rendering – React“. [Online]. Dostupno na: https://reactjs.org/docs/conditional-rendering.html. [Pristupljeno: 10-7-2021]

[30] „npm Blog Archive: Introducing npx: an npm package runner“. [Online]. Dostupno na: https://blog.npmjs.org/post/162869356040/introducing-npx-an-npm-package-runner. [Pristupljeno: 11-7-2021]

[31] „Getting Started | Create React App“. [Online]. Dostupno na: https://create-react-app.dev/docs/getting-started. [Pristupljeno: 11-7-2021]

[32] „Introduction to MongoDB Realm for Web Developers — MongoDB Realm“. [Online]. Dostupno na: https://docs.mongodb.com/realm/get-started/introduction-web/. [Pristupljeno: 11-7-2021]

[33] „bcryptjs: Documentation“, *Openbase*. [Online]. Dostupno na: https://openbase.com/js/bcryptjs/documentation. [Pristupljeno: 13-7-2021]

POPIS KRATICA

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
ECMA	European Computer Manufacturers Association
ES	ECMAScript
FAQ	Frequently asked questions
FERIT	Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JSX	JavaScript XML
RAM	Random Access Memory
SQL	Structured Query Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

POPIS SLIKA

SAŽETAK

U ovom radu razvijena je web aplikacija za povezivanje brucoša i starijih studenata s ciljem savjetovanja. Aplikacija omogućuje i komunikaciju s registriranim studentima putem foruma. Za izradu aplikacije ključne korištene tehnologije su JavaScript, biblioteka React i baza podataka MongoDB. Na početku, rad se osvrće na postojeća rješenja za povezivanje studenata. Zatim je promotrena potrebna teorijska podloga za uspješan razvoj aplikacije. Nakon toga, navedeni su i opisani najvažniji postupci u razvoju aplikacije koji omogućuju glavne funkcionalnosti. Na kraju je prikazan rad aplikacije počevši s prijavom ili registracijom korisnika. Nakon prijave u sustav, korisnik popunjava upitnik o vlastitim podacima te odlaskom na stranicu svog profila može vidjeti popis dostupnih savjetnika ili brucoša. Ako nema popunjenu kvotu, brucoš može slati zahtjeve za savjetovanjem, a savjetnik dobivene zahtjeve prihvatiti ili odbiti. Svaki korisnik može ostavljati pitanja i komentare na forumu ili pronaći odgovore u odjeljku često postavljenih pitanja.

Ključne riječi: brucoši, JavaScript, MongoDB, React, web aplikacija

ABSTRACT

Application for advising freshmen

This paper aimed to develop a web application, which will connect freshmen and senior students the latter of whom will advise the former group. The application also enables communication with registered students through a forum. The key technologies used to create the application are JavaScript, React library and MongoDB database. Firstly, the paper reviews existing solutions for connecting students. Secondly, the necessary theoretical framework for the successful application development was studied. Next, the most important procedures in the application development, which enable the main functionalities, are listed and elaborated on. Lastly, the application, starting with the login or registration of a user, is presented. After logging into the system, the user fills out a personal data questionnaire. Clicking on their profile page, they can see a list of available advisors or freshmen. If their quota is not filled, the freshman can send advisement requests, which can be accepted or rejected by the advisor. Each user can post questions and comments to the forum or find answers in the frequently asked questions section.

Keywords: freshmen, JavaScript, MongoDB, React, web application

ŽIVOTOPIS

Laura Mandić rođena je 27. prosinca 1999. u Osijeku. Pohađala je Osnovnu školu Nikole Tesle Mirkovci, nakon koje upisuje Tehničku školu Ruđera Boškovića Vinkovci, smjer tehnička gimnazija. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja sudjeluje u mnogim natjecanjima, uključujući županijska natjecanja iz matematike, informatike, engleskog jezika, hrvatskog jezika, kemije, biologije, badmintona i stolnog tenisa. Na državno natjecanje iz informatike plasira se tri puta te osvaja prvo mjesto 2016. godine, a zatim i 2018. godine. Nakon završetka srednje škole, upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Tijekom studija povremeno je zaposlena kao demonstratorica na kolegijima Osnove elektrotehnike 1, Programiranje 1, Programiranje 2, Objektno orijentirano programiranje te Digitalna elektronika. Na trećoj godini studija od Fakulteta prima Nagradu za uspješnost u studiranju.

Laura Mandić

PRILOZI

Priloženi disk sadržava:

1. Završni rad „Aplikacija za savjetovanje brucoša“ u .docx formatu
2. Završni rad „Aplikacija za savjetovanje brucoša“ u .pdf formatu
3. Programski kod aplikacije