Web aplikacija za objavu poslova

Balat, Matej

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

Stručni studij Elektrotehnika, smjer informatika

WEB APLIKACIJA ZA OBJAVU POSLOVA

Završni rad

Matej Balat

Osijek, 2020.

SADRŽAJ

1. UVOD

U ovom završnom radu opisane su funkcionalnosti i postupci izrade web aplikacije za objavu poslova. Funkcionalnosti ove web aplikacije omogućuju korisnicima pronalazak poslova, mogućnost izdvajanja poslova u dvije zasebne liste, mogućnost komentiranja i ocjenjivanja profila kompanije. Predviđena su dva korisnička profila. Profil posloprimca i profil poslodavca. Korisnik kao posloprimac uređuje svoj profil i pronalazi poslove koje naknadno može spremati u zasebne liste. Takve liste će olakšati pregled poslova. Posloprimac kreira svoj profil i profil svoje kompanije na koju dodaje poslove.

U drugom poglavlju opisan je pregled postojećih web stranica koje imaju približnu tematiku i temeljni cilj kao web aplikacija za objavu poslova. U ovom poglavlju utvrđuje se nedostatak funkcionalnosti i informacija koje nude postojeći web portali.

U trećem poglavlju opisane su sve tehnologije za izradu serverskog i korisničkog dijela ove web aplikacije. Međusobna komunikacija između serverske i korisničke strane je ključna u izradi web aplikacije. Pravilna struktura koda i odabir pogodnih tehnologija olakšava ostvarenje takve komunikacije. Opisane tehnologije i strukture omogućuju lakše pisanje i uređivanje koda.

U četvrtom poglavlju opisane su funkcionalnosti aplikacije uz primjere s korisničkog sučelja. U ovom poglavlju pojašnjeni su svi dijelovi korisničkog sučelja zbog jednostavnije uporabe web aplikacije. Uz svako tekstualno pojašnjenje priložena je i slika korisničkog sučelja.

1.1. Zadatak završnog rada

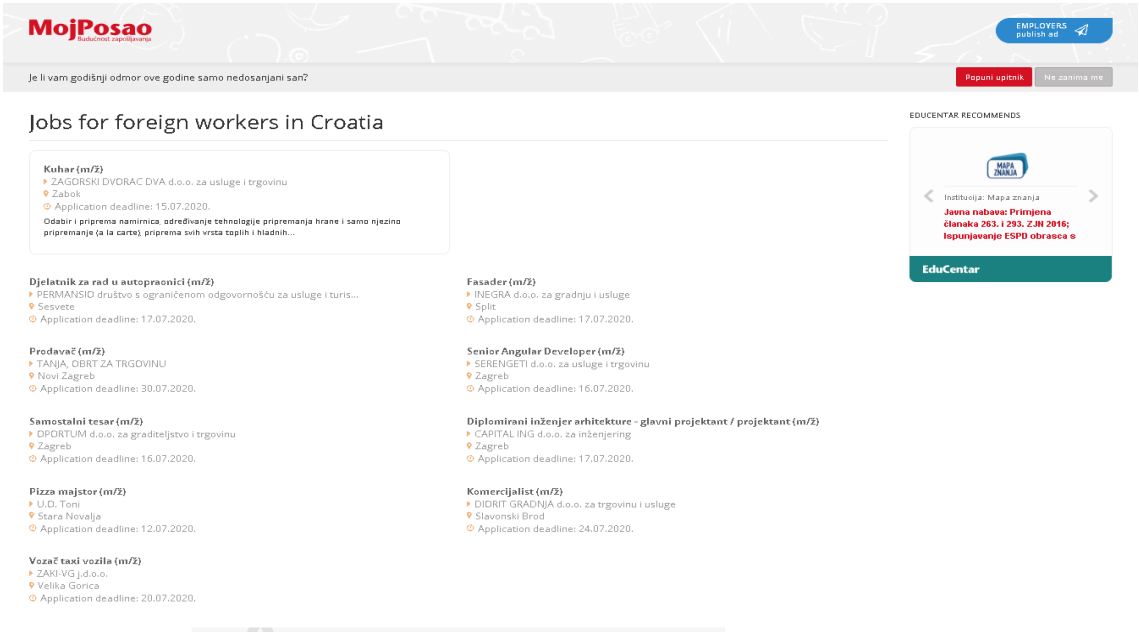
Opisati postupak ponude i potražnje poslova uz prednosti obavljanja navedenih akcija putem web aplikacije. Navesti funkcionalnosti koje treba imati web aplikacija pomoću koje se mogu ponuditi ili tražiti poslovi. Kreirati bazu podataka u kojoj će se pohraniti sve informacije. Aplikacija treba podržavati dva korisnička profila: poslodavac i posloprimac.

2. PREGLED POSTOJEĆIH RJEŠENJA

U ovom poglavlju navedene su prednosti i nedostatci već postojećih web portala. U nastavku je navedeno par primjeraka postojećih web stranica koje se mogu koristiti za kreiranje temeljnih funkcionalnosti.

2.1. Web stranica Moj Posao

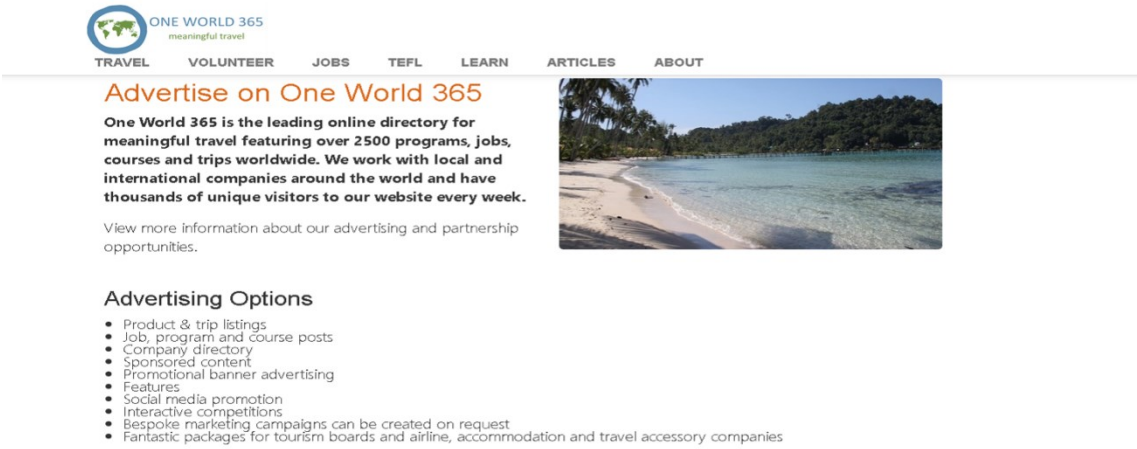
Moj Posao je web stranica koja nudi mnogo rješenja na tematiku zapošljavanja isključivo za zaposlenike u Republici Hrvatskoj [1]. U kolekciji poslova se nalazi 4000 različitih poslova koji su kategorizirani, podijeljeni po županijama i gradovima. Od 4000 poslova samo 10 poslova je kategorizirano za radnike iz inozemstva kako je prikazano na slici 2.1. Nedostatak stranice je prijevod sadržaja na engleski jezik. Osim poslova, web stranica Moj posao sadrži aktualne vijesti koje utječu na razvoj poslova u Republici Hrvatskoj i savjete za posloprimce i poslodavce.



Slika 2.1 Rezultat pretraživanja internetske stranice *moj-posao.net[1]*

2.2. Web stranica One World 365

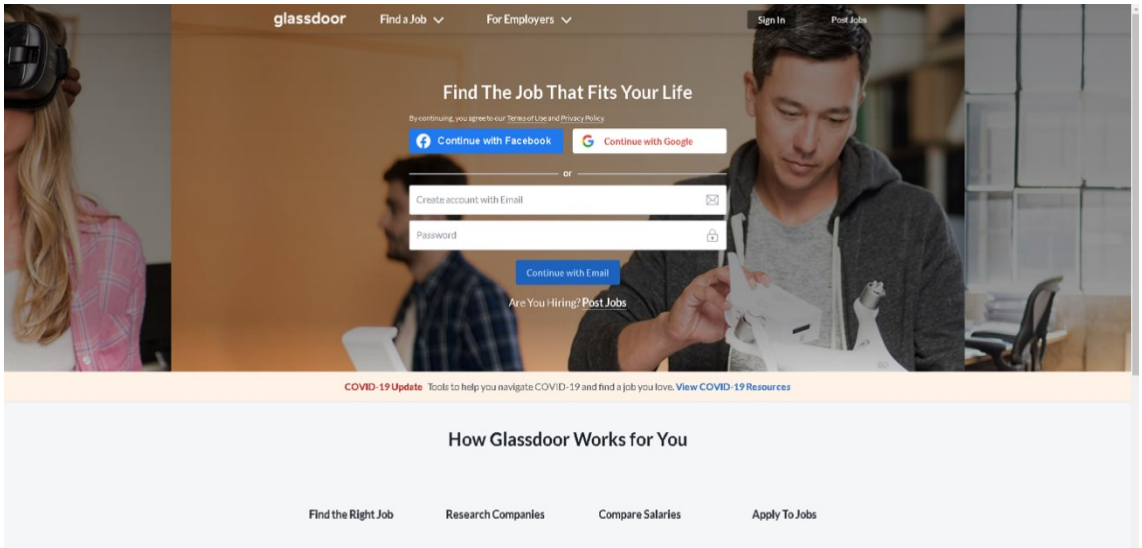
One Word 365 je web stranica čija je naslovna stranica prikaza na slici 2.2. One Word 365 je jednostranična web stranica koja nudi pretragu poslova za neregistrirane korisnike [2]. Osim poslova One World 365 nudi 2500 programa za putovanja, tečaja i poslova za učenje jezika u inozemstvu. Aktivnosti u ponudi se protežu na sve kontinente s mogućnosti boravka od jednog tjedna do šest mjeseci.



Slika 2.2 Naslovnica internetske stranice *oneworld365.org[2]*

2.3. Web stranica Glassdoor

Glassdor je web stranica čija je naslovna stranica prikazana na slici 2.3. Glassdor je naprednija web stranica u usporedbi s moj-posao.net i oneworld365.org. Pretragu može obavljati i neregistrirani korisnik, a registrirati se mogu poslodavci i posloprimci. U registraciji se mogu dodavati dokumenti sa certifikatima. Web stranica omogućava pretragu poslova po lokacijama, spremanje kompanija čije obavijesti o zapošljavanju želite primati i informacije o pitanjima srazgovora ua posao [3].



Slika 2.3 Naslovnica internetske stranice *glassdoor.com[3]*

3. STRUKTURE I TEHNOLOGIJE ZA IZRADU WEB APLIKACIJE

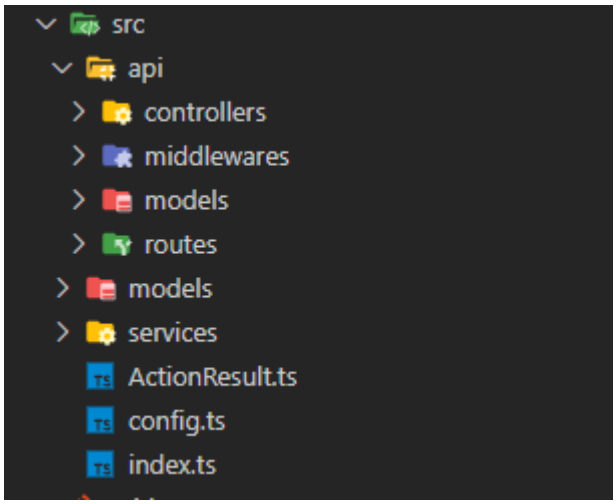
U ovom poglavlju pobliže su opisane tehnologije i strukture za realizaciju web aplikacije. Objasnit će se kako dijelovi koda funkcioniraju na serverskom i klijentskom dijelu aplikacije.

3.1. Serverska strana aplikacije

Serverska strana aplikacije obrađuje podatke koje korisnik šalje putem zahtjeva (engl. *request*). Podaci koje serverska strana aplikacije primi se spremaju u bazu podataka, a zatim se sa serverske strane šalje odgovor servera (engl. *response*). Tehnologije za serverski dio web aplikacije su Javascript transpiler Typescript, programski okvir Express.js i MySQL baza podataka. U web aplikaciji se kreira web servis za razmjenu podataka između serverskog i korisničkog dijela aplikacije. Takav servis poznatiji je pod nazivom REST (engl. *Representation state transfer*) web servis. U završnom radu arhitektura REST web servisa je podijeljena u tri sloja kao na slici 3.1. Troslojna podjela REST web servisa:

- *models* sloj opisuje podatke
- *services* sloj obavlja logičke operacije
- *controllers sloj* obrađuje zahtjeve i ovisno o rezultatu šalje odgovore

Struktura projekta kreirana je po primjeru iz literature [6].



Slika 3.1 Struktura datoteka troslojne arhitekture servisa

3.1.1. Typescript

Typescript je transpiler za Javascript [4]. Kod odrade Typescript koda prevoditelj prevodi Typescript kod u Javascript kod. Typescript je nadogradnja za Javascript jer uvodi koncept objektno orijentiranog programiranja u Javascript. Uvođenjem objektno orijentiranog principa podrazumijeva se uvođenje objekata, klasa, nasljeđivanja, polimorfizma, apstrakcije i enkapsulacije. Typescript programski jezik čini osnovu razvoja serverskog i korisničkog dijela web aplikacije za objavu poslova.

3.1.2. Express

Express je razvojno okruženje koje nudi mogućnosti za razvoj i funkcionalnost web servera [5]. Pojednostavljuje izradu i organizaciju API-a (engl. *Application Programming Interface*). U izradi web aplikacije se koristi jer definira tablicu rutiranja kao na odsječku programskog koda 3.1, koja se koristi za izvođenje različitih akcija na osnovu HTTP(engl. *Hypertext Transfer Protocol*) zahtjeva i URL-a (engl. *Uniform Resource Locator*) zahtjeva kao na odsječku programskog koda 3.2.

```
import { Router } from 'express'
import companyRouter from './CompanyRouter';
import jobRouter from './JobRouter'
import userCompanyRouter from './UserCompanyRouter';
import userJobRouter from './UserJobRouter';
import userRouter from './UserRouter';
const router = Router();
router.use("/jobs", jobRouter);
router.use("/companies", companyRouter);
router.use("/user-companies", userCompanyRouter);
router.use("/users", userRouter);
router.use("/user-jobs", userJobRouter);

export default router;
```

Odsječak programskog kôda 3.1 Kod za definiranjeExpress routera

```
import { Router } from "express";
import UserController from "../controllers/UserController";
import validateBody from "../middlewares/validadteBody";
import { UserLoginRest } from "../models/User/UserLogin";
import { UserRegisterRest } from "../models/User/UserRegister";
import auth from '../middlewares/auth';

const userController = new UserController();

const userRouter = Router();

userRouter.get("/profile", auth(), userController.GetAsync);
userRouter.get("/profile/:id", auth(), userController.FindAsync);
userRouter.post("/login", validateBody(UserLoginRest), userController.LoginAsync);
userRouter.post("/register", validateBody(UserRegisterRest), userController.RegisterAsync);
userRouter.put("/edit-profile", auth(), userController.PutAsync);

export default userRouter;
```

Odsječak programskog kôda 3.2 Kod za definiranje zahtjeva

3.1.3. REST API

Pojam API predstavlja programsko sučelje aplikacije koje je spomenuto kod Express razvojnog okruženja. U razvoju web aplikacije je izgrađen REST API za posluživanje podataka kako je objašnjeno. [6]

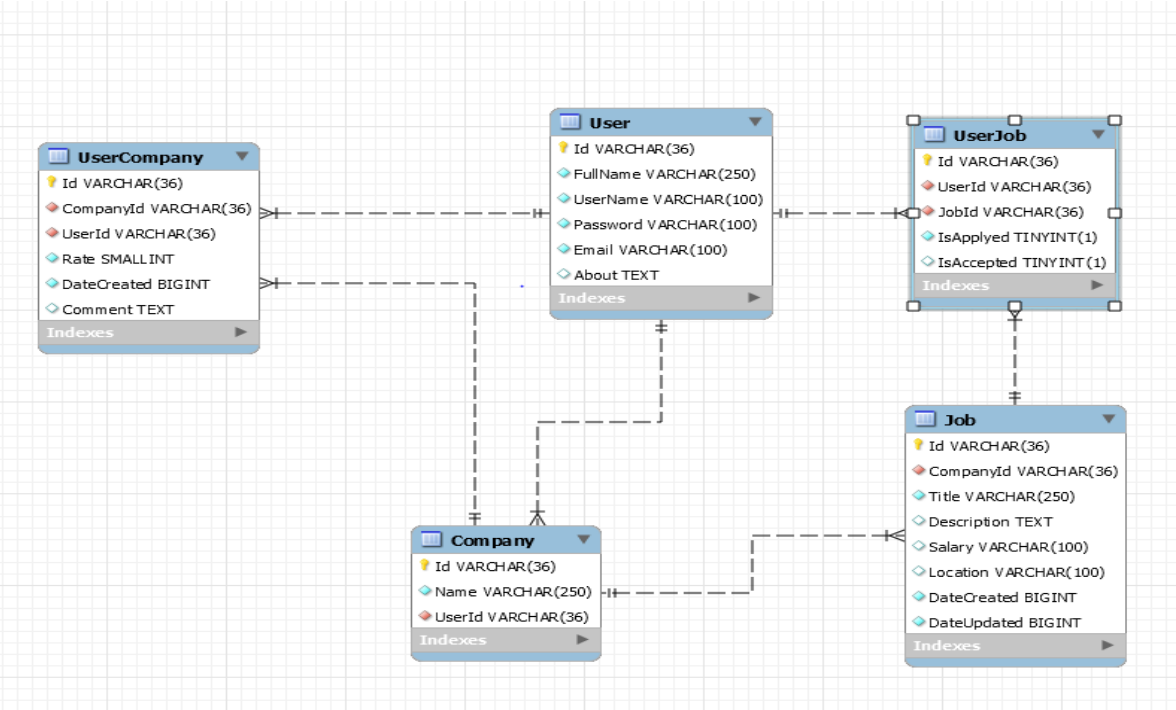
REST je web arhitektura koja koristi HTTP protokol za predstavljanje resursa. Arhitektura se primjenjuje u *router*-u, instanciranjem *Router()* objekta iz *express* modela. Primjena ovakvog principa omogućuje definiranje HTTP metoda i ruta.
Koristit ćemo REST API jer se svakom resursu pristupa preko zajedničkog sučelja pomoću HTTP metoda kojima server osigurava pristup resursima.
U REST arhitekturi koristi ćemo metode:

- GET – koristi se za *read only* pristup resrsu,
- PUT– koristi se za kreiranje novog resursa ili uređivanje već postojećeg resursa,
- POST– koristi se za stvaranje novog resursa i
- DELETE – koristi se za brisanje postojećeg resursa.

3.1.4. MySQL baza podataka

MySQL je relacijski i upravljački sustav baze podataka zasnovan na SQL-u, odnosno strukturiranom jeziku upita (engl. *Structured Query Language*) [8]. Sustav koji upravlja bazama zove se DBMS (engl. *Database Management System*), a služi za pisanje naredbi koje pozivaju operacije nad sadržajem u bazi podataka. MySQL koristi relacijski model kojim se definiraju odnosi između podataka. U ovom završnom radu MySQL kod se koristi za kreiranje tablica i veza. Upiti nisu pisani sa SQL kodom već se interpretiraju iz Typescript koda u SQL kod.

U ovom završnom radu izrađena je baza podataka pod nazivom “JobZilla” i sastoji se od pet tablica kao na slici 3.2.



Slika 3.2 Dijagram baze podataka

3.1.4.1. Tablica *User*

U tablicu *User* spremiti će se podaci za identifikaciju korisnika s podatkom *id* kao primarnim ključem. Ostali atributi tablice *User* su ime i prezime *FullName*, korisničko ime *UserName*, zaporka *Password*, elektronička pošta *Email* i portfolio *About*. Tablica *User* povezana je s tablicom *UserCompany, UserJob* i *Company* u relacijskim vezama jedan naprema više.

3.1.4.2. Tablica *Company*

U tablicu *Company* spremljeni su svi podaci o kompaniji i korisniku koji je kreirao kompaniju. Tablica *Company* sadrži jedinstveni identifikacijski primarni ključ *Id* i povezana je sa tablicom *User* pomoću vanjskog ključa *UserId*. Preostali atribut tablice *Company* je naziv komapnije *Name.* Tablica *Company* sadrži relacijsku vezu jedan naprema više s tablicama *UserCompany* i Job te više naprema jedan s tablicom *User*.

3.1.4.3. Tablica *Job*

U tablicu *Job* spremljeni su svi podaci o poslu koji pripada određenoj kompaniji. Tablica *Job* sadrži jedinstveni identifikacijski primarni ključ *Id* i povezana je s tablicom *Company* pomoću vanjskog ključa *CompanyId*. Preostali atributi tablice *Job* su naslov *Title,* opis *Description,* zarada *Salary,* mjesto *Location,* datum kreiranja *DateCreated* i datum uređivanja *DateUpdated.* Tablica

Job sadrži relacijsku vezu jedan naprema više s tablicom *UserJob* i više naprema jedan s tablicom *Company.*

3.1.4.3. Tablica *UserCompany*

Tablica *UserCompany* sadrži informacije o kompaniji koju korisnici komentiraju i ocjenjuju te prikazuje sve komentare, ocjene i prosjek ocjena koju su korisnici zadali.

Tablica *UserCompany* sadrži jedinstveni identifikacijski primarni ključ *Id* i povezana je s tablicom *Company* pomoću vanjskog ključa *CompanyId* i s tablicom *User* pomoću vanjskog ključa *UserId*. Preostali atributi tablice *UserCompany* su ocjena *Rate,* datum kreiranja *dateCreated* i komentar *Comment.* Tablica *UserCompany* sadrži relacijsku vezu više naprema jedan s tablicom *User* i tablicom *Company.*

3.1.4.3. Tablica *UserJob*

Tablica *UserJob* sadrži informacije o poslu kojeg korisnik može dodati u listu poslova koji se prate, prijaviti, prihvatiti ili odustati od spremanja ili prijave. Tablica *UserJob* sadrži jedinstveni identifikacijski ključ *Id* i povezana je s tablicom *User* pomoću vanjskog ključa *UserId* i s tablicom *Job* pomoću vanjskog ključa *JobId.* Preostali atributi tablice *UserJob* su stanje prijave *IsApplyed* i stanje potvrde IsAccepted. Tablica *UserJob* sadrži relacijsku vezu više naprema jedan s tablicom *User* i tablicom *Job.*

3.1.5. Sequelize

ORM (engl. *Object Relation Mapping*) je oblik tehnike mapiranja koji nam omogućuje slanje upita i rukovanje podacima iz baze podataka korištenjem objektno orijentiranog programiranja [9]. Veća učinkovitost je postignuta pri slanju upita za više tablica, sinkronizaciji asocijacije i validacijama. U ovom završnom radu korištena je ORM biblioteka Sequelize. Sequelize se inicijalizira unutar razvojnog okruženja Express kao na slici 3.3. Sequelize je korisna biblioteka jer omogućava definiranje modela pomoću dekoratora. Dekoratori su funkcije koje provjeravaju vrijednost i tip podatka iz zahtjeva. Sequelize sadrži implementirane metode koje služe za komunikaciju s bazom bez pisanja SQL upita [7].

U projektu su definirani kontroleri i servisi za svaki model koji je kreiran. U kontrolerima se pozivaju metode iz servisa i ovisno o dospjelim podacima se manipulira s odgovorima servera HTTP statusnim kodovima.

U servisu su implementirane metode za dohvaćanje podataka iz baze podataka.

```typescript
import { Sequelize } from 'sequelize-typescript'
import { DB_DATABASE, DB_PASSWORD, DB_USERNAME } from "../config";

export const sequelize = new Sequelize(
    {
        dialect: 'mysql',
        database: DB_DATABASE,
        username: DB_USERNAME,
        password: DB_PASSWORD,
        models: [__dirname + '/src/services/Models/'],
        define: {
            timestamps: false,
            freezeTableName: true
        }
    }
);
```

Slika 3.3 Kod za inicijalizaciju Sequelize ORM-a i spajanje na bazu podataka

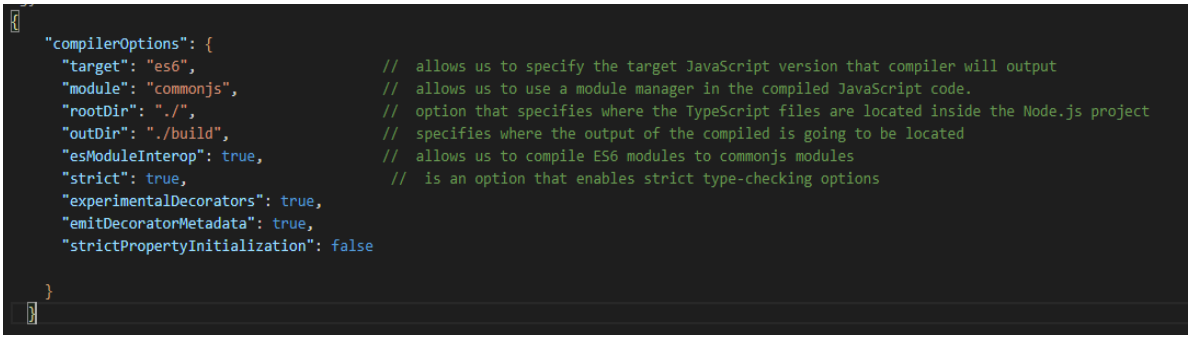
3.1.6. Konfiguracija

Konfiguracija projekta sadržana je u dokumentu *package.json* i *tsconfig.json*. U datoteci *package.json* definirane su skripte s paketima koje se koriste u izradi aplikacije, skripte za pokretanje aplikacije i informacije o autoru projekta kao što je prikazano na odjsečku programskog koda 3.3.

```
{
 "name": "JobZilla",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "NODE_ENV=development nodemon src/index.ts",
  "build": "tsc --project ./"
 },
 "keywords": [],
 "author": "Matej Balat",
 "license": "ISC",
 "dependencies": {
  "bcrypt": "^5.0.1",
  "body-parser": "^1.19.0",
  "class-transformer": "^0.3.1",
  "class-validator": "^0.13.1",
  "dotenv": "^8.2.0",
  "express": "^4.17.1",
  "jsonwebtoken": "^8.5.1",
  "lodash": "^4.17.21",
  "mysql2": "^2.2.5",
  "nodemon": "^2.0.7",
  "reflect-metadata": "^0.1.13",
  "sequelize": "^6.6.2",
  "sequelize-typescript": "^2.0.0-beta.1",
  "uuid": "^8.3.2"
 },
 "devDependencies": {
  "@types/bcrypt": "^3.0.1",
  "@types/body-parser": "^1.19.0",
  "@types/cors": "^2.8.10",
  "@types/express": "^4.17.11",
  "@types/jsonwebtoken": "^8.5.1",
  "@types/lodash": "^4.14.170",
  "@types/node": "^14.14.37",
  "@types/uuid": "^8.3.0",
  "@types/validator": "^13.1.3",
  "cors": "^2.8.5",
  "ts-node": "^9.1.1",
  "typescript": "^4.2.3"
 }
}
```

Odsječak programskog kôda 3.3 Primjer konfiguracije projekta datoteke package.json

Datoteka tsconfig.json sadrži opcije kompajlera potrebne za prevođenje aplikacije kao na slici 3.4.



Slika 3.4 Primjer konfiguracije tsconfig.json datoteke

3.2. Klijentska strana aplikacije

U izradi klijentske strane web aplikacije korištena je JavaScript biblioteka pod nazivom React u kombinaciji s Typescript skriptnim jezikom i Material-UI bibliotekom. U prvoj fazi izrade klijentske strane izrađuje se grafičko sučelje putem kojeg korisnik može koristiti funkcionalnosti aplikacije. Pri izradi grafičkog sučelja veliku važnost se pridodaje samom dizajnu radi ugodnijeg korištenja i preglednosti web aplikacije. Nakon izrade grafičkog sučelja izrađuju se modeli za jednostavniji prikaz podataka na stranici i pomoćne klase koje odrađuju pozive na serverski dio aplikacije. Modeli su definicije tipova podataka koji se čitaju iz baze, poznatiji su po nazivu sučelja (engl. *Interface*). U zadnjoj fazi, nakon povezivanja klijentske i serverske strane, ispituje se funkcionalnost u komunikaciji ovih dvaju strana.

3.2.1. React

React.js je JavaScript biblioteka za izradu korisničkog sučelja pomoću definiranih klasa koje nude rješenja za probleme s frontend programiranjem. React-u se, osim mogućnost izgradnje dinamičke i interaktivne web aplikacije, pripisuje brzina i skalabilnost, a uz to sve posjeduje i veliku zajednicu programera. Komponente prikazuju svoje korisničko sučelje u render funkcijama koristeći TSX za spajanje HTML strukture s logikom [12]. Specifičnost Reacta-a je u slaganju korisničkog sučelja iz manjih dijelova koda koji se mogu više puta iskoristiti. Takvi dijelovi koda se nazivaju komponente (engl. *components*). Primjer komponente vidljiv na slici 3.5.

```
export default ({columns, rows, actions}: ITableProps) =>
{
    return(
        <TableContainer>
            <Table stickyHeader size="small">
                <TableHead>
                    <TableRow>
                        {columns.map(column => (
                            <TableCell key={column}>
                                <Typography variant="h6">{column}</Typography>
                            </TableCell>
                        ))}
                        {
                            actions && actions.onDelete && <TableCell>Actions</TableCell>
                        }
                    </TableRow>
                </TableHead>
                <TableBody>
                    {rows.map((row) => (
                        <TableRow key={row.id}>
                            {
                                Object.entries(row).map(([key, value]) => {
                                    return <TableCell key={key} align="left">{value}</TableCell>
                                })
                            }
                            {
                                actions && actions.onDelete &&
                                <TableCell>
                                    <IconButton aria-label="Delete" onClick={() => actions.onDelete!(row.id)}>
                                        <DeleteOutlineTwoToneIcon />
                                    </IconButton>
                                </TableCell>
                            }
                        </TableRow>
                    ))}
                </TableBody>
            </Table>
        </TableContainer>
    )
};
```

Slika 3.5 Primjer dinamičke komponente Table

Nakon izrade komponenti obavlja se povezivanje pomoću *ReactDOM.render()* funkcije iz react.dom biblioteke kao na slici 3.6. Renderiranje komponenti unutar *ReactDOM.render()* funkcije se postiže prosljeđivanjem dvaju argumenata. Prvi argument predstavlja sadržaj koji će se renderirati, a drugi element predstavlja lokaciju u DOM-u (engl. *Document Object Model*) gdje se prvi argument treba prikazati.

```
ReactDOM.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Slika 3.6 Primjer *ReactDOM.render()* funkcije

U završnom radu sve komponente su renderirane iz glavne <App /> roditeljske komponente. U React aplikacijama postoji hijerarhija komponenti. Iz navedenog razloga komponente se dijele na roditeljske komponente i elemente potomke , a renderiranje takve hijerarhije odvija se u smjeru od roditeljske komponente prema komponenti potomka. Pri renderiranju tok podataka se odvija od vrha prema dnu kako bi svaka roditeljska komponenta prosljeđivala podatke komponenti potomka. Između komponenti prosljeđujemo parametre koji se nazivaju *props* kao na slici 3.7.

```
<Table columns={columns} rows={data!} actions={actions} />
```

Slika 3.7 Primjer prosljeđivanja *props*-a u roditeljsku komponentu, komponente potomka Table

React omogućava renderiranje većeg broja komponenti potomka u nekoj drugoj komponenti. Postupak se provodi tako da se generira niz komponenti spremljenih u varijablu. Zatim se pomoću *map()* funkcije prolazi nizom elemenata i kreira novi niz koji sadrži vraćene vrijednosti kao na slici 3.8. Vraćene vrijednosti se na korisničko sučelje šalju u obliku odgovora serverskog dijela aplikacije. Kod korištenja funkcije *map()* korišten je poseban *prop* parametar koji se naziva *key.* Parametar *key* se koristi kako bi se kreirala jedinstvena veza za svaku instancu komponente koja sadrži jedinstvenu vrijednost.

```
<TableRow>
    {columns.map(column => (
        <TableCell key={column}>
            <Typography variant="h6">{column}</Typography>
        </TableCell>
    ))}
    {
        actions && actions.onDelete && <TableCell>Actions</TableCell>
    }
</TableRow>
```

Slika 3.8 Primjer korištenja funkcije *map()* i jedinstvenog prop parametra *key*

3.2.2. Material-UI

Material-UI je biblioteka koja olakšava izradu React aplikacije. Sadrži gotove komponente i jednostavnu implementaciju vizualnog dijela aplikacije [11].

4. OPIS KORIŠTENJA WEB APLIKACIJE

Poglavlje sadržava detaljan opis uporabe web aplikacije. Funkcionalnosti su poredane po radnom toku (engl. *Work-flow*). Ovakav poredak je neizostavan zbog korisnikove sigurnosti i ranjivosti podataka.

4.1. Prijava i registracija

Prije pristupa podacima aplikacije korisnik se identificira s postojećim računom i podacima koje je poslao pri registraciji. Identifikacija se izvršava slanjem korisničkog imena *Username* i zaporke *Password*. Odjava se obavlja odabirom *LOGOUT* gumba iz navigacijskog zaglavlja vidljivog na slici 4.2.

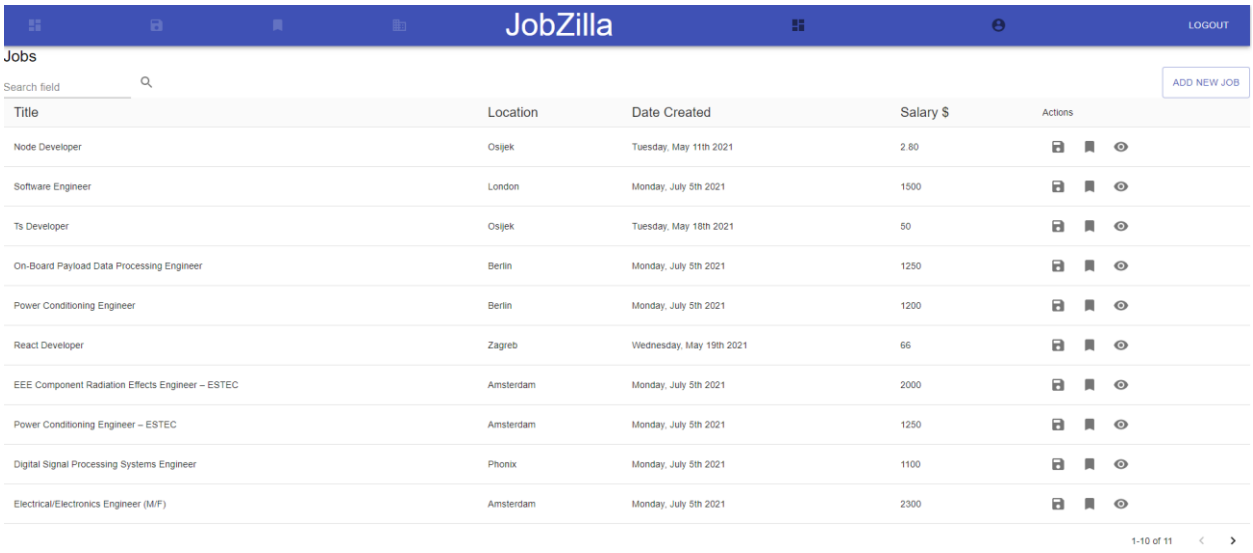
Slika 4.1 Identifikacijska korisničkog računa

Identifikacijski račun se kreira ispunjavanjem svih polja u formi. Podaci koji će kasnije biti vidljivi svim korisnicima aplikacije se mogu naknadno izmijeniti, dok se podaci potrebni za identifikaciju ne mogu naknadno mijenjati. Korisničko ime *username* i elektronička pošta ne mogu koristiti više od jednog korisnika s toga svaki korisnik mora imati jedinstveno korisničko ime i elektroničku poštu.

Slika 4.2 Kreiranje identifikacijskog računa

4.1. Tablica poslova

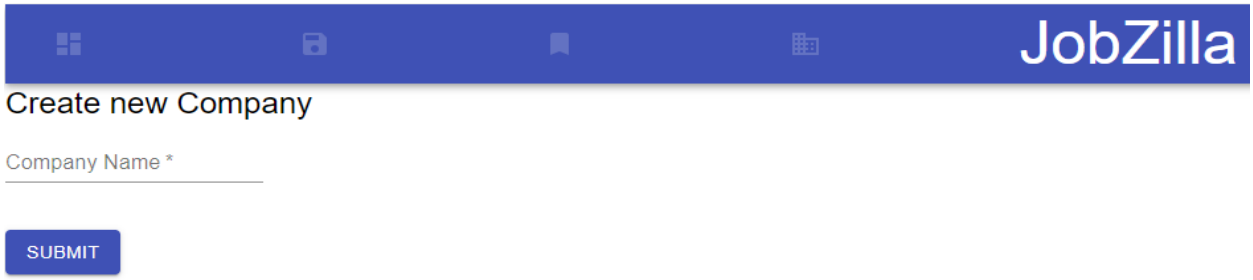
Tablica poslova prikazuje naziv posla, lokaciju, datum kreiranja i zaradu. Stranica tablice može sadržavati deset redova koji prezentiraju podatke pojedinog posla. Stranice tablice se mogu izmjenjivati pomoću paginacije. Poslovi se mogu pretraživati po nazivu posla. Svaki redak uz pojedine informacije pruža funkcionalnosti stupca *Actions*. Prva ikonica u stupcu *Actions* omogućava spremanje posla, druga ikonica omogućava prijavu posla i treća ikonica omogućava detaljan pregled posla.



Slika 4.3 Lista postojećih poslova

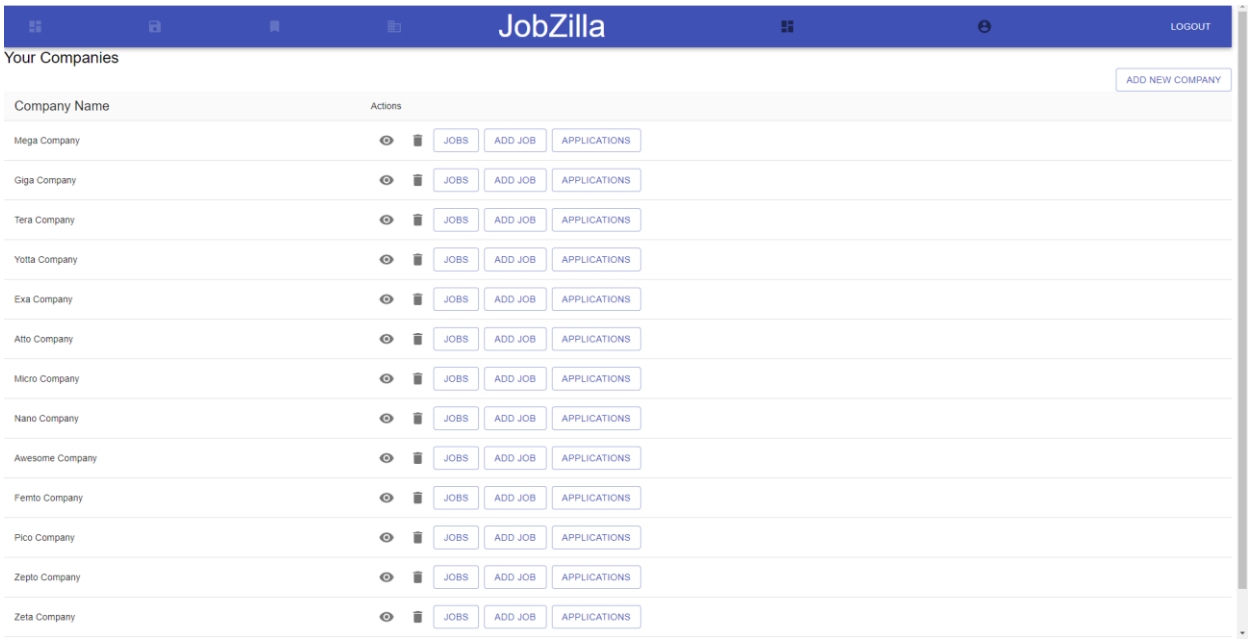
4.2. Kreiranje kompanije

Svaki korisnik ima mogućnost kreiranja kompanije, a samo kreiranje kompanije se izvršava ispunjavanjem obrasca. Obrazac se sastoji od tekstualnog polja za unos proizvoljnog naziva kompanije, a potvrda za kreiranje ostvaruje odabirom gumba *SUBMIT*.



Slika 4.4 Obrazac za kreiranje kompanije

Korisnik može imati više kompanija i na svaku kompaniju može dodati više poslova. Na slici 4.5 prikazana je tablica korisnikovih kompanija. Podaci u tablici su prikazani u obliku liste. Tablica kompanija prikazuje naziv kompanije i sve radnje koje vlasnik kompanije može obavljati. Prva ikonica u stupcu *Actions* vodi na stranicu profila kompanije koja je vidljiva svim korisnicima, a sadržava prosječnu ocjenu i komentare o čemu će više biti rečeno u poglavlju 4.5. Druga ikona u stupcu *Actions* omogućava brisanje profila kompanije, treći gumb *JOBS* vodi na stranicu s listom poslova koji su aktivni na odabranoj kompaniji, četvrti gumb *ADD JOB* otvara obrazac za kreiranje novog posla o čemu će više biti rečeno u poglavlju 4.3. Peti gumb *APPLICATIONS* otvara tablicu sa svim prijavama za poslove po odabranoj kompaniji o čemu će više biti rečeno u poglavlju 4.4. i 4.5.

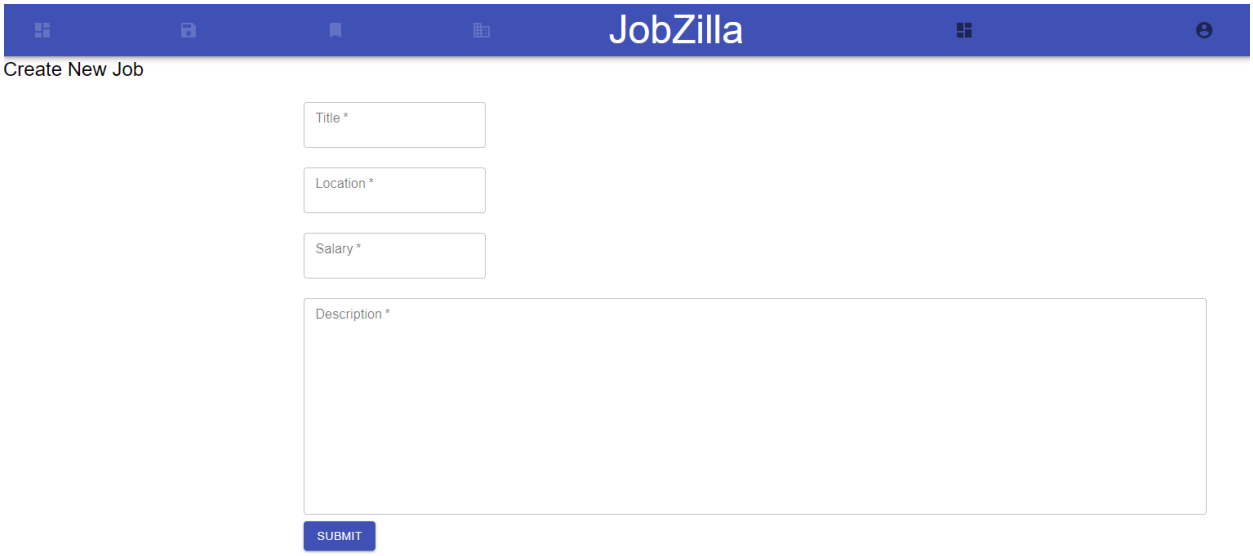


Slika 4.5 Tablica kompanija

4.3. Kreiranje posla

Akcija kreiranja posla omogućena je korisniku samo ako je prethodno kreirana kompanija. Obrascu za kreiranje posla se pristupa odabirom *ADD JOB* gumba koji se nalazi uz naziv kompanije na tablici kompanija. Odabirom gumba *ADD JOB* otvara se obrazac kao na slici 4.6. Obrazac sadrži polja za unos naziva posla, lokacije, zarade, opisa i gumba za potvrdu kreiranja. Sva polja su obavezna sadržavati informacije prije odabira gumba za kreiranje. Tako kreirani posao se sprema u listu poslova odabrane kompanije, a pristup podatcima je moguć sa svim korisnicima. Informacije o poslu se ne mogu naknadno uređivati i korisnik koji je kreirao posao

može u bilo kojem trenutku obrisati posao. Posao se može obrisati u slučaju da korisnik obriše kompaniju. Ako se obriše kompanija, obrisati će se svi poslovi vezani za tu kompaniju.



Slika 4.6 Obrazac za kreiranje posla

4.4. Spremanje i prijavljivanje posla

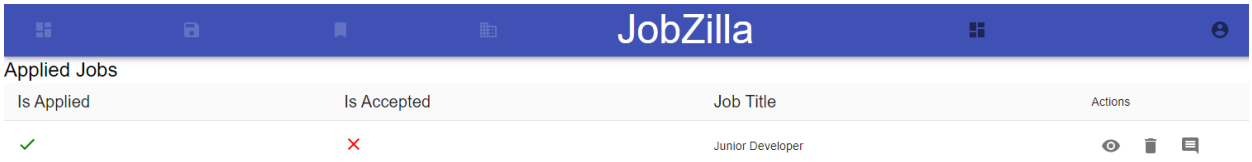
Korisnik u ulozi posloprimca može spremiti željeni posao u tablicu spremljenih poslova ili se prijaviti za posao te se takav posao sprema u tablicu prijavljenih poslova. Spremanje posla se obavlja odabirom ikone *Save* uz željeni posao. Zatim se posao sprema u tablicu spremljenih poslova kao na slici 4.7. Svaki korisnik ima posebnu tablicu spremljenih poslova i samo on može uređivati tablicu.

Slika 4.7 Tablica spremljenih poslova

Tablici spremljenih poslova se pristupa odabirom *Saved* ikone iz navigacijskog izbornika. Tablica spremljenih poslova prikazuje podatke o tome je li korisnik prijavio posao, je li posloprimac prihvatio njegovu prijavu, naziv posla i naziv kompanije. Uz podatke o pojedinom poslu omogućene su i radnje koje korisnik može obavljati u stupcu *Actions*. Prva ikonica *View* omogućava pregled posla, druga ikonica *Apply* omogućava prijavu posla i treća ikonica delete uklanja posao iz tablice spremljenih poslova. Svrha prikaza informacija ovakve tablice je da korisniku smanji i olakša izbor poslova.

Prijava posla se obavlja odabirom ikone *Apply* uz naziv posla. Posao se može prijaviti iz tablice *Dashboard* ili iz tablice *Saved*. Ako korisnik prijavi posao iz tablice *Dashboard* posao se sprema u tablicu sa spremljenim poslovima kao na slici 4.8, a zatim se posao sprema u tablicu sa prijavljenim poslvima kao na slici 4.9.

Slika 4.8 Prijavljeni posao iz tablice spremljenih poslova

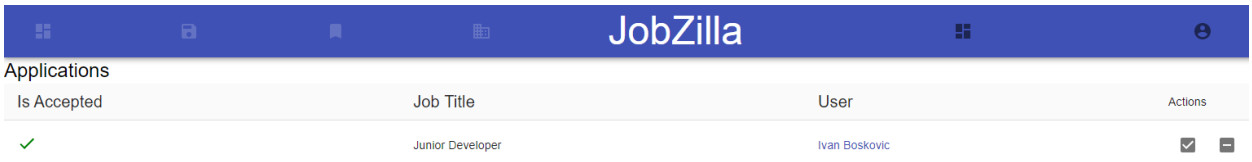


Slika 4.9 Tablica prijavljenih poslova

Tablici prijavljenih poslova se pristupa odabirom *Applied* ikone iz navigacijskog izbornika. Tablica prijavljenih poslova sadrži informacije je li posao prijavljen, je li poslodavac odobrio prijavu i naslov posla. Uz podatke o pojedinom poslu omogućene su i radnje koje korisnik može obavljati u stupcu *Actions*. Prva ikonica *View* omogućava pregled posla, druga ikonica *Delete* omogućava uklanjanje posla i ikonica *Rate* omogućava ocjenjivanje i komentiranje posla o čemu će više biti rečeno u poglavlju 4.5. Ako korisnik obriše posao iz tablice prijavljenih poslova, posao će se obrisati i u tablici spremljenih poslova. Isto vrijedi ako je posao spremljen i prijavljen, a brisanje se izvršava na tablici spremljenih poslova.

4.5. Prihvaćanje korisnikovog zahtjeva

Nakon prijavljivanja posla prijava se sprema u tablicu potvrde o prijavljenim poslovima kao na slici 4.10. Tablici potvrda o prijavljenim poslovima se pristupa odabirom gumba *Applications* iz tablice kompanija. Ovoj akciji može pristupiti samo korisnik koji posjeduje kompaniju. Tablica o prijavljenim poslovima prikazuje informacije je li prijava prihvaćena ili odbijena, naziv posla i korisničko ime osobe koja je podnijela prijavu. Korisničko ime je ispisano u obliku poveznice koja vodi na korisnički profil. Uz informacije svaka prijava sadrži kolonu *Actions* s dvije ikone. Prva ikona *Accept* služi za prihvaćanje zahtijeva i druga ikona *Refuse* služi za odbijanje zahtjeva.



Slika 4.10 Tablica potvrda o prijavljenim poslovima

4.6. Ocjenjivanje i komentiranje kompanije

Ocjenjivanje i komentiranje kompanije omogućeno je nakon prihvaćanja korisničkog zahtjeva. Korisnik pristupa obrascu za ocjenjivanje i komentiranje tako da se pozicionira na tablicu sa prijavljenim poslovima i odabere ikonu *Rate*. Odabirom akcije otvara se obrazac sa poljima kao na slici 4.11.

Slika 4.11 Obrazac za ocjenjivanje i komentiranje kompanije

Obrazac se sastoji od polja za unos ocjene i komentara. Sva polja su obavezna sadržavati informaciju prije odabira gumba za kreiranje. Nakon ispunjavanja obrasca takva ocjena i komentar se dodaju na profil kompanije. Ocjena koja se unosi u polje može biti cjelobrojni broj s vrijednosti između 1 i 5. Profil kompanije sačinjavaju ocjene s komentarima te aritmetička sredina svih ocjena.

Slika 4.12 Profil kompanije

5. Zaključak

U ovom završnom radu su opisane i objašnjene funkcionalnosti i postupci izrade web aplikacije za objavu poslova. Web aplikacija za objavu poslova olakšava pronalazak poslova i posluživanje informacija s ostalim korisnicima. Ova web aplikacija prezentira jednostavnost pri korištenju i laku izradu serverskog dijela aplikacije korištenjem Express razvojnog okruženja s MySQL bazom podataka te razvoj korisničkog dijela aplikacije korištenjem React biblioteke. Opisom radnog toka korisničkog sučelja i testiranjem funkcionalnosti može se zaključiti da su podatci svakog korisnika sigurni te da web aplikacija omogućuje lakši i sigurniji pronalazak poslova.

LITERATURA

[1] Internetska stranica MojPosao - https://www.moj-posao.net/ datum posjete: 15.7.2020.

[2] Internetska stranica One World 365 - http://www.oneworld365.org/ datum posjete: 15.7.2020.

[3] Internetska stranica Glassdoor - https://www.glassdoor.com/index.htm datum posjete: 15.7.2020.

[4] Anonimno, What is Typescript? - https://www.typescriptlang.org/ datum posjete: 18.3.2021.

[5] Express dokumentacija - https://expressjs.com/ datum posjete: 18.3.2021.

[6] Anonimno , Building An APO With Node.js, Express And Typescript- https://javascript.plainenglish.io/create-a-typescript-express-server-in-minutes-7d34c306f60/ datum posjete: 18.3.2021.

[7] Max A Raju, sequelize-typescript - https://www.npmjs.com/package/sequelize-typescript datum posjete: 23.5.2021.

[8] Anonimno, MySQL - What is mySQL and why do I need it? - https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/ datum posjete: 23.5.2021.

[9]Anonimno, ORM - What is an ORM, how does it work, and how should I use one? - https://stackoverflow.com/questions/1279613/what-is-an-orm-how-does-it-work-and-how-should-i-use-one datum posjete: 23.5.2021.

[10] React.js dokumentacija - https://reactjs.org/docs/getting-started.html datum posjete: 29.5.2021.

[11] Material-ui dokumentacija - https://material-ui.com/getting-started/learn/ datum posjete: 30.5.2021.

SAŽETAK

U ovom završnom radu izrađena je i modelirana web aplikacija za pomoć pronalaska posloprimaca u inozemstvu. Korisničko sučelje i programska podrška na serverskoj strani omogućava registraciju poslodavaca, prijavu poslodavaca, dodavanje novih poslova, prikaz spremljenih poslova te ocjenjivanje kompanija. U izradi web aplikacije korištena je React.js biblioteka za izradu korisničkog sučelja, Node.js izvršno okruženje, Express.js programski okvir i MySQL baza podatak za programsku podršku na serverskoj strani.

Ključne riječi: Express.js, MySQL, Node.js, React.js, web aplikacija

ABSTRACT

Web application for job posting

A web application that helps people find employees abroad has been modeled and developed within this bachelor thesis. User interface and software support enables employer registration, employer application, adding new jobs, viewing saved jobs and allows employees to grade a company. The application was developed using React.js library for creating user interface, Node.js runtime environment, Express.js framework and MySQL database for storing data.

Keywords: Express.js, MySQL, Node.js, React.js, web application