Razvoj mobilnih web aplikacija korištenjem React Native platforme

Sušanj Samolov, Raul

SVEUČILIŠTE U RIJECI

ODJEL ZA INFORMATIKU

Izrada mobilne web aplikacije korištenjem React Native platforme

Završni rad

Izradio: Raul Sušanj Samolov Mentor: dr.sc. Igor Jugo

Rijeka, rujan 2018.

SADRŽAJ

1. UVOD

Sadržaj ovoga završnog rada se temelji na dokumentaciji praktičnog dijela završnog rada tj. Izrade mobilne aplikacije pomoću web tehnologija te opisu strukture, svrhe i korištenje aplikacije. U ovome radu ćemo opisati i tehnologije koje su korištene tijekom razvoja i razvoja same aplikacije.

Naziv aplikacije je „Studentski posao“ te služi kao mobilni portal za sve studente kako bi im se olakšao pronalazak studentskog posla. Osim za studente, mobilna aplikacija olakšava i pojednostavljuje posao studentskim centrima i poslodavcima diljem Republike Hrvatske. Poslodavci ovim putem imaju mogućnost bez zahtjeva studentskog centra objaviti oglas za studentski posao sa svim potrebnim informacijama koje se zahtijevaju prilikom objave samoga oglasa. S druge strane djelatnicima studentskih službi ova aplikacija smanjuje obujam posla te im daje mogućnost fokusa na druge poslove.

Motivacija za izradu ovakve aplikacije je nastala iz više razloga i situacija s kojima sam se susreo kao student Sveučilišta u Rijeci tijekom korištenja usluga Studentskog Centra Rijeka. Prva i glavna motivacija izrade ove aplikacije je svojevrsna želja digitalizirati sve što je u papirnatome obliku te olakšati populaciji u nabavci i pristupačnosti materijala za posao, školovanje i slobodno vrijeme te smanjiti svakodnevni stres ljudi prilikom obavljanja administracije. U ovom trenutku želim izraziti veliku pohvalu studentskim centrima što postoji mogućnost podizanja studentskih ugovora putem mreže te me upravo ta mogućnost navela na razmišljanje o jednoj kompletiranoj aplikaciji putem koje bi studenti imali detaljan uvid u sve usluge koje pruža studentski centar. Kada govorimo o uslugama studentskog centra, misli se na prehranu, smještaj, organizirana putovanja te mogućnost rada za vrijeme studija.

Konkretno ovdje navedena aplikacija rješava samo jedan segment prije navedene ideje s obzirom na to da sveukupna ideja zahtjeva rad više ljudi, više vremena te veće resurse. Mobilna aplikacija „Studentski posao“ za korištenje je najinteresantnija i najkorisnija studentima i poslodavcima, a gledano s poslovnog aspekta kao potencijalni proizvod na tržištu interes bi mogla probuditi u svim studentskim centrima te Ministarstvu obrazovanja Republike Hrvatske.

2. TEHNOLOGIJE

Tehnologije korištene za izradu mobilne aplikacije su sve redom web tehnologije te se prvenstveno koriste za izradu web aplikacija. Kako je današnja ideja olakšati razvoj aplikacija kako web tako i mobilnih usmjerio sam svoju pažnju na jednu relativno novu tehnologiju koja je međutim zastupljena u poznatim aplikacijama koje koristi svakodnevni korisnik pametnog telefona. Kao glavna baza ove mobilne aplikacije korišten je razvojni okvir React Native koji je u ovome radu kasnije u detalje opisan. Međutim važno je navesti najveću prednost ove razvojne okoline, a to je da se pisanjem jednog koda osigura razvoj za sve trenutno aktivne mobilne platforme kao što su Android, Ios i Windows Mobile. Tijekom razvoja aplikacije uzeto je u obzir da su tehnologije međusobno kompatibilne te da im je zajednička stavka JavaScript kao programski tj. skriptni jezik koji je trenutno za vrijeme pisanja ovog rada najpopularniji jezik na svijetu te mu popularnost iz dana u dan raste.

Ostale tehnologije koje ćemo detaljnije u nastavku opisati, a korištene su za izradu aplikacije jesu:

- React
- React Native
- NodeJS
- NPM
- Express
- GraphQL
- PostgreSQL
- Knex

2.1. React i React Native

React[1] je razvojni okvir za stvaranje korisničkog sučelja te isključivo služi za razvoj front-end tj. klijentske strane aplikacije. Koristi se pretežno za izradu web aplikacija koje se pokreću u web pregledniku poput Chroma, Mozille Firefox, Safari i sl. Svojom funkcijom naravno može poslužiti i u izradi desktop aplikacija te mobilnih aplikacija. Trenutno se bilježi da je to najpopularniji i najviše korišteni razvojni okvir na svijetu pored Angulara i Vue.js-a. React je karakteriziran s mnogim prednostima poput stvaranja vlastitih komponenti koje je moguće iskoristiti na više mjesta u istome projektu ili u drugim projektima te su te komponente lako prilagodljive ovisno o potrebi aplikacije. Druga korist jesu već postojeće komponente koje se nalaze u samom razvojnom okviru poput lista, dugmadi, tablica i sl. Međutim najvažnija prednost Reacta je brzina i način osvježavanja korisničkog sučelja. React postupa tako da osim stvarnog DOM-a (Document Object Model) sadrži i virtualni DOM u kojeg sprema prethodno stanje. Kada dolazi do neke promjene unutar aplikacije korisničkom ili funkcionalnom interakcijom, React uspoređuje stvarni DOM i vlastiti virtualni DOM te primjenjuje promjeni samo na elementu tj. komponenti koja se razlikuje od one u prethodnom DOM-u. Pomoću tako moćne funkcionalnosti, React nam omogućava maksimalnu brzinu osvježavanja korisničkog sučelja. U stvarnosti niti ne dolazi do osvježavanja sučelja već su sve komponente pohranjene na klijentskoj strani. Osim samih funkcionalnih prednosti Reacta imamo i veliku društvenu mrežu korisnika Reacta i React Nativa koji razvijaju mnogobrojne pakete i komponente kako bi olakšali sebi i drugima razvoj aplikacija. Samu tehnologiju razvio je Facebook te ju koristi na vlastitim proizvodima, a među poznatijim korisnicima su Airbnb, Uber, Instagram, Discord i drugi. Iako se u ovom radu React spominje kao razvojni okvir te se može takav iskaz pronaći u velikom broju literature, React je tehnički razvojna biblioteka, a ne okvir, međutim uz korištenje kompatibilnih paketa koji su odvojeni od Reacta s razlogom možemo React kategorizirati u razvojne okvire poput rivala Angulara. Paketi za rutanje, navigaciju i sl. su odvojeni kako bi platforma bila primjenjiva na desktop, web i mobile aplikacije uz uvjet da ne komplicira razvoj.

React Native[2] nije potrebno dodatno opisivati jer radi na istome principu osim nekoliko izuzetaka specifičnih za razvoj mobilnih aplikacija. React Native omogućava s razvojem

[1] React - React, A JavaScript library for building user interfaces. https://reactjs.org/ (2018.)
[2] React Native - React Native, Build native mobile apps using JavaScript and React. https://facebook.github.io/react-native/ (2018.)

jednog koda istovremeno zadovoljiti uvjete trenutno aktualnih platforma poput Androida, Ios-a i Windows Mobilea. React Native se od Reacta razlikuje po komponentama i načinu stiliziranja komponenti. Osim React Native slični razvojni okviri su Ionic, Cordova i NativeScript, ali trenutno je React Native jedini pomoću kojeg se mogu razviti nativne aplikacije za pojedine platforme dok drugi okviri simuliraju taj osjećaj korisniku prilikom korištenja aplikacije. Važno je napomenuti da niti jedna tehnologija nije savršena pa tako i React tj. React Native međutim trenutno se drži kao naj najefikasnija tehnologija.

2.2. NodeJS, NPM i Express

Gledano u prošlost PHP[3] se smatrao najpopularnijim i gotovo jedinim rješenjem za stvaranje back-end segmenta neke web aplikacije. Danas je PHP i dalje zastupljen na jako velikom broju aplikacija i sustava te je rangiran iznad programskih jezika poput Rubya[4] ili Phytona[5] (u svrhu web aplikacija). Kako je kroz vrijeme JavaScript kao skriptni jezik postajao sve popularniji i raširenij među programerima bilo je vrijeme da se pokuša front-end tehnologiju svrstati i u back-end. S tom idejom je nastao NodeJS[6] koji je u današnje vrijeme sve popularniji te laganim korakom prestiže i PHP. NodeJS služi za izradu poslužitelja tj. poslužitelj strane neke aplikacije te je kompatibilan s bilo kojom klijentskom tehnologijom i s bilo kojom bazom podataka. Glavna prednost NodeJS-a je korištenje JavaScripta te je s time olakšano učenje razvoja aplikacija jer sada novom programeru nije potrebno znanje dvaju programskih jezika (iako je poželjno) već je dovoljno znati osnovne koncepte programiranja i JavaScripta. Osim NodeJS-a koristi se i NPM[7] tj. Node Package Manager koji služi za instalaciju mnogobrojnih gotovih biblioteka koda za različite projekte.

Pored povećane jednostavnosti razvoja uz ovakvu tehnologiju, programer današnjice si nastoji olakšati posao što je više moguće, pa tako se stvaraju pretprocesori za CSS poput Sassa[8] i Lessa[9], front-end razvojni okviri, gotove biblioteke komponenata i u ovom

[3] PHP - http://php.net/ (2018.)
[4] Ruby - https://www.ruby-lang.org/en/ (2018.)
[5] Python - https://www.python.org/ (2018.)
[6] NodeJS - https://nodejs.org/en/ (2018.)
[7] NPM - https://www.npmjs.com/ (2018.)
[8] Sass - https://sass-lang.com/ (2018.)
[9] Less - http://lesscss.org/ (2018.)

slučaju razvojni okvir za back-end tj. NodeJS okvir pod imenom Express[10]. Jedina prednost Expressa je što sadrži već gotove univerzalne funkcije u sebi koje smanjuju broj linija koda i pomažu programeru u razvoju i poštivanju zadanih vremenskih rokova.

U aplikaciji „Studentski posao“ NodeJS je korišten kao poslužitelj koji pokreće cijelu poslužitelj stranu međutim radi se o jednom malom sloju te kasnije preuzima GraphQL glavnu ulogu tako da poslužitelj stranu ove aplikacije možemo nazvati GraphQL poslužiteljom.

2.3. GraphQL

Kako bi razumjeli što je GraphQL tj. GraphQL API i kako funkcioniraju ti koncepti, moramo prvo razumjeti što je RESTfull API i što je uopće API. Kada su se prije razvijale web aplikacije koristio se klasični pristup dohvaćanja podataka koji je funkcionirao tako da bi se svakim pozivom klijentske strane pozivala cijela HTML datoteka te je ta metoda nekada krajnjim korisnicima zadavala glavobolje s obzirom na to da je uz lošiju Internet vezu znalo dolaziti do situacija da se čeka po koju minutu za prikaz jednog dijela web aplikacije tj. stranice bolje nazvano.

Danas se uglavnom koriste API (Application Programming Interface) koji je sastavljen na poslužitelj strani aplikacije. API služi za vraćanje podataka iz baze podataka na klijentsku stranu te je cijeli sadržaj pohranjen u JSON datoteku koja je minimalne veličine te je za nju potrebno manje vremena za „prikaz“. Cijelo oblikovanje klijentske strane se odvija na klijentskoj strani te je prilikom prikazivanja aplikacije samo jednom napravljen poziv na jednu datoteku (index.html) koja sadrži poveznice na ostatak sadržaja. Kada je potrebno prikazivanje dinamičkih podataka koristimo API za brže prikazivanje samih podataka na sučelju.

Sada možemo pojasniti i pojam RESTfull API. RESTfull API je princip pisanja API-a. Radi tako da se unutar koda definiraju sve potrebne rute aplikacije, poznatije krajnjem korisniku kao linkovi. Unutar bloka koda koji je zabilježen određenom rutom se vrši sva potrebna radnja, od dohvaćanja, brisanja, izmjenjivanja pa sve do logike unutar samog koda. Gledano na prošlost razvoja aplikacija RESTfull API je uveliko pripomogao u brzini odvijanja radnji poput navedenih međutim uvijek je moguće bolje.

[10] Express - https://expressjs.com/ (2018.)

Kako bi maksimalno optimizirao aplikaciju te ju učinio bržom i kako koristim React Native odlučio sam se za opciju sa GraphQl[11] poslužiteljom tj. API-em. GraphQL osim što je poput Reacta i React Nativa proizvod od Facebooka te je ujedno kompatibilan s njima, njegova velika prednost je smanjenje zahtjeva s klijentske strane. Dok za RESTfull API možemo reći da je linearan i radi poziv po poziv za JSON datoteke, GraphQL je upravo suprotno. GraphQL api je mrežaste strukture te je moguće dolaziti pomoću jednog poziva do više različitih točaka podataka. Kako bi pojednostavili ovaj koncept zamislimo mrežu s više točaka. Svaka točka u sebi sadrži neki entitet koji je povezan s nekim drugim entitetom. Kada radimo upit za podatke povezujemo se na jednu točku koju smo definirali u kodu. Pomoću tog jednog poziva i spajanja na navedenu točku, jednostavnim upitima možemo pristupati svim ostalim točkama koje su povezane s glavnom ulaznom točkom, a točke koje se nalaze hijerarhijski ispod glavne ulazne točke su opet povezane s točkama iste ili niže razine. Važno je napomenuti da se GraphQL prvenstveno koristi na poslužitelj strani dok se na klijentskoj strani tj. front-endu koriste paketi ovisno o tehnologiji kao npr. u slučaju naše aplikacije korišten je react-graphql paket i GraphQL paket kojim se stvaraju upiti.

2.4. PostgreSQL

Prilikom odabira baze podataka odlučio sam se za PostgreSQL[12]. Prednosti PostgreSQLa u usporedbi s MySQL[13], Oracle[14] ili NoSql bazom su sljedeće:

- Open Source je za razliku od MySQLa ili Oraclea
- Gubitak podataka prilikom neispravnih upita je nemoguć
- Restrikcije upita su strože te je minimalna mogućnost pogreške
- Koristi SQL sintaksu
- Veća mogućnost relacija

[11] QraphQL - https://graphql.org/ (2018.)
[12] PostgreSQL - PostgreSql: The world's most advanced open source relational database. https://www.postgresql.org/ (2018.)
[13] MySQL - https://www.mysql.com/ (2018.)
[14] Oracle - https://www.oracle.com/database/index.html (2018.)

- Kako je najveći dio aplikacije prikazivanje i pisanje podataka PostgreSQL je brži u usporedbi s ostalima
- Sigurnost je statistički bolja
- Podržava veći broj tipova podataka
- Podržava JSON, XML i ostale NoSql tipove podataka
- Podržava Materijalizirane prikaze i privremene tablice dok ostali samo privremene tablice
- Veći broj ugrađenih funkcija za manipulaciju podataka i izvođenje statističkih analiza direktno u bazi podataka
- Mogućnost dodavanja vlastitih funkcija i tipova
- Kompatibilno s većim brojem programskih jezika

2.5. Knex

Knex[15] je jedna od biblioteka koja služi za pojednostavljivanje same sintakse u kodu. Knex služi klijentskoj strani koda kao okvir za sql upite. Usporediti ga se može sa NoSql jezikom tj. okvirom MongoDB[16]. Glavna uloga Knexa je da se uz pomoć gotovih funkcija koje imaju intuitivne nazive za programere skrate linije koda. Funkcije u Knexu jednostavno zamjenjuju tradicionalne sql upite međutim moguće ih je i kombinirati.

Jednostavan primjer Knex funkcija bi bio sljedeći:**SQL:**

SELECT * FROM table_name;

SELECT first_name FROM table_name WHERE id=1341;

Knex:

knex.select();

knex.select('first_name').where({id=1341});

** knex prije funkcije je proizvoljan naziv te obilježava u kodu konekciju na bazu.

Npr**.**

[15] KnexJS - https://knexjs.org/ (2018.)
[16] MongoDB - https://www.mongodb.com/ (2018.)

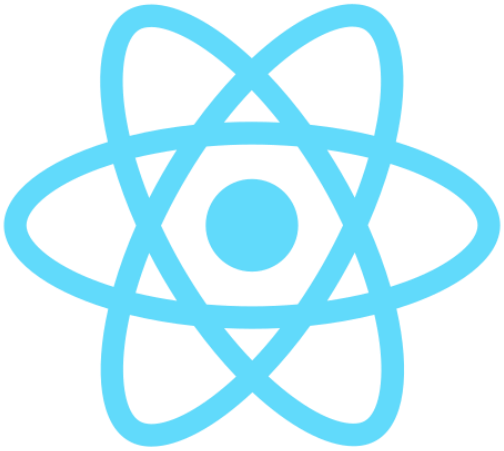
```
const knex = require('knex')({
  client:'pg',
  connection:{
    host: 'localhost',
    port: 5432,
    user: 'postgres',
    password: 619hdH9kJj',
    database: 'sc_db'
  }
});
```

2.6. Mogućnosti dodavanja tehnologija

Neke od mogućnosti za optimizaciju koda i same aplikacije je odabir još jedne tehnologije koja nije korištena u ovome projektu zbog jednostavnosti same aplikacije te bi dolazilo do komplikacija prilikom razvoja. Na klijentskoj strani koda korištena je tehnika pisanja kontejnera koji je centraliziran i služi kao „ljepilo“ između klijent i poslužitelj strane. Za projekt ove veličine i ovakvih zahtjeva je takav način i više nego dovoljan međutim ukoliko bi ovakva aplikacija zaživjela na tržištu potrebno je napraviti neke izmjene uključujući odabir novog poveznog sloja klijenta i poslužitelja. Trenutne mogućnosti su bezbrojne, ali favoriti postoje. Prvo imamo Relay[17] koji bi činio savršenog trećeg iz redova Facebooka, ali nažalost pri prevelikim aplikacijama zna stvarati probleme i nije idealan u stvarnim situacijama. Druga mogućnost bi bio Apollo[18] koji je poput Relaya razvojni okvir za kontejnere te se bolje ponaša u većem broju situacija nego Relay. Zadnja mogućnost bi bilo korištenje Reduxa[19] koji se razlikuje od prethodno navedenih tehnologija te ne simulira kontejner već privučene podatke slaže

[17] Relay - https://facebook.github.io/relay/ (2018.)
[18] Apollo - https://www.apollographql.com/ (2018.)
[19] Redux - https://redux.js.org/ (2018.)

u zasebne objekte. Zove ga se i „State managerom“ s obzirom da u React i React Nativu imamo *state* (eng.stanje) koji ćemo objasniti naknadno.



Slike 1 Logotip platforme React

3. IZRADA MOBILNE APLIKACIJE „STUDENTSKI POSAO“

3.1. Struktura

Kada govorimo o strukturi mobilne aplikacije Studentski Posao možemo reći da se sastoji od tri dijela, a to su klijent strana (front-end), poslužitelj strana (back-end) i baza podataka. Za sada preskočimo bazu podataka jer ćemo više o njoj kasnije te se koncentrirajmo trenutno na klijent i poslužitelj stranu.

3.1.1.Klijent

Za početak pojašnjenje što je točno klijent strana. Klijent strana ili poznatija u svijetu pod engleskim nazivom front-end strana je u grubo opisano sve što korisnik aplikacije vidi i sve s čime korisnik ima interakciju kao npr. dugmadi, polja za unos podataka, spuštanje zaslona i slične interakcije. U aplikaciji Studentski Posao možemo pronaći

jasnu strukturu aplikacije zato što koristimo React Native koji nas na dobar način tjera biti uredan i discipliniran prilikom konstruiranja i arhitekture aplikacije.

Kako React Native radi s komponentama koje se najčešće nazivaju i ekranima najveći dio projekta zauzimaju ekrani koji su logički raspoređeni u datotečnom sustavu. Mape se nazivaju po funkcionalnosti određenog ekrana, a sami ekrani dobivaju opisni naziv pisan CamelCase principom isto po svojoj funkcionalnosti, ali im se na kraju dodaje još riječ Component (eng. komponenta). Osim ekrana tj. komponenti na klijentskoj stani projekta imamo još nekoliko određenih datoteka i mapa. Moramo imati odvojenu mapu (u ovom slučaju je to API mapa) u kojoj se nalazi glavni univerzalni kontejner koji služi za logiku dohvaćanja i prikazivanja podataka zatim moramo imati u toj mapi datoteku s glavnim upitom na bazu te index datoteku kako bi lakše raspoznavali gdje se što nalazi. Na klijentskoj strani projekta dobro je imati odvojenu mapu sa svim potrebnim datotekama koje pokreću navigaciju kroz aplikaciju te najvažniji pokretač cijele klijentske strane je App.js koja je povezana na glavnu indeks datoteku. Osim navedenih datoteka imamo i one krucijalne kao što su package.json u kojemu se nalaze sve poveznice na instalirane pakete, a instalirane pakete pronalazimo u generički proizvedenoj mapi node_modules.

3.1.2. Poslužitelj

S druge strane kada spominjemo naš API tj poslužitelj stranu aplikacije možemo napomenuti da je njena struktura puno kompleksnija. Poslužitelj strana sadrži tipove koji se nalaze u zasebnim mapama s nazivom intuitivnim korisniku tj. programeru. Svaka mapa sadrži tip entiteta u kojemu su definirani atributi, zatim se u istoj mapi nalazi polje entiteta u kojemu vraćamo tražene podatke te je poželjno imati entitet za prikupljanje svih elemenata nekog tipa. Osim tipova i entiteta možemo imati mutacije. Mutacije su jednostavne CRUD funkcionalnosti koje služe za manipulaciju podataka. Osim navedenih datoteka i mapa imamo i ključne datoteke kao što su schema.js koja nije ništa drugo nego skup tipova i mutacija te imamo knex.js datoteku koja služi za povezivanje na bazu podataka.

Dakle kao zaključak strukture projekta kada govorimo o klijent i poslužitelj strani možemo reći da je poslužitelj strana ona koja vrši cijelu logiku aplikacije od dohvaćanja

podataka do manipulacije samih podataka, a klijent strana je ona koja dohvaća potrebne podatke sa poslužitelj strane te ih prikazuje u sučelju na način koji je predviđen.

3.2. Funkcionalnost aplikacije

Trenutno je aplikacija u razvoju, ali sadrži već najvažnije dijelove. Funkcija ovakve aplikacije je da skrati vrijeme traženja studentskog posla, podnošenja zahtjeva za objavljivanje oglasa te modernizacija. Kako bih opisao tok razvoja same aplikacije potrebno je opisati potencijalni scenariji korištenja aplikacije sa strane studenta i poslodavca.

Zamislimo sljedeću situaciju, poslodavac ima previše posla i želi primiti studenta za rad, a kako bi skratio putovanje i pozive prema studentskoj službi, poslodavac stvara jednom korisnički račun na aplikaciji Studentski Posao te objavljuje oglas za posao koji je putem obaveznih polja za unos podataka ispunio.

Student može pregledavati oglase za posao bez potrebne registracije te su mu dostupne sve glavne informacije i uvjeti. U slučaju da je student registrirani korisnik, on može ispuniti svoj online portfelj unutar aplikacije te na oglasu za posao putem jednog klika šalje svoje podatke poslodavcu na email. Sve daljnje radnje studenta i poslodavca su trenutno van domene funkcionalnosti aplikacije. Kasnijoj verziji je moguće nadodati i razviti sustav za dopisivanje.

3.3. Razvojno okruženje

Tijekom razvoja mobilne aplikacije “Studentski posao” korišteni su alati za razvoj koji su ili otvoreni izvor tj. Open source ili dobiveni studentskom licencom na korištenje u razdoblju od godinu dana. Kao razvojno okruženje koristio sam sljedeće alate:

- Visual Paradigm 15.0
- Visual Studio Code
- JetBrains WebStorm
- pgAdmin 4
- JetBrains DataGrip

- Android Studio i Xcode
- Expo
- Git Bash
- GraphiQl
- AVD (Android Virtual Device)

3.3.1. Visual Paradigm

Visual Paradigm je alat koji služi za grafički nacrt tj. UML shemu relacijske baze podataka i shemu aplikacije te njenih relacija.

3.3.2. Visual Studio Code

Visual Studio Code je uređivač teksta namijenjen za pisanje koda. To je jedan među prvim Open Source proizvodima od Microsofta te je stekao veliku popularnost među programerima jer je jednostavan, brz i efikasan. Prilikom izrade aplikacije korišten je samo za testiranje određenih manjih segmenata aplikacije te nije korišten u krajnjem razvoju mobilne aplikacije.

3.3.3. JetBrains WebStorm

JetBrains WebStorm je kategoriziran u najpopularniji IDE (Integrated Development Environment) te je korišten kao glavni uređivač koda za ovaj projekt. WebStorm je dobiven pomoću studentske licence koja vrijedi 365 dana od aktivacije te je s tom istom licencom moguće koristiti sve proizvode od JetBrainsa. WebStorm je karakterističan što okuplja sve funkcionalnosti većine uređivača teksta i ostalih IDE-a. Funkcionalnosti koje valja napomenuti jesu integrirani terminal koji je moguće otvoriti više puta te ga prepoloviti na pola ako je potrebno pratiti više akcija od jednom. Druga vrlo vrijedna funkcionalnost je korisničko sučelje za git koje olakšava sve osnovne akcije git terminala, a ponajviše pomaže prilikom konflikata u kodu kojih u ovoj aplikaciji nije bilo jer je radila samo jedna osoba na tome. Treća funkcionalnost je s moje subjektivne

strane najkorisnija a to je istovremeno izmjenjivanje varijabli i imena datoteka. Prilikom izmjene naziva datoteke mijenja se naziv iste datoteke na svim mjestima u projektu, a tako i varijabli. Zadnje navedena funkcionalnost je bila od najveće koristi u ovome projektu s obzirom na to da se nazivlje datoteka često mijenjalo ovisno o odvajanju većih komponenti u manje komponente. Ukupni zaključak za WebStorm IDE je da se radi o jednom naprednom i moćnom alatu za programera današnjice.

3.3.4. pgAdmin 4

Kako bi izradili bazu te ju uspješno pokrenuli lokalno za razvoj aplikacije, potreban nam je određeni alat. Alat koji je korišten za stvaranje i pokretanje PostgreSQL baze je pgAdmin koji je najpopularnije GUI sučelje za razvoj PostgreSQL baza. Za pgAdmin kažemo da je ujedno i pokretač poslužitelja za bazu podataka. Kako pgAdmin nažalost ipak nije najjednostavniji alat za korištenje potrebno je bilo korištenje jednostavnijeg alata kao što je DataGrip od JetBrainsa.

3.3.5. JetBrains DataGrip

DataGrip nije ništa drugo nego IDE za baze podataka. Krase ga opcije i funkcionalnosti kao prije navedeni WebStorm s obzirom da se radi o istome proizvođaču softvera. Funkcionalnosti kao što je izmjenjivanje tekstova istovremeno, predlaganje već postojećih naziva i sl. samo su neke od prednosti korištenja DataGripa. Stvaranje konekcija na poslužitelj tj. bazu podataka je vrlo jednostavna te nije ograničeno samo na PostgreSQL već i sve ostale vrste baza podataka. Moguće je imati otvorene konekcije na više različitih poslužitelja istovremeno te jedna od najvećih prednosti prilikom stvaranja baze je mogućnost dijeljenja zaslona s prikazom tablica na dvije ili više integriranih prozora. Ovaj alat u kombinaciji sa WebStormom za mene osobno predstavlja savršenu razvojnu okolinu za jednog programera radilo se o webu, mobilnim aplikacijama ili znanstvenim izračunima.

3.3.6. Android Studio

U razvoju aplikacije „Studentski Posao“ Android Studio nije korišten međutim tijekom daljnjeg razvoja bi bio krucijalan za završetak projekta. Razloga ima mnogo, ali jedan

od najvažnijih je taj da nažalost React Native još nije u potpunosti razvijen kako bi nadomijestio nativni razvoj aplikacija. Srećom moguće je kombinirati React Native te isto tako dijelove koda pisati u nativnom okruženju kao što su Android Studio (Java[20]) i Xcode (Swift[21]) za IOS. Najčešći slučaj korištenja nativnog okruženja je kada se želi napraviti tkz. Splash Screen tj. ekran koji se pojavljuje na samome početku aplikacije sve dok se React Native aplikacija ne učita u potpunosti. U suprotnome vidjeli bi samo bijeli zaslon te bi korisnik odmah odustao od korištenja aplikacije. Još jedan razlog korištenja nativnog okruženja je nekonzistentnost u razvoju React Nativa. Prva zamisao Facebooka je bila da će React Native biti samo razvojno okruženje za IOS te je naknadno obrađen za razvoj Androida. Zbog navedene činjenice Android ponekad kasni u razvoju te moramo uzeti u obzir da Android uređaja ima puno više vrsta nego IOS uređaja kojih je svega desetak, što naravno olakšava i razvoj aplikacija.

Kada je potrebno razvijati određenu nativnu komponentu nije nužno pokretati Android Studio ili Xcode, već je moguće sve to odraditi unutar projekta koji je već otvoren u određenom uređivaču teksta. Ukoliko je projekt kreiran putem npm-a ili automatskog kreiranja WebStorm IDE-a onda možemo pronaći datoteke android i ios u projektu, a ukoliko stvaramo projekt pomoću react-native-cli-a onda je potrebno pokrenuti naredbu unutar projekta „npm eject“ koja će se pobrinuti da raspakira projekt iz Expo okvira.

3.3.7. Expo

Prethodno smo spomenuli riječ Expo[22] što je naziv jedne tehnologije razvijene od strane Facebooka kako bi se pojednostavio razvoj aplikacija u React Nativu. Expo je okvir unutar kojeg se nalazi bazna struktura projekta te brine o svim dodatnim poslovima za programera kao što je održavanje i obnavljanje paketa, pokretanje aplikacije, debagiranje i ostale nužne, ali zamarajuće radnje. Projekt „Studentski Posao“ je izrađen uz pomoć Expo okvira kako bi se pojednostavio i ubrzao tok razvoja aplikacije.

[20] Java - https://www.java.com/en/ (2018.)
[21] Swift - https://developer.apple.com/swift/ (2018.)
[22] Expo - https://expo.io/ (2018.)

3.3.8. Git bash

Git bash služi kao ljuska koja pruža unix naredbe i prvenstveno služi za pokretanje skripti i git naredbi.

3.3.9. GraphiQL

Kada je potrebno testirati poslužitelj stranu aplikacije dok još uvijek nemamo klijent stranu gdje bi mogli vidjeti rezultate kao što je bio slučaj u ovoj aplikaciji, onda je za GraphQL api najbolji alat GraphiGL. Graphiql je web aplikacija razvijena od Facebooka koja služi kao gui sučelje za pokretanje GraphQL upita i mutacija tj. za pokretanje gotovih API-a. Graphiql osim što nudi unos upita i mutacija te njihov rezultat, pruža i dokumentaciju samoga API-a ukoliko je u kodu dodan opis svake mutacije i upita.

3.3.10. AVD (Android Virtual Device)

AVD je u prijevodu Android virtualni uređaj koji djeluje na računalu kao virtualna mašina sa sučeljem u obliku pametnog telefona. Prilikom razvoja aplikacije vrlo je zahtjevno pokretati aplikaciju na pravom uređaju iako je i to moguće te neki favoriziraju takav razvoj, ali brži i kvalitetniji način razvoja je u virtualnoj android mašini koja ima sve funkcionalnosti pravoga uređaja. Performanse virtualnog uređaja ovise o performansama računala te ovisno o kapacitetima virtualni android mijenja svoje ponašanje. AVD je sastavni dio Android Studia te se može od tamo i pokretati međutim brži način je preko komandne linije u Git bashu.

3.3.11. Operacijski sustav Windows Professional 10

Prilikom izrade aplikacije korišten je Laptop Lenovo Z50 s Intelovim i5 procesorom četvrte generacije 1.70GHz – 2.40GHz, 8GB radne memorije i 1TB memorije na tvrdom disku. Laptop koristi Windows 10 Professional kao glavni operativni sustav te se i u njemu razvija aplikacija. Važno je napomenuti kako operativni sustav poput Windowsa nije idealan za razvoj web aplikacija, mobilnih aplikacija ili bilo kojeg softvera ukoliko se ne radi o programskom jeziku .NET[23]. U ovom slučaju korišten je Windows jer je glavna svrha računala odrađivanje poslova u studijske svrhe te iz tog razloga nisam prelazio na drugi operativni sustav. Većina programera danas koristi Linux

[23] .Net - https://www.microsoft.com/net (2018.)

distribucijske sustave kao npr. Ubuntu[24], Debian[25] ili neku drugu distribuciju, a programeri koji imaju priliku raditi na Appleovim uređajima poput MacBooka koriste XOS operativni sustav koji je vrlo jednostavan za korištenje, baziran je na unixu, ali ga je moguće pokretati samo na Apple uređajima koji cijenovno nisu dostupni bilo kome.

Savršena okolina za razvoj softvera ne postoji. Svaki programer ima svoj način razvijanja softvera te učeći i po iskustvu bira programske jezike, tehnologije i okruženje u kojemu će razvijati osim ako je zaposlenik neke tvrtke koja od njega ili nje zahtjeva da koristi određenu okolinu.

3.4. Arhitektura i pokretanje projekta

Pokretanje projekta je vrlo jednostavno međutim prvo da objasnim kako se započinje projekt. Prvi dio je razvijanje ideje i bilježenje svakog mogućeg detalja. Kada se stvori određena slika aplikacije u glavi, vrijeme je da se svrstaju prije navedeni detalji u kategorije, a zatim pretvore u veće koncepte. Drugi korak je skiciranje sučelja tj. ekrana na papir i razmatranje rasporeda aplikacije i njegovog toka.

Nakon što je gore navedeni dio obavljen potrebno je razmisliti koji će nam podaci biti potrebni te je nužno napraviti okvirnu shemu buduće baze podataka. Kada je shema dovršena imamo već dobro uvježbani tok aplikacije te je vrijeme za izgradnju aplikacije.

Tijekom početka razvoja aplikacije baza i struktura aplikacije će se često mijenjati, ali to je uredu. Kada smo dovršili bazu vrijeme je za izradu poslužitelj strane koja je kostur aplikacije i koja pokreće sve funkcionalne dijelove korisniku vidljive. Prije u poglavlju Struktura je navedeno što radi klijent, a što poslužitelj strana pa to ovdje nećemo ponavljati. Ukratko kada smo odredili određene tipove podataka u API-u, onda odlučujemo gdje su nam potrebne mutacije jer neki tipovi neće imati potrebu za stvaranjem, brisanjem i izmjenjivanjem. Dovršeni API i bazu podataka je potrebno dobro testirati, tek kada je to dobro obavljeno možemo prijeći na klijentsku stranu te napraviti sve potrebne korake kako bi se spojili na API.

[24] Ubuntu - https://www.ubuntu.com/ (2018.)
[25] Debian - https://www.debian.org/ (2018.)

Poslužitelj strana aplikacije je kreirana pomoću integriranog alata za stvaranje NodeJS projekta u WebStormu, dok je klijent strana stvorena pomoću Expo okvira tj. terminal naredbe create-react-native-app ime_aplikacije.

Da bi pokrenuli projekt moramo provjeriti koje su nam zadane naredbe za takve radnje. Naredbe se nalaze u package.js datoteci te je uvijek jedna za pokretanje poslužitelja, a najčešće je to npm start osim ako ih neko izmjenjuje, pa tako imamo:

- npm run development – pokretanje projekta u razvojnoj okolini
- npm run production – pokretanje projekta u publiciranoj okolini

Za pokretanje prije navedenog virtualnog Android uređaja potrebno je putem Android Studia ili komandne linije pokrenuti virtualni uređaj.

Naredba za pokretanje AVD-a u datoteci gdje se nalazi:

- emulator –avd naziv_virtualnog_uređaja

Kako bi API uspješno radio ne smijemo prije pokretanja samog API-a pokrenuti i poslužiteljna kojem nam se nalazi baza, radilo se o lokalnom poslužitelju, vanjskom poslužiteljuili oblaku.

Nakon što smo pokrenuli poslužiteljs bazom podataka i naš API možemo pokrenuti i AVD ili spojiti pravi mobilni uređaj na računalo, ali obavezno prije odobriti opciju za debagiranje.

Kada je sve navedeno pokrenuto možemo pokrenuti i klijent stranu koja se pokreće na dva načina. Prvi način je ako smo aplikaciju kreirali na klasičan način ili putem WebStorma, tada nam je naredba za pokretanje aplikacije react-native run-android, a ako se radi o Expo okviru dovoljno je u terminal upisati npm run android.

3.5. Sučelja, dizajn i tok aplikacije

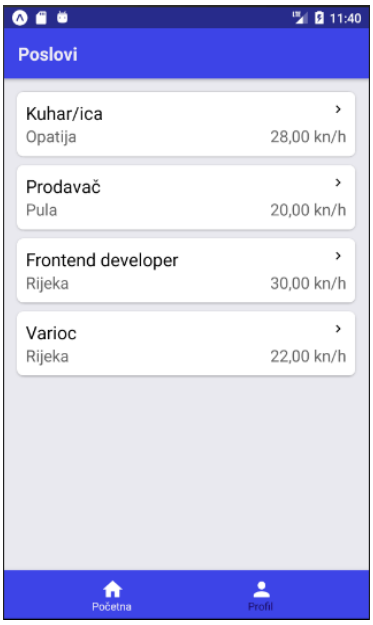
Koliko mali bili ekrani mobilnih uređaja toliko je i zahtjevno napraviti idealan dizajn aplikacije. Potrebno je u jako malo prostora postaviti sve zamišljene funkcionalnosti, a da je iskustvo korisnika na visokoj razini te da je intuitivno što ponekad predstavlja veliki problem. S druge strane je dobro što su se kroz godine razvili neki standardi

dizajniranja aplikacija. Jedan od najvažnijeg standarda je dizajn navigacije koja je krucijalna u aplikaciji. U aplikaciji „Studentski posao“ korištena je tab navigacija tj. kartična navigacija te se konkretno koriste samo dvije kartice što je minimalno potrebno za tako jednu aplikaciju. Prva kartica je ujedno i početni zaslon aplikacije gdje se nalaze aktualni oglasi za posao dok se u drugoj kartici nalazi korisnički profil ukoliko je korisnik prijavljen, a ako nije prijavljen nudi mu se opcija da se prijavi ili registrira ako još nema korisnički račun.

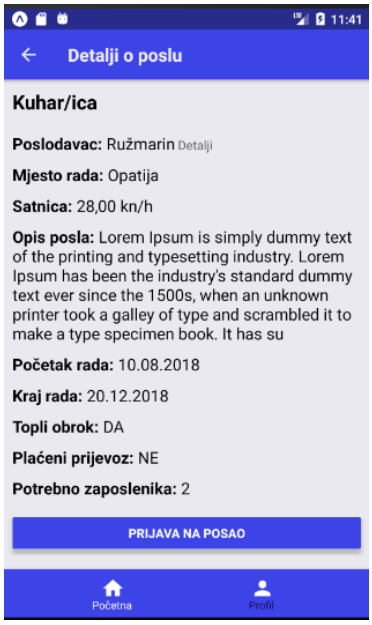
Na početnom ekranu kao što je spomenuto vidljivi su zadnji oglasi za posao koji su objavljeni. Svaki oglas prikazan je u zasebnoj kartici te su u njoj istaknute informacije kao što je naziv posla, mjesto rada i satnica. Klikom na određeni posao moguće je vidjeti sve potrebne detalje o poslu, kao što je naziv poslodavca, adresa, kontakt, opis posla i ostalo. Na dnu ekrana s detaljima oglasa se nalazi dugme ukoliko je korisnik prijavljen te je moguća prijava putem njega tj. slanje životopisa direktno poslodavcu koji je objavio oglas. Ako student želi saznati više o poslodavcu dovoljan je klik na natpis „Detalji“ pored naziva poslodavca.

Prilikom prijave korisnika aplikacije, korisnik bira dali je student ili poslodavac. Kod prijave kao poslodavac radi se automatska kontrola validnosti podataka u već postojećoj bazi Studentskog centra te se na taj način autorizira radi li se zaista o navedenom poslodavcu i dali taj poslodavac postoji. Nakon registracije korisnik student može popuniti svoj profil s osnovnim i detaljnim podacima te priložiti životopis ili motivacijsko pismo, dok poslodavac može objavljivati oglase. Kada poslodavac objavljuje oglas mora ispuniti sva obavezna polja kako bi student imao detaljan uvid u oglas posla.

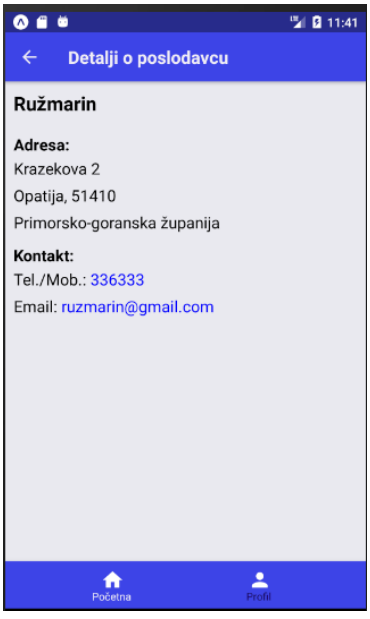
Dodatne manje funkcionalnosti aplikacije jesu klikom na kontakt broj poslodavca otvaranje aplikacije za pozive sa pretpunjenim brojem telefona ili klikom na email poslodavca otvaranje email klijenta na mobilnom uređaju.



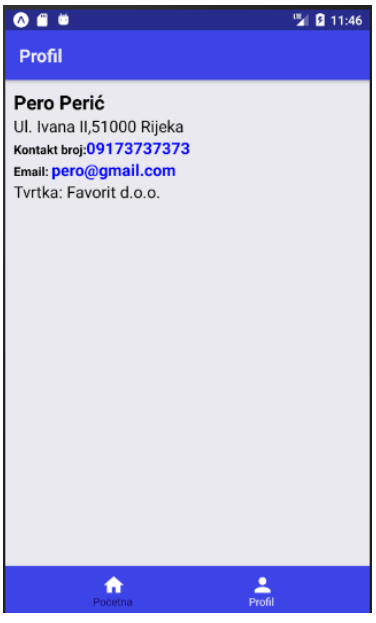
Slika 3 Sučelje: Početna/Poslovi



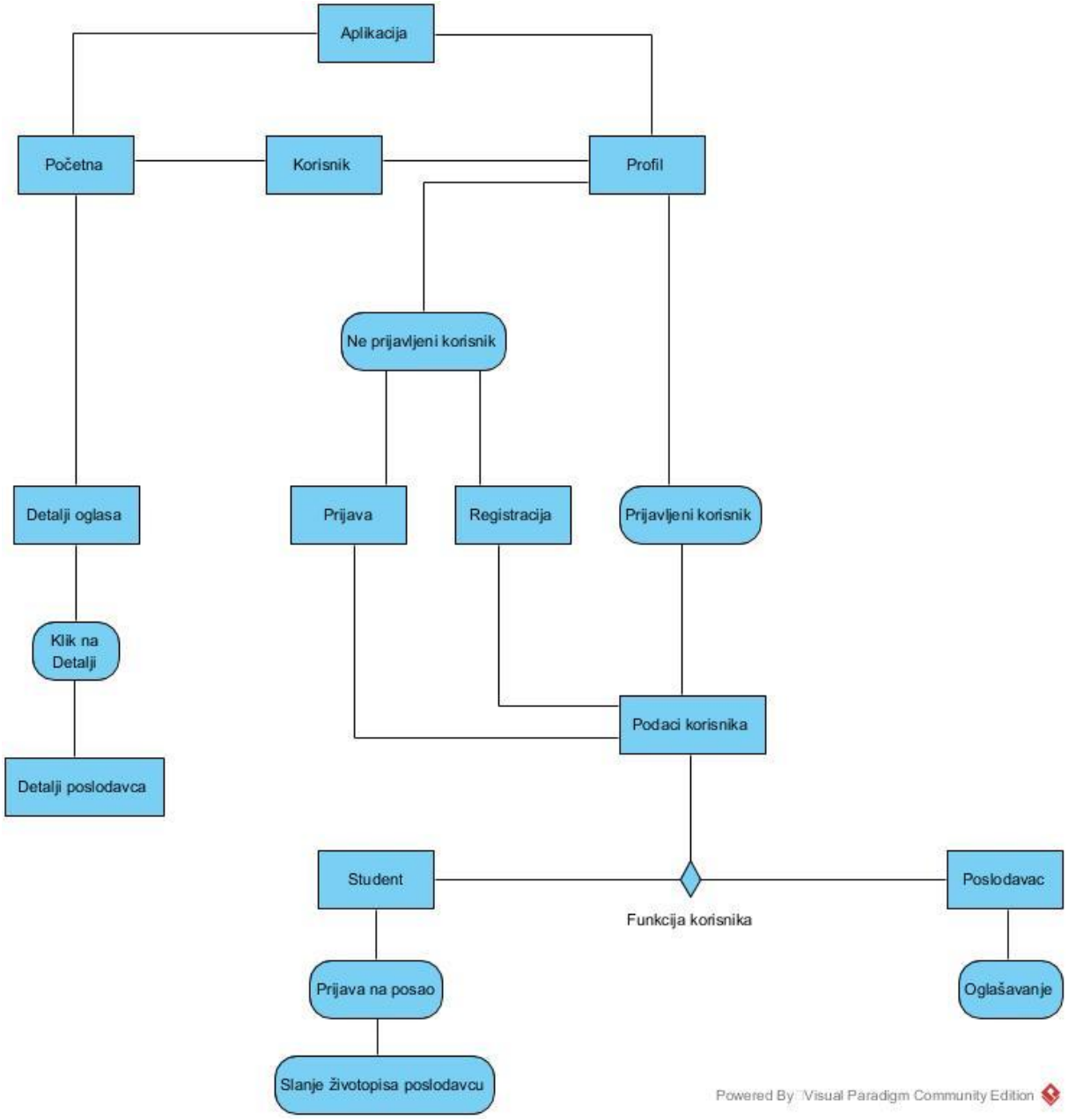
Slika 4 Sučelje: Detalji o poslu



Slika 5 Sučelje: Detalji o poslodavcu



Slika 6 Sučelje: Profil



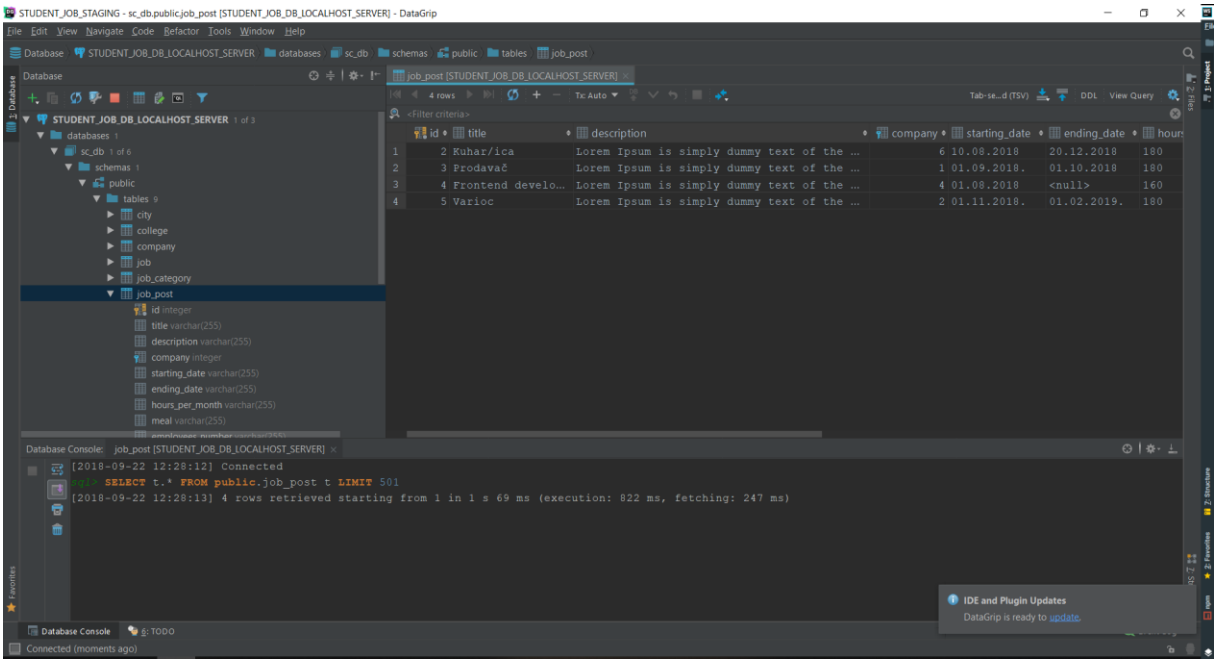
Slika 7 Shema toka aplikacije Studentski posao

3.6. Baza podataka

Najvažnija puzla aplikacije je temelj, a to je baza podataka. Struktura baze podataka je vidljiva na dalje navedenoj grafičkoj shemi te su vidljive i relacije. Tablice su provedene kroz normalne forme te su izmijenjene tijekom razvoja i testiranja aplikacije. U bazi se nalaze i neke trenutno suvišne tablice, ali će poslužiti u budućnosti prilikom daljnjeg razvoja aplikacije. Primjer takve tablice je Job tablica koja je trenutno nepotrebna međutim ubuduće kroz objavljivanje oglasa poslodavaca moći će se pohranjivati naslov posla u navedenu tablicu te kasnije olakšati korisniku unos naziva posla s padajućim izbornikom koji nudi od prijašnjih korisnika validne nazive posla.

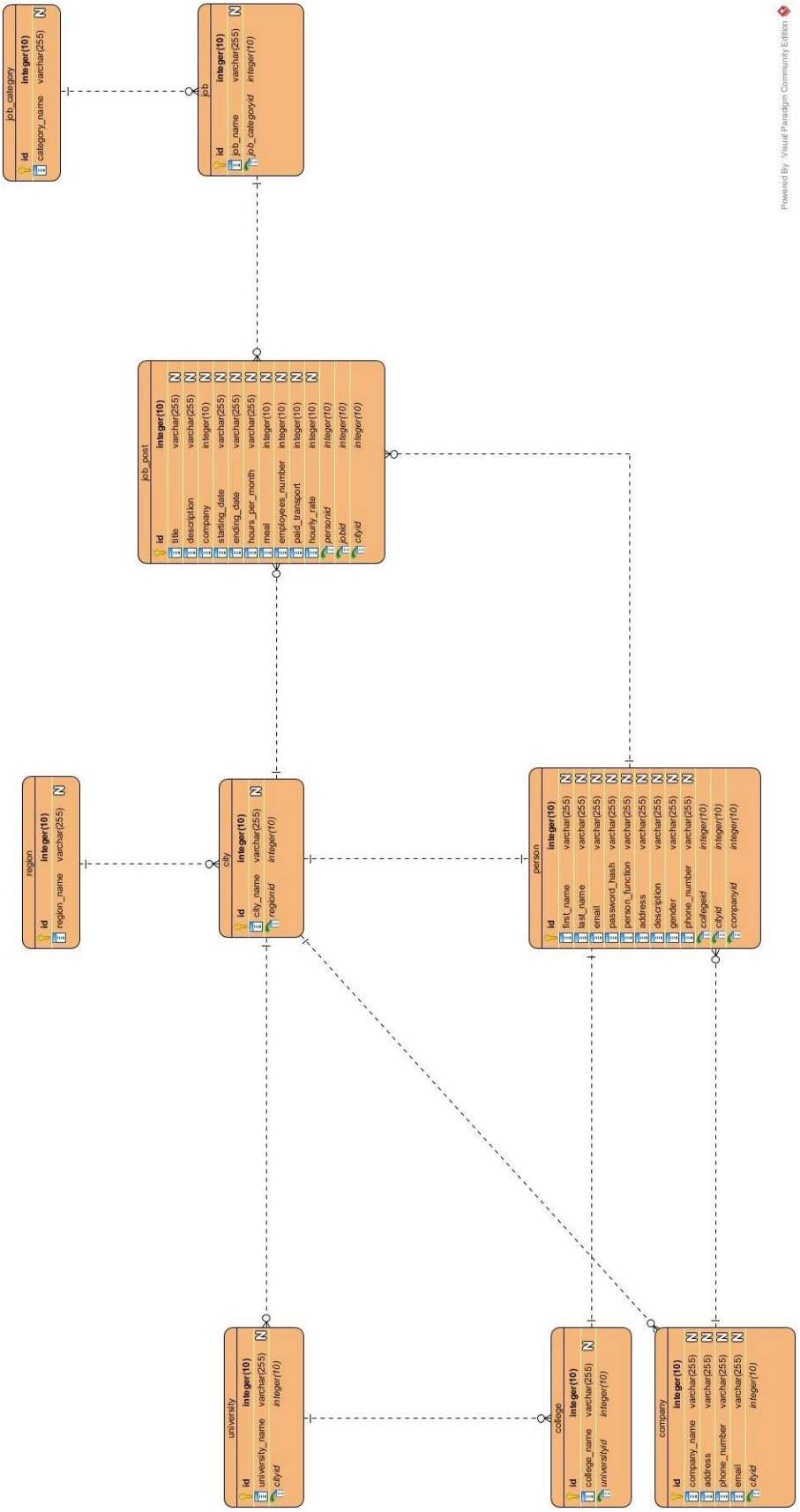
Baza podataka aplikacije „Studentski posao“ se sastoji od ukupno devet tablica koje su međusobno relacijski povezane. Kao što je prije navedeno korištena je PostgreSQL baza podataka, pgAdmin 4[26] kao pokretač lokalnog poslužiteljagdje se nalazi baza podataka te JetBrains DataGrip za lakše rukovanje poslužiteljomi bazom podataka.

Većim dijelom su tipovi podataka varchar s limitacijom od 50 do 255 znakova dok su svi identifikatori tipa int od 11 dozvoljenih znakova te su automatski inkrementalni.



Slika 8 Prikaz sučelja razvojnog alata JetBrains DataGrip

[26] pgAdmin 4 - https://www.pgadmin.org/ (2018.)



Slika 9 Shema baze podataka aplikacije Studentski Posao

3.7. Budući razvoj aplikacije

Kako aplikacija nije u potpunosti dovršena teško je govoriti o vremenu, resursima i budućem stanju aplikacije međutim teoretski možemo razviti aplikaciju da bude spremna za tržište.

Ideje za aplikaciju jesu podizanje ugovora za studente, prilaganje pdf dokumenata, filtriranje poslova po parametrima lokacije, vrste posla, datumu objave i sl., detaljnije uređivanje profila, slika na profilu, integrirani sustav za dopisivanje, ocjenjivanje poslodavca, objavljivanje dojmova o poslodavcu ili studentu kao djelatniku i još mnoge druge funkcije. Navedene ideje se odnose samo na aplikaciju kao „Studentski posao“. S velikim vremenskim okvirom, zainteresiranim timom studenata, programera moguće je proširiti mogućnosti aplikacije te napraviti jedinstveni sustav za studentske centre, studentska naselja i same studente.

Neke od mogućnosti:

- Sustav za smještajne jedinice – prijavljivanje na natječaj, prijavljivanje u objekt, otključavanje vrata sobe studenata pomoću unikatnog qr koda.
- Sustav za prehranu – prikaz dnevnog jelovnika, cjenika, radnog vremena i obavijesti, prikazivanje lokacija i trenutnog reda u liniji za uzimanje hrane i pića.
- Sustav za praćenje i menadžment financijskog budžeta studenta
- Sustav za praćenje studentskih aktivnosti
- Sustav za prijevozna sredstva – produljenje pokazne karte, kupnja karte za vožnju, vozni redovi i uživo praćenje autobusa (postojeća aplikacije tvrtke Ericsson)
- Sustav za razmjenu studenata, putovanja i sportskih aktivnosti
- Integrirani portal za skripte, materijale i ostalu literaturu (proširena postojeća skripta.hr)
- Sustav za upise i prijavljivanje ispita (postojeći studomat)
- Sustav za kolegije (MudRi/Merlin)

4. ZAKLJUČAK

Kao zaključak naveo bih da je ovaj završni rad kako praktični dio tako i pismeni bio uzbudljiv i vrlo poučan. Napravljen je dobar temelj za aplikaciju te vrijedi nastaviti razvoj. Za izradu kompletne aplikacije potrebno je više osoba, programera, sistem administratora, ljudi sa znanjem o projektiranju baza podataka i ostali. Smatram da ova aplikacija ima veliki potencijal na tržištu te se nadam da će neko to uočiti kako bi doprinijeo projektu. Većina dijelova za ideju ove aplikacije je uzeto iz svakodnevice jednog prosječnog studenta koji se susreće s problemima koji su pomoću ovakve aplikacije lako rješivi. U ovaj rad je uloženo jako puno truda i za priznati je da je pisanje ovog završnog rada i projekta bilo neprocjenjivo iskustvo koje me razvilo kao akademskog građana, a posebno kao studenta informatike i na kraju programera. Zaključak je da je od iznimne važnosti dobar odabir tehnologija s kojima se razvija aplikacija te ako se u počecima razvoja ne odaberu kompatibilne tehnologije može doći do velikih problema. Odabir tehnologija ne mora nužno biti kao u izradi aplikacije Studentski posao, ali trenutno za vrijeme razvoja ove aplikacije su navedeni alati najpopularniji i najviše zastupljeni u industriji, samim time je društvena zajednica jako velika te olakšava rješavanje problema. Neovisno o ovome završnom radu osobnu ću nastaviti razvijati aplikaciju Studentski Posao jer osim što u njoj vidim potencijal od velike mi je pomoći za naučiti nešto novo te se susresti sa situacijama koje moram prevladati, a moguće je da će mi na važnijim projektima zatrebati takvo iskustvo. Pisanje koda se uvijek može poboljšati te savršen kod ne postoji međutim ako se krene s dobrim temeljima kasnije je lakše refaktorirati i optimizirati kod. Često ćemo se susresti s problemima i neće nam biti jasno zašto naša logika ne funkcionira i često programeri krive tehnologiju ili razvojno okruženje, ali valja zapamtiti da kod radi što mu kažemo, a ne što želimo.

LITERATURA

1. React, A JavaScript library for building user interfaces. https://reactjs.org/ (2018.)
2. React Native, Build native mobile apps using JavaScript and React. https://facebook.github.io/react-native/ (2018.)
3. GraphQL, Describe your dana. https://graphql.org/ (2018.)
4. PostgreSql: The world's most advanced open source relational database. https://www.postgresql.org/ (2018.)
5. NodeJS. https://nodejs.org/en/ (2018.)
6. Express. https://expressjs.com/ (2018.)
7. Knex, A SQL query builder for JavaScript. https://knexjs.org/ (2018.)