Sustav za dostavu paketa zasnovan na umrežavanju korisnika i lokacijskim podacima

Vrsaljko, Maja

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

SUSTAV ZA DOSTAVU PAKETA ZASNOVAN NA UMREŽAVANJU KORISNIKA I LOKACIJSKIM PODACIMA

Rijeka, studeni 2021.

Maja Vrsaljko
0069077833

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski sveučilišni studij računarstva

Diplomski rad

SUSTAV ZA DOSTAVU PAKETA ZASNOVAN NA UMREŽAVANJU KORISNIKA I LOKACIJSKIM PODACIMA

Mentor: doc. dr. sc. Sandi Ljubić

Rijeka, studeni 2021.

Maja Vrsaljko
0069077833

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, studeni 2021.

_____

Ime Prezime

Zahvala

Najveće zahvale upućujem svojim roditeljima na pruženoj mogućnosti školovanja i beskonačnoj podršci.

Zahvaljujem se prijateljicama Azri Subašić i Ivani Baćac na pomoći tijekom svih godina studija.

Hvala doc. dr. sc. Sandiju Ljubiću na mentoriranju i pruženoj slobodi pri izradi rada.

Sadržaj

Popis slika **viii**

Popis tablica **ix**

Popis isječaka programskog koda **x**

1 Uvod **1**

2 Postojeći sustavi zasnovani na umrežavanju i lokaciji **3**

2.1 Usluge zasnovane na lokaciji 4

2.2 Wolt 5

2.3 Glovo 6

2.4 Bolt 7

3 Implementacija izvornog sustava za dostavu paketa **8**

3.1 Arhitektura sustava 9

3.1.1 Tehnološki stog 10

3.1.2 Sigurnost 11

3.2 Poslužitelj 16

3.2.1 Struktura 17

3.2.2 REST API pristupne točke 18

3.3 Klijent 21
3.3.1 Struktura 21
3.3.2 Implementacija korisničkog sučelja 27
3.4 Implementacija značajnijih funkcionalnosti 30
3.4.1 Obavijesti 31
3.4.2 Umrežavanje korisnika 32
3.4.3 Praćenje lokacije paketa u stvarnom vremenu 33
3.4.4 Ocjenjivanje korisnika 35
3.5 Problemi pri razvoju sustava 36

4 Karakteristični slučajevi korištenja **37**
4.1 Prijava i registracija u sustav 37
4.2 Upravljanje vlastitim oglasima 38
4.3 Pretraga i prijava na postojeće oglase 40
4.4 Slanje i primanje paketa 41
4.5 Upravljanje vlastitim profilom 43

5 Zaključak **45**

Bibliografija **46**

Sažetak **50**

Popis slika

3.1 Arhitektura izvornog sustava 10
3.2 Arhitektura implementirnog Spring Boot Securitya u sustavu 14
3.3 Dijagram toka pristupa zaštićenim i nezaštićenim rutama 16
3.4 ERD baze poslužitelja 18
3.5 MVC arhitektura sustava 19
3.6 Stablo komponenti klijentske aplikacije 22
3.7 Usporedba Flux i Redux arhitekture 23
3.8 Zaslon sa svim oglasima klijentske aplikacije u vertikalnoj i horizontalnoj orijentaciji 28
3.9 Zaslon za stvaranje novog oglasa na različitim mobilnim uređajima . 30

4.1 Zasloni prijave i registracije u sustav 38
4.2 Zasloni s pregledom svih oglasa i ponudama na specifični oglas . . . 39
4.3 Zasloni s naprednom tražilicom i detaljima jednog oglasa 41
4.4 Zasloni svih dostava i praćenjem paketa u stvarnom vremenu 42
4.5 Zaslon za ocjenjivanje usluge korisnika 43
4.6 Zasloni s osobnim profilom i uređivanjem osobnih podataka 44

Popis tablica

3.1 REST pristupne točke poslužitelja 20

Popis isječaka programskog koda

3.1 JSON odgovor poslužitelja nakon uspješne autentikacije 12
3.2 Zaglavlje JWTa 13
3.3 Sadržaj JWTa 13
3.4 Redux funkcija za stvaranje akcije 25
3.5 Axios funkcija za dohvat podataka s poslužitelja 25
3.6 Funkcija za stvaranje Redux *slicea* 26
3.7 Komponenta za prikaz liste oglasa 29
3.8 Funkcija za slanje mobilnih obavijesti Expo Notifications APIu 32
3.9 Funkcija za slanje geolokacijskih informacija u stvarnom vremenu . . 35

Poglavlje 1

Uvod

Mobilni uređaji i internet revolucionizirali su komunikaciju i životni stil modernog čovjeka. Zahvaljujući mobilnim uređajima korisnici imaju mogućnost pristupiti informacijama gdje god se nalazili. Već od samih začetaka interneta znanstvenici su brzo zaključili da je ogromnu količinu informacija potrebno filtrirati i individualizirati [1]. Danas, primjerice rezultati Google tražilice za upit "restorani" će izgledati drugačije korisniku iz Rijeke u usporedbi s korisnikom iz Zadra. Google će korisniku prvo prikazati restorane u mjestu gdje se nalazi. Ovakvo filtriranje informacija, između ostalog, zasnovano je na lokaciji uređaja kojoj razni servisi, s prethodnim dopuštenjem, pristupaju.

Usluge zasnovane na lokaciji se mogu definirati kao servisi koji pružaju korisnicima informacije u ovisnosti o lokaciji mobilnog uređaja i kontekstu korisnika (dob, spol, interesi itd.) [1]. Ne mora nužno biti riječ o mobilnom uređaju, već bilo koji uređaj koji ima Global Positioning System (GPS) prijemnik. Takvi servisi široko su rasprostranjeni i primjenu pronalaze u raznim područjima poput navigacijskih sustava, asistirane vožnje, asistencije u zdravstvu i sportu, video igrama zasnovanim na istraživanju svijeta itd. U posljednjih deset godina društvene mreže omogućile su korisnicima da na razne načine prošire svoj sadržaj s novom dimenzijom, lokacijom. Društvene mreže zahvaljujući poznavanju lokacije svojih korisnika mogu povezati korisnike u blizini, obavijestiti ih o događajima koje bi ih mogle interesirati ili im prikazivati specifični skup oglasa.

U ovom radu implementiran je primjer sustava zasnovanog na lokaciji s elementima društvene mreže. Sustav je namijenjen korisnicima koji nude usluge prijevoza paketa ostalim korisnicima ili potražuju dostavu za vlastite pakete. Dostavu je moguće pratiti na mapi u stvarnom vremenu zahvaljujući pristupu lokaciji mobilnog uređaja dostavljača. Elementi društvene mreže implementiranog sustava su ocjenjivanje usluge korisnika, praćenje korisnika i interakcija s oglasima ostalih korisnika. Sustav je implementiran na principima Representational State Transfer (REST) arhitekture korištenjem modernih tehnologija. Poslužiteljska strana sustava implementirana je u Spring Bootu, a klijentska aplikacija u React Nativeu.

Poglavlje 2

Postojeći sustavi zasnovani na umrežavanju i lokaciji

Posljednjih godina popularne društvene mreže poput Facebooka, Twittera ili Instagrama uključile su razne značajke zasnovane na lokaciji u svoj postojeći sadržaj. Korisnici mogu dodati geografske značajke kao oznake na objavljene fotografije ili videozapise, podijeliti svoju lokaciju u stvarnom vremenu, potražiti događaje u njihovoj blizini, pretraživati ljude, oglase ili grupe prema lokaciji [1]. Korištenje lokacije u društvenim mrežama relativno je nova pojava, lokacija se inicijalno primarno koristili u navigacijskim sustavima.

U nastavku je detaljnije opisan razvoj usluga zasnovanih na lokaciji i dan je primjer nekoliko sustava koji nude usluge prijevoza raznih vrsta paketa i koji su motivirali implementaciju izvornog sustava. Neke od popularnih aplikacija u Hrvatskoj koje koriste geolokacijske informacije za dostavu svojih usluga i elemente društvene mreže su aplikacije za dostavu hrane poput Wolta i Glova i aplikacije za prijevoz ljudi Bolt ili Uber. Praćenje poštanskih pošiljki i dalje ovisi o pojedinoj dostavnoj službi i njihovoj infrastrukturi. U Hrvatskoj većina dostavnih službi prvo svoje korisnike obavijesti o zaprimljenom paketu kojem je dodijeljena identifikacijska oznaka. Korisnici tad putem njihovih web ili mobilnih aplikacija mogu pratiti paket preko dane identifikacijske oznake paketa. Paket se ne prati u stvarnom vremenu, već dostavljač ažurira lokaciju paketa po dolasku na određenu kontrolnu točku.

2.1 Usluge zasnovane na lokaciji

Usluge zasnovane na lokaciji (engl. Location-based services (LBS)) je generalni pojam koji obuhvaća razne programske alate koji koriste geolokacijske podatke za pružanje usluga svojim korisnicima. LBS ima primjenu u različitim područjima poput marketinga i oglašavanja, zdravstva, osobnog života, zabave i sl. Razvoj LBSa započeo je integracijom podataka iz satelita, mobilne mreže i naprednih grafičkih sučelja mobilnih uređaja [2]. Sve do kraja 1990-tih mobilni uređaji s vrlo jednostavnim grafičkim sučeljima podržavali su samo komunikaciju pozivom ili SMS-om. Već tad se mogla zaprimiti korisnička lokacija mobilnog uređaja koristeći lokalizaciju mobilne mreže. Ovakva jednostavna lokalizacija se najčešće koristila u slučajevima nužde. Početkom 21. stoljeća, razvojem grafičkog sučelja i sve široj rasprostranjenosti Wireless Application Protocol (WAP)a, koji je mobilnih uređajima omogućio pristup internetu započela je komercijalna upotreba LBSa [2]. Jedna od prvih takvih bila je FriendZone usluga koja je korisnicima omogućavala anonimno čavrljanje s ostalim korisnicima u blizini.

Inicijalno, za lociranje mobilnog uređaja koristila se zemaljska mobilna mreža. Pozicioniranje uređaja vršilo se mjerenjem kašnjenja signala na putu od najbližeg telefonskog tornja do uređaja [3]. Takvo pozicioniranje je često dosta neprecizno, ovisi o jačini signala i temelji se na lokaciji najbližeg tornja. Za preciznije lociranje ovom tehnikom radi se interpolacija signala iz više obližnjih tornjeva. Prednost korištenja ove tehnike je što ne zahtijeva nikakve dodatne hardverske ili softverske implementacije, već se u potpunosti oslanja na postojeću mrežnu infrastrukturu i danas se ponekad koristi u slučajevima nužde. Moderni mobilni uređaji su opremljeni GPS prijemnicima koji omogućavaju precizno pozicioniranje uređaja. GPS je razvilo američko ministarstvo obrane 1970. godine u vojne svrhe te se 1980. godine počinje koristiti u civilne svrhe [4]. Danas postoji 32 satelita u orbiti dediciranih za geolociranje korisnika od kojih su 24 uvijek aktivna [5]. Svaki satelit ima svoj unutarnji atomski sat i odašilje kodirani signal na specifičnoj frekvenciji koji sadrži informacije o njegovom položaju u određenom vremenskom trenutku. GPS prijemnik u mobilnom uređaju pronalazi najvidljivije satelite od kojih prikuplja podatke. Razlika u vremenu kada je signal primljen i kad je emitiran ukazuje na put koji je taj signal

prešao, odnosno udaljenost od satelita od mobilnog uređaja. Za uspješno pozicioniranje potrebni su podaci iz barem tri satelita uz pomoć kojih se onda, postupkom triangulacije, određuje položaj korisnika. Prednosti korištenja GPSa su očigledno precizno geolociranje korisnika, dok su mane velika potrošnja baterije mobilnog uređaja. Za što ispravniji rad GPSa potrebno je pregledno nebo bez većih prepreka što u velikim urbanim sredinama zna biti problem. U takvim situacijama koristi se Assisted Global Positioning System (AGPS) koji GPS prijemniku šalje dodatne podatke o lokaciji dobivene iz razmijenjenih signala s obližnjim telefonskim tornjevima [6].

Usluge zasnovani na lokaciji mogu se podijeliti u četiri osnovne kategorije [2]. Prvi su servisi koji pružaju pomoć pri navigaciji poput Google Mapsa i Apple Mapsa. Zatim kategorija servisa za praćenje u stvarnom vremenu poput servisa za praćenje prometa, pronalaska članova obitelji itd. Informacijski servisi pružaju informacije o specifičnom sadržaju koji je zanimljiv korisnika npr.turistički vodiči kroz gradove. Posljednji su aplikacijski servisi koji obuhvaćaju sve ostale aplikacije zasnovane na lokaciji poput raznoraznih igara, mobilnog oglašavanja, sustava za preporuku itd.

2.2 Wolt

Wolt je finska tehnološka tvrtka osnovna 2014. godine čiji je najpoznatiji proizvod istoimena mobilna aplikacija za dostavu hrane [7]. Aplikacija omogućava korisnicima narudžbu hrane iz restorana u mjestu u kojem se nalaze. Korisnici zatim status narudžbe i lokaciju dostavljača mogu pratiti u stvarnom vremenu. Wolt je pojednostavio proces dostave hrane tako da je umrežio restorane, dostavljače i korisnike na jedno mjesto pa tako danas broji preko pet milijuna registriranih korisnika, šest tisuća partner restorana i dvanaest tisuća partnera dostavljača u preko dvadeset država Europe. U Hrvatskoj je dostupan u većim gradovima poputa Zagreba, Rijeke, Osijeka, Splita, Zadra i Slavonskog Broda. Aplikacija je pojednostanila proces dostave osiguravši restoranima dostavljače.

Wolt je dostupan na Android i iOS mobilnim platformama i na svim popularnim internetskim preglednicima poput Google Chromea, Firefoxa, Microsoft Edga.

Prema StackShareu Wolt aplikacija implementirana je korištenjem React i React Native radnog okvira na klijentskoj strani i korištenjem Python programskog jezika na poslužiteljskoj strani. Podaci se čuvaju u PostgreSQL, MongoDB i Redis bazama. Prednosti korištenja Wolt aplikacije je već spomenuti proces narudžbe i dostave hrane, dok je nedostatak porast cijena usluge budući da putem Wolt aplikacije korisnik plaća dio usluge Wolt kompaniji i restoranu iz kojeg naručuje.

2.3 Glovo

Glovo je proizvod istoimene španjolske razvojne tvrtke osnovane u Barceloni 2015. godine [8]. Glovo je mobilna aplikacija, slična Woltu koja povezuje korisnike, poduzeća i dostavljače te omogućuje kupnju, primanje i slanje bilo kojeg proizvoda iz ponude aplikacije. Glovo korisnicima omogućava narudžbu hrane iz raznih restorana, namirnica iz supermarketa ili čak kupnju farmaceutskih proizvoda. Odnedavno Glovo u Hrvatskoj nudi usluge osobnog kurira za dostavu manjih paketa unutar istog grada. Sve narudžbe funkcioniraju na istom principu, dostava se prati na Google Maps karti u stvarnom vremenu, a po završetku dostave korisnik ocjenjuje uslugu dostavljača. Prisutan je u 24 države diljem svijeta u preko 400 gradovua. U Hrvatskoj je dostupan u većim gradovima. U svibnju 2021. godine Glovo je kupio konkurentsku platformu Pauza.hr koja je do tad bila u vlasništvu tvrtke Delivery Hero.

Mobilna aplikacija zatvorenog koda dostupna je na Android i iOS platformama, a iste usluge dostupne su i kroz web preglednike. Prema StackShareu aplikacija je implementirana korištenjem Python i Java programskih jezika te NodeJs i Django radnih okvira. Prednosti korištenja Glovo aplikacije su slične kao i kod Wolt aplikacije. Jednostavni proces narudžbe hrane i namirnica, plaćanje putem interneta i raznolika ponuda, dok je nedostatak veća cijena proizvoda u odnosu na cijenu istih proizvoda u restoranima ili dućanima.

2.4 Bolt

Bolt je estonska tvrtka osnovna 2013. godine koja korisnicima nudi usluge taksija, prijevoza paketa, hrane i iznajmljivanja romobila [9]. Svoje usluge nude u 45 država svijeta i imaju oko 75 milijuna korisnika globalno i oko 1.5 milijuna vozača [9]. U Hrvatskoj su dostupne dvije aplikacije, Bolt za usluge taksija i iznajmljivanja romobila i Bolt Food & Delivery za usluge dostave hrane i paketa. Sve usluge Bolt kompanije zasnovane su na lokaciji. Primjerice, korisnik kroz Bolt aplikaciju, na karti, pretražuje najbliži romobil. Korisnik zatim iznajmljuje romobil na određeni period vremena, a Bolt prati lokaciju romobila i korisnika. Također, na sličan način funkcionira Boltov prijevoz. Korisnik na karti vidi Bolt automobile koji mu mogu ponuditi prijevoz do željene lokacije. Bolt vozači svojim osobnim automobilima nude usluge taksija ostalim korisnicima i za svoje potrebe imaju Bolt Courier aplikaciju. Bolt Delivery usluga nudi prijevoz manjih paketa do 20 kilograma unutar istog grada. Unutar aplikacije pošiljatelj i primatelj mogu podijeliti plan rute dostave i u stvarnom vremenu pratiti kretanje pošiljke i termin isporuke.

Bolt je kompanija zatvorenog koda i tehnološki stog njihovih aplikacija nije javno poznat. Mobilne aplikacije su dostupne na Android i iOS platformama, a usluge taksija su dostupne i kroz web preglednike gdje se može naručiti vožnja bez registracije u sustav. Prednosti korištenja Bolt platforme za korisnika je dostupnosti više različitih usluga kroz jedan sustav, a nedostaci su cijene taksi usluga koje znatno variraju (ovisno o potražnji) te nepovjerenje u Bolt vozače koji nisu licencirani taksisti.

Poglavlje 3

Implementacija izvornog sustava za dostavu paketa

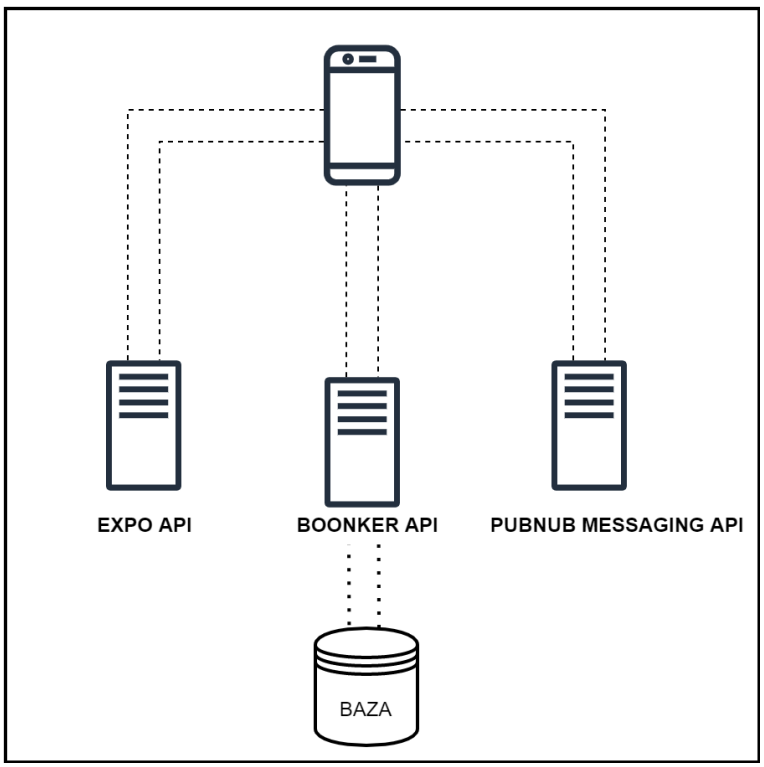
Izvorni sustav za dostavu paketa zasnovan na lokaciji i umrežavanju korisnika implementiran je u dvije aplikacije, klijentskoj i poslužiteljskoj. Sustav omogućava prijavu korisnika putem obrasca za prijavu korištenjem korisničkog imena i lozinke. Prijavljeni korisnik tada može pregledavati oglase svih ostalih korisnika. Pregled svih oglasa može se suziti pregledom samo oglasa korisnika koje prati ili naprednijom pretragom po raznim parametrima. Korisnik može objavljivati vlastite oglase u kojima nudi usluge prijevoza ili traži prijevoz za svoje pakete. Objavljene oglase može pregledavati, uređivati ili brisati. Korisnici se na aktivne oglase mogu prijavljivati, odnosno slati ponude koje autor oglasa prihvaća ili odbija. Prihvaćanjem ponude na oglas definira se nova dostava paketa u dogovorenom terminu. Vozač pokreće dostavu na zakazani datum i time aktivira zaslon putem kojeg se može pratiti lokacija paketa u stvarnom vremenu. Korisnici se međusobno ocjenjuju kroz obrazac za unos ocjene i komentara. Sve recenzije, broj pratitelja i sljedbenika pojedinog korisnika vidljive su na osobnim profilima.

3.1 Arhitektura sustava

Izvorni sustav za dostavu paketa implementiran je na temeljima REST arhitekture i sastoji se od klijentske i poslužiteljske aplikacije. REST je teorijski model programske arhitekture namijenjen ostvarivanju distribuiranih sustava na internetu. Web servisi korisnicima nude ograničenu količinu resursa do kojih se dolazi slanjem Hypertext Transfer Protocol (HTTP) zahtjeva na specifični Uniform Resource Locator (URL). Poslužiteljska aplikacija zadužena je za pohranjivanje i obradu podataka, dok je zadaća klijentske aplikacije prikaz tih podataka u korisničkom sučelju. Klijent i poslužitelj komuniciraju HTTP protokolom, primjerice, klijent šalje GET HTTP zahtjev na URL /boonker/api/advertisements/1, a poslužitelj mu vraća HTTP odgovor koji u tijelu odgovora sadrži JavaScript Object Notation (JSON) objekt sa svim traženim podacima i standardni HTTP status kod 200 koji označava uspješno izvršen zahtjev.

Kako bi se sustav mogao nazvati REST sustavom postoji nekoliko načela koje mora zadovoljiti. Prvo i osnovno načelo REST arhitekture kaže da za pristup svakom resursu mora postojati jedinstveni URL kojim se dohvaća specifični resurs. Nadalje, resursi moraju biti logično povezani; u izvornom sustavu za svakog korisnika postoje recenzije, te recenzije nemaju smisla ako nemaju referencu na korisnika. Za pristup resursima predviđa se upotreba GET, POST, PUT i DELETE HTTP standardnih metoda koje se nerijetko koriste nedosljedno. Flickr web servis u svojoj dokumentaciji navodi da za brisanje resursa koristi GET metodu umjesto standardne DELETE. REST načela također navode da je potrebno omogućiti višestruke reprezentacije HTTP odgovora u Extensible Markup Language (XML) zapisu ili JSON formatu [10] [11]. Web servisi često nisu dosljedni i ne implementiraju sva teorijski opisana načela pa se nazivaju RESTful servisima. Implementirani izvorni sustav nema podršku za višestruku reprezentaciju resursa pa je točnije zvati ga RESTful sustavom.

Pojednostavljena arhitektura izvornog sustava prikazana je na Slici 3.1. Klijentska aplikacija, osim s prethodno opisanom poslužiteljskom aplikacijom, komunicira s Expo i PubNub servisima. Expovi servisi nude jedno rješenje za Android i iOS platformu što omogućava ubrzan razvoj nativnih aplikacija. Klijent izvornog sustava



Slika 3.1 Arhitektura izvornog sustava

komunicira s Expo Location Application Programming Interface (API)em za upravljanje lokacijom i Expo Notification APIem za implementaciju mobilnih obavijesti. PubNub je platforma koja omogućava komunikaciju u stvarnom vremenu unutar aplikacija. Klijent se pretplaćuje na specifičan kanal na kojem objavljuje i prima poruke putem a za razmjenu poruku zasnovanog na konceptu objave i pretplate poruka (engl. *publish/subscribe messaging* API).

3.1.1 Tehnološki stog

Klijentska strana sustava implementirana je u React Native radnom okviru u JavaScriptu. React Native je popularni radni okvir prisutan u današnjem razvoju mobilnih aplikacija te je primarno odabran kako bi sustav bio podržan na Android i iOS platformama. Za stvaranje HTTP zahtjeva prema poslužitelju korištena je

Axios knjižnica, a za upravljanje stanjem aplikacije koristio se Redux Toolkit. Čitanje geolokacijskih informacija i implementacija primanja i slanja mobilnih obavijesti ostvarena je uz podršku Expo radnog okvira. Expo je platforma otvorenog koda koja sadrži razne alate i servise za univerzalni razvoj mobilnih aplikacija poput upravljanja lokacijom, plaćanja, autorizacije itd. Komponente korištene u aplikaciji dio su React Native Paper i React Native knjižnice zasnovane na Googlevom Material dizajnu, responzivne su i podržane na iOS-u i Androidu. Za upravljanje obrascima koristila se Formik knjižnica u kombinaciji s knjižnicom za validiranje polja Yup. Praćenje lokacije korisnika u stvarnom vremenu ostvarilo se uporabom PubNub platforme koja pruža API za razmjenu poruka.

Poslužiteljska strana sustava implementirana je u Spring Boot radnom okviru u Javi 11. Sigurnosni aspekt sustava implementiran je korištenjem Spring Boot Security radnog okvira. Za upravljanje podacima pohranjenim u H2 bazi koristio se Spring Data JPA radni okvir.

3.1.2 Sigurnost

Proces autentikacije iznimno je važan element informacijske sigurnosti budući da predstavlja prvi korak interakcije korisnika sa sustavom [12]. Autentikacija je mehanizam utvrđivanja identiteta korisnika pomoću različitih korisničkih akreditacija. Ovisno o potrebama i osjetljivosti sustava, autentikacija korisnika može se izvoditi u nekoliko provjera tzv. autentikacijskih faktora koje korisnik mora zadovoljiti. Najjednostavniji oblik potvrde korisničkog identiteta je jednofaktorska autentikacija koja provjerava par vrijednosti; korisničku oznaku i lozinku. Jednofaktorska autentikacija je popularni i široko rasprostranjeni mehanizam koji se koristi u različitim aplikacijama niskih sigurnosnih zahtjeva. Društvene mreže poput Facebooka, Twittera i Instagrama implementiraju opisani mehanizam. Višefaktorske autentikacije obuhvaćaju dodatne provjere pomoću token uređaja, jednokratnih SMS lozinki, potvrde identiteta putem e-pošte, biometrijskih provjera lica, otiska prsta, govora itd. [13]. Složeniji mehanizmi provjere koriste se u aplikacijama koje sadrže osjetljive korisničke podatke poput bankovnih aplikacija. Implementirani sustav koristi jednofaktorsku autentikaciju korisničkim imenom i lozinkom. U trenutku uspješne autentikacije ko-

risnika vrši se i autorizacija kojom se utvrđuje korisnička uloga u aplikaciji. Svaki korisnik, sukladno svojoj ulozi u aplikaciji ima pravo pristupa određenim resursima aplikacije.

Korisnik se u aplikaciju prijavljuje samo jednom, unosom korisničkog imena i lozinke, pri čemu, po uspješnoj autentikaciji poslužitelj vraća JSON odgovor dan u nastavku u Ispisu 3.1.

```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.
    eyJzdWIiOiJtdnJzYWxqa28iLCJpc3MiOiJib29ua2VyIi
    wiaWF0IjoxNjMzODkzMzAxLCJleHAiOjE2MzM5MDE5NDF9.
    -5g6SphdnUo-zVGGc0usvaDgve-mObWAYrCaj0cSA-v4KH
    12j8uV6ER60lo_e-g7GnfydAAbxTrybVH9pF0WyQ",
    "type": "Bearer",
    "id": 1,
    "username": "mvrsaljko",
    "email": "majavrsaljko78@gmail.com",
    "roles": [
        "ROLE_USER"
    ],
    "fullName": "Maja Vrsaljko"
}
```

Ispis 3.1 JSON odgovor poslužitelja nakon uspješne autentikacije

Odgovor, nevidljiv korisničkom sučelju, sadrži osnovne informacije o korisniku. Informacije o korisnike prikazane u 3.1, a to su jedinstveni identifikator u polju *id*, puno ime i prezime u polju *fullName*, korisničko ime u polju *username*, adresu elektroničke pošte u polju *email* i najvažnije polje *token* koje sadrži base64 šifrirani JSON Web Token (JWT). Prilikom svakog idućeg poziva poslužitelju, klijent mora slati JWT u zaglavlju HTTP zahtjeva pomoću *Bearer* sheme kako bi se autenticirao. JWT je kompaktni, samostalni JSON objekt definiran otvorenim Request for Comments (RFC) 7519 standardom koji služi za prijenos sigurnih informacija HTTP protokolom [14]. U svom kompaktnom obliku JWT je strukturiran od tri niza znakova odvojena točkom. Prvi niz je zaglavlje JWTa koji tipično sadrži informacije o

tipu tokena i kriptografskom algoritmu koji se koristi za šifriranje potpisa tokena. Dekodirano zaglavlje iz Ispisa 3.1 prikazano je na Ispisu 3.2, tip tokena je JWT, a za šifriranje se koristi HS512 kriptografski algoritam.

```
{
  "typ": "JWT",
  "alg": "HS512"
}
```

Ispis 3.2 Zaglavlje JWTa

Drugi niz znakova je sadržaj JWTa koji obuhvaća niz izjava (engl. *claims*) koje najčešće opisuju korisnika i aplikaciju koja je izdala JWT. Izjave se kategoriziraju kao registrirane, javne i privatne [14]. Registrirani zahtjevi su definirani RFCom 7519, neobavezni su i koriste se kako bi se olakšala i standardizirala upotreba JWTa. Neke od takvih izjava su "iss" kojim se definira entitet koji je izdao JWT, "sub" koji definira subjekta kojem je namijenjen token, "exp" koji označava vrijeme valjanosti tokena itd. Javne izjave su definirane od strane korisnika JWTa i registriraju se u Internet Assigned Numbers Authority (IAN) JWT Claims registru; primjer takvih izjava su "email" ili "zoneinfo". Posljednje izjave su privatne koje se usuglašavaju između stranaka koje koriste JWT; radi se o specifičnim izjavama koje se definiraju za potrebe aplikacije. Dekodirani sadržaj tokena iz Ispisa 3.1 je prikazan u nastavku i sadrži registrirane izjave o subjektu "mvrsaljko" kojem je namijenjena, izdavaču "boonker" aplikaciji, datumu izdavanja "iat" i vremenskom trajanju "exp".

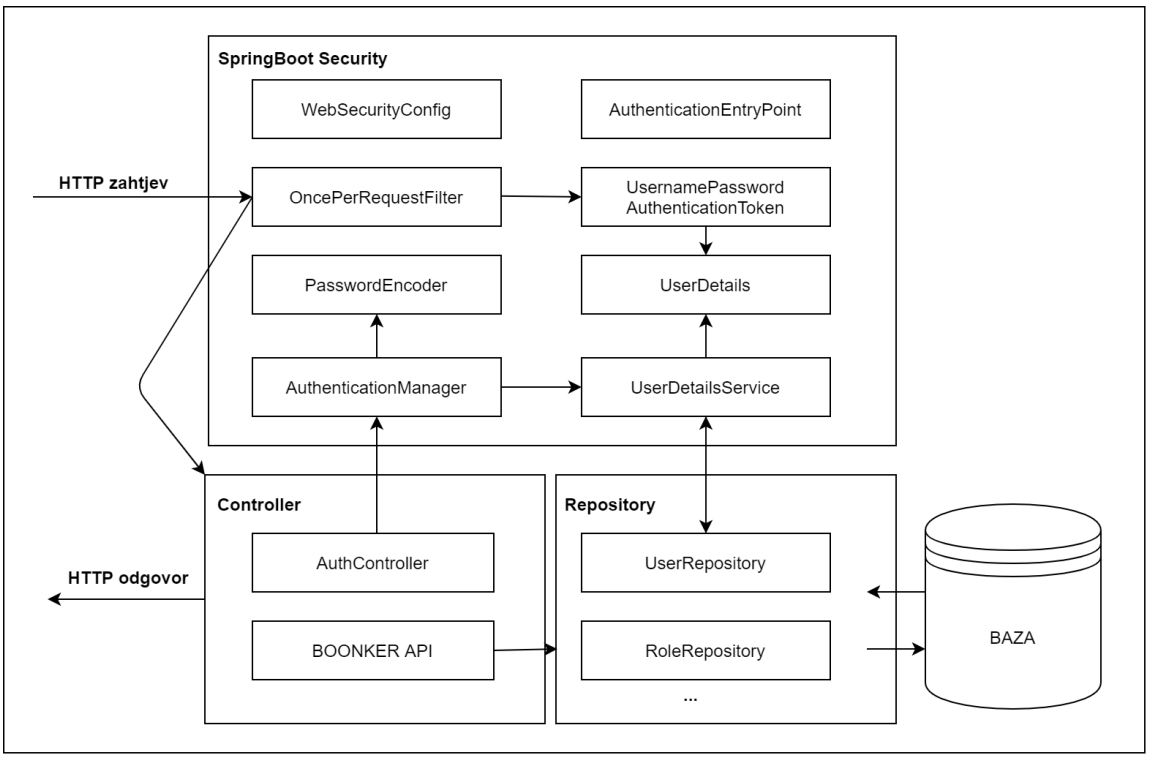
```
{
  "sub": "mvrsaljko",
  "iss": "boonker",
  "iat": 1633893301,
  "exp": 1633901941
}
```

Ispis 3.3 Sadržaj JWTa

Posljednji niz znakova tokena je potpis kojim se verificira autentičnost poruke. JWT zaglavlje i sadržaj se potpisuju s kriptografskim algoritmom definiranim u zaglavlju

JWTa i tajnom lozinkom.

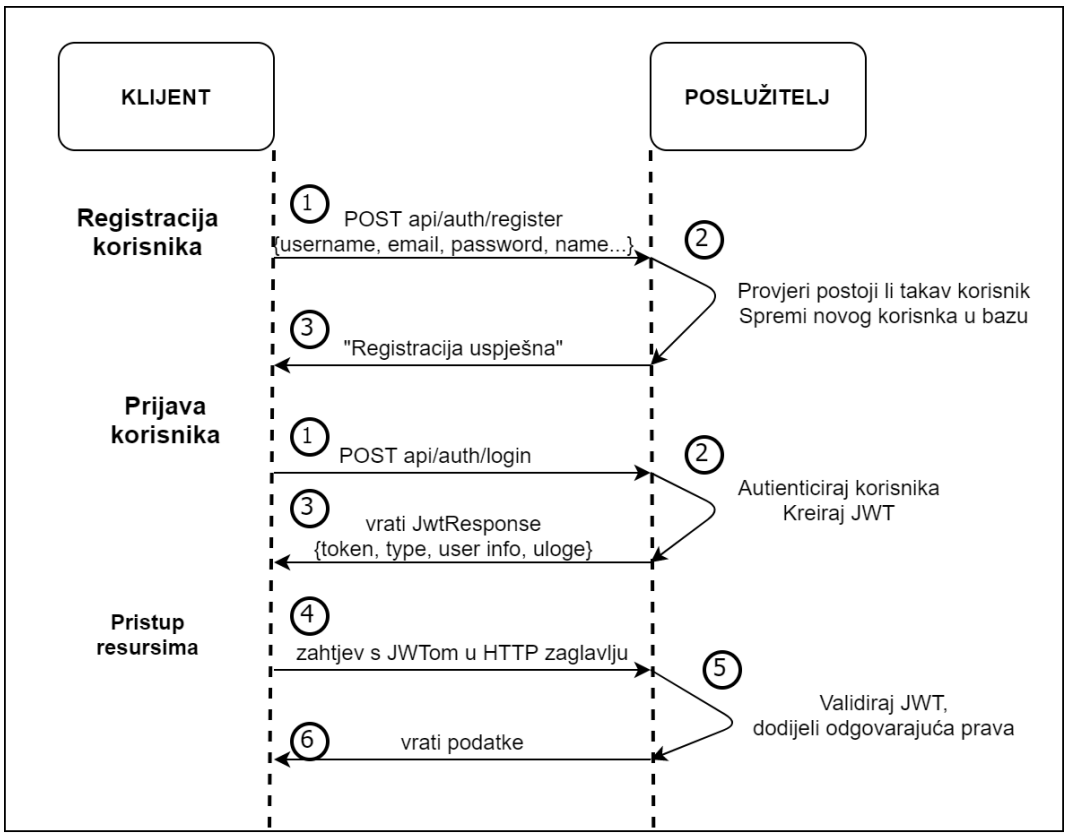
Sustav autentikaciju i autorizaciju implementira korištenjem Spring Boot Security radnog okvira. Spring Boot Security je moćan, fleksibilan i pouzdan radni okvir za implementaciju sigurnosne zaštite Java aplikacija. Komponente sustava koje sudjeluju u implementaciji prikazane su na Slici 3.2. WebSecurityConfig je najbit-



Slika 3.2 Arhitektura implementirnog Spring Boot Securitya u sustavu

niji razred koji se koristi za konfiguraciju globalnih postavki HttpSecuritya poput Cross-Origin Resource Sharing (CORS)a, politike stvaranje sjednica i sl. U njemu su također definirane rute kojima se može pristupiti bez autentikacije i upravljanje svim ostalim zaštićenim rutama. Svi dolazeći HTTP zahtjevi prvo prolaze kroz web filter OncePerRequestFilter koji provjerava autentikacijski sadržaj HTTP zahtjeva. OncePerRequestFilter parsira JWT iz zaglavlja HTTPa, validira ispravnost tokena, izvlači subjekt iz sadržaja JWTa i zatim provjerava postoji li takav subjekt tj. korisnik u sustavu uz pomoć UsernamePasswordAuthenticationToken razreda. Ukoliko

su svi uvjeti zadovoljeni zahtjev se preusmjerava na odgovarajući kontroler, a u suprotnom se baca iznimka o neovlaštenom pristupu. Ako je zahtjev poslan na neku od nezaštićenih ruta za prijavu ili registraciju tada se zahtjev preusmjerava na AuthController. Pri prijavi korisnika, u tijelu zahtjeva šalje se korisničko ime i lozinka koje AuthenticationManager koristi za provjeru. Ako postoji takav korisnik, generira se JWT token s njegovim ulogama, a u suprotnom baca se odgovarajuća iznimka (korisnik ne postoji, pogrešna lozinka i sl.). Pri registraciji korisnika tijelo zahtjeva sadrži razne podatke o korisniku važne za sustav. Prije stvaranja novog korisnika, provjera se postoji li već takav korisnik s istim korisničkim imenom, e-poštom ili mobilnim brojem. Po uspješnom završetku svih provjerava novi korisnik se sprema u bazu, lozinka se šifrira s PasswordEncoderom i dodjeljuje mu se početna korisnička uloga. Opisani proces pristupa zaštićenim i nezaštićenim rutama vizualiziran je na dijagramu toka na Slici 3.3



Slika 3.3 Dijagram toka pristupa zaštićenim i nezaštićenim rutama

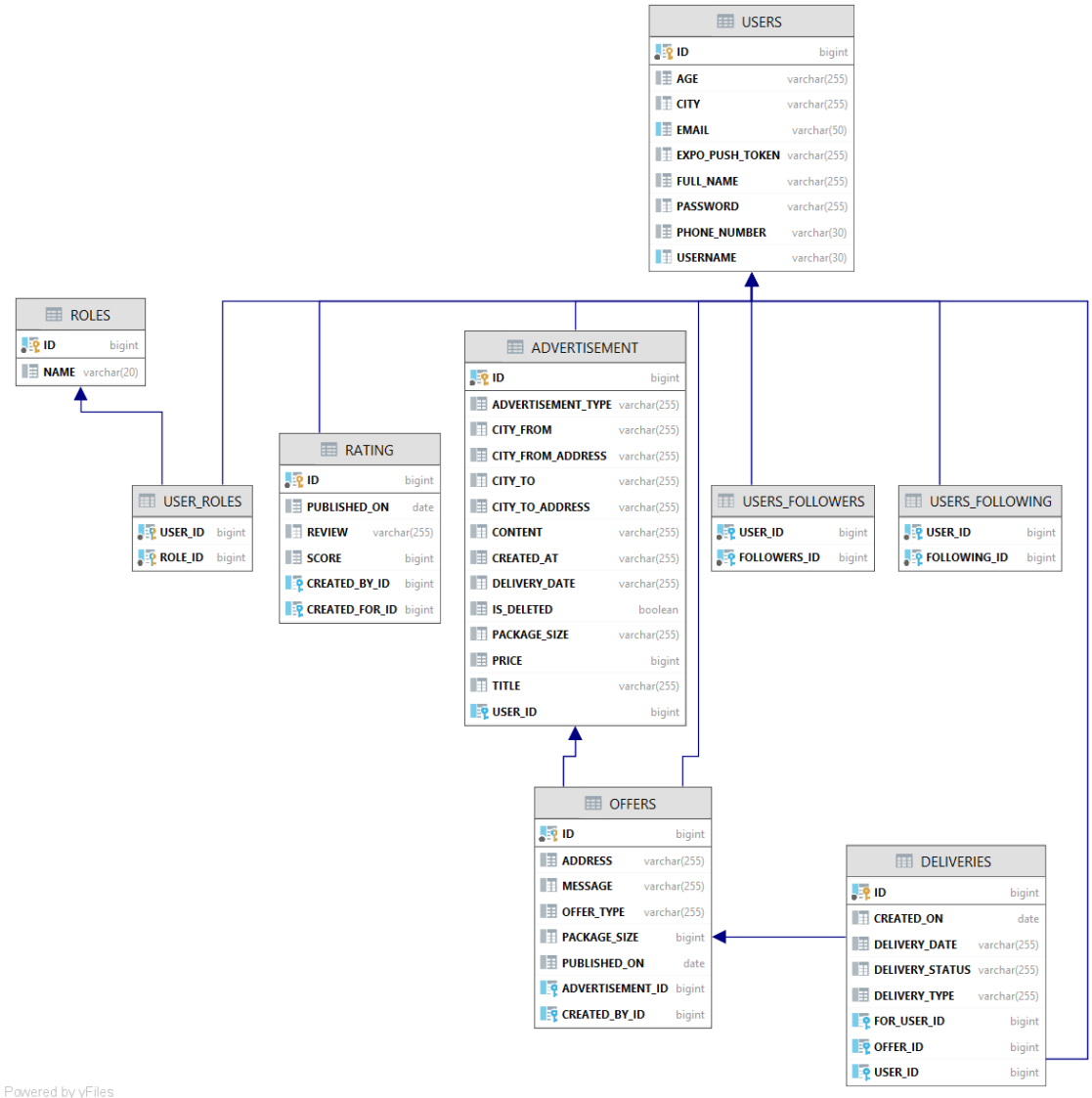
3.2 Poslužitelj

Implementirani poslužitelj zadužen je za upravljanje svim procesima i podacima sustava. Podaci se trajno spremaju u H2 bazu koja se sastoji od manjeg broja modela i jednostavnih relacija. Entity Relationship Diagram (ERD) prikazan je Slici 3.4. Ključna tablica USERS pohranjuje informacije o korisniku koji je povezan sa svim ostalim modelima u sustavu. Svaki korisnik može biti u ulozi ROLE_USER, ROLE_ADMIN ili oboje. Popis svih mogućih uloga pohranjen je u tablici ROLES, a u tablici USER_ROLES pohranjene su uloge za svakog pojedinačnog korisnika. Ocjenjivanje usluge korisnika sprema se u tablici RATING koja pohranjuje ID autora recenziju, ocjenu i datum objave. Korisnikova praćenja i pratitelje pohranjuje se u tablice USERS_FOLLOWING i USERS_FOLLOWERS. Svi oglasi koje korisnik

objavljuje pohranjuju se u tablicu ADVERTISEMENT koja sadrži razne podatke svakog oglasa. Ponude na specifične oglase spremaju se u tablicu OFFERS koja je povezana vezom Više-Na-Jedan s tablicom ADVERTISEMENT. Konačno, prihvaćanjem određene ponude stvara se unos u tablici DELIVERIES koja prati stanje i status dostave. Podaci iz baze se ne brišu, već se za operacije brisanja postavlja zastavica IS_DELETED. Ako se pobriše oglas, vrijednost zastavice IS_DELETED u tablici ADVERTISEMENT postavit će se na vrijednost *true* pa će oglas i sve ponude vezane za njega postat nevidljive sustavu.

3.2.1 Struktura

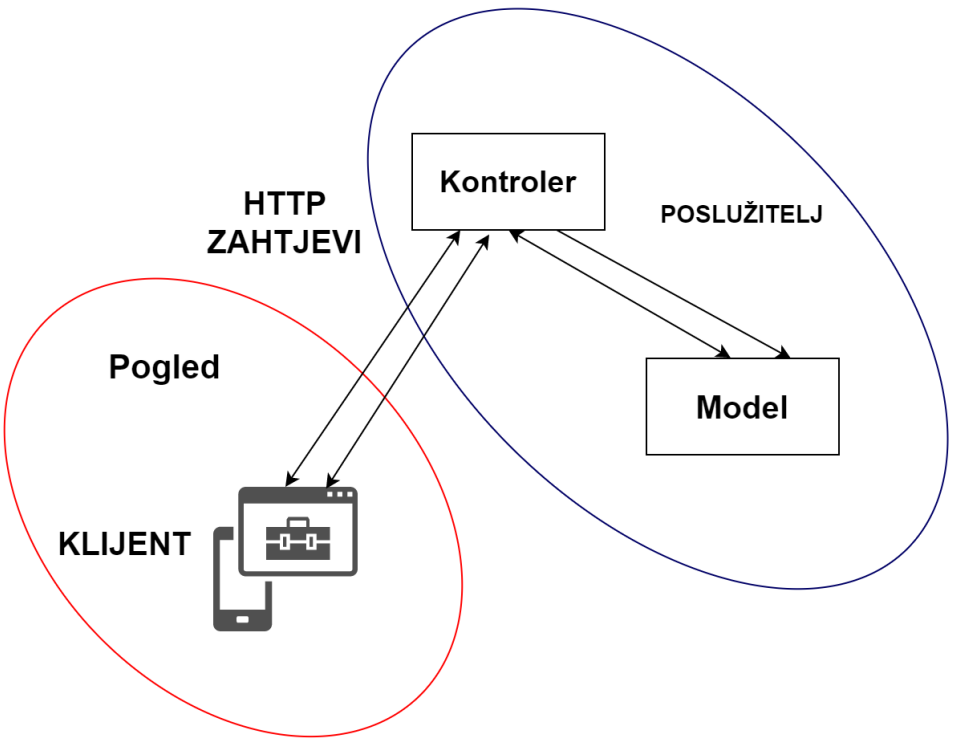
Poslužitelj je implementiran na temeljima Model-View-Controller (MVC) arhitekturalnog obrasca za moderni razvoj aplikacija. MVC raščlanjuje sustav u tri logičke komponente: model, pogled (engl. *view*) i kontroler gdje svaka od navedenih komponenti upravlja posebnim aspektom sustava [15]. Model sadrži podatke i poslovnu logiku sustava, kontroler upravlja korisničkim zahtjevima, a pogled prikazuje modelirane podatke. Osnovna ideja MVCa je izolirati poslovnu logiku i obradu podataka od korisničkog sučelja kako bi se olakšao razvoj, testiranje i održavanje sustava. Odnos komponenti u MVC arhitekturi implementiranog sustava prikazan je na Slici 3.5. Komponentu pogleda implementira klijentska aplikacija koja je zadužena za oblikovanje korisničkog sučelja s React Native komponentama. Kontroler komponentu implementiraju Spring Boot REST kontroleri koji ovisno o HTTP zahtjevu dohvaćaju resurse iz modela. Odnos kontrolera i modela na poslužitelju dodatno je raščlanjen na aplikacijski, uslužni i podatkovni sloj. Takvo raslojavanje implementirano je prema Data Access Object (DAO) obrascu koji omogućava izoliranje aplikacijskog i uslužnog (nekad zvanog i poslovnog) sloja od podatkovnog [16]. Osnovna ideja je odvojiti operacije stvaranja, čitanja, ažuriranja i brisanja podataka u bazi od poslovnih procesa. Aplikacijski sloj je ulazna točka u sustav i sastoji se od niza kontrolera koji upravljaju REST API pristupnim točkama.



Slika 3.4 ERD baze poslužitelja

3.2.2 REST API pristupne točke

REST API pristupne točke su niz URLova kojima klijentske aplikacije pristupaju resursima. Pristupne točke koje su dostupne u poslužiteljskoj aplikaciji dane su u Tablici 3.1. URLovi prefiksirani s /api/auth su nezaštićene pristupne točke o



Slika 3.5 MVC arhitektura sustava

kojima je bilo riječ u Potpoglavlju 3.1.2. Svi ostali, zaštićeni URLovi imaju bazni put /boonker/api/ nakon čega slijedi sufiks ovisno o kojem resursu se pristupa, npr. svi URLovi za pristup podacima o korisnicima imaju putanju /boonker/api/users/.

Na poslužiteljskoj strani kontroleri su logički podijeljeni prema poslovnim cjelinama koje obrađuju poput RatingControllera koji je zadužen za sve zahtjeve koji obrađuju ocjenjivanje korisnika. Svi takvi zahtjevi imaju URL koji započinje putanjom /boonker/api/rating. Kontroler validira ispravnost dolaznog zahtjeva i zatim poziva odgovarajuću metodu servisnog sloja. Na primjer, za stvaranje nove recenzije klijent šalje POST HTTP zahtjev s ispunjenim tijelom na /boonker/api/rating/

URL. Nakon sigurnosne provjere, zahtjev dolazi na RatingController koji provjerava ispravnost pristiglog RatingRequest objekta kojeg zatim dalje preusmjerava u servisni sloj. RatingService zadužen je za oblikovanje poslovne logike recenzije. Radi se mapiranje RatingRequest objekta na odgovorajući Rating objekt i dodaje se ocjena recenzije u ukupnu ocjenu korisnika. Konačno, novi Rating objekt prosljeđuje se podatkovnom sloju u RatingRepository koji sprema objekt u tablicu RATING.

Tablica 3.1 REST pristupne točke poslužitelja

Metoda	URL
POST	/api/auth/login
POST	/api/auth/register
GET	/boonker/api/advertisements
GET	/boonker/api/advertisements/my
GET	/boonker/api/advertisements/{id}
DELETE	/boonker/api/advertisements/{id}
PUT	/boonker/api/advertisements/{id}
GET	/boonker/api/deliveries/activate/{id}
GET	/boonker/api/deliveries/driving
GET	/boonker/api/deliveries/finish/{id}
GET	/boonker/api/deliveries/incoming
GET	/boonker/api/deliveries/{id}
POST	/boonker/api/offers/
GET	/boonker/api/offers/{advertisementId}
GET	/boonker/api/offers/{id}/accept
GET	/boonker/api/offers/{id}/decline
GET	/boonker/api/ratings/user/id
GET	/boonker/api/users/{id}
POST	/boonker/api/users/expo
GET	/boonker/api/users/{id}
GET	/boonker/api/users/{id}/advertisement
GET	/boonker/api/users/{id}/follow
GET	/boonker/api/users/{id}/followers
GET	/boonker/api/users/{id}/following
GET	/boonker/api/users/{id}/unfollow

Klijentska aplikacija za slanje HTTP zahtjeva koristi JavaScript Axios knjižnicu. Riječ je o HTTP klijentu zasnovanom na obećanjima (engl. *promise-based*) koji

osim slanja HTTP zahtjeva omogućava transformiranja HTTP odgovora, tj. transformiranje JSON objekata u oblik razumljiv klijentu, presretanje HTTP zahtjeva i upravljanje Promise APIem. Slanje HTTP zahtjeva je asinkrona operacija; kako bi klijent nesmetano nastavio s izvršavanjem operacija Axios vraća Promise. Promise je objekt koji reprezentira eventualni uspješni izvršetak asinkrone operacije i može biti u jednom od tri stanja: čekanje, ispunjeno ili odbačeno (*pending, fulfilled, rejected*).

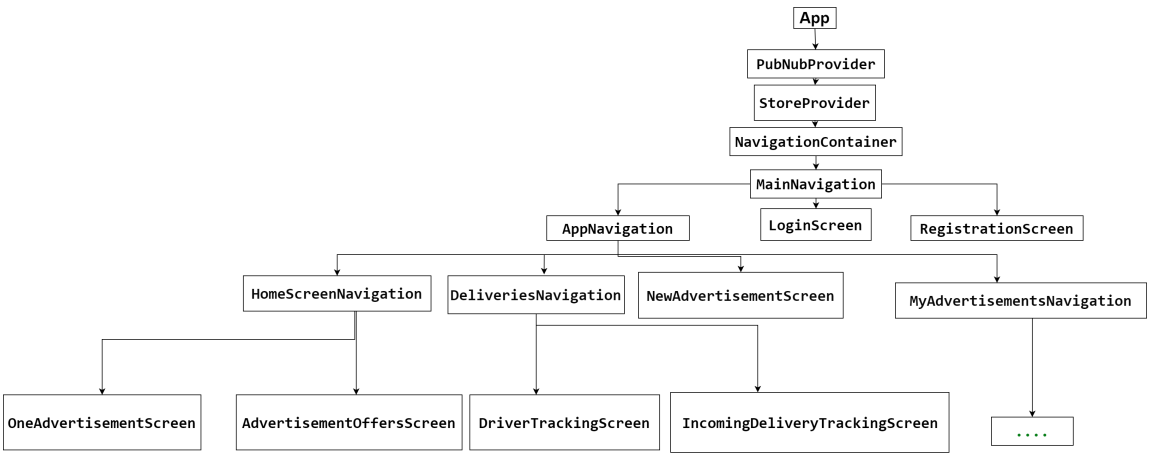
3.3 Klijent

Klijentska aplikacija realizira mobilno korisničko sučelje za Android i iOS platforme implementirano u React Nativeu. React Native je JavaScript radni okvir otvorenog koda koji omogućava brzi razvoj nativnih aplikacija [17]. Radi se o popularnom radnom okviru koji omogućava razvojnim programerima korištenje ReactJS web principa u mobilnom okruženju. Zadaća implementirane aplikacije je renderiranje korisničkog sučelja, upravljanje korisničkim interakcijama i komunikacija s poslužiteljskom aplikacijom. Klijent ne sudjeluje u obradi i pohrani podataka, već ih samo, prethodno obrađene od strane poslužitelja, prikazuje korisniku.

3.3.1 Struktura

ReactJS i React Native, za razliku od većine današnjih popularnih radnih okvira, ne nameću specifičnu arhitekturu. Autori radnog okvira navode da je ReactJS samo pogled koji služi kao spremnik za korisničko sučelje [17]. Korisničko sučelje je sastavljeno od niza komponenti koje zajedno čine React stablo. Komponenta je strukturalna jedinica kojom se opisuje jedna vizualna cjelina koja može biti minimalna poput tekstualne labele ili gumba ili složenija poput registracijskog obrasca. Osnovna ideja je implementirati nezavisne komponente, labavo povezane s ostatkom stabla, koje se mogu višestruko iskoristiti na raznim zaslonima. Pojednostavljeni prikaz stabla komponenti implementiranog u klijentskoj aplikaciji dan je na Slici 3.6. App komponenta stable je glavna, korijenska komponenta iz koje se račvaju sve ostale komponente sučelja. PubNubProvider i StoreProvider su specifične komponente čija je zadaća opskrbiti svu djecu u stablu s posebnim funkcijama i podacima pa tako

StoreProvider osigurava pristup zajedničkom lokalnom skladištu aplikacije, a PubNubProvider pruža podatke o spajanju na komunikacijske kanale za prijenos poruka u stvarnom vremenu. Navigacijske komponente poput MainNavigation, AppNavigation ili HomeScreenNavigation implementiraju gotove komponente React Navigation knjižnice za koordinaciju određenim dijelom korisničkog sučelja. Sve komponente koje se renderiraju na korisničkom sučelju sufiksirane su riječju Screen poput UserProfileScreen i UserRatingScreen i one su kompozicija više manjih komponenti poput kartica, gumbova, labela, lista i sl.



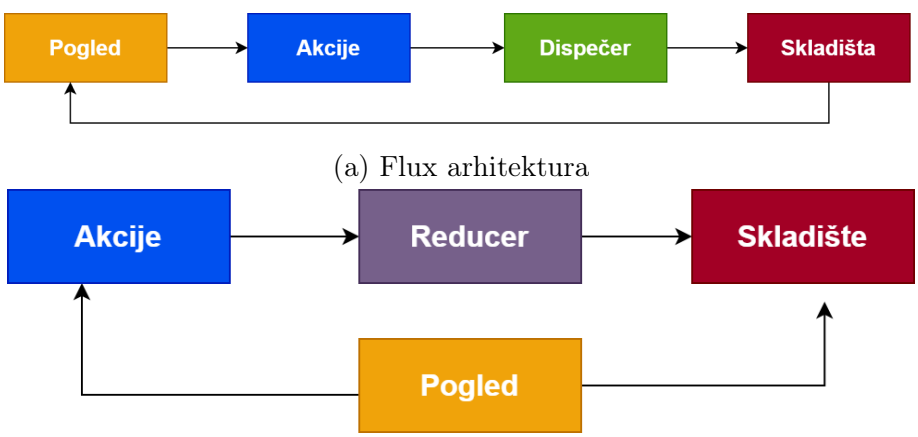
Slika 3.6 Stablo komponenti klijentske aplikacije

Svaka komponenta u React/React Nativeu može čuvati svoje stanje. Stanje komponente su podaci potrebni za rad aplikacije, npr. komponenta korisničkog profila čuva podatke o korisniku koje prikazuje na sučelju. Osim čuvanja podataka, komponenta prati i promjene nad tim podacima i u slučaju određenih akcija korisnika renderira promjene na sučelju. React dolazi s gotovim mehanizmom za čuvanje podataka tzv. lokalnim skladištem. Administriranje lokalnim skladištem u velikim aplikacijama zahtjevno je i složeno u čemu pomaže implementacija React arhitekturalnih obrazaca poput Fluxa, Reduxa i Context API-a.

Facebookov razvojni tim razvio je Flux kao alternativu MVC arhitekturi koja se pokazala nestabilnom, suviše složenom i teškom za održavanje za klijentske aplikacije budući da su promjene na korisničkom sučelju puno češće u odnosu na poslužitelj-

ske sustave [18]. Flux arhitektura dozvoljava tok podataka u samo jednom smjeru, od roditelja prema djeci i ažurira sučelje samo na promjenu stanja tj. podataka u stanju komponenti. Osnovne komponente Fluxa su pogled (engl. *view*), akcije, dispečer (engl. *dispatcher*) i skladište (engl. *store*). Korisnik interakcijom sa sučeljem npr. klikom na obrazac okida određenu akciju. Akcija prikupi podatke i proslijedi ih u dispečer funkciju koja na temelju tipa akcija određuje u koje skladište je potrebno poslati podatke. Skladište je u osnovi objekt koji čuva podatke u obliku ključ-vrijednost. Ažuriranje skladišta uzrokuje i ažuriranje korisničkog sučelja. Flux arhitektura podržava jedno ili više skladišta, može čak imati i skladište za svaku komponentu. Flux arhitektura pojednostavila je upravljanje stanjem komponenti u React aplikacijama, ali se s vremenom pokazalo da dolazi s puno koda kojeg je teško održavati (engl. *boilerplate code*).

2015. godine Dan Abramov i Andrew Clark razvili su Redux knjižnicu koja je zasnovana na Flux ideji s novim komponentama koje pojednostavljuju cijeli proces [18]. U Reduxu postoji jedno globalno skladište svih podataka cijele aplikacije, funkcije koje upravljaju promjenama (engl. *reducer functions*) nad podacima te akcije i pogled. Sve komponente imaju pristup centralnom skladištu i promjene nad tim podacima propagiraju se cijelim React stablom. Opisane Flux i Redux arhitektura prikazane su na Slici 3.7.



(a) Flux arhitektura

(b) Redux arhitektura

Slika 3.7 Usporedba Flux i Redux arhitekture

Klijentska aplikacija za upravljanjem stanjem komponenti koristi Redux Toolkit. Redux Toolkit je dodatno pojednostavljeni paket koji pruža APIe za konfiguraciju Redux komponenti i omogućava da se globalno skladište podijeli na logičke dijelove (engl. *slices*). Moguće je imati dio skladišta koji čuva samo podatke o korisniku, zatim dio skladišta zadužen za podatke o oglasima, dostavama itd. Redux Toolkit dolazi s raznim funkcijama od kojih su neke *createStore()* za stvaranje skladišta i *createReducer()* za konfiguraciju reducer funkcija. Svaki zaslon klijentske aplikacije ima pripadajući dio u globalnom Redux skladištu koji se stvara konfiguracijom *slicea*. Primjerice, za dohvat svih aktualnih oglasa s poslužiteljske aplikacije prikazanih na Slici 3.8 koristi se *AllAdvertisements.slice.js* datoteka koja sadrži asinkronu akciju, *slice* i inicijalno stanje *slicea* u globalnom skladištu. Asinkrona Redux funkcija za stvaranje akcije za dohvat svih oglasa dana je Ispisom 3.4.

```
export const getAllAdvertisements = createAsyncThunk(
  "getAllAdvertisements",
   async (searchAdvertisementQueryRequest) => await
      getAdvertisements(searchAdvertisementQueryRequest);
);
```

Ispis 3.4 Redux funkcija za stvaranje akcije

Redux funkcija *createAsyncThunk(..)* kao argumente prima naziv akcije u obliku niza znakova i *callback* funkciju koja vraća *Promise*. U ovom slučaju akcija se zove *getAllAdvertisement* i funkcija *createAsyncThunk(...)* će prema tom nazivu generirati tri stanja akcije; *pending, fulfilled* i *rejected*. Zatim se poziva *getAllAdvertisements callback* funkcija koja za argumente uzima parametre pretrage svih oglasa spremljenih u objektu *searchAdvertisementQueryRequest*. Parametri se unose u traci za pretraživanje, a po prvom učitavanju su prazni i funkcija vraća listu svih oglasa. Funkcija *getAdvertisements(...)* za dohvat svih podataka s poslužitelja koristi Axios knjižnicu i prikazana je u nastavku.

```
export async function getAdvertisements(
   searchAdvertisementQueryRequest) {
  const authorization = await authHeader();
  const response = await api.post("/advertisements/all",
     searchAdvertisementQueryRequest, {headers: authorization});
  return response.data;
}
```

Ispis 3.5 Axios funkcija za dohvat podataka s poslužitelja

U Ispisu 3.5 je vidiljivo da je za uspješan poziv prema poslužitelju potrebno dodati autorizacijski *Bearer* u zaglavlje HTTP zahtjeva koji se čita iz lokalnog skladišta klijenta. Zatim se kreira POST HTTP zahtjev prema odgovarajućem URLu koji u tijelu zahtjeva može sadržavati parametre pretrage oglase. Parametri pretrage se spremaju u searchAdvertisementQueryRequest objektu. Poslužitelj vraća listu oglasa u JSON odgovoru.

Definirana akcija, prethodno opisana u Ispisu 3.4 *getAllAdvertisement(...)* se poziva na zaslonu koji prikazuje sve oglase pomoću posebne React funkcije *useEffect(...)* unutar koje se koristi *dispatch()* funkcija. Konačno, funkcija za stvaranje *slicea getAllAdvertisementsSlice(...)* prima četiri osnovna argumenta prikazana u Ispisu 3.6.

```
export const getAllAdvertisementsSlice = createSlice({
  name: "getAllAdvertisementsSlice",
  initialState: initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder.addCase(getAllAdvertisements.fulfilled, (state,
        action) => {
      state.advertisements = action.payload;
      state.error = false;
    });
    builder.addCase(getAllAdvertisements.rejected, (state, action
        ) => {
      state.error = true;
    });
  },
});
```

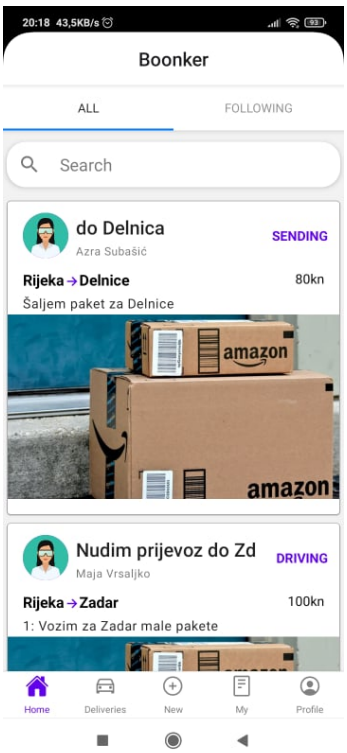
Ispis 3.6 Funkcija za stvaranje Redux *slicea*

Osim imena, funkcija prima početno stanje *slicea* koje će se pohraniti u skladište pri inicijalizaciji aplikacije. Inicijalno stanje je objekt koji se sastoji od polja *advertisement* u koje se pohranjuje lista oglasa i boolean polje *error* u koje se zapisuje ako je došlo do neke pogreške. Argument *reducers* prima funkcije koje upravljaju sinkronim akcijama, dok je *extraReducers* argument predviđen za asinkrone pozive. Tako se u ovom slučaju kontroliraju dva statusa akcije, ispunjeni odnosno *fulfilled* u čijem se slučaju skladište popunjava s listom oglasa. U slučaju da je akcija odbijena odnosno u stanju *rejected* skladište se popunjava samo s *error* poljem s vrijednošću *true*.

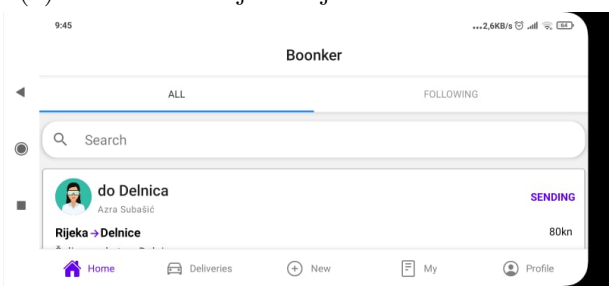
3.3.2 Implementacija korisničkog sučelja

Korisničko sučelje većinski je implementirano korištenjem ugrađenih komponenti React Native knjižnice. React Native pruža velik broj komponenti koje su responzivne i podržane na Android i iOS platformama [17]. Radni okvir razlikuje nekoliko kategorija komponenti: osnovne, kontrolne, liste te specifične za Android ili iOS. Osnovne komponente obuhvaćaju komponente koje će se zasigurno koristiti u većini mobilnih aplikacija poput Viewa, Texta, TextInputa, Imagea i ostalih. View je temeljna komponenta koja služi kao spremnik za raspoređivanje ostalih komponenti (engl. *layout*). Kontrolne komponente upravljaju interakcijom korisnika sa sučeljem i u toj kategoriji je najvažnija Button komponenta. U kategoriji lista nalaze se FlatList i SectionListe koje služe za prikaz velikog broj kategoriziranih podataka. Prednost korištenja ugrađenih komponenti je ubrzan razvoj responzivnih zaslona i rad s pouzdanim funkcijama koje su već dovoljno optimizirane za velike količine podataka. Osim ugrađenih komponenti, klijent koristi React Native Paper i Expo knjižnice komponenti. React Native Paper je kolekcija funkcionalnih i responzivnih komponenti koje se temelje na Googleovom Material dizajnu. U kolekciji se nalaze većina komponenti koje se koriste u modernom razvoju mobilnih aplikacija poput kartica, trake za pretraživanje, potvrdnog okvira, tablica, ikona, izbornika, dijaloga i ostalih. Expova knjižnica komponenti nudi kompleksnije komponente za rad sa specifičnim funkcionalnostima mobilnih uređaja. Klijentska aplikacija koristi MapView komponentu za prikaz Google Mapsa na sučelju i DateTimePicker komponentu za odabir datuma. Osim spomenutih, Expo sadrži komponente za rad sa senzorima, mobilnom kamerom, dokumentima, SMS-om, mrežom, plaćanjem itd.

Na Slici 3.8 prikazan je vertikalna i horizontalna orijentacija naslovnog zaslona mobilne aplikacije koju klijent vidi nakon uspješne autentikacije. Na naslovnom zaslonu korisnik vidi sve aktualne oglase za slanje ili prijevoz paketa. Na dnu ekrana nalazi se navigacijska traka s kojom korisnik navigira kroz sve značajne funkcionalnosti aplikacije. Prva ikona s labelom *Home* obojena je ljubičastom što označava da je to trenutno aktivni zaslon. Iduća ikona automobila s labelom *Deliveries* vodi na zaslon koji prikazuje sve korisnikove vožnje i dostave. Središnja ikona plusa na traci vodi na zaslon za stvaranje novog oglasa, koju slijedi ikona oglasa s labelom *My*, a koja prikazuje sve objavljene oglase korisnika. Posljednja ikona s labelom *Profile*



(a) Vertikalna orijentacija zaslona



(b) Horizontalna orijentacija zaslona

Slika 3.8 Zaslon sa svim oglasima klijentske aplikacije u vertikalnoj i horizontalnoj orijentaciji

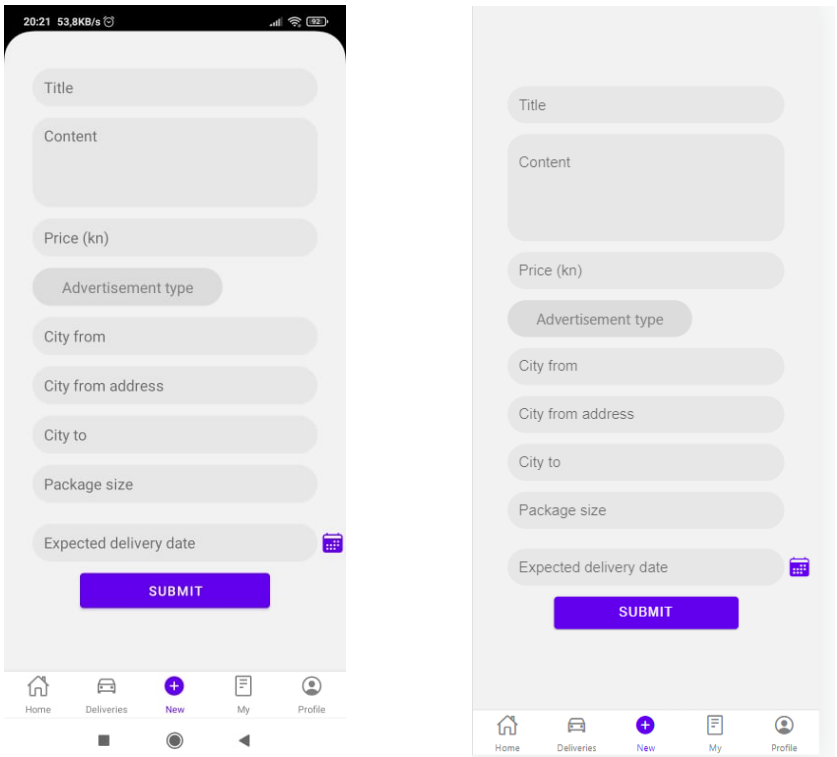
vodi na osobni profil korisnika. Ikone dolaze iz paketa Ionicons radnog okvira. Na vrhu zaslona nalaze se dvije kartice s tekstom *All* i *Following* koje naslovni zaslon raspodjeljuju na dva. Korisnik klizanjem prstom u lijevo ili desno mijenja aktivnu karticu, odnosno mijenja listu oglasa koje vidi; sve aktualne oglase ili samo listu oglasa korisnika koje prati. Središnji dio ekrana prikazuje listu kartica pri čemu je

svaka kartica jedan oglas.
Za prikaz liste oglasa iskoristila se FlatList komponenta koja za prvi argument prima polje oglasa u JSON formatu pristiglih s poslužitelja. Drugi argument je komponenta Item koja definira vizualni prikaz jednog oglasa i sastoji se od kompozicije Card komponente i osnovne View komponente. FlatList komponenti se također može definirati i ponašanje prilikom osvježavanja ekrana; kada korisnik prstom povuče zaslon prema dolje dogodit će se novi poziv prema poslužitelju koji će dohvatiti sve najnovije oglase.
Lista oglasa ponavlja se na raznim zaslonima u aplikaciji i na isti način prikazuje oglase, samo se razlikuje skup podataka koji se prikazuje, npr. na zaslonu MyAdvertisementScreen se prikazuju samo oglasi koje je korisnik stvorio. Logika prikaza oglasa enkapsulirana je u posebnu komponentu specifičnu za klijenta (AllAdvertisement) koja kao argument prima polje oglasa i *callback* funkcije koje upravljaju osvježavanjem ekrana. AllAdvertisement komponenta, prikazana u Ispisu 3.7, zatim implementira prethodno opisanu FlatList komponentu. Ovakvom ekstrakcijom zajedničkih elemenata zaslona pojednostavljuje se razvoj novih zaslona i održavanje koda.

```
<AllAdvertisements
  advertisements={advertisements}
  onRefresh={() => onRefreshData()}
  refresh={refresh}
/>
```

Ispis 3.7 Komponenta za prikaz liste oglasa

Kao što je već spomenuto, većina komponenti korištenih u razvoju mobilne aplikacije dolazi iz gotovih knjižnica i kolekcija komponenti. Takve gotove komponente su responzivne i samim tim zasloni koji ih koriste su responzivni i dostupni na mobilnim uređajima raznih veličina. Na Slici 3.9 prikazan je isti zaslona stvaranja novog oglasa na dva različita uređaja iPHONE X i Xiaomi Redmi Note 9. Oba zaslona sadrže iste funkcionalnosti s istim komponentama.



(a) Xiaomi Redmi 9 (b) Iphone X

Slika 3.9 Zaslon za stvaranje novog oglasa na različitim mobilnim uređajima

3.4 Implementacija značajnijih funkcionalnosti

Sustav implementira prijavu korisnika u sustav korištenjem JWT tokena opisanog u prethodnom Potpoglavlju 3.1.2. Po uspješnoj prijavi korisnik može upravljati oglasima, ponudama, dostavama ili vlastitim profilom. Nad tim entitetima implementirane su Create, Read, Update, Delete (CRUD) operacije na poslužitelju zahvaljujući kojima korisnik može pregledavati, pretraživati, ažurirati ili brisati određene entitete sustava. Na poslužiteljskoj strani implementirana je logika umrežavanja i interakcije korisnika. Na klijentskoj strani implementirano je praćenje lokacije paketa u stvarnom vremenu; poslužiteljska aplikacija obavještava se samo o početku i završetku dostave, dok klijentska aplikacija tijekom dostave komunicira s PubNub APIem. Korisnika se putem mobilnih obavijesti informira o aktivnostima ostalih korisnika u sustavu. Implementacija klijentske i poslužiteljske strane važnijih funkcionalnosti

sustava opisana je u nastavku.

3.4.1 Obavijesti

Klijentska aplikacija obavještava korisnika o važnijim događajima poput početka dostave, novog pratitelja ili nove ponude na oglas putem mobilnih obavijesti. Mobilne obavijesti implementirane su korištenjem Expo Notifications APIa. Knjižnica pruža podršku za slanje i primanje instantnih ili zakazanih obavijesti. Prednost korištenja ovog APIa je u jednom rješenju za Android i iOS mobilne platforme.

Pri prvoj uspješnoj prijavi u aplikaciju, sustav će zatražiti dozvole za slanje i primanje obavijesti i nakon toga generirati Expo Push token pozivom *getExpoPushTokenAsync()* asinkrone funkcije. Expo Push token je jedinstveni token generiran za svaki mobilni uređaj prema kojem se identificira primatelj obavijesti. Ako korisnik promijeni mobilni uređaj ili reinstalira aplikaciju promijenit će se i Expo Push Token, stoga se, za svaki slučaj, traži generiranje tokena prilikom svake prijave u sustav. Ako nije došlo do nikakve promjene Expo će vratiti isti token. Token se čuva na poslužiteljskoj strani u USERS tablici. Prilikom slanja instantne obavijesti drugom korisniku prvo se radi dohvat njegovog tokena iz baze i zatim se šalje zahtjev Expo Notifications APIu s njegovim tokenom i sadržajem obavijesti. Na primjer, kada korisnik zaprati nekog drugog korisnika obavijest se šalje na sljedeći način. Prvo se poziva poslužitelj s funkcijom *getExpoPushToken(id)* prikazanoj u Ispis 3.8 koja će s URLa boonker/users/expo/id dohvatiti Expo Push Token korisnika s danim ID-em, u ovom slučaju ID korisnika kojeg se upravo zapratilo. Poslužitelj će zatim vratiti traženi token koji će se proslijediti u *sendPushNotification(...)* funkciju koja osim tokena kao argument prima naslov obavijesti "*New follower*" i tijelo obavijesti "*<currentUser> just followed you!*". Ime trenutnog korisnika čuva se u objektu currentUser koji je pohranjen u lokalnom skladištu. U *sendPushNotification(...)* funkciji šalje se HTTP POST zahtjev Expo APIu. Osim navedenih argumenta kroz API se mogu poslati dodatne informacije kroz *data* polje u JSON formatu, definirati zvuk obavijesti ili prioritet u odnosu na ostale obavijesti aplikacije. Expo će prema Expo Push Tokenu identificirati primatelja obavijesti i ako se radi o Android korisniku proslijedit će obavijest Firebase Cloud Messaging servisu,

a ako se radi o iOS sustavu na Apple Push Notification service (APNS). Expovo enkapsuliranje slanja obavijesti na različite platforme olakšava i ubrzava razvoj.

```
...
const token = await getExpoPushToken(id);
await sendPushNotification(token.expoPushToken,"New follower!",
   currentUser.fullName + " just followed you!");
...

export async function sendPushNotification(expoPushToken, title,
   body, data) {
  const message = {
  to: expoPushToken,
  sound: "default",
  title: title,
  body: body,
  data: data
  };
  await fetch("https://exp.host/--/api/v2/push/send", {
    method: "POST",
    body: JSON.stringify(message),
  });
}
```

Ispis 3.8 Funkcija za slanje mobilnih obavijesti Expo Notifications APIu

3.4.2 Umrežavanje korisnika

Mogućnost povezivanja korisnika osnova je sustava zasnovanih na konceptima društvenih mreža. Povezivanje se ostvaraje mehanizmima praćenja, svaki korisnik može zapratiti neograničen broj korisnika i imati neograničeni broj pratitelja. Međusobnim praćenjem korisnici ostvaruju veze za lakše buduće interakcije. U klijentskoj aplikaciji korisnik može pregledati oglase zapraćenih korisnika. Korisnik može zapratiti drugog korisnika odlaskom na njegov profil i pritiskom na gumb s labelom *Fol-*

low. Pritiskom na taj gumb šalje se GET HTTP zahtjev prema poslužitelju na URL /boonker/users/{id}/follow gdje je {id}, ID korisnika kojeg se želi zapratiti. Na poslužitelju u modelu User definirane su dvije liste List<User> following i List<User> followers. Liste sadrže korisnike koje korisnik prati ili korisnike koji su zapratili korisnika. Liste u modelu su označene anotacijom @ManyToOne i zahvaljujući toj anotaciji Hibernate, Javin radni okvir koji pojednostavljuje interakciju s bazom, generira dvije dodatne tablice USER_FOLLOWING i USER_FOLLOWERS u koje se spremaju podaci o praćenjima korisnika. U Java kodu, za dodavanje novog korisnika u listu praćenih ljudi dovoljno je pretražiti bazu prema ID-u tog korisnika i u pronađeni objekt tipa User dodati u List<User> following. Dodatno, zapraćenom korisniku potrebno je dodati tog korisnika u listu njegovih pratitelja tj. u List<User> followers. Za kraj, potrebno je pozvati *save()* metodu UserRepositorya nad trenutnim korisnikom i korisnikom kojeg se zapratilo kako bi se ažuriralo stanje u bazi. Jednom zapraćeni korisnik se može prestati pratiti pritiskom na gumb s labelom *Unfollow* kojim se šalje GET HTTP zahtjev na URL /boonker/api/users/id/unfollow. Zapisi o praćenju uklanjaju se iz prethodno spomenutih tablica. Popis svih korisnika koje korisnik prati ili popis njegovih pratitelja mogu se vidjeti na korisničkom profilu pritiskom na labele *Following* ili *Followers* koje prikazuju listu svih takvih korisnika. Na poslužitelj se tad šalje GET HTTP zahtjevi /boonker/users/id/following ili /boonker/users/id/followers za pretragu baze za željenom listom korisnika.

3.4.3 Praćenje lokacije paketa u stvarnom vremenu

Najzanimljivija funkcionalnost klijentske aplikacije je praćenje dostave u stvarnom vremenu. Praćenje lokacije mobilnog uređaja korisnika prilikom dostave paketa ostvaren je kombinacijom Expo Location APIa i PubNub React knjižnice za razmjenu poruka putem APIa za objavljivanje i primanje poruka. Expova knjižnica omogućava čitanje geolokacijskih informacija, geografske širine i dužine uređaja (engl. **latitude and longitude**). Kao i ostale Expove knjižnice nudi jedno rješenje za Android i iOS platforme. PubNub komunikacijska platforma nudi gotove funkcije za razmjenu poruka. Klijentska aplikacija dostavljača stvara kanal koji nosi ime prema identifikatoru aktualne dostave. Na novostvoreni kanal, na svaku promjenu lokacije, objavljuje se

poruka u JSON formatu s trenutnim geolokacijskim informacijama. Klijentska aplikacija primatelja paketa pretplaćena je na taj isti kanal i osluškuje pristigle poruke te na svaku novu promjenu ažurira korisničko sučelje s novom lokacijom dostavljača. Lokacija dostavljača prikazana je na MapView komponenti koja koristi Google Maps.

Cijeli proces započinje kada dostavljač započne vožnju pritiskom na gumb s labelom *START DRIVING* kojom se na poslužitelj šalje GET HTTP zahtjev na URL /boonker/api/deliveries/activate/{id} kojim se status modela Delivery u bazi mijenja u ACTIVE. Na započetu vožnju primatelju se šalje mobilna obavijest koja ga informira o započetoj vožnji koju sad može pratiti u stvarnom vremenu. React *useEffect(...)* funkcija koja dohvaća trenutnu lokaciju dostavljača opisana je u Ispisu 3.9. Prije samog dohvata geoinformacijskih podataka potrebno je zatražiti dopuštenje za korištenje lokacije uređaja pomoću funkcije *requestForegroundPermissionsAsync()*. Jednom kad je čitanje lokacije dozvoljeno, metoda *watchPositionAsync()* pretplaćuje se na promjene lokacije uređaja i na svaku promjenu ažurira location varijablu stanja komponente. Također, na promjenu lokacije objavljuje se poruka na kanal pomoću *pubnub.publish()* funkcije.

Na primateljskoj strani implementirana je *listener* funkcija koja na istom kanalu zaprima nove poruke prema kojima ažurira svoju *location* varijablu stanja. Kada dostavljač dođe na odredište pritiskom na gumb s labelom *FINISH DELIVERY* završava dostavu i prijenos poruka. Na poslužitelj na URL /boonker/api/delivery/{id}/finish šalje se GET zahtjev kojim se status modela Delivery mijenja u FINISHED. Na komunikacijski kanal odašilje se posljednja poruka sadržaja *isFinished* koja *listener* funkciju obavještava da je dostava završena.

```
React.useEffect(() => {
  (async () => {
    ...
    await Location.watchPositionAsync({}, (location) => {
      setLocation({
        latitude: location.coords.latitude,
        longitude: location.coords.longitude,
      });
      if (isFinishedDriving === false) {
        pubnub.publish({
          channel: channelName,
          message: {
            latitude: location.coords.latitude,
            longitude: location.coords.longitude,
          },
        });
      }
    });
  })();
}, []);
```

Ispis 3.9 Funkcija za slanje geolokacijskih informacija u stvarnom vremenu

3.4.4 Ocjenjivanje korisnika

Po završetku dostave korisniku, dostavljaču i primatelju, se otvara dijalog putem kojeg ocjenjuje ostvarenu interakciju s drugim korisnikom. Uslugu ocjenjuje brojčano od jedan do pet i neobaveznim komentarom. Pritiskom na gumb s labelom *Submit* šalje se POST HTTP zahtjev na /boonker/api/ratings/user/{userId} koji u tijelu zahtjeva sadrži recenziju. Na poslužitelju se recenzija sprema u tablicu RATING koja je vezom Više-Na-Jedan povezana s tablicom USER. Ukupna ocjena korisnika računa se kao aritmetička sredina svih recenzija upućenih njemu. Ocjena se prikazuje na korisničkom profilu gdje je moguće vidjeti i sve komentare na njegovu uslugu.

3.5 Problemi pri razvoju sustava

Problemi pri implementaciji sustava tipični su, nepredvidivi i mogu biti različiti, a neki od njih su problemi s razvojnom okolinom, programskim jezikom, mobilnom platformom uređaja ili korištenim knjižnicama. Na poslužiteljskoj strani nije bilo većih problema budući da se koristio standardizirani i općeprihvaćeni način implementacije REST sučelja u Spring Boot aplikacijama. Na klijentskoj aplikaciji bilo je raznih problema, ponajviše s različitim JavaScript knjižnicima. React Native radni okvir nudi samo osnovne funkcionalnosti pa je za sve ostale, naprednije funkcionalnosti potrebno pronaći rješenje u vanjskim knjižnicima. Verzije takvih knjižnica često ne prate i verzije React Nativea pa je potrebno pronaći niz verzija koje se međusobno ne pobijaju i ne uzrokuju konflikte. Osim toga, testiranje nativnih aplikacije je zahtjevno budući da postoji veliki broj mobilnih uređaja različitih veličina zaslona i različitih verzija Android ili iOS platformi. Android Emulator pomaže pri testiranju različitih mobilnih uređaja, ali samo u jednoj mjeri. Zbog ograničenja emulatora mnoge funkcionalnosti se nisu mogle testirati, poput primanja i slanja obavijesti ili praćenje lokacije paketa. Emulator se zbog toga koristio za testiranje vizualnih komponenti uređaja, a za testiranje funkcionalnosti koristili su se stvarni fizički uređaji.

Poglavlje 4

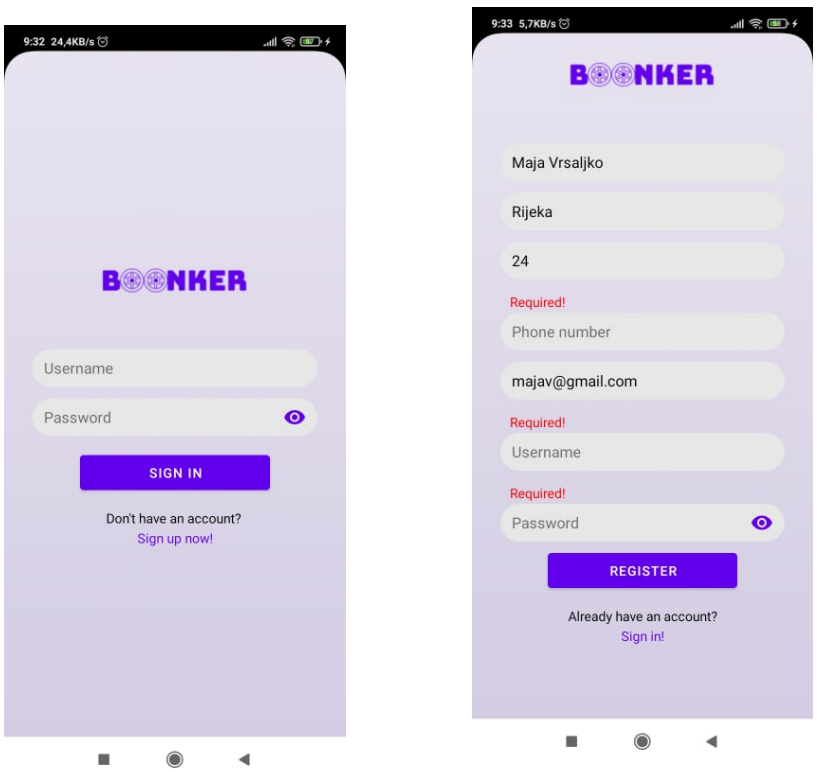
Karakteristični slučajevi korištenja

Funkcionalnosti sustava se iz korisničke perspektive mogu podijeliti na pet većih cjelina o kojima je već bilo govora iz tehničke perspektive. Prvi susret korisnika s aplikacijom je registracija u sustav, nakon čega slijedi prijava. Upravljanje vlastitim oglasima uključuje pregled, objavu, uređivanje i brisanje oglasima i upravljanje ponudama na specifični oglas. Korisnik može pretraživati postojeće oglase i slati ponude na iste. U sklopu pretrage tuđeg sadržaja može zapratiti autore oglasa. Dogovorene dostave može pratiti u stvarnom vremenu i ocjenjivati usluge ostalih korisnika. Naposljetku, korisnik može sve osobne informacije definirane prilikom registracije promijeniti uređivanjem vlastitih podataka na svom korisničkom profilu.

4.1 Prijava i registracija u sustav

Prilikom prvog pokretanja klijentske aplikacije korisnik vidi zaslon za prijavu u sustav prikazan na Slici 4.1a. Ako je korisnik već registriran, u sustav se prijavljuje korisničkim imenom i lozinkom. Za uspješnu prijavu mora unijeti oba podatka točno, u suprotnom poslužitelj vraća HTTP status kod 401 s porukom *Unauthorized error: Bad credentials* koja se prevodi u korisniku razumljivu poruku *Wrong password or username*. U slučaju da korisnik nije registriran, pritiskom na link s tekstom *Sign up now!* otvara se registracijski zaslon prikazan na Podslici 4.1b. Za uspješnu registraciju korisnik mora popuniti sva polja, u suprotnom, kao što je prikazano na

Slici 4.1b, klijentska aplikacija vraća validacijske greške koje upozoravaju korisnika da nije popunio sva obvezna polja.



(a) Prijava u sustav

(b) Registracija s validacijskim greškama

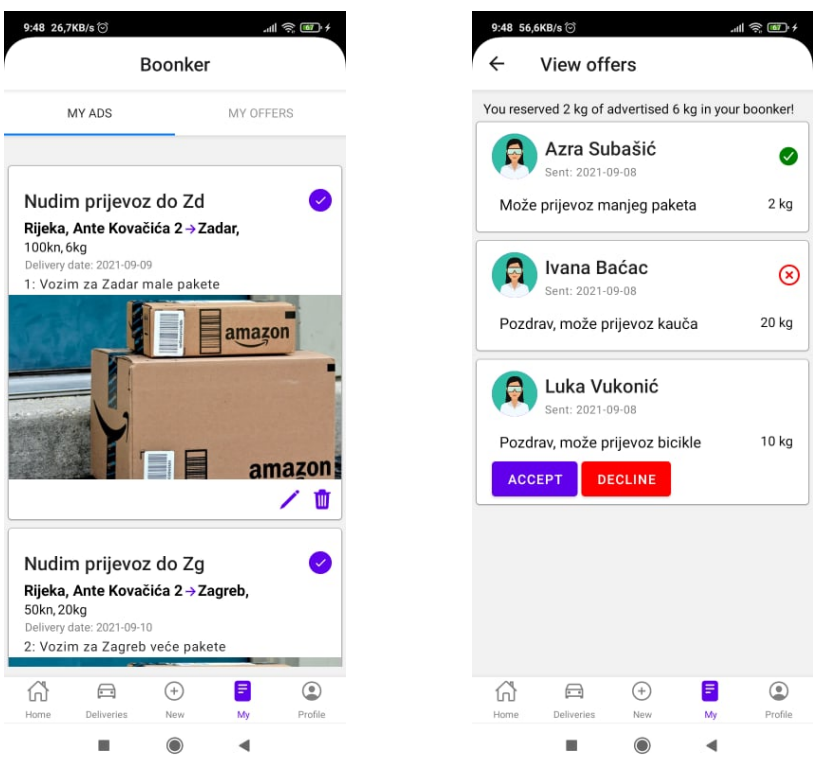
Slika 4.1 Zasloni prijave i registracije u sustav

4.2 Upravljanje vlastitim oglasima

Korisnik pritiskom na ikonu plusa u donjoj navigacijskoj traci otvara zaslon za stvaranje novog oglasa koji je već spomenut u Potpoglavlju 3.3.2 na Slici 3.9. Za stvaranje novog oglasa korisnik mora popuniti sva obvezna polja, definirati naslov, sadržaj i cijenu usluge i zatim, ovisno o tipu oglasa, koji može biti DELIVER PACKAGE ili SEND PACKAGE definirati početnu i odredišnu adresu prijevoza. U slučaju da je tip oglasa DELIVER PACKAGE korisnik nudi usluge prijevoza i treba

samo definirati početnu adresu na kojoj može pokupiti pakete, a odredišnu adresu definiraju ostali korisnici u svojim ponudama na oglase. Također, treba definirati koliko velike pakete može prevesti u kilogramima. Ako je tip oglasa SEND PACKAGE tad korisnik mora definirati i početnu i odredišnu adresu jer traži uslugu prijevoza tog paketa. Za kraj, potrebno je odabrati datum dostave.

Stvoreni oglasi mogu se pregledati pritiskom na ikonu oglasa s tekstualnom labelom *My* u donjoj navigacijskoj traci koja otvara zaslon sa sadržajem bitnim korisniku. Zaslon je prikazan u nastavku na Slici 4.2a. Na vrhu zaslona nalaze se dvije kartice *MY ADS* i *MY OFFERS* koje raspodjeljuju zaslon na dva dijela. Na prvom zaslonu prikazana je lista svih oglasa koje je korisnik stvorio, a na drugom, do kojeg se dolazi prelaskom prsta po ekranu u lijevo, lista svih ponuda na tuđe oglase.



(a) Zaslon s pregledom svih oglasa (b) Zaslon s ponudama na oglas

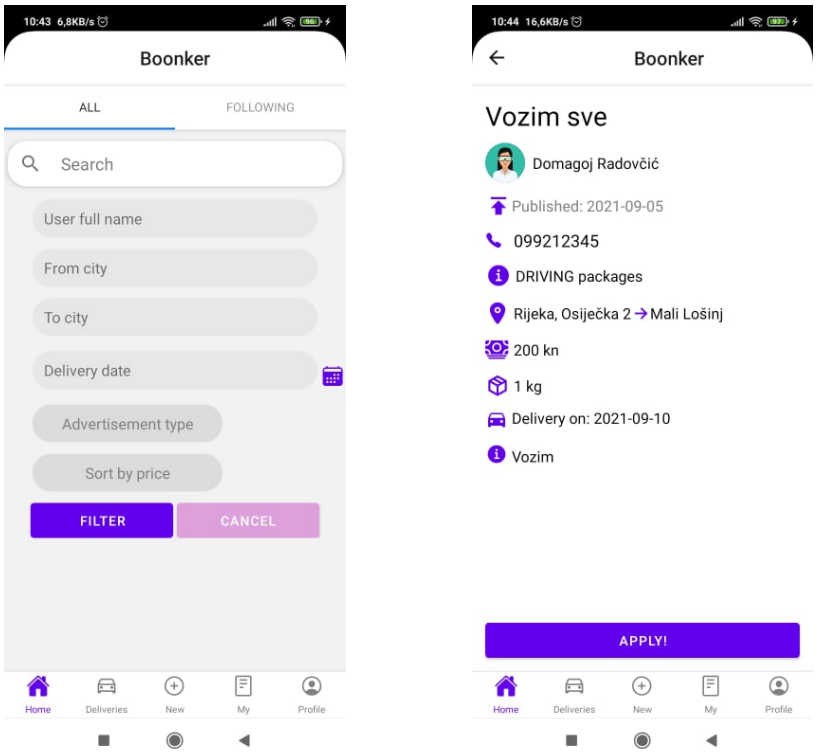
Slika 4.2 Zasloni s pregledom svih oglasa i ponudama na specifični oglas

Na zaslonu koji prikazuje sve oglase, svaki oglas je prikazan unutar kartice u kojoj

su prikazani detalji. U desnom gornjom kutu nalazi se ikona kvačice koja označava da je oglas aktivan. U dnu svake kartice nalaze se ikone olovke i kantice za smeće. Pritisak na ikonu olovke otvara zaslon za uređivanje tog specifičnog oglasa. Urediti se mogu gotovo svi podaci oglasa. Pritiskom na ikonu kantice za smeće otvara se dijalog koji traži od korisnika potvrdu o brisanju odabranog oglasa. Obrisani oglas ne briše se trajno iz baze podataka, već se samo označava s istinitom zastavicom IS_DELETED. Odabirom cijele kartice otvara se novi zaslon prikazan na Slici 4.2b koji prikazuje listu svih ponuda na odabrani oglas. Svaka ponuda se može prihvatiti ili odbiti. Prihvaćanjem ponude, kartica dobiva zelenu ikonu kvačice i veličina tog paketa dodaje se u ukupnu veličinu rezerviranog prostora. U slučaju odbijanja ponude otvara se dijalog koji traži potvrdu i razlog odbijanja. Razlog odbijanja se šalje odgovarajućem korisniku.

4.3 Pretraga i prijava na postojeće oglase

Pretraga postojećih oglasa svih korisnika prikazana je naslovnoj strani *Home* kartice donje navigacijske trake. Sadrži sve aktivne oglase svih korisnika sortirane od najnovijih do najstarijih. O zaslonu je već bilo riječ u Potpoglavlju 3.3.2 na Slici 3.8. Svi oglasi ili samo oglasi korisnika koje se prati mogu se pretražiti pritiskom na *Search* traku za pretraživanje koja otvara zaslon s naprednom tražilicom prikazan na Slici 4.3a. Lista svih oglasa može se pretražiti po raznim parametrima, imenom i prezimenom korisnika, prema vrsti oglasa, datumu dostave ili gradu odredišta ili polazišta. Lista se također može sortirati uzlazno ili silazno po cijeni dostave. Pritiskom na karticu oglasa otvara se novi zaslon koji sadrži detalje odabranog oglasa. Jedan takav zaslon prikazan je na Slici 4.3b gdje su prikazane dodatne informacije o oglasu, moguća veličina paketa, datum objave i dostave. Pritiskom na broj telefona autora oglasa aplikacija preusmjerava korisnika na poziv. Za slanje ponude na oglas potrebno je pritisnuti gumb s labelom *Apply* koji otvara dijalog u kojeg korisnik upisuje svoju ponudu, veličinu paketa koji želi poslati i po potrebi adresu. Korisnik status poslane ponude može vidjeti u prethodno spomenutoj *MY OFFERS* kartici.

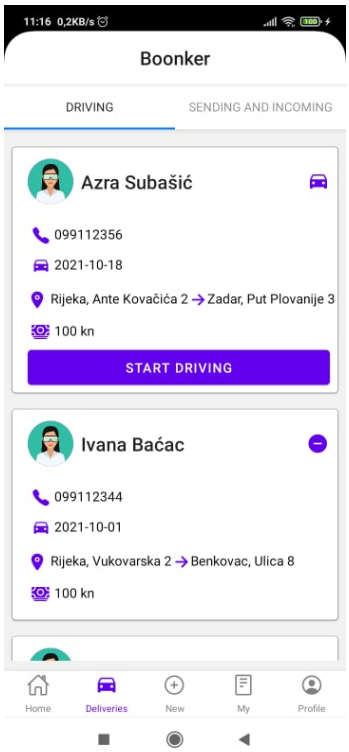


(a) Zaslon s naprednom tražilicom (b) Zaslon s detaljima oglasa

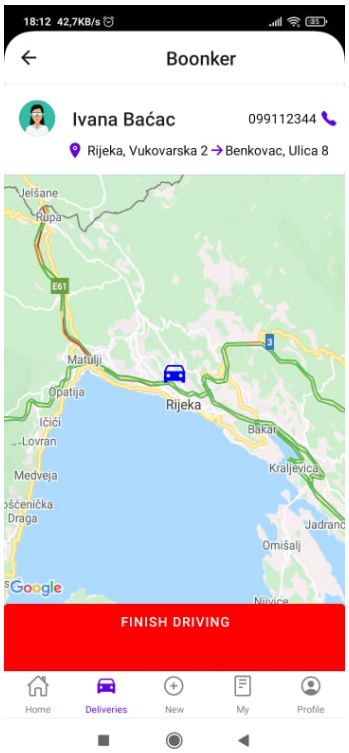
Slika 4.3 Zasloni s naprednom tražilicom i detaljima jednog oglasa

4.4 Slanje i primanje paketa

Kao što je u prethodnim poglavljima opisano, korisnik kroz klijentsku aplikaciju može slati pakete ili nuditi uslugu dostave ostalim korisnicima. Status svih dostava dan je drugoj kartici donje navigacijske trake *Deliveries* koja otvara zaslon prikazan na Slici 4.4a. U gornjem dijelu zaslona nalaze se dvije kartice *DRIVING* koja prikazuje listu dostava za čiju vožnju je zadužen korisnik i druga *SENDING AND INCOMING* za sve dostave koje pristižu korisniku. Na kartici *DRIVING* oglasi koji u gornjem desnom uglu kartice imaju ikonu zelene kvačice su izvršene dostave, ikona kruga s crticom je dostava koja je bila zakazana, ali nije izvršena. Kartice koje imaju ikonu automobila na redu su za dostavu taj dan. Dostava započinje vožnjom, odnosno pritiskom na gumb *START DRIVING* koja otvara novi zaslon prikazan na Slici 4.4b. Na zaslonu se prati lokacija dostavljača u stvarnom vremenu koja je iz



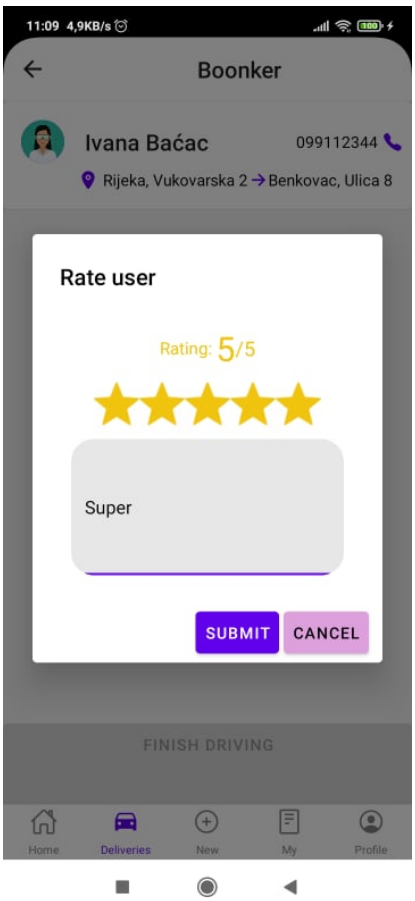
(a) Zaslon s listom svih dostava



(b) Zaslon s praćenjem lokacije paketa u stvarnom vremenu

Slika 4.4 Zasloni svih dostava i praćenjem paketa u stvarnom vremenu

tehničke perspektive detaljnije opisana u Potpoglavlju 3.4.3. Na vrhu zaslona nalaze se podaci o primatelju, ime prezime, adresa dostave i mobilni broj kojeg se može nazvati pritiskom na ikonu telefona. Isti zaslon vidi primatelj u kartici *SENDING AND INCOMING*. Jednom kada je dostavljač stigao na odredište i dostavio paket, pritiskom na gumb *FINISH DRIVING* završava se dostava. Konačno, otvara se dijalog za unos recenzije drugog korisnika prikazan na Slici 4.5. Korisnik unosi ocjenu od 1 do 5 i može ostaviti neobavezni dodatni komentar, a isti dijalog vidi i drugi korisnik. Ostavljena recenzija prikazuje se na korisničkom profilu.

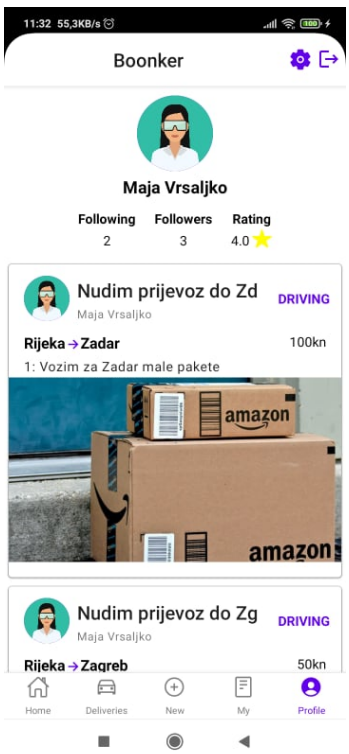


Slika 4.5 Zaslon za ocjenjivanje usluge korisnika

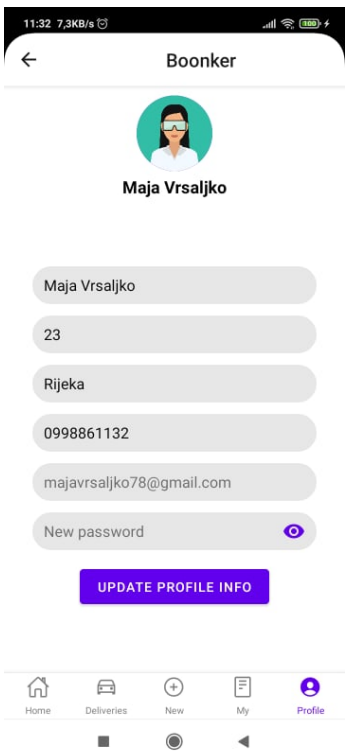
4.5 Upravljanje vlastitim profilom

Korisnik svoj osobni profil može vidjeti na posljednjoj kartici na donjoj navigacijskoj traci označenom ikonom profila čovjeka s tekstualnom labelom *Profile*. Zaslon osobnog profila prikazan je na Slici 4.6a. Korisnik na tom zaslonu može ponovno vidjeti sve svoje oglase, pratitelje, sljedbenike i recenzije. U gornjem desnom uglu zaslona nalaze se ikone postavki i izlaza iz aplikacije. Pritiskom na ikonu postavki otvara se zaslon za uređivanje osobnih podataka prikazan na Slici 4.6b. Korisnik može urediti svoje ime i prezime, godine, grad i kojeg dolazi ili postaviti novu lozinku. Pritiskom na ikonu izlaza odjavljuje se iz aplikacije i vraća na početni ekran

za prijavu.



(a) Osobni profil



(b) Uređivanje osobnih podataka

Slika 4.6 Zasloni s osobnim profilom i uređivanjem osobnih podataka

Poglavlje 5

Zaključak

U ovom radu implementiran je sustav za dostavu paketa zasnovan na lokaciji s elementima društvene mreže. Sustav je namijenjen korisnicima koji nude usluge prijevoza paketa ostalim korisnicima ili potražuju dostavu paketa. Sustav je implementiran na REST principima korištenjem modernih tehnologija (Spring Boot na poslužiteljskoj strani i React Native na klijentskoj). Implementirane su osnovne i najvažnije funkcionalnosti za rad sustava poput registracije i prijave, međusobne interakcije korisnika, dodavanja novih i pretrage postojećih oglasa, vrednovanja usluge pojedinog korisnika i praćenja lokacije paketa u stvarnom vremenu. Postojeći sustavi za dostavu paketa popularni u Hrvatskoj nude usluge samo unutar istog grada, dok implementirani sustav nudi mogućnost dostave unutar cijele Hrvatske.

Idući logični korak u razvoju je postaviti sustav na testnu okolinu na odgovarajućem servisu poput Microsoft Azurea. Nakon toga slijedi detaljno testiranje sustava, posebice klijentske aplikacije na različitim mobilnim uređajima, u stvarnim uvjetima. Povratne informacije korisnika mogu uvelike pomoći pri daljnjem razvoju i poboljšanju sustava.

Bibliografija

[1] J. M. K. Haosheng Huang, Georg Gartner *et al.*, "Location based services: ongoing evolution and research agenda," *Journal of Location Based Services*, vol. 2, no. 2, pp. 63–93, 8 2018.

[2] S. Steiniger, M. Neun, and A. Edwardes, *Foundations of Location Based Services*, 01 2006.

[3] Wikipedia, "Mobile phone tracking," https://en.wikipedia.org/wiki/Mobile_phone_tracking#Network-based, 2021, s interneta; pristupljeno 20.10.2021.

[4] ——, "Global Positioning System," https://en.wikipedia.org/wiki/Global_Positioning_System, 2021, s interneta; pristupljeno 21.10.2021.

[5] E. Howell, "Navstar: GPS Satellite Network," https://www.space.com/19794-navstar.html, 2018, s interneta; pristupljeno 20.10.2021.

[6] J. Hildenbrand, "How does GPS work on my phone?" https://www.androidcentral.com/how-does-gps-work-my-phone, 2020, s interneta; pristupljeno 21.10.2021.

[7] Wikipedia, "Wolt," http://hr.wikipedia.org/w/index.php?title=Wolt&oldid=6014883, 2021, s interneta; pristupljeno 21.10.2021.

[8] ——, "Glovo," http://hr.wikipedia.org/w/index.php?title=Glovo&oldid=6083904, 2021, s interneta; pristupljeno 21.10.2021.

[9] ——, "Bolt," http://en.wikipedia.org/w/index.php?title=Bolt%20(company)&oldid=1049964481, 2021, s interneta, pristupljeno 18.10.2021.

[10] L. Brdar, "Rest api kao poslužiteljski dio jednostranične web aplikacije," Završni rad, Tehnički fakulet, 2016.

[11] S. Kuserbajn, "Rest arhitektura," Završni rad, Veleučilište u Šibeniku, 2016.

[12] T. P. i L. Jelenković, "Autentifikacija i autorizacija korisnika na jednom mjestu," 2007. , s Interneta, https://bib.irb.hr/datoteka/299708.06_ISS_1043.pdf

[13] A. Sabol, "Autentikacija i autorizacija pomoću json web tokena (jwt)," Diplomski rad, Fakulet informatike, Sveučilište Jurja Dobrile, 2019.

[14] "Introduction to JSON Web Tokens," https://jwt.io/introduction, 2021, s interneta, pristupljeno 15.10.2021.

[15] "Spring MVC introduction," https://www.studytonight.com/spring-framework/spring-mvc-introduction, 2021, s interneta, pristupljeno 15.10.2021.

[16] Baeldung, "The DAO Pattern in Java," https://www.baeldung.com/java-dao-pattern, 2021, s interneta, pristupljeno 15.10.2021.

[17] R. N. Community, "React Native Introduction," https://reactnative.dev/docs/getting-started, 2021, s interneta, pristupljeno 19.10.2021.

[18] "MVC vs Flux vs Redux – The Real Differences," https://www.clariontech.com/blog/mvc-vs-flux-vs-redux-the-real-differences, 2020, s interneta, pristupljeno 23.10.2021.

Pojmovnik

AGPS Assisted Global Positioning System. 5

API Application Programming Interface. 10, 11, 17, 18, 21, 24, 30, 31, 33

APNS Apple Push Notification service. 32

CORS Cross-Origin Resource Sharing. 14

CRUD Create, Read, Update, Delete. 30

DAO Data Access Object. 17

ERD Entity Relationship Diagram. 16

GPS Global Positioning System. 1, 4, 5

HTTP Hypertext Transfer Protocol. 9, 10, 12, 14, 17, 19–21, 25, 31, 33–35, 37

IAN Internet Assigned Numbers Authority. 13

JSON JavaScript Object Notation. 9, 12, 21, 25, 29, 31, 34

JWT JSON Web Token. 12–15, 30

LBS Location-based services. 4

MVC Model-View-Controller. 17, 22

REST Representational State Transfer. 2, 9, 17, 18, 36, 45

RFC Request for Comments. 12, 13

URL Uniform Resource Locator. 9, 18–20, 25, 31, 33, 34

WAP Wireless Application Protocol. 4

XML Extensible Markup Language. 9

Sažetak

U radu su istraženi i opisani postojeći višekorisnički sustavi za dostavu paketa, hrane ili prijevoza ljudi zasnovani na lokaciji. S obzirom na prednosti i nedostatke dostupnih rješenja, implementiran je izvorni sustav za dostavu paketa zasnovan na lokacijskim podacima i umrežavanju korisnika. Dan je pregled korištenog tehnološkog stoga, arhitekture sustava i sigurnosnih aspekata autentikacije i autorizacije. Na poslužiteljskoj strani sustava opisana je implementirana struktura MVC i pristupne točke zasnovane na REST arhitekturi, a na klijentskoj strani realizacija responzivnog korisničkog sučelja za mobilne uređaje. Demonstrirani su svi važni slučajevi korištenja, poput registracije i prijave, međusobne interakcije korisnika, dodavanja novih i pretrage postojećih oglasa, praćenja lokacije paketa u stvarnom vremenu, primanja notifikacija te ocjenjivanja usluge korisnika. Uz opise karakterističnih slučajeva korištenja prikazane su i odgovarajuće zaslonske snimke.

Ključne riječi **— sustav za dostavu paketa, usluge zasnovane na lokaciji, društveno umrežavanje, mobilna aplikacija, arhitektura REST, React Native, Spring Boot**

Abstract

In this thesis, the existing multi-user location-based systems for the delivery of packages, food or transport of people are investigated and described. Given the advantages and drawbacks of the available solutions, the original system for package delivery based on location data and user networking has been implemented. An overview of the utilized technological stack, system architecture and security aspects of authentication and authorization is given. While the backend part of the system involves MVC pattern and REST endpoints, the frontend part refers to the implementation of the responsive user interface for mobile devices. Important use cases are demonstrated, such as registration and login, user interaction, adding new and searching existing ads, real-time package tracking, receiving notifications, and user rating. In addition to the description of typical use cases, the corresponding

screenshots are also presented.
Keywords **— parcel delivery system, location-based services, social networking, mobile application, REST architecture, React Native, Spring Boot**