|  |  |  |
| --- | --- | --- |
| Grb Univerziteta | Grb PMF-aUniverzitet u Novom Sadu  Prirodno-matematički fakultet  Departman za matematiku i informatiku |  |
|  |  |  |

Pane Odalović

**Aplikacija za rezervaciju restorana implementirana upotrebom Android i SpringBoot tehnologija**

-Diplomski rad-

Novi Sad, 2018

**SADRŽAJ**

[PREDGOVOR **5**](#_Toc518991664)

[1.UVOD **7**](#_Toc518991665)

[2. ANDROID PLATFORMA 9](#_Toc518991666)

[2.1 ARHITEKTURA PLATFORME 9](#_Toc518991667)

[2.1.1 LINUX KERNEL 11](#_Toc518991668)

[2.1.2 HARDWER ABSTRACTION LAYER 11](#_Toc518991669)

[2.1.3 ANDROID RUNTIME 11](#_Toc518991670)

[2.1.4 NATIVE LIBRARIES 12](#_Toc518991671)

[2.1.5 JAVA API FRAMEWORK 12](#_Toc518991672)

[2.1.6 SYSTEM APPLICATIONS 13](#_Toc518991673)

[2.2 KONCEPTI ANDROID APLIKACIJE 13](#_Toc518991674)

[2.2.1 ANDROID MANIFEST 13](#_Toc518991675)

[2.2.2 OSNOVNE KOMPONENTE 14](#_Toc518991676)

[2.2.3 RESURSI 16](#_Toc518991677)

[2.2.4 SIGURNOST 16](#_Toc518991678)

[2.2.5 PAKOVANJE APLIKACIJE 16](#_Toc518991679)

[3. OSTALE KORIŠĆENE TEHNOLOGIJE 17](#_Toc518991680)

[3.1 SPRING BOOT 17](#_Toc518991681)

[3.2 FIREBASE 18](#_Toc518991682)

[3.2.1 FIREBASE CLOUD MESSAGING 19](#_Toc518991683)

[3.3 GOOGLE MAPS 19](#_Toc518991684)

[3.4 ANDROID SQLite 20](#_Toc518991685)

[3.5 REST PROTOKOL 21](#_Toc518991686)

[4. ARHITEKTURA APLIKACIJE 23](#_Toc518991687)

[4.1 DIJAGRAM SLUČAJEVA KORIŠĆENJA ANDROID APLIKACIJE 23](#_Toc518991688)

[4.2 DIJAGRAM SLUČAJEVA KORIŠĆENJA SERVERSKE APLIKACIJE 25](#_Toc518991689)

[4.3 DIJAGRAM PAKETA 27](#_Toc518991690)

[4.4 DIJAGRAM KLASA 27](#_Toc518991691)

[4.5 FIZIČKI MODEL BAZE PODATAK 32](#_Toc518991692)

[5. IMPLEMENTACIJA APLIKACIJE 35](#_Toc518991693)

[5.1 SPRING BOOT 35](#_Toc518991694)

[5.2 FIREBASE 37](#_Toc518991695)

[5.3 GOOGLE MAPS 40](#_Toc518991696)

[5.4 REST ARHITEKTURA 41](#_Toc518991697)

[6. PRIKAZ RADA APLIKACIJE 45](#_Toc518991698)

[6.1 PRIKAZ POČETNIH ACTIVITY KOMPONENTI 45](#_Toc518991699)

[6.2 PERSPEKTIVA ADMINISTRATORA 46](#_Toc518991700)

[6.3 PERSPEKTIVA KORISNIKA 49](#_Toc518991701)

[7. ZAKLJUČAK 55](#_Toc518991702)

[8.LITERATURA 57](#_Toc518991703)

[BIOGRAFIJA 59](#_Toc518991704)

# PREDGOVOR

*Android[1]*palatforma je zasnovana na *Linux* operativnom sistemu i namenjena je za mobilne uređaje kao što su mobilni telefoni i tablet računari. Za razvoj *Android* aplikacije se uglavnom koristi *Java* programski jezik. Platforma je otvorenog koda, što znači da svako može da je prilagodi svojim potrebama. *Android* platforma iz godine u godinu postaje sve popularnija i koristi se u svim oblastima. Pored toga svakodnevno se pojavljuju aplikacije za različite namene, i to aplikacije za zabavu, aplikacije za edukaciju, kao i aplikacije za komunikaciju.

*Spring Boot[8]* framework trenutno je jedna od najpopularnijih tehnologija za implementaciju serverske strane aplikacija. Ovaj *framework* zasnovnan je na *Java* platformi, što znači da bilo koje aplikacije koje su pisane u programskom jeziku Java mogu da koriste ovaj *framework. Spring Boot framework* je platforma otvorenog koda, kao što važi i za *Android* paltformu.

Za razvoj aplikacije za rezervaciju restorana, najpogodnija je Android platforma, zato što aplikacija treba da bude mobilna i u svakom trenutku dostupna korisniku. Osnovu koju ova aplikacija pruža jeste da korisnik sa bilo kog mesta može da rezerviše mesto u željenom restoranu i da pozove prijatelje na druženje. Korisnik na osnovu svoje lokacije može da pretražuje restorane u blizini i da ima detaljn prikaz na mapi.

Diplomski rad sastoji se od sledećih poglavnja:

1. U *uvodnom* delu opisujemo cilj diplomskog rada.
2. Poglavlja *Android platforma* opisuje *Android* platformu, arhitekturu platforme, osnovne komponente.
3. U poglavlju *ostale korišćene tehnologije* opisane su sve tehnologije koje su korišćene u izradi aplikacije, naglasak je na *Spring Boot* framework-u.
4. *Arhitektura aplikacije* tu se nalazi sama arhitektura aplikacije, kako je aplikacija stavljena u pakete, dijagrami klasa, fizički model baze podataka i dijagrami slučajeva korišćenja.
5. Poglavlje *implementacija* se sastoji od prikaza samog koda aplikacija, uz detaljan opis.
6. Poglavlje *prikaz rada aplikacije* sadrži upustvo za korišćenje aplikacije, iz perspektive korisnika i administratora.
7. U *zaključku* se nalaze sve uočene prednosti *Android* platforme i *Sprig Boot* framework-a.
8. U *literaturi* je spisak svih korišćenih literatura pri izradi aplikacije.

# UVOD

U današnje vreme, činjenica je da svaki čovek ima neku vrstu mobilnog uređaja. Većina tih uređaja koriste *Andorid* operativni sistem. Zato su Android aplikacije doživele veliku ekspanziju. Zbog toga aplikacija za rezervaciju restorana je napravljena kao Android aplikacija. Aplikacija omogućava svim korisnicima pregled svih restorana na jednom mestu, i rezervaciju istih.

Aplikacija za rezervaciju restorana predstavlja mobilnu aplikaciju, koja je u svakom trenutku dostupa korisnicima. Korisnici aplikacije imaju pristup mapama, gde im je prikazana lokacija na kojoj se nalaze, kao i restorani u njihovoj blizini. Svaki korisnik može da odabere koji tip restorana želi da mu se prikaže na mapi i može da odredi do koje udaljenosti od lokacije telofona da se prikazuju restorani.

Rezervacije za restorane se prave od strane jednog korisnika. Korisnik odabere željeni restoran, a time je omogućen pristup pozivima sa mobilnog uređaja. Odabirom poziva korisnik rezerviše mesto u restoranu. Da bi korsnik pozvao svoje prijatelje, tj. poslao im obaveštenje o rezervaciji restorana i vremenu trajanja rezervacije, korisnik mora da zna e-mail adrese svojih prijatelja. Preko e-mail adrese dodaje prijatelje u rezervaciju. Kad se napravi rezervacija obaveštenja se šalju na mobilne uređaje i na e-mail adrese pozvanih korisnika.

U aplikaciji svaki korisnik može da oceni restorane. Prilikom pregleda restorana, može da vidi ocenu restorana i na osnovu toga korisnik ima uvid u mišljenje ostalih korisnika aplikacije o nekom restoranu.

Svaki korisnik ima listu omiljenih restorana. Time je omogućeno korisniku da lakše nalazi restorane koje često posećuje i koji se njemu sviđaju. Ta lista može biti modifikovana tako što restorani mogu biti dodati ili uklonjeni iz iste.

# 2. ANDROID PLATFORMA

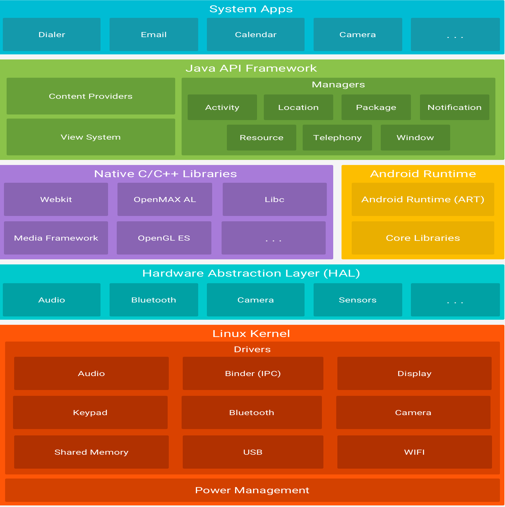
*Android* je najpopularnija platforma za razvoj mobilnih aplikacija, koje se izvršavaju na različitim uređajima kako što su: mobilni telefoni, tableti, itd. Gotovo svaki čovek ima jedan Android uređaj i svakim danom im se povećava brojnost. Taj podatak nam govori koliko je Android platforma popularna u današnje vreme i da će vremenom postati sve popularnija. Androidnudi dodatne alatke koje programeru olakšavaju pravljenje Andorid aplikacije. Ponuđena je mogućnost automatskog prilagođavanja grafičkog korisničkog interfejsa na različitim uređajima, kao i skup dodatnih resursa za optimizaciju na različitim uređajima koji zavise od karakteristika uređaja. Pored toga programeri imaju punu kontorlu nad svojom aplikacijom i na taj način je postignuto kreiranje jedinstvenog koda.

*Android Studio[2]* je jednostavno razvojno okruženje koje Androidnudi za razvoj mobilnih aplikacija, koje u sebi sadrži sve potrebne alate. *Andrioid Studio* je zasnovan na *IntelliJ* IDEA i nudi mogućnost razvoja aplikacije u *Java* i *Kotlin* programskom jeziku, kao i integraciju C i C++ jezika. Poseduje *Android Emulator* koji nam simulira rad uređaja za koji pravimo aplikaciju. Ne moramo posedovati sve uređaje za koje pravimo Android aplikaciju, već nam to *Android Emulator* obezbeđuje. *Android* je integrisan i sa *Git[12]* sistemom. To znači da postoji mogućnost paralelnog razvijanja jedne aplikacije od strane više programera sa različitih uređaja, integracija koda.

## 2.1 ARHITEKTURA PLATFORME

*Android platforma* zasnovana je na *Linux Kernel-u. Android platforma* se sastoji iz sledećih delova:

* Linux kernel
* Hardware abstraction layer (HAL)
* Android runtime (ART)
* Native libraries
* Java API framework
* System applications



Slika 2.1.1.-Arhitektuka Android platforem

Ova struktura je prikazana na slici 2.1.1. Arhitektura platforme se vidi, na pomenutoj slici, sa komponentama svakog dela arhitekture. U nastavku ovog poglavlja biće opisani svi delovi arhitekture.

### 2.1.1 LINUX KERNEL

*Linux kernel*  je temelj *Android platforme* na koji se oslanjaju komponente višeg nivoa kada trebaju pristupiti funkcionalnostima hardvera. Pruža standardne servise oparativnog sistema, kako što su: upravljanje memorijom i procesima, prava pristupa, drajveri za različite uređaje, itd. Ovaj deo arhitekture vodi računa o komunikaciji između procesa, deljenju memorije, upravljanju struje, itd.

### 2.1.2 HARDWER ABSTRACTION LAYER

*Hardwer abstraction layer,* sloj iznad *Linux kernel-a,* obezbeđuje interfejs preko kog se iz jezika višeg nivoa može pristupiti određenoj hardverskoj komponenti uređaja kao što je: kamera, bluetooth, određeni senzor. Za svaku hardversku komponentu postoji modul koji implementira interfejs, preko kog se pristupa toj komponenti. Kad želimo da pristupimo hardverskoj komponenti iz višeg nivoa, poziva se *API* višeg nivoa. Androidsistem učitava modul za određenu komponentu koja se poziva i izvršava se pozivna funkcija.

### 2.1.3 ANDROID RUNTIME

*Android runtime* komponenta na trećem nivou Android arhitekture, iznad *Hardwer abstraction layer* sloja. Ova komponenta predstavlja okruženje u kom se izvršava aplikacija. *Android runtime* izvršava *DEX* fajlove, *bytecode* format koji izvršava Androidaplikacija u ovoj komponenti. *DEX* fajlovi nastaju kompajliranjem *Java* izvornog koda. *Android runtime* se koristi od verzije *Android 5.0.* Sve verzije posle koriste isto *Android runtime*, a verzije pre *Android 5.0* koriste *Dalvik Virtual Machine.* Prednosti *Android runtime* je trenutno izvršavanje koda, optimizovanje čišćenja memorije, bolja podrška i debagovanje aplikacija iz *Android Studio-a.*

### 2.1.4 NATIVE LIBRARIES

Osnovne komponente kao što su *Android runtime* i *Hardwer abstraction layer* razvijene su u programskom jeziku C, koji zahteva izvorne biblioteke napisane u programskom jeziku C i C++. Android operativni sistem nudi *Java* API setvise preko kojih se može pristupiti nekom od ovih *native* biblioteka. Postoji još jedan način za pristup funkcionalnostima koje nude ove biblioteke, a to je *Android NDK.*

### 2.1.5 JAVA API FRAMEWORK

Sve funkcionalnosti Androidsistemasu dostupane preko API(**A**pplication **P**rogramming **I**nterface) *servisa* napisanih u programskom jeziku Java*.* Programeri mogu brzo kreirati aplikacije ponovnom upotrebom osnovnih, modularnih komponenti i servisa, u koje spadaju:

* *Views System* -predstavlja proširen skup grafičkih komponenti koje korisnik koristi pri razvijanju grafičkog korisničkog interfejsa.
* *Resource Manager* - preko njega se pristupa resursima koji nisu izvorni kod.
* *Notification Manager* - omogućava kreiranje obaveštenja za korisnika i prikazuje ih na statusnoj liniji telefona.
* *Activity Manager* -upravlja životnim ciklusom aplikacije, i navigira između komponenti unutar aplikacije ili između dve aplikacije.
* *Content Provider* -upravlja podacima aplikacije, gde svaka aplikacija može da pristupi podacima iz druge aplikacije.

### 2.1.6 SYSTEM APPLICATIONS

*System aplications* nalazi se na samom vrhu arhitekture Androidaplikacije. Androidima veliki broj aplikacija drugih proizvođača koje su dostupne putem prodavnice aplikacije, kao što je *GooglePlay* (prodavnica aplikacija kreirana od strane Google-a) ili putem instalacije *APK* datoteka aplikacije sa web-sajtova drugih proizvođača. Aplikacija GooglePlay omogućava listu dostupnih aplikacija koje su kompatibilne za odabrani uređaj. Neke od mogućih aplikacija koje su dostupne korisnicima su pretraživači, komunikacione mreže, kamere, itd. Sve ove aplikacije koje se mogu preuzeti su na istom nivou kao i aplikacije koje korisnici instaliraju na svojim uređajima. To znači da svaka preuzeta aplikacija može biti zamenjena nekom drugom aplikacijom koja obavlja istu funkciju, a napravljena je od strane korisnika.

Aplikacije za Android se razvijaju u Java programskom jeziku, korišćenjem *Android razvojnog paketa* (**Android S**oftware **D**evelopment **K**it, **SDK**).

## 2.2 KONCEPTI ANDROID APLIKACIJE

Androd aplikacije ima osnovne koncepte potrebne za rad same aplikacije, a to su: ***Android manifest, osnovne komponente, resurse, sigurnost*** i ***pakovanje aplikacije***.U ovom poglavlju sledi opis svih nabrojanih koncepata.

### 2.2.1 ANDROID MANIFEST

Svaka Androidaplikacija u svom korenu mora da ima fajl *AndroidManifest.xml*. Sve komponente koje sadrži jedna aplikacija se definišu u okviru *AndroidManifest.xml* fajla. Komponente aplikacije se aktiviraju preko asihrone poruke – *intent*. Pored navođenja komponenti u Manifest fajlu deklarišu se:

* Hardver i softver koji aplikacija koristi ili zahteva, kao što su kamera, multi-touch ekran ili bluetooth usluge.
* API biblioteke sa kojima aplikacija treba da se poveže.
* Minimalan API nivo, koji zavisi od toga koje API servise aplikacija koristi.
* Sve dozvole koje aplikacija zahteva.
* Sve informacije o aplikaciji.

Primer kako izgleda *AndrodiManifest.xml* fajl dat je na listingu 2.2.1

* <?xml version="1.0" encoding="utf-8"?>  
  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  
   package="com.example.pane.myapplication">  
    
   <application  
   android:allowBackup="true"  
   android:icon="@mipmap/ic\_launcher"  
   android:label="@string/app\_name"  
   android:roundIcon="@mipmap/ic\_launcher\_round"  
   android:supportsRtl="true"  
   android:theme="@style/AppTheme">  
   <activity android:name=".MainActivity">  
   <intent-filter>  
   <action android:name="android.intent.action.MAIN" />  
    
   <category android:name="android.intent.category.LAUNCHER" />  
   </intent-filter>  
   </activity>  
   </application>  
    
  </manifest>

Listing 2.2.1-Primer AndroidManifest.xml fajle

### 2.2.2 OSNOVNE KOMPONENTE

Komponente predstavljaju osnovne elemente svake *Android*aplikacije. One određuju ponašanje i karakteristike aplikacije, pa samim tim sistem može da ostvari komunikaciju sa aplikacijom preko njenih komponenti. Android aplikacije mogu da imaju 4 vrste komponenti:

* *Activity* (aktivnost)
* *Service* (servis)
* *Broadcast* reciver (prijemnik poruka)
* *Content* provider (provajder sadržaja)

**Aktivnost** predstavlja osnovnu komponentu za interakciju sa korisnikom. Jedan ekran sa korisničkim interfejsom predstavlja aktivnost. Iako su aktivnosti u okviru jedne aplikacije napravljene i regulisane tako da predstavljaju logičku celinu, svaka od njih je nezavisna. Pa tako je lako zaključiti da različite aplikacije mogu da pokreću aktivnosti iz drugih aplikacija. Aplikacija se sastoji od više povezanih **Activity klasa**, a jedna Activity klasa predstavlja jedan ekran aplikacije. Pri pokretanju aplikacije prikazuje se *glavna* Activity klasa. Zbog čitkosti i preglednosti koda obično jedna Activity klasa implementira jednu akciju koju korisnik može da uradi.

Svaka aktivnost može da pokrene **servis**. Servis je komponenta koja ne pruža korisnički interfejs. Ova komponenta se izvršava u pozadini i time servis omogućava realizaciju dugotrajnih operacija, kao i izvršavanje udaljenih procesa. Takođe, upotrebom servisa omogućeno je izvršavanje paralelnih radnji.

**Broadcast** **reciver** je komponenta koja reaguje na događaje i obrađuje događaje. Ova komponenta je usko vezana za prenos sistemske objave i notifikacije. Većina broadcastreciver*-*a potiče iz sistema, npr. obaveštenje da li je baterija slaba. Svaki događaj može da bude kreiran od strane sistema ili od strane druge aplikacije. Ti događaji se implementiraju u **Intent klasama**.

**Content provider** je komponenta koja upravlja podacima. Ti podaci predstavljaju deljeni skup podataka aplikacije. To su podaci koji su skladišteni u fajl sistemu, bazi, na web-u ili na bilo kojoj perzistentnoj lokaciji kojoj aplikacija može da pristupi. Preko provajdera sadržaja, aplikacija može da vrši upit nad podacima. Takođe, ova komponenta omogućava razmenu podataka između aplikacija.

### 2.2.3 RESURSI

Svaka Androidaplikacija osim koda sadrži i odgovarajuće resurse. To su delovi aplikacije koji se ne kompaliraju. U resurse spadaju layout fajlovi, stringovi, slike, animacije, meniji stilovi, boje, itd. Korisnik može da napravi različite vrste XML formata(čitljiv od strane kompajlera). Najčešći zapisi u XML fajl jeste za tekstualne resurse. Sa korištenjem resursa, olakšava se ažuriranje različitih karakteristika aplikacije bez modifikovanja koda. Resursi omogućavaju i optimizaciju aplikacije za različite konfiguracije uređaja, kao što su različiti jezici i veličine ekrana (***portrait* *i* *landscape* *layout***). Ovo je važan aspekt razvoja Android aplikacije zbog prilagođavanja različitim tipovima uređaja.

### 2.2.4 SIGURNOST

Sigurnost je jedna od bitnih karakteristika Android-a. Napravljen je takav sistem koji ograničava pristup komponentama. Odnosno, aplikacija koristi samo one komponente koje su joj potrebne. Unutar Androidsistema svaka aplikacija se izvršava u ograničenom prostoru, tako da je tim ostvarena nezavisnost aplikacija u samom sistemu. Sigurnost je realizovana na tome da svaka aplikacija traži pristup podacima. Taj pristup može biti automatski odobren od strane sistema ili će se prepustiti korisniku da sam odlučuje.

### 2.2.5 PAKOVANJE APLIKACIJE

**APK** (Android Application Package) je paket Android aplikacije. Kreirana aplikacija se pakuje u .*apk* fajl. To je kao JAR fajl za Android aplikacije i sadrži bajtkod aplikacije, korišćenih biblioteka i resursa. Sa instaliranjem APK na uređaju, aplikacija koja je upakovana može da se instalira. Međutim, za njenu objavu potrebno je izgenerisati jedinstven ključ.

# 3. OSTALE KORIŠĆENE TEHNOLOGIJE

U ovom poglavlju su opisane ostale korišćene tehnologije. To su: *spring boot, firebase, google maps, andrioid SQLite i rest protokol.* Svaka tehologije ja predstavljene, na koji način radi i kako se koristi.

## 3.1 SPRING BOOT

Serverski deo aplikacije je napisan u programskom jeziku Java koristeći *Spring Boot Framework*. Spring Boot Framework pruža obiman model programiranja i konfiguracije namenjen razvoju modernih *Enterprise Java* aplikacija. Ključni element Spring Boot Frameworka je infrastrukturna podrška na aplikativnom nivou. Omogućuje razvojnim timovima da se fokusiraju na aplikacionu biznis logiku, bez nepotrebne povezanosti sa *Deployment* okruženjem .

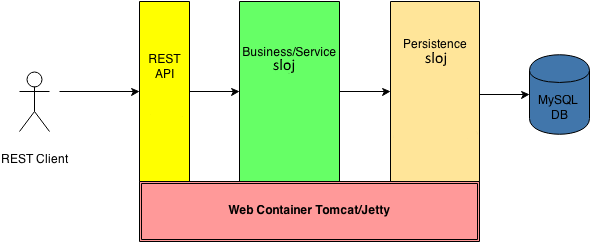
Serverski deo aplikacije je razdvojen u tri primarna sloja: *Presentation* (*REST API*), *Service/Business* i *Persistence.*

*Presentation* sloj vrši interakciju sa *Service/Business* slojem. Glavni zadatak ovog sloja jeste posredovanje između klijenta i *Service/Business* sloja. Ovaj sloj sa klijentima vrši interakciju obično putem HTTP protokola koristeći HTML, XML (Extensible Markup Language) ili JSON oblik reprezentacije podataka. Skup klasa koje se nalaze u ovom sloju su obično kontroleri, handler-i za izuzetke i greške, filteri itd. Klase Presentation sloja ne bi trebale da sadrže biznis logiku.

*Service/Business* sloj je zadužen za vršenje biznis logike, kao što je na primer u slučaju rezerzacije potrebno da se pošalju mejlovi i notifikacije svim pozvanim korisnicima iz rezervacije. Ovaj sloj ima direktnu interakciju sa *Persistence* slojem.

*Persistence* sloj služi za pristup bazi podataka. Zadatak ovog sloja je da kreira skup interfejsa koji će služiti kao apstrakcija pristupu podataka, dok će implementacija biti zadužena za konkretno korišćenu tehnologiju.

Na slici 3.1. prikazani su slojevi serverskog dela aplikacije.



Slika 3.1. –Slojevi serverskog dela aplikacija

## 3.2 FIREBASE

*Firebase* predstavlja platformu za mobilne uređaje koja omogućava razvijanje aplikacije. Ova platforma sadrži nekoliko dobro definisanih funkcija koje se mogu kombinovati. Funkcije su najčešće vezane za pozadinske usluge mobilnih uređaja. *Firebase* pruža mnoge usluge:

* *Analitika* - omogućava uvid u korišćenje aplikacije, kao i u angažovanje korisnika.
* *Hosting* – predstavlja servis za skladištenje statičkog sadržaja. Statičke datoteke koje podržavaju skladištenje su CSS, HTML, JavaScript. Firebase hosting je nastao po zahtevu korisnika zbog toga što je pored baze podataka u realnom vremenu bilo potrebno mesto za skladištenje njihovog sadržaja.
* *Baze podataka u realnom vremenu* – Servis programerima aplikacija pruža API koji omogućava sinhronizovanje aplikacionih podataka preko klijenta i njihovo skladištenje na tzv. “oblak”. Omogućene su biblioteke za rad sa iOS-om, JavaSript-om, Java-om, Android-om. Pored toga, omogućen je pristup REST API-u. REST API koristi serverski protokol za slanje događaja, što predstavlja API za kreiranje HTTP konekcije koje se koriste za primanje notifikacija sa servera.
* *Autentifikacija* – Servis koji može da autentifikuje korisnike koristeći kod sa klijentske strane. Firebase autentifikacija omogućava pristup preko društvenih mreža (Twitter i Facebook), Google i GitHub. Pored toga, omogućen je pristup Firebase-u direktno preko email-a i šifre.

### 3.2.1 FIREBASE CLOUD MESSAGING

**F**irebase **C**loud **M**essaging (**FCM**) predstavlja platformu za razmenu poruka. Sa ovom platformom se obaveštavaju klijenti aplikacija da su podaci dostupni za sinhronizaciju. Firebase server prepoznaje korisnika preko dodeljenih tokena. Serverski deo aplikacije mora da komunicira sa Firebase serverom. Ta se komunikacija ostvaruje preko generisanja potrebnih ključeva, i to se radi preko zvaničnog sajta Firebasa-a. Postoje 3 načina slanja poruka, a to su slanje pojedinačnom uređaju, slanje grupi uređaja, kao i slanje uređajima koji su pretplaćeni na aplikaciju. Implementacija Firebase Cloud Messaging uključuje dve bitne komponente za slanje i primanje poruka i to:

* Server aplikacije koji je u okruženju Cloud Functions za Firebase i on kreira i šalje poruke,
* Klijentska aplikacija koja prima poruke može da bude Android ili iOS.

Kad klijent šalje poruku, šalje se *json* fajl sa željenim podacima, serveru Firebase-a sa ključem za autentifikaciju. Server izgeneriše poruku i prosleđuje poruku određenom korisniku preko njegovog tokena.

## 3.3 GOOGLE MAPS

**Google maps** su deo ponuđenih Google tehnologija. Predstavljaju aplikacioni servis Internet mapa za desktop i mobilne uređaje. Korisnicima je omogućen pristup satelitskim snimcima, mapama gradova, ali i mogućnost pregleda ulica. Mape su napisane uz pomoć JavaScript-a i XML-a. Uz pomoć koda napravljeni su alati koji su omogućili uvođenje ikonice za proizvoljne lokacije, koordinate lokacija i metapodatke. Takođe, neki krajnji korisnici su iskoristili i preokrenuli alat kako bi dobili skripte za klijentsku stranu, što je omogućilo korisniku ili web-sajtu da uključi proširene ili prilagođene funkcije na interfejsu Google mapa.

Postoji podrška za mape koje se ugrađuju zahvaljujući *aplikacionom programskom interfejsu* (eng. **A**pplication **P**rogramming **I**nterface **- API**). Pored API-a postoji i lokator poslovnih objekata i drugih organizacija u brojnim zemljama širom sveta.

## 3.4 ANDROID SQLite

SQLite je programska biblioteka koja implementira sistem za upravljanje relacionim bazama podataka. Koncept baze podataka SQLite je, za razliku od drugih sistema klijent-server, povezan sa kodom aplikacija, umesto da pravi jezgro za samostalan proces sa kojim aplikacija komunicira. Povezivanjem sa kodom ova biblioteka postaje sastavni deo aplikacije. SQLite podržava različite komande SQL-a (eng**. S**tructured **Q**uery **L**anguage) ali sa nekim izuzecima. To je u stvari upotreba neuobičajene topologije za SQL. Neki računarski procesi ili niti mogu pristupati istim bazama podataka bez problema. Nekoliko mogućnosti čitanja se mogu ostvariti paralelno. Upisivanje je moguće ukoliko trenutno nije u toku pokušaj pristupa bazi, inače pokušaj upisa propada sa porukom greške.

SQLite se koristi sa velikim brojem programskih jezika, kao što su: C, C++, Java, Delphi, PHP, Objective-c, C#, Basic, itd. Pored toga postoje ActiveX omotači koji omogućavaju pristup SQLite-u na Windows-u preko JavaScript-a. Ovo omogućava i korištenje baza podataka kod HTML aplikacija.

Da bi se pristupilo SQLite bazi podataka preko Android sistema, potreban je poseban binarni sistem koji implementira interfejs između baze podataka i korisnika. Verzija koja se trenutno koristi je 3, tj. *sqlite3*. Instalacija se odvija u direktorijumu unutar putanje /system/xbin.

## 3.5 REST PROTOKOL

**Re**presentational **S**tate **T**ransfer (REST) predstavlja koncept u kome jedinstveni URL-ovi domena predstavljaju, zapravo, objekte kojima upravljamo kroz HTTP metode. Pojednostavljeno rečeno, REST predstavlja stil arhitekture koji se sastoji od skupa komponenti, konektora i elemenata koji se nalaze unutar distribuiranog sistema. Platformska i jezička nezavisnost važi i za REST kao i za ostale servise. Jedan od najvećih zagovornika REST protokola je Google, i to servisi kao što su Google Translate API, Google Earth API, Google Maps API itd. Osnovne karakteristike Web arhitekture su:

* **Performanse** – Označava parametre i tehničke podatke. Način interakcije komponenti utiče na performansu.
* **Skalabilnost** – Služi za podršku velikoj količini podataka i interakciji između komponenata.
* **Jednostavnost**
* **Modifikacija** – Lakoća izvođenja promena unutar koda.
* **Vidljivost**
* **Prenosivost** –Mogućnost prenošenja koda sa jednog mesta na drugo.
* **Pouzdanost** – Otpornost na neuspjeh na sistemskom nivou prilikom grešaka unutar komponenti, konektora i podataka.

REST protokol omogućuje jednostavnu implementaciju, komunikaciju između tehnologija, kao i jedan od najbitnijih aspekata Interneta, a to je sigurnost. REST je preuzeo dobre stvari iz SOAP protokola kao što su neutralnost, tj. mogućnost programskih jezika kao što su C/C++, C#, Java i slični da komuniciraju. REST protokol je usko povezan sa HTTP protokolom. REST zahteva od korisnika da eksplicitno koriste HTTP metode na način koji je propisan sa definicijom protokola. Osnovni REST dizajn princip uspostavlja 1-1 vezu između CRUD operacije (kreiranje, čitanje, modifikovanje, brisanje) i odgovarajućih HTTP metoda. Tako, prema ovom mapiranju važi:

* + za kreiranje resursa na serveru koristiti POST
  + za preuzimanje resursa koristiti GET
  + za menjanje stanja resursu ili njegovu modifikaciju koristiti PUT
  + za uklanjanje resursa koristiti DELETE

REST je najbolji način za razvijanje Web servisa koji su jednostavni za implementaciju i održavanje. Za razliku od SOAP baziranih Web servisa , ne postoji zvanični standard za RESTful Web API-e. To je zato što je REST arhitektonski stil, dok je SOAP sam po sebi protokol. REST nije sam po sebi standard, ali RESTful implementacije koriste standard, kao što je *HTTP, URI, JSON*  i *XML*.

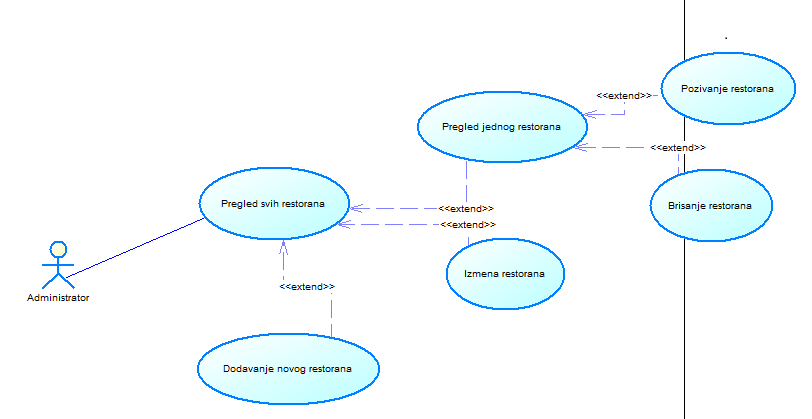
# 4. ARHITEKTURA APLIKACIJE

Serverska strana aplikacije je rađena u *Spring Boot* framework-u. U nastavku sledi arhitektura aplikacije, kao i sve funkcionalnosti koje pruža ova serverska aplikacija.

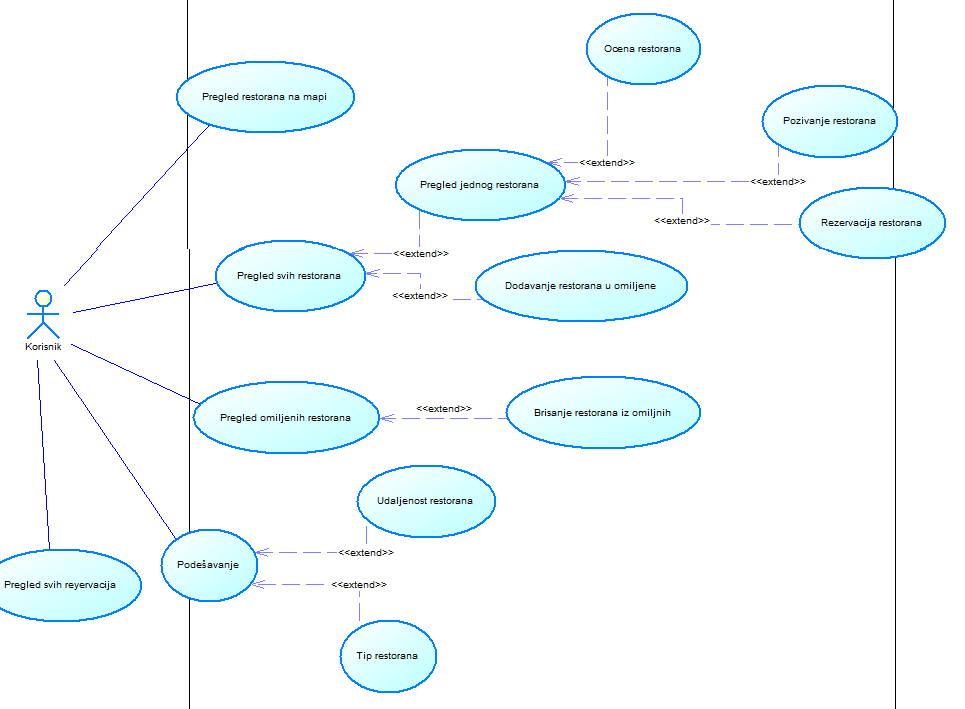
## 4.1 DIJAGRAM SLUČAJEVA KORIŠĆENJA ANDROID APLIKACIJE

U aplikaciji postoje dve perspektive: administrator i korisnik, i svaka perspektiva ima svoj skup funkcionalnosti koji su predstavljeni na dva dijagram slučajeva korišćenja. Na slici 4.1.1 je prikazana perspektiva *administratora* i sve njegove funkcionalnosti. Na slici 4.1.2. prkazana je perspektiva *korisnika,* sa svi njegovim funkcionalnostima.

Slika 4.1.1.-Dijagram sa perspektivom administrator



Administrator može da pregleda sve restorane, da klikne na jedan restoran i da mu se prikažu informacije o tom restoranu. Može da pozove restoran ili da izbriše restoran. Svaki restoran može da izmeni i da doda novi restoran.



Slika 4.1.2.-Dijagram sa perspektivom korisnika

Korisnik vidi mapu sa svim restoranima na kojoj su isti označeni, vidi pregled jednog kao i svih restorana. Korisnik može da oceni svaki restoran, pozove restoran i napravi rezervaciju za restoran. Korisnik ima pregled omiljenih restorana. Omiljeni restorani su predstavljeni listom koja može biti modifikovana sa dodavanjem ili uklanjanjem restorana. Može da podesi da mu se na mapi vide restorani za koje je odabrao tip i udaljenost od trenutne pozicije. Korisnik može da vidi sve rezervacije u kojim učestvuje.

## 4.2 DIJAGRAM SLUČAJEVA KORIŠĆENJA SERVERSKE APLIKACIJE

Dijagram slučajeva korišćenja serverske aplikacije predstavlja sve moguće API-e koje pruža server. Kao što se može videti na slici 4.2.1. klijent je Android aplikacija, sa metodama koje ta aplikacija poziva sa servera.



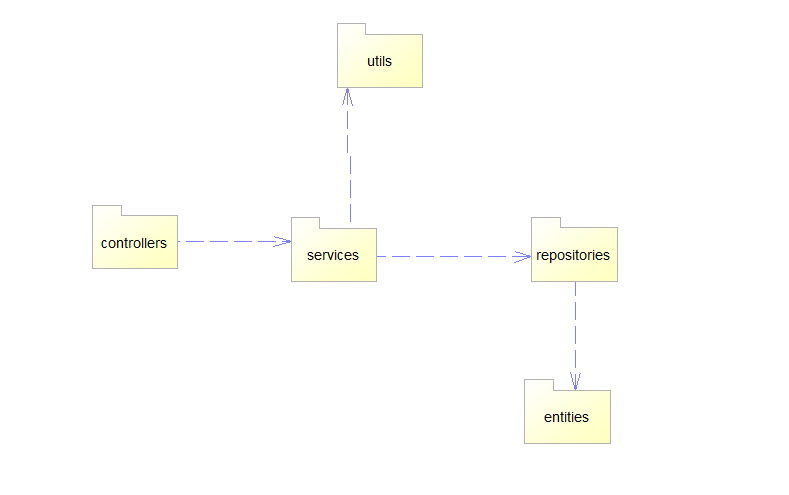
Slika 4.2.1.- Dijagram klasa serverske aplikacije

Na slici 4.2.1. su sve metode. U nastavku ovog poglavlja, svaka metoda biće opisana:

* *Zahtev za logovanje* -pozivom ove metode klijent može da proveri da li u aplikaciji postoji korisnik sa prosleđenim korisničkim imenom i lozinkom.
* *Zahtev za registraciju* - klijent može da registruje korisnika u aplikaciju, sa prosleđenim parametrima korisnika.
* *Zahtev za sve rezevacije -* ova metoda vraća sve rezervacije na koje je korisnik pozvan ili koje je on napravio.
* *Zahtev za brisanje tokena* –za svakog korisnika koji je ulogovan u aplikaciju na svom Andrid uređaju, čuva se token, preko kog se šalju obaveštenja pri rezervaciji. Klijent može da zatraži da se taj token obriše kad se korisnik odjavi sa aplikacije.
* *Zahtev za tip restorana* -klijentska aplikacija zahteva da vidi kog je tipa jedan restoran kad korisnik pregleda detalje restorana.
* *Zahtev za sve tipove* -kad se dodaje novi restoran, klijentska aplikacija zahteva od servera da se vrate svi tipovi restorana kako bi korisnik imao uvid u sve tipove i odabrao kod je tipa restoran.
* *Zahtev za sve restorane* - ova metoda vraća sve restorane, tj. one koji su registrovani u aplikaciji.
* *Zahtev za omiljene restorane -* svaki korisnik aplikacije ima listu omiljenih restorana. Kad korisnik pregleda listu omiljenih restorana, klijent šalje zahtev serveru preko ove metode za listu omiljenih restorana.
* *Zahtev za dodavanje restorana u omiljene* - svaki korisnik može da doda restoran u omiljene. Klijentska aplikacija šalje zahtev preko ove metode.
* *Zahtev za brisanje restorana iz omiljenih* - ova metoda nam ukljanja restoran iz liste omiljenih restorana.
* *Zahtev za jedan restoran­* - korisnik pregleda jedan restoran. Serveru se šalje zahtev preko ove metode za vraćanje jednog restorana.
* *Zahtev za brisanje restorana* -administrator aplikacije želi da obriše restoran iz aplikacije. Klijentska aplikacija šalje zahtev za brisanje serverskoj aplikaciji preko ove metode.
* *Zahtev za dodavanje restorana* - ova metoda nam dodaje restorane u aplikaciju kad klijent pozove ovu metodu na serveru.
* *Zahtev za izmenu restorana* -preko ove metode na serveru može da se izmeni restoran.
* *Zahtev za sve ocene restorana* -ova metoda nam vraća sve ocene za restoran koji je prosleđen.
* *Zahtev za izmenu ocene* -svaki korisnik aplikacije može da izmeni ocenu koji je dao restoranu. Kad se to desi poziva se ova metoda na server.
* *Zahtev za dodavanje ocene* -klijent šalje zahtev za dodavanje ocene restoranu preko ovo metode na serveru.

## 4.3 DIJAGRAM PAKETA

*Dijagram paketa* nam služi za prikazivanje arhitekture aplikacije. Na njemu su prikazane sve zavisnosti između paketa. Na slici 4.3.1. je prikazan dijagram paketa serverske aplikacije. U nastavku sledi opis dijagrama paketa.



Slika 4.3.1.-Dijagram paketa

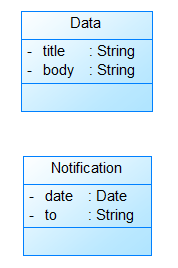
Klase koje se nalaze u paketu “controllers” koriste klase iz paketa “services”, kako što je prikazano na slici 4.3.1. Takođe, na slici se može videti da je paket “services” povezan još sa paketima “utils” i “repositories”, a paket “repositories” je povezan sa paketom “entities”. Klase koje komuniciraju su klase paketa koji su međusobno povezani.

## 4.4 DIJAGRAM KLASA

Dijagram klasa aplikacije je podeljen po paketima. U ovom poglavlju ćemo opisati i prikazati sve dijagrame klasa. Serverska strana aplikacije za rezervaciju restorana je grupisana u pet paketa. Za svaki paket je prikazan dijagram klasa:

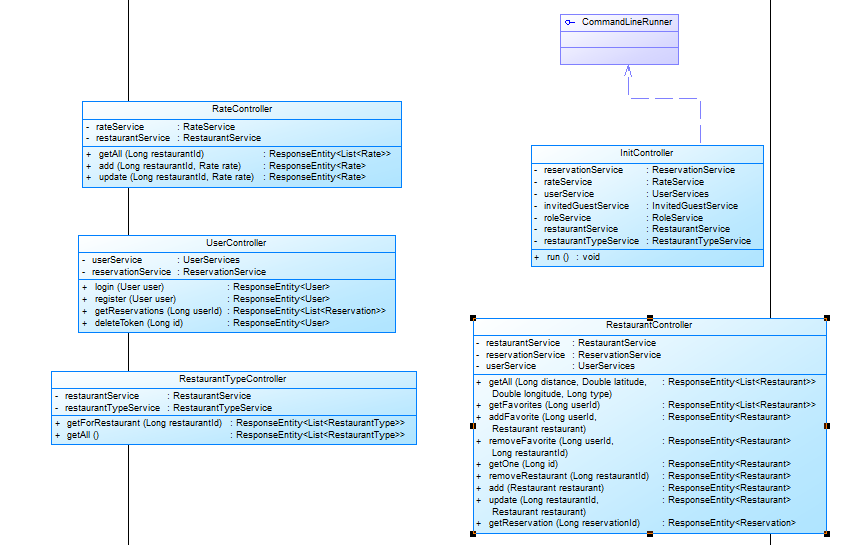
* Dijagram klasa za paket *utils*
* Dijagram klasa za paket *controllers,*
* Dijagram klasa za paket *entities,*
* Dijagram klasa za paket *services*
* Dijagram klasa za paket *repository*

U paketu *repository* se nalaze interfejsi za komunikaciju sa bazom podataka. Svi interfejsi nasleđuju interfejs ***JpaRepository.***



Na slici 4.4.1. je prikazan dijagram klasa za paket “utils”. Na ovom dijagramu vidimo da imamo dve klase *Data* i *Notification.* U klasi *Data* postoje dva parematra *title* i *body*, oba parameter su tipa ***String****.*Dok klasa *Notification* ima dva parametra, jedan je *date* tipa ***Date,*** a drugi je *to* tipa ***String.***

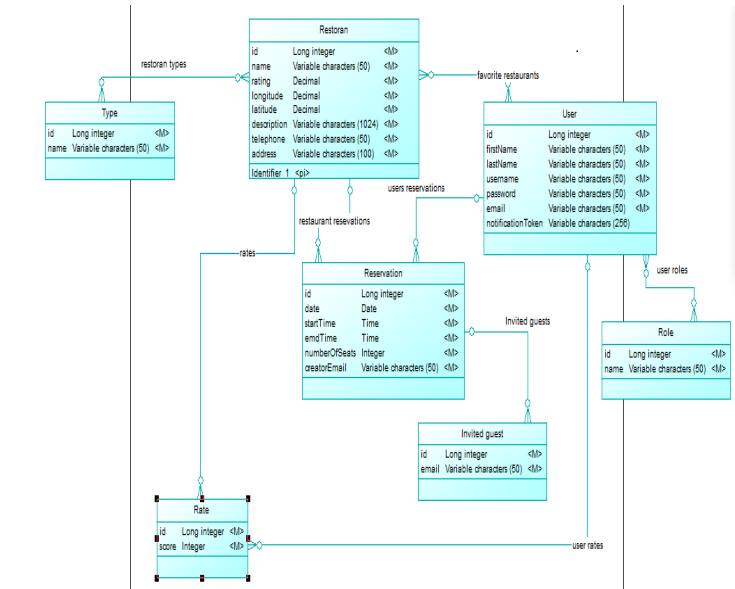
Slika 4.4.1. -Dijagram klasa za paket utils



Slika 4.4.2.-Dijagram klasa za paket controllers

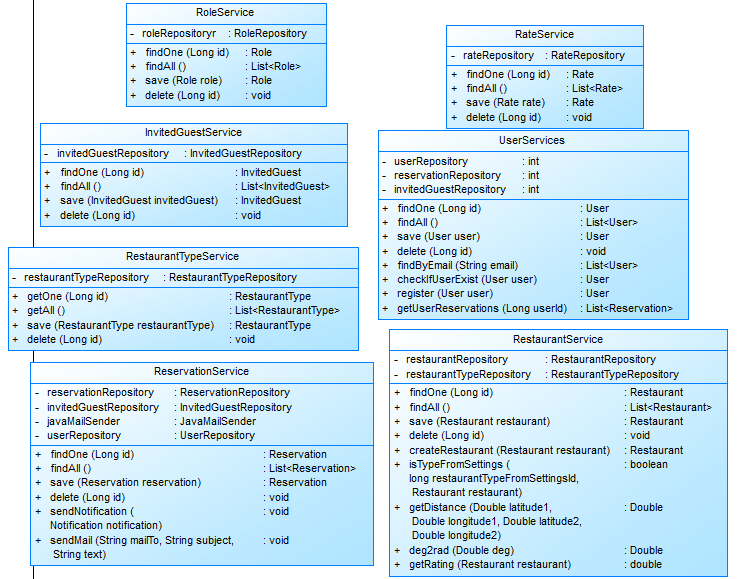
Na slici 4.4.2. je prikazan dijagram klasa za paket *conttrollers*. Na dijagramu se mogu videti klase: ***InitContorller, RateController, UserController, RestaurantController i RestaurantTypeContoller.*** U svakoj klasi se vide parametri koje poseduje klasa, sve metode klase, parametri metoda i povratne vrednosti svake metode u klasi. ***InitController*** nasleđuje interfejs *CommanLineRuner.*

Slika 4.4.3. -Dijagram klasa za paket entities



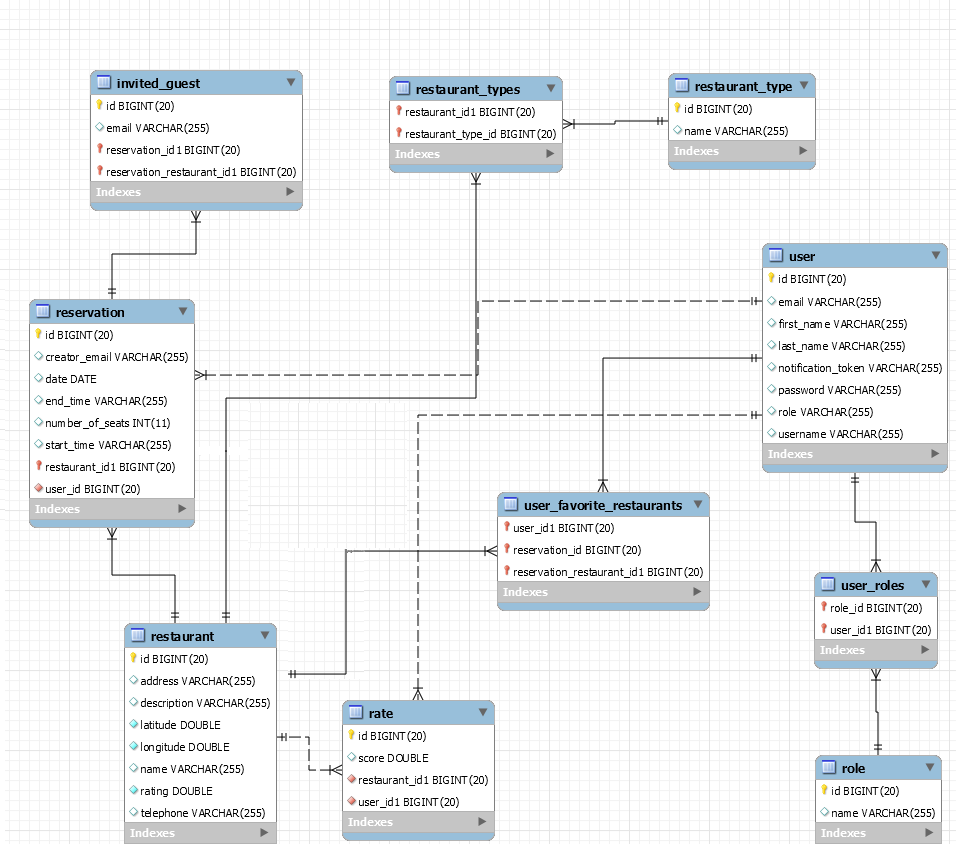
Na slici 4.4.3. je prikazan er-dijagram za paket *entities.* Na slici se vidi da ima sedam klasa. ***Restaurant, Type, User, Reservation, Role, InvitedGuest i Rate.*** Na slici se vide klase sa svim parametrima koje poseduju. Klase su uvezane međusobno, odnosno, svaka klasa ne sadrže samo proste tipove podataka, nego i tipove koji su generisani u aplikaciji. Klasa ***Restaurant*** sadrži listu klase ***Type.*** Preko ovih klasa se definiše fizički model u bazi podataka, što ce biti opisano u ovom poglavlju.

Slika 4.4.4.- Dijagram klasa za paket services



Na slici 4.4.4. je prikazan dijagram klasa iz paketa *services.* U ovom paketu ima sedam klasa, a to su: ***RoleService, RateService, InviteGuestService, UserService, RestaurantTypeService, RestauranService*** i ***ReservationService.*** Svaka klasa ima parametre koje koristi. Ove klase koriste interfejse iz pakata *repository,* pa svi parametri u klasi su iz tog paketa. Svaka klasa ima osnovne metode, kao što su: save, delete, getAll i getOne. Pored toga su implemetirane još neke metode, koje se koriste pri radu aplikacije. Metode su prikazane na slici sa ulaznim parametrima, kao i sa povratnim vrednostima.

## 4.5 FIZIČKI MODEL BAZE PODATAK



Slika 4.5.1.-Fizički model baze podataka

Na slici 4.5.1. Prikazan je fizički model baze podataka. Svaki tabela ima listu parametara i svoj jedinstveni primarni ključ, preko kog se indentifikuje. U bazi postoje sledeće tabele:

* *restaurant*
* *rate*
* *reservation*
* *user*\_*favorite*\_*restaurants*
* *role*
* *user\_roles*
* *user*
* *restaurant*\_*type*
* *restaurant*\_*types*
* *invited*\_*guest*

**Veze između entiteta**:

Veza između “*restaurant*” i “*restaurant\_type*” je ***“ManyToMany”***, što bi značilo da jedan restoran može da ima jedan ili više tipova i da jedan tip može da ima jedan ili više restorana. Kad se pravi veza više na više onda se pojavljuje među tabela, koja nam pamti jedinstvene primarne ključeve povezanih tabela. Na ovom primeru to je *“restaurant\_types”.* Restoran je još povezan sa rezervacijom, tu je veza “***OneToMany***”, što znači da svaki restoran ima jednu ili više rezervacija, a rezervacija ima tačno jedan restoran. Tabela “*rate*” je veza između tabela “*user*” i “*restaurant*”. Svaki korisnik daje ocenu za restoran. Svaki restoran ima više ocena i svaki korisnik može da oceni više restorana. Tabela “*user*” je vezana sa tabelom “*role*” vezom “***ManyToMany***”. Svaki korisnik ima jednu ili više uloga, a svaka uloga ima jednog ili više korisnika, povezna tabela “user\_roles”. Korisnik je povezan još sa rezervacijom, veza je “***OneToMany***”. Svaki korisnik ima jednu ili više rezervacija, a rezervacija ima tačno jednog korisnika. Tabela “*user*” je vezana i sa tabelom “*user\_favorite\_restaurants*”, a ovo je povezana tabela između korisnika i restorana. Svaki korisnik ima jedan ili više omiljenih restorana. Veza između rezervacije i liste prijatelja je takva da svaka rezervacija ima jednog ili više pozvanih prijatelja, dok pozvani prijatelj ima samo jednu rezervaciju.

# 5. IMPLEMENTACIJA APLIKACIJE

U poglavlju četiri, detaljno je opisana arhitektura aplikacije za rezervaciju restorana. U ovom poglavlju biće prikazana implementacija ključnih klasa, prikazan kod aplikacije uz detaljan opis šta taj kod radi i na koji račin taj kod funkcioniše.

## 5.1 SPRING BOOT

Svaka *Spring Boot* aplikacija sastoji se iz tri sloja. U poglavlju tri, je predstavljen osnovni concept *Spring Boot* boblioteke, a u ovom poglavlju. Kroz sva tri sloja aplikacije se prolazi na primeru *Restaurant.* Prvi sloj aplikacije su *controleri.* Na listingu 5.1.1 prikazana je metoda za dodavanje restorana.

@PostMapping("/restaurants")  
public ResponseEntity<Restaurant> add(@RequestBody Restaurant restaurant) throws Exception{  
 Restaurant createdRestaurant = restaurantService.createRestaurant(restaurant);  
 return new ResponseEntity<>(createdRestaurant, HttpStatus.*OK*);  
}

Listing 5.1.1.-Metoda za dodavanje restorana u kontrolerima

Na listingu 5.1.1. se može videti da je metoda za dodavanje restorana anotirana sa anotacijom *postMapping* koja u sebi sadrži parametar “/restaurants”. Da bi bio ispoštovan ***REST*** protokol, odabrana je anotacija *POST.* Paremetar u ovoj anotaciji predstavlja putanju do ove metode u kontroleru pomoću koje klijent pristupa željenom resursu. Metoda prima model *restaurant* koji klijent šalje putem HTTP protokola u JSON format, koji se nalazi u telu post zahteva. Kada zahtev stigne na server, ovaj JSON fajl se pomoću *objectMapera* mapira na model Restaurant. Kad se uspešno primi zahtev, ovaj model se prosleđuje servisu, odnosno drugom sloju aplikacije, koji je prikazan na listingu 5.1.2.

public Restaurant createRestaurant(Restaurant restaurant) {  
 String address = restaurant.getAddress() + ", Srbija";  
 GeoApiContext context = new GeoApiContext.Builder()  
 .apiKey("AIzaSyA74dmDSIsfMCsaRN0wZkIu\_qwfDBvIcHQ")  
 .build();  
 try {  
 GeocodingResult[] results = GeocodingApi.*geocode*(context, address).await();  
 Gson gson = new GsonBuilder().setPrettyPrinting().create();  
 restaurant.setLatitude(Double.*parseDouble*(gson.toJson(results[0].geometry.location.lat)));  
 restaurant.setLongitude(Double.*parseDouble*(gson.toJson(results[0].geometry.location.lng)));  
 }catch (Exception e){  
 System.*err*.println("GEO COORDINATES ERROR : ");  
 e.printStackTrace();  
 }  
 List<RestaurantType> restaurantTypes = restaurant.getTypes();  
 restaurant.setTypes(new ArrayList<>());  
 restaurant = restaurantRepository.save(restaurant);  
 restaurant.setTypes(restaurantTypes);  
 restaurant = restaurantRepository.save(restaurant);  
 return restaurant;  
}

Listing 5.1.2.-Metoda za dodavanje restorana u servisima

Kao što je navedeno u poglavlju tri, servis sloj služi za biznis logiku aplikacije. A na ovom primeru kreiranja restorana, biznis logika predstavlja dobavljanje koordinata za prikaz restorana na mapi. Koristimo GoogleAPI za geo-koordinate. Pored toga koristi se ključ koji smo generisali za korišćenje API servisa. Nakon što se uspešno dobave koordinate setujemo na model Restaurant. Zatim ovako kreiran model se prosleđuje sledećem sloju aplikacije, koji ja zadužen za snimanje modela u bazu podataka. Ovaj sloj vodi računa o komunikaciji sa bazom podataka. Zadatak ovog sloja je i da napravi SQL naredbe koje je neophodno izvršiti na strani baze, kako bi se željena aktivnost odradila. Ovaj sloj je prikazan na listingu 5.1.3.

public interface RestaurantRepository extends JpaRepository<Restaurant, Long> {  
}

Listing 5.1.3.-Sloj repozitoriju

Kao što smo prikazali metodu za dodavanje restorana, postoje i metode za izmenu, brisanje i dobavljanje restorana. Ove metode prestavljaju osnovne CRUD operacije nad restoranom.

## 5.2 FIREBASE

Androidaplikacija za rezervaciju restorana ima svoj deo serverske aplikacije. Serverska aplikacija mora da komunicira se ***Firebase*** serverom i da razmenjuje poruke sa istim. Da bi aplikacija koristila ***Firebase*** potrebno je dodati u *gradle-u* dve biblioteke prikazano na listingu 5.2.1.

implementation 'com.google.firebase:firebase-core:16.0.0'  
implementation 'com.google.firebase:firebase-messaging:17.0.0'

*Listing 5.2.1.- Biblioteke za Firebase u gradle*

Sa servera ***Firebase*** koristi se ***Cloud Messaging*** preko kog se šalju obaveštenja serveru aplikacije. Potrebno je dodati ***Cloud Messaging*** u android aplikaciju, tako što se izgeneriše JSON fajl sa ključevima napravljenim na sajtu *Firebase*. JSON fajl sa ključevima prikazan je na listingu 5.2.2.

{  
 "project\_info": {  
 "project\_number": "918312886642",  
 "firebase\_url": "https://restaurants-64179.firebaseio.com",  
 "project\_id": "restaurants-64179",  
 "storage\_bucket": "restaurants-64179.appspot.com"  
 },  
 "client": [  
 {  
 "client\_info": {  
 "mobilesdk\_app\_id": "1:918312886642:android:e78b783de25f449b",  
 "android\_client\_info": {  
 "package\_name": "com.mutacije.restaurant"  
 }  
 },  
 "oauth\_client": [  
 {  
 "client\_id": "918312886642-g16c3k34nq5pt03oncuvivkfng4rge6i.apps.googleusercontent.com",  
 "client\_type": 3  
 }  
 ],  
 "api\_key": [  
 {  
 "current\_key": "AIzaSyB18Lijx\_mnUGWe8ts4njIWWcVxh1AV4J8"  
 }  
 ],  
 "services": {  
 "analytics\_service": {  
 "status": 1  
 },  
 "appinvite\_service": {  
 "status": 1,  
 "other\_platform\_oauth\_client": []  
 },  
 "ads\_service": {  
 "status": 2  
 }  
 }  
 }  
 ],  
 "configuration\_version": "1"  
}

*Listing 5.2.2.- JSON file za Firebase sa ključevima*

Kada se aplikacija pokrene na uređaju za korisnika koji se registrovao u aplikaciju, čuva se token, preko koga se razmenjuju poruke. Korisnik dobija obaveštenja na uređaj na kom se poslednji put ulogovao i pamti se token od tog uređaja. Token se dobija tako što se na *FireBaseInstanceId* pozove metoda *getInstane* i na ovu metodu se pozove *getToken,* prikazano na listingu 5.2.3.

@Override  
public void onTokenRefresh() {  
 String newToken = FirebaseInstanceId.*getInstance*().getToken();  
 L.*e*("NEW TOKEN : " + newToken);  
}

*Listing 5.2.3.- Metoda koja na vraća token*

Potrebno je da se implementira metoda za primanje poruka klase *MyFirebaseMessagingService* koja nasleđuje klasu ***FirebaseMessagingService***. Pored toga je potrebno i da se redefiniše metoda iz te klase koja se zove ***onMessageRecived***.Ova metoda je prikazana na listingu 5.2.4.

@Override  
public void onMessageReceived(RemoteMessage remoteMessage) {  
 L.*e*("Message received");  
 String title = remoteMessage.getData().get(*MESSAGE\_TITLE*);  
 String body = remoteMessage.getData().get(*MESSAGE\_BODY*);  
  
 Intent intent = new Intent(this, DrawerActivity.class);  
 intent.putExtra(DrawerActivity.*EXTRA\_INVITATION*, true);  
   
 intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(this, 0, intent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this, *CHANNEL\_ID*)  
 .setSmallIcon(R.drawable.*ic\_group\_black*)  
 .setContentTitle(title)  
 .setContentText(body)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*)  
 .setContentIntent(pendingIntent)  
 .setAutoCancel(true);  
  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(this);  
 notificationManager.notify((new Random()).nextInt(), builder.build());  
  
 L.*e*(remoteMessage.getData().toString());  
}

*Listing 5.2.4.- Metoda za primanje obaveštenja*

U aplikacije se koristi *Log* za ispisivanje poruka iz paketa *android.util.log.* Poruka ima dva dela ***title*** i ***body***.Na početku se pravi intent objekta, koji definiše koja stranica aplikacije će da se otvori, kad stigne obaveštenje. Za pravljenje notifikacije koristi se *NotificationBilder* i u njemu se setuju parametri koji su potrebni za pravljenje notifikacije. Kad se pravi rezervacija za određeni restoran, klijent pošalje zahtev serverskoj aplikaciji , server šalje poruku ***Firebase***-u, a on šalje poruku svim korisnicima koji su označeni u rezervaciji, preko korisničkih tokena. Kada korisnik primi poruku od ***Firebase*** koristi se ova metoda i tu se kreira obaveštenje. Server pristupa ***Firebase***  serveru preko API-a, koji je prikazan na listingu 5.2.5, i potrebno je poslati ključ u *header* elementu, koji nam je izgnerisao firebase kad smo dodali projekat na njihovom sajtu.

String url = "https://fcm.googleapis.com/fcm/send";

*Listing 5.2.5..- API za pristum serveru Firebase*

Kad klijent aplikaacije pošalje serveru zahtev za notifikaciju, na serveru se poziva metoda *sendNotification,* koja komunicira sa *FireBase,* prikazana je ne listingu 5.2.5.

private void sendNotification(Notification notification){  
 RestTemplate restTemplate = new RestTemplate();  
 String requestJson = "";  
 String url = "https://fcm.googleapis.com/fcm/send";  
 HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.APPLICATION\_JSON);  
 headers.set("Authorization","key=AAAA1c-2oXI:APA91bGRYHTwIFtFsdEh3FRM64cZ8BJmqr69syTusE-60O9Wsc0bzwHUKPVeT1VZCuAsv2T85j5A2-pSj4WLxREV7FeB8G4SdGBUkhAHQvCZvbf\_7kzjItAka4cYHPkTczXRFFah7fmOtgQL2T4KVRKQHBfngeWWOg");  
 try{  
 requestJson = (new ObjectMapper()).writeValueAsString(notification);  
 }catch (Exception e){  
 System.err.println("FAILED TO CREATE JSON REQUEST");  
 e.printStackTrace();  
 }  
 HttpEntity<String> entity = new HttpEntity<String>(requestJson,headers);  
 restTemplate.postForObject(url, entity, String.class);  
 System.err.println("POST SENT FOR NOTIFICATION");  
}

*Listing 5.2.6..- Metoda za komunikaciju sa FireBase*

Metoda prima parameter Notifikacija, potrebno je notifikaciju prebaciti u *json* fajl, kako bi se preosledila *firebase* serveru. Potrebno je setovati kljuc u *header* elemenu, za autorizaciju.

## 5.3 GOOGLE MAPS

*Google maps Api* pruža korištenje *Google Maps* servisa na *Androd* platformi, usluge koje pruža servis su: *prikaz mape, rukovanje kamerom i postavljanje oznaka.* Da bi *Android* aplikacija mogla da koristi uslugeovog servisa, potrebno je uraditi sledeće korake:

* Preuzeti GooglePlay servis SDK. Na listingu 5.3.1. prikazana je biblioteka iz *gradle-a* preko koje se dobavlja navedeni SDK.

implementation 'com.google.android.gms:play-services-maps:15.0.1'

*Listing 5.3.1..- Biblioteka za Google Play service SDK*

* Sledeći korak jeste da se preuzme ključ za Google Maps API. Potrebno je na zvaničnom sajtu *Google Maps Platforame* genirisati ključ za pristup API-u.
* Potom je potrebno izgenerisani ključ uneti u *AndoridManifest.xml* fajl Andorid aplikacije. Prikazano je na listingu 5.3.1. Ključ se nalazi u string resursima.

android:name="com.google.android.geo.API\_KEY"  
android:value="@string/google\_maps\_key" />

*Listing 5.3.2..- Andorid manifest, ključ za pristup GoogleMaps*

* U Android aplikaciji se napravi *activity maps,* pomoću kog se vrši prikaz same mape. Na listing 5.3.3. prikazan je *layout* aktivnosti za prikaz mape.

<?xml version="1.0" encoding="utf-8"?>  
<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
xmlns:map="http://schemas.android.com/apk/res-auto"  
xmlns:tools="http://schemas.android.com/tools"  
android:id="@+id/map"  
android:name="com.google.android.gms.maps.SupportMapFragment"  
android:layout\_width="match\_parent"  
android:layout\_height="match\_parent"  
tools:context=".activities.MapsActivity" />

*Listing 5.3.3..- Layout maps activity*

Kad se urade svi koraci, uspešno se integrisale mape zajedno sa aplikacijom. Prilikom prikaza mape, postoji mogućnost da se dodaju markeri na mapi, koji su korišćeni u aplikaci za prikaz svih resotrana. Kod za prikaz markera je prikazan na listingu 5.3.4.

for (Restaurant restaurant : this.restaurants) {  
 LatLng latLng = new LatLng(restaurant.getLatitude(), restaurant.getLongitude());  
 Marker marker = map.addMarker(new MarkerOptions().position(latLng).title(restaurant.getName()));  
 marker.setTag(restaurant);  
}

*Listing 5.3.4..- Dodavanje markera na mapu*

## 5.4 REST ARHITEKTURA

Androdiaplikacija komunicira sa serverskom aplikacijom putem *REST* protokola. U ovo poglavlju biće prikazano i objašnjeno pravljenje *rest* klijenta u Androidaplikaciji. Na listingu 5.4.1. prikazan je način na koji je kreiran *post* zahtev na klijentskoj strani.

public static void addRestaurant(Restaurant restaurant, Callback callback){  
 String restaurantJson = "";  
 try{  
 restaurantJson = (new ObjectMapper()).writeValueAsString(restaurant);  
 }catch (Exception e){  
 L.*e*("Restaurant Service add restaurant failed to serialize restaurant");  
 e.printStackTrace();  
 return;  
 }  
 RequestBody requestBody = RequestBody.*create*(MediaTypeConstants.*JSON*, restaurantJson);  
 Request request = new Request.Builder()  
 .url(Environment.*BASE\_URL* + "api/restaurants")  
 .post(requestBody)  
 .build();  
 OkHttpClientInstance.*getInstance*().newCall(request).enqueue(callback);  
}

*Listing 5.4.1..- Metoda za post zahtev*

Kad klijent šalje zahtev za dodavanje restorana u Android aplikaciju, metoda sa listinga 5.4.1, šalje zahtev serverskoj aplikaciji. Na početku metode se kreira model *Restaurant* u obliku JSON formata. Prebacivanje iz objektnog modela u JSON se vrši preko *ObjectMapera.* Nakon što se uspešno kreira JSON, potrebno je da se napravi zahtev. Da bi se kreirao zahtev potrebno je prvo napraviti *body* element HTTP zahteva. Prilikom kreiranja tela zaglavlja potrebno je navesti u kom format se nalaze podaci koji se stavljaju u telu zaglavlja. Zatim se kreira zahtev. Da bi zahtev bio uspešno kreiran potrebno je postaviti vrednost za *url,* kako bi se znalo na koju adresu je zahtev upućen. Nakon toga definišemo tip zahteva kroz koji se proseđuje prethodno napravljen body. Na samom kraju metode poziva se ***OkHttpClient*** koji predstavlja klijenta za server aplikacije i preko njega se šalje zahtev serveru. Slanje zahteva se radi asihrno, što bi značilo da se u pozadini aplikacije pravi nova nit, sa koje se šalje zahtev, a glavna nit nastavlja dalje izvršavanje aplikacije. Metoda ima drugi parametar *callback*. Ovaj parametar služi kad se zahtev izvršava na drugoj niti, da se dobije povratna informaciju od servera. U zavisnosti od uspešnosti povratne informacije, treba da se obradi zahtev na klijentskoj strani.

Ovim je pokazan *POST* zahtev. Postoje još tri vrste zahteva, a to su: *PUT, GET, DELETE. Put* ja isti kao i prikazani *post* zahtev. Na listingu 5.4.2. prikazan je *get* zahtev.

public static void getOne(Long restaurantId, Callback callback){  
 Request request = new Request.Builder()  
 .url(Environment.*BASE\_URL* + "/api/restaurants/" + restaurantId)  
 .build();  
 OkHttpClientInstance.*getInstance*().newCall(request).enqueue(callback);  
}

*Listing 5.4.2..- Metoda za get zahtev*

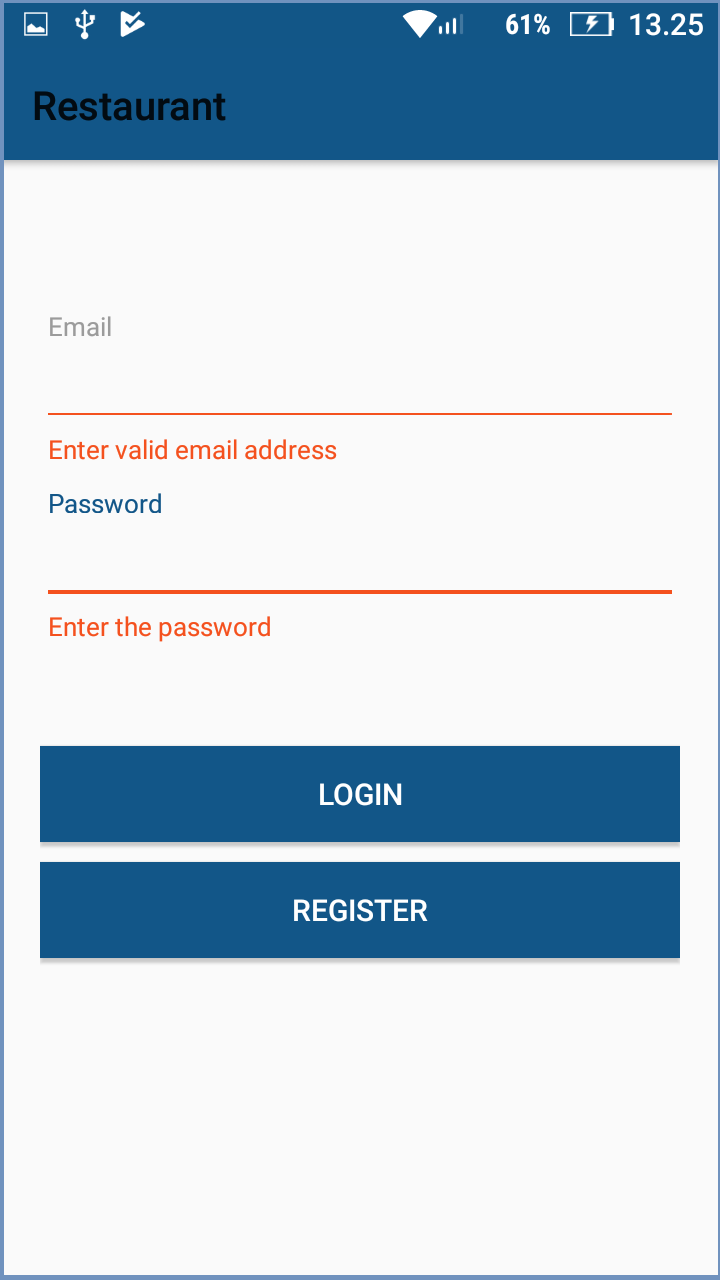
U ovoj metodi nije potrebno definisati body, jer za metodu je potrebno uspešno kreirati link sa parametrom, preko kog dobavljamo restoran sa servera. Isto se pravi klijent i za operaciju *delete.*

# 6. PRIKAZ RADA APLIKACIJE

Do sad je bila obrađena arhitektura i implementacija same aplikacije u *Android* i *Spring Boot.* U ovom poglavlju biće prikazan rad aplikacije kroz prolazak kroz sve ekranse forme aplikacije. U aplikaciji postoje uloge: *administratora* i *korisnika*, i u zavisnosti od toga ko je ulogovan određene *Activity* komponente se prikazuju.

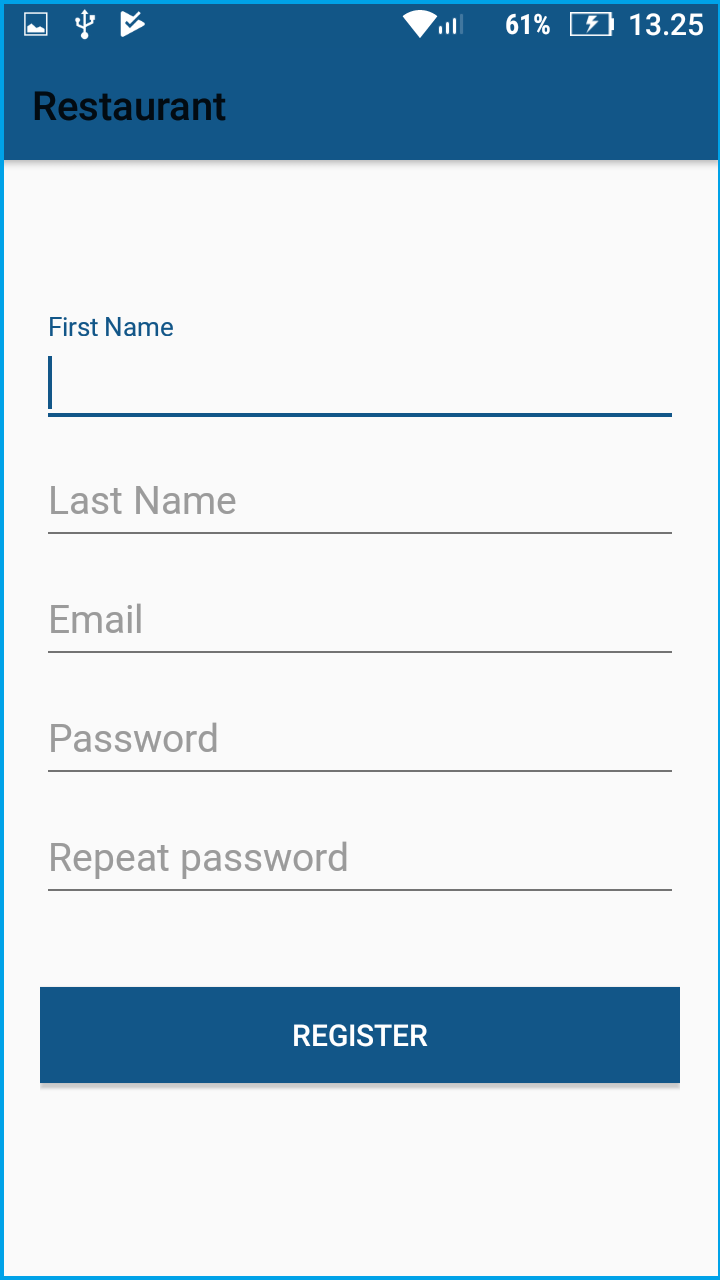
Na početku su opisane početne *Activity* komponente koje služe za *login* i *registraciju,* a u drugom delu su opisane *Activity* komponente za admina. U poslednjem delu su opisane sve komponente za korisnika.

## 6.1 PRIKAZ POČETNIH ACTIVITY KOMPONENTI



Na slici 6.1.1. prikazana je forma koja se dobija kad se pokrene aplikacija. Korisnik treba da unese svoje *e-mail* i *password* i da pritisne dugme *login* kako bi se ulogovao u aplikaciju. Ukoliko nema nalog treba da pritisne dugme *register* i onda se otvara forma za registraciju. Ta forma je prikazana na slici 6.1.2.

Slika 6.1.1- Početna forma aplikacije

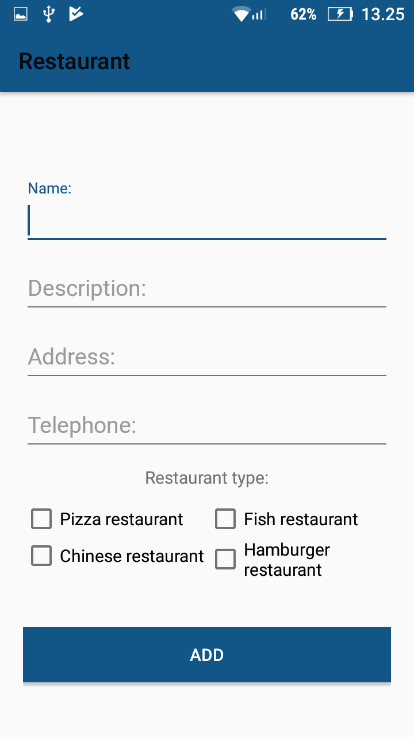
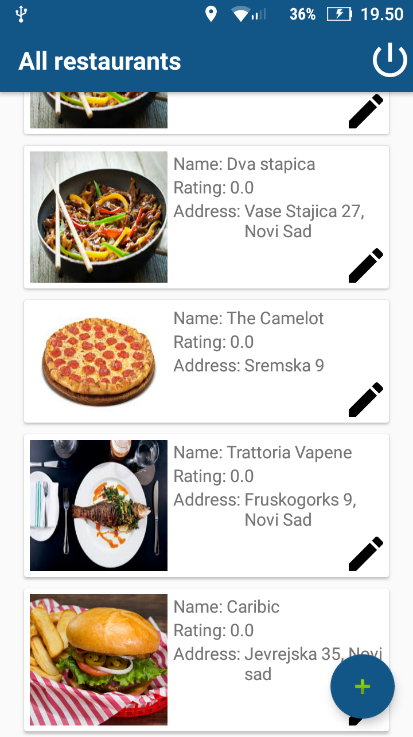


Na slici 6.1.2 prikazana je forma za registraciju. Korisnik treba da unese svoje podatke i *password*, kao i da ponovi *password.* Pritiskom na dugme *register*, korisnik se registruje u aplikaciju i vraća se na formu sa slike 6.1.1.

Slika 6.1.2. – Registraciona forma aplikacije

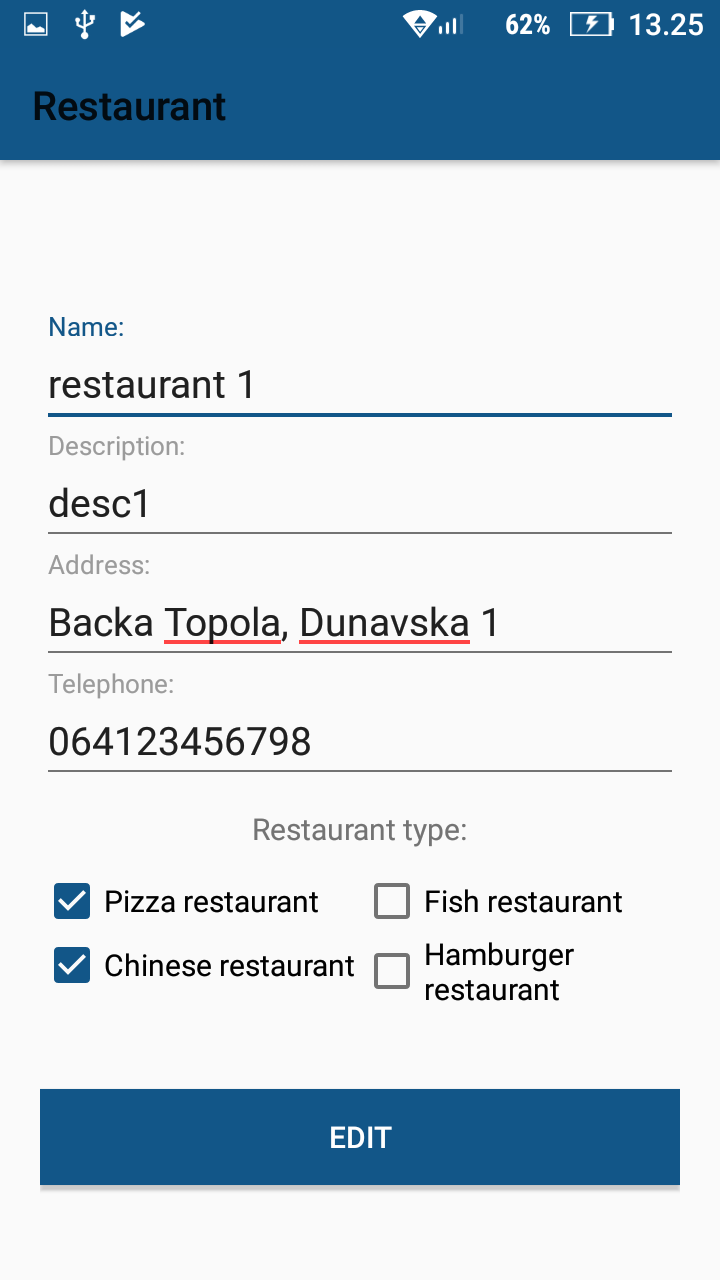
## 6.2 PERSPEKTIVA ADMINISTRATORA

Na slici 6.2.1. je prikazana početna stranica kad se administrator uloguje. Vidi se prikaz svih restorana koji su registrovani u aplikaciji. Vidi se slika restorana, u zavisnosti kom tipu pripada, ime restorana, ocena i adresa. Pritiskom na taster “plus”, otvara se forma za dodavanje restorana. Forma je prikazana na slici 6.2.2. Pritiskom na olovku možemo da izmenimo restoran. Tada se otvara forma za izmenu restorana, slika 6.2.3. Ako pritisnemo na restoran, otvara nam se forma za detaljan prikaz restorana kao na slici 6.2.4. Pritiskom na dugme u desnom uglu, korisnik se izloguje iz aplikacije i vraća se na formu sa slike 6.1.1.



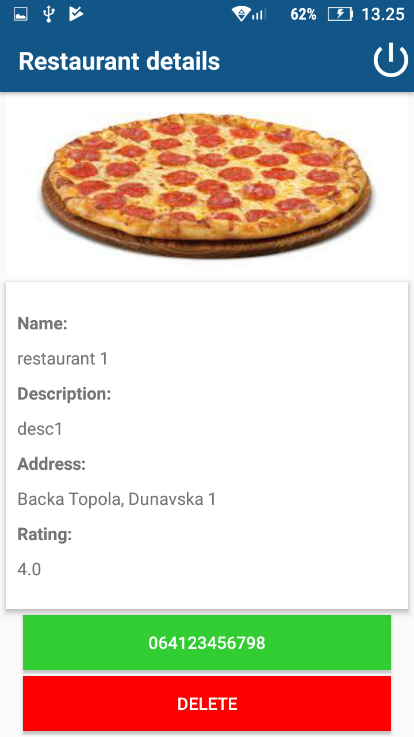
Slika 6.2.1. -Administrator home Slika 6.2.2.- Forma za dodavanje restorana

Slika 6.2.2. je forma za dodavanje novog restorana. Tu je potrebno popuniti sva polja, odabrati tip restorana i potom pritisnutni dugme “ADD”. Dodaje se restoran i otvara se forma “Admin home” slika 6.2.1.



Slika 6.2.3. je forma za izmenu postojećeg restorana. Prikazani su svi podaci koji su trenutno sačuvani. Izmene podataka se vrše pritiskom na određeno polje. Pritiskom na dugme „EDIT“, restoran će se izmeniti. Potom se otvara forma „Admin home“ slika 6.2.1.

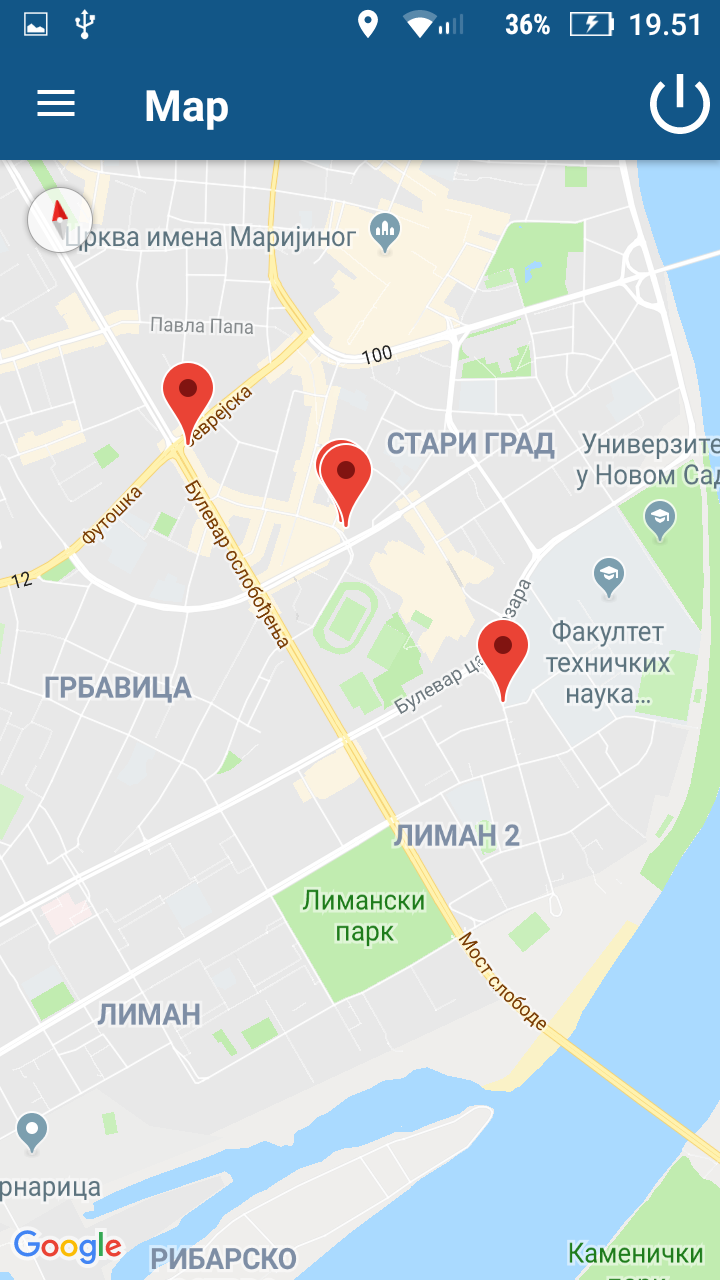
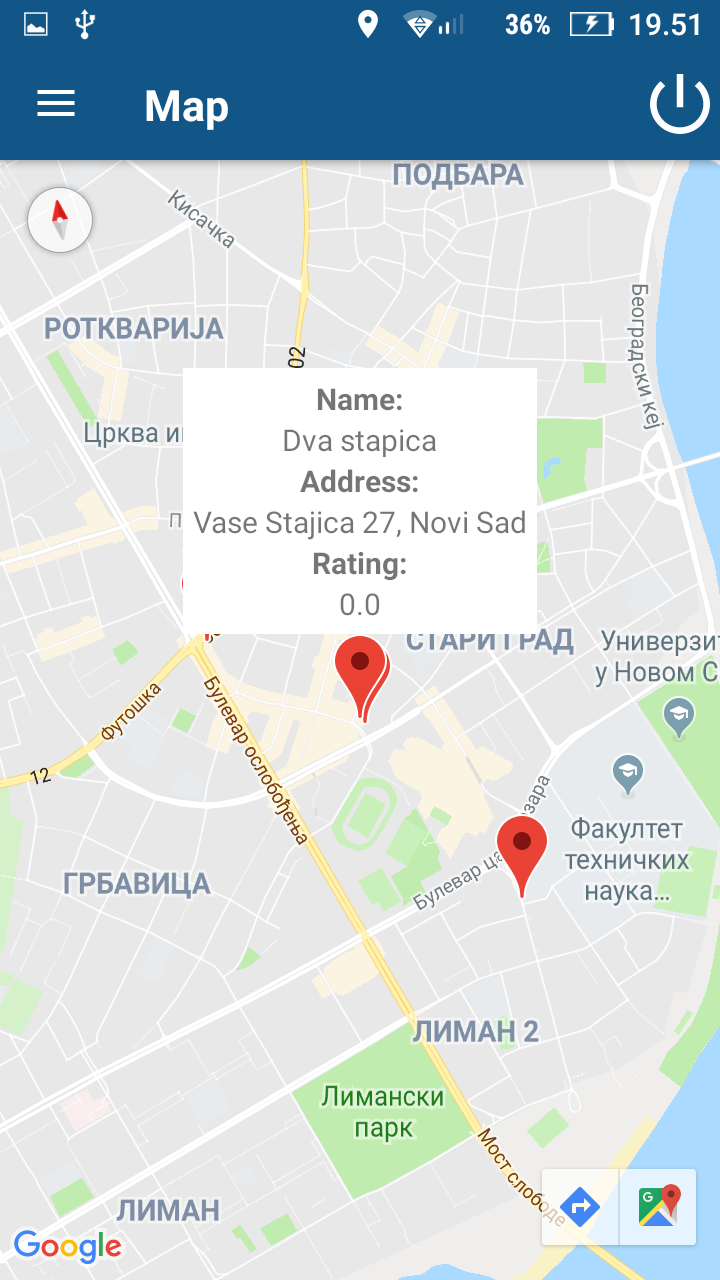
Slika 6.2.3. –Forma za izmenu restorana

**Slika 6.2.4. je forma ze detaljan prikaz restorana. U ovoj formi su prikazani svi podaci o restoranu, ime, opis, adresa i ocena. Ako se klikne na zeleno dugme sa brojem telefona, poziva se restoran. Kad se klinke na dugme „DELETE“ restoran se briše iz aplikacije i otvara se forma „Administrator home“, slika 6.2.1.

Slika 6.2.4. –Forma za prikaz restorana

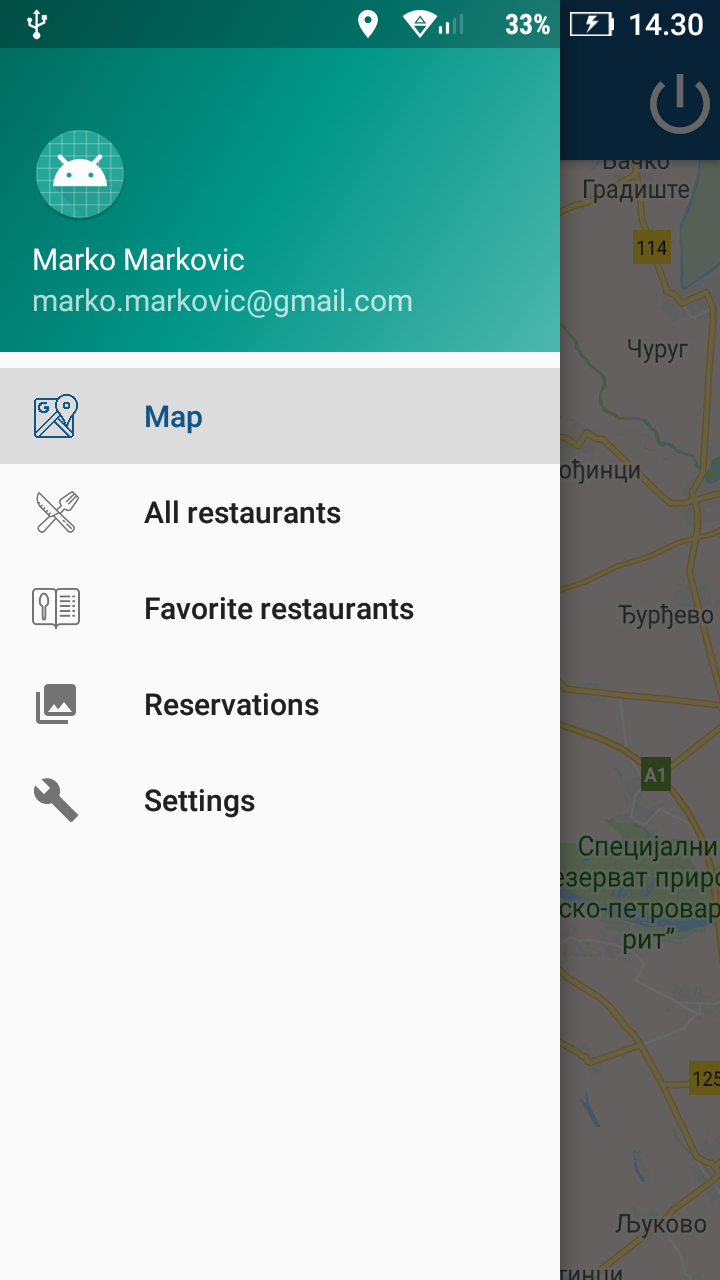
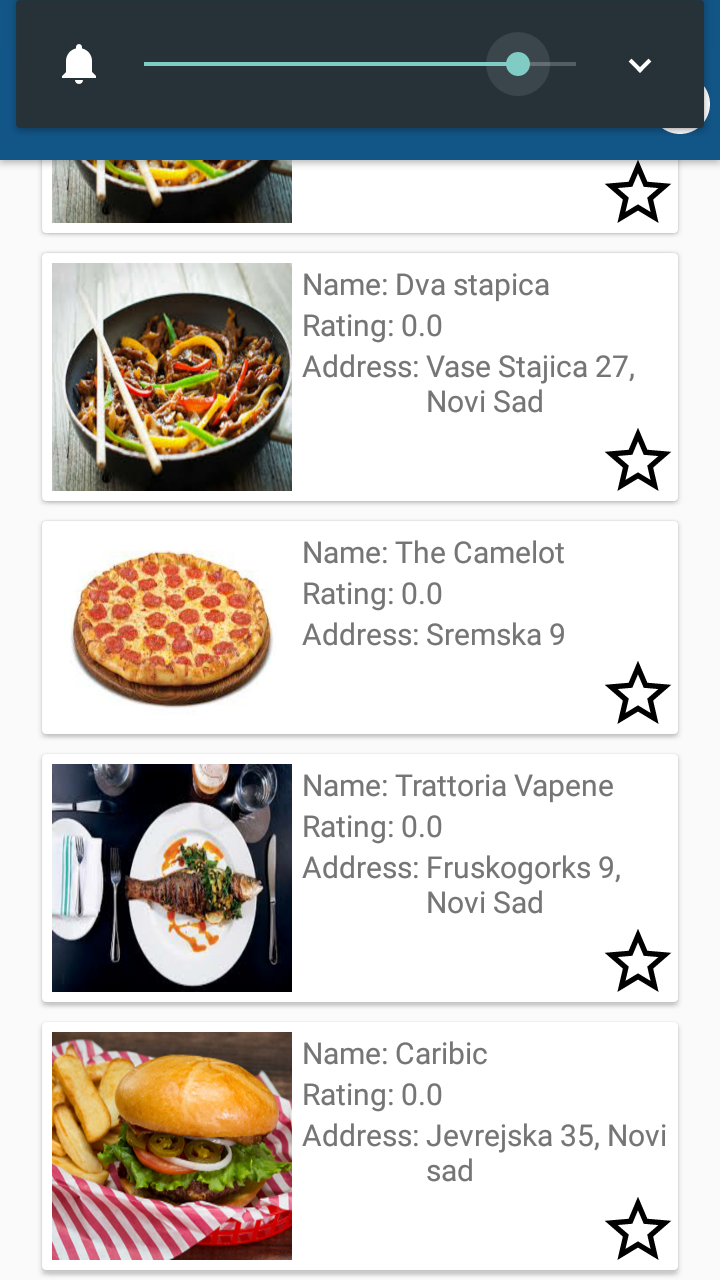
## 6.3 PERSPEKTIVA KORISNIKA

Slika 6.3.1. je forma “Korisnik home”. Tu se prikazuju svi restorani na mapi. Crvenim “pinovima” su označeni restorani. Kade se klikne na “pin” prikažu se informacije kao na slici 6.3.2. Pritiskom na dugme u desnom gornjem ćošku, izlogujemo se iz aplikacije i vraćamo se na stranicu logovanje (slika 6.1.1). Pritiskom na strelice sa leve strane ili prevlačenjem prsta sa leva na desno pojavljuje se “navigation drawer” kao na slici 6.3.3.



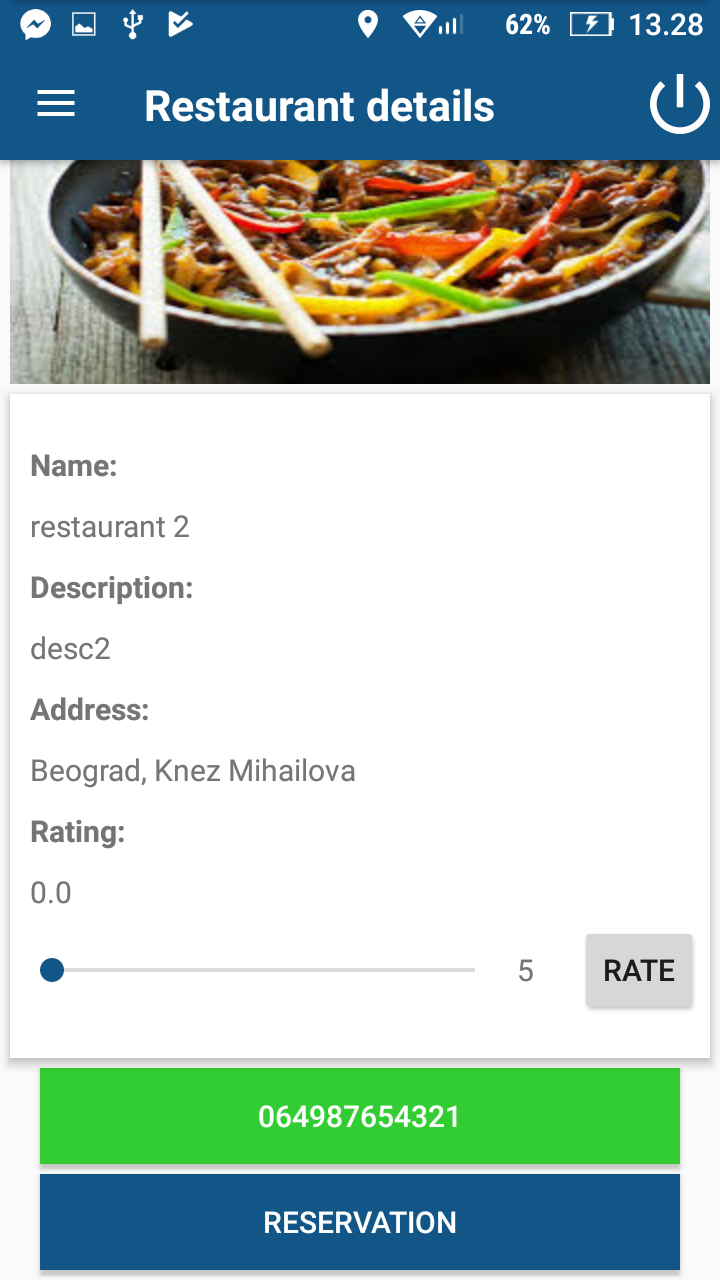
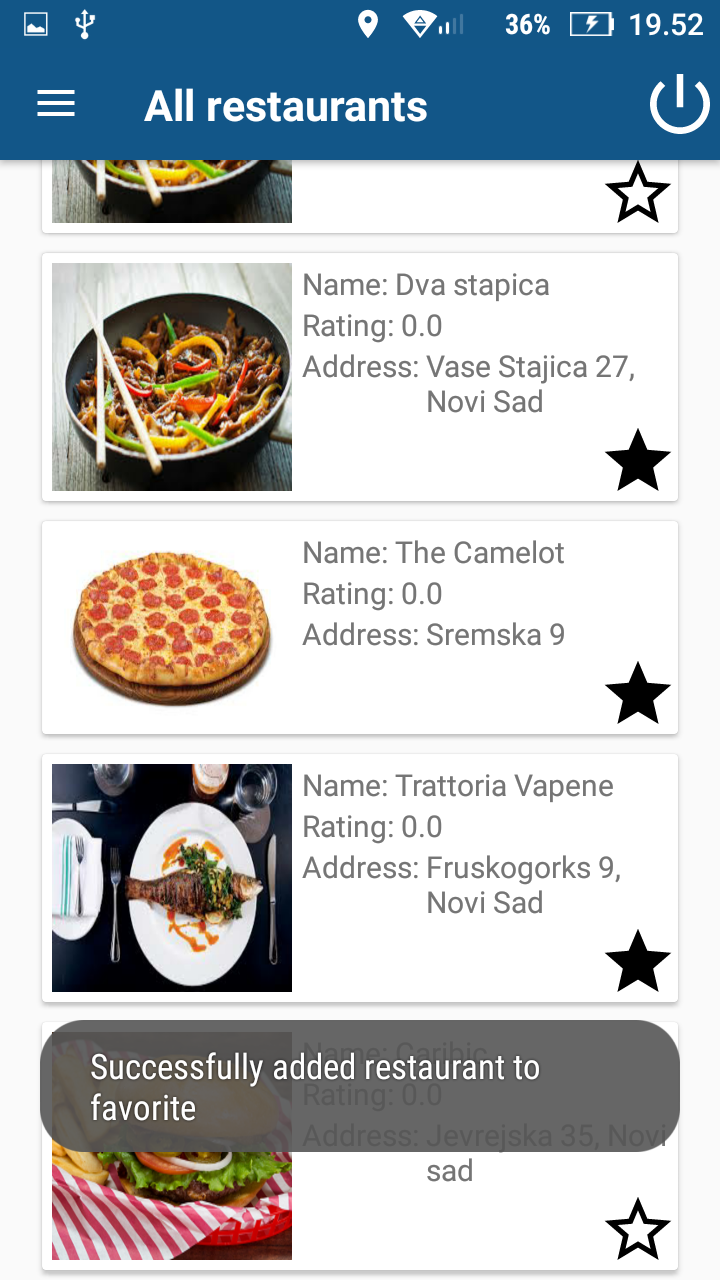
*Slika 6.3.1.-Forma Korisnik home Slika 6.3.2.-Forma sa opisom*

Slika 6.3.3. je forma “navigation drower” služi za navigaciju pomoću koje se korisnik kreće kroz aplikaciju. Forma sadrži ime, prezime i e-mail korisnika. Kade se klikne na “Map” otvara se forma “korisnik home”, kao na slici 6.3.1. Kade se klikne na “All restaurants” otvara se forma “all restaurants” kao na slici 6.3.4. Pritiskom na “favorite restaurants” prelazimo na formu “favorite restaurants” kao na slici 6.3.8. Pritiskom na taster “reservations” prelazimo na formu “reservations” slika 6.3.9. Pritiskom na tester “settings” prelazimo na formu “settings” prikazanu na slici 6.3.10. Kad korisnik prevuče prstom sa desna na levo, otvara se forma “Korisnik home”, slika 6.3.1.



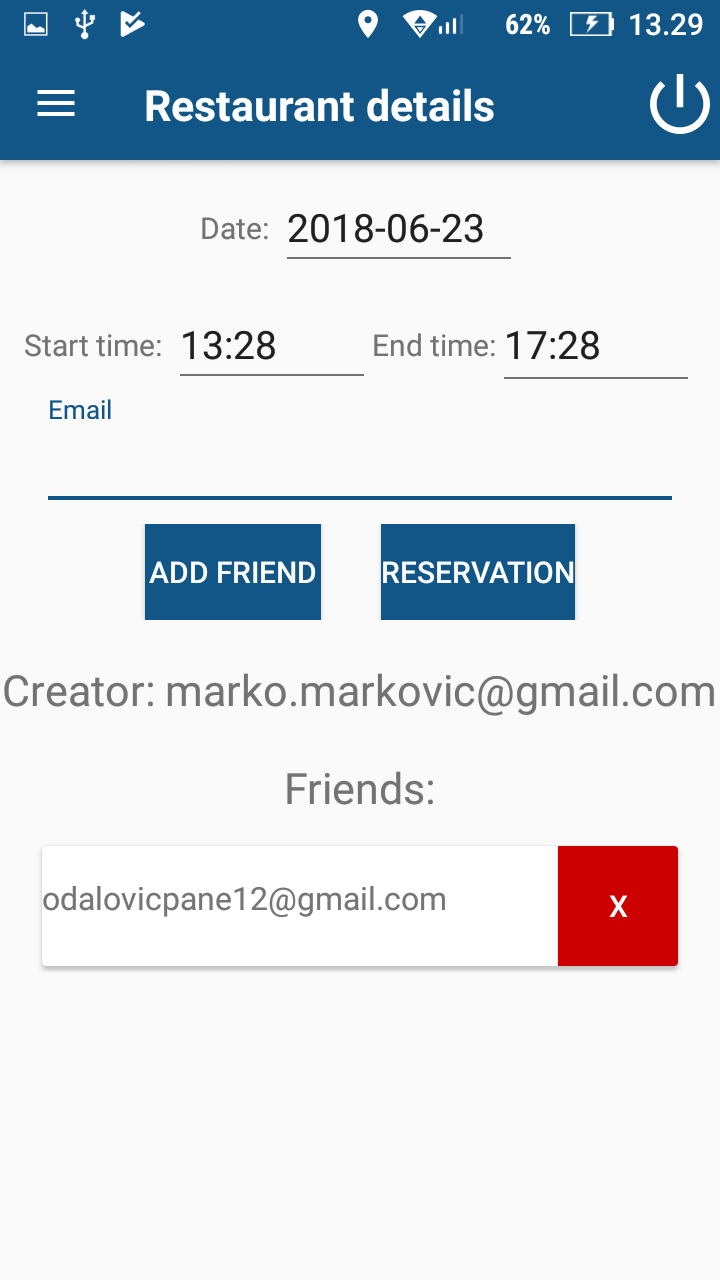
Slika 6.3.3.-Forma “navigation drower” Slika 6.3.4.-Forma “all restaurants”

Slika 6.3.4. je forma za prikazivanje svih restorana gde su prikazani svi restorani koji su registrovani u aplikaciji. Kada korisnik klikne na zvezdu, dodaje restoran u svoje omiljene restorane, prikazano na slici 6.3.5. Kada korisnik klikne na restoran, prikazuje mu se detaljan opis restorana, slika 6.3.6.



Slika 6.3.5. Forma “all restaurants” dodavanje u omiljene

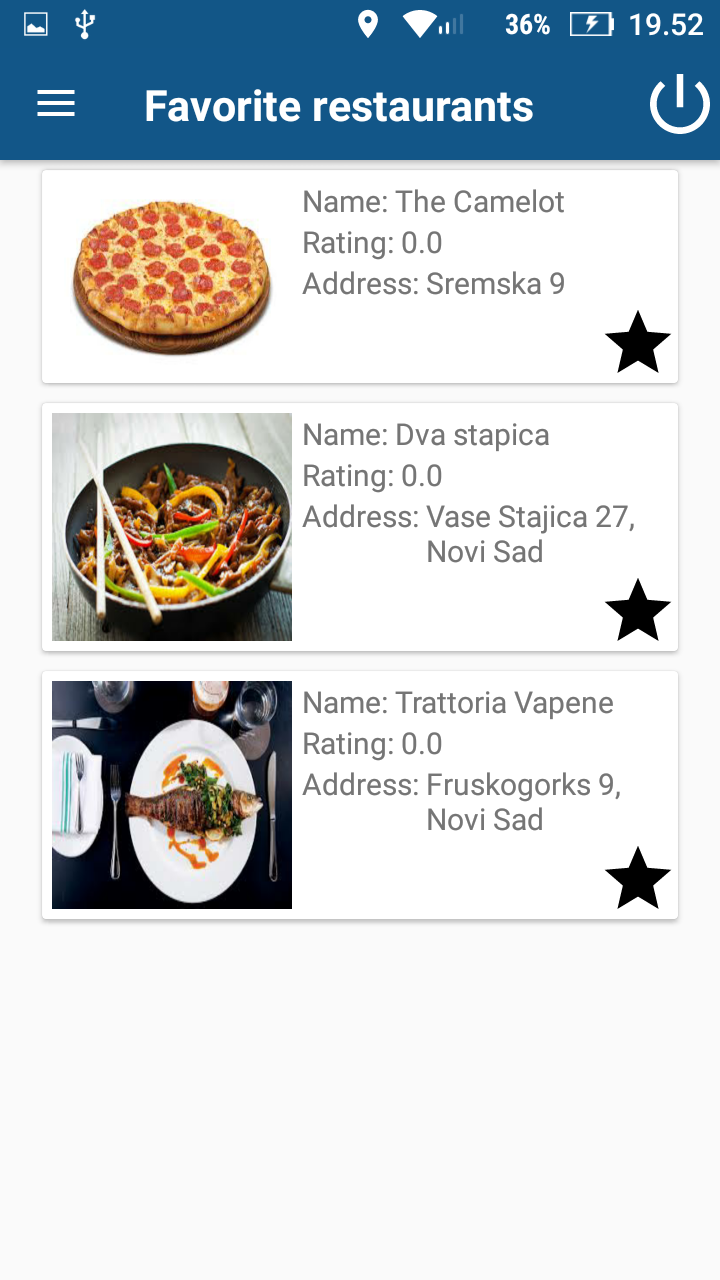
*Slika 6.3.6.-Forma detaljan prikaz restorana*



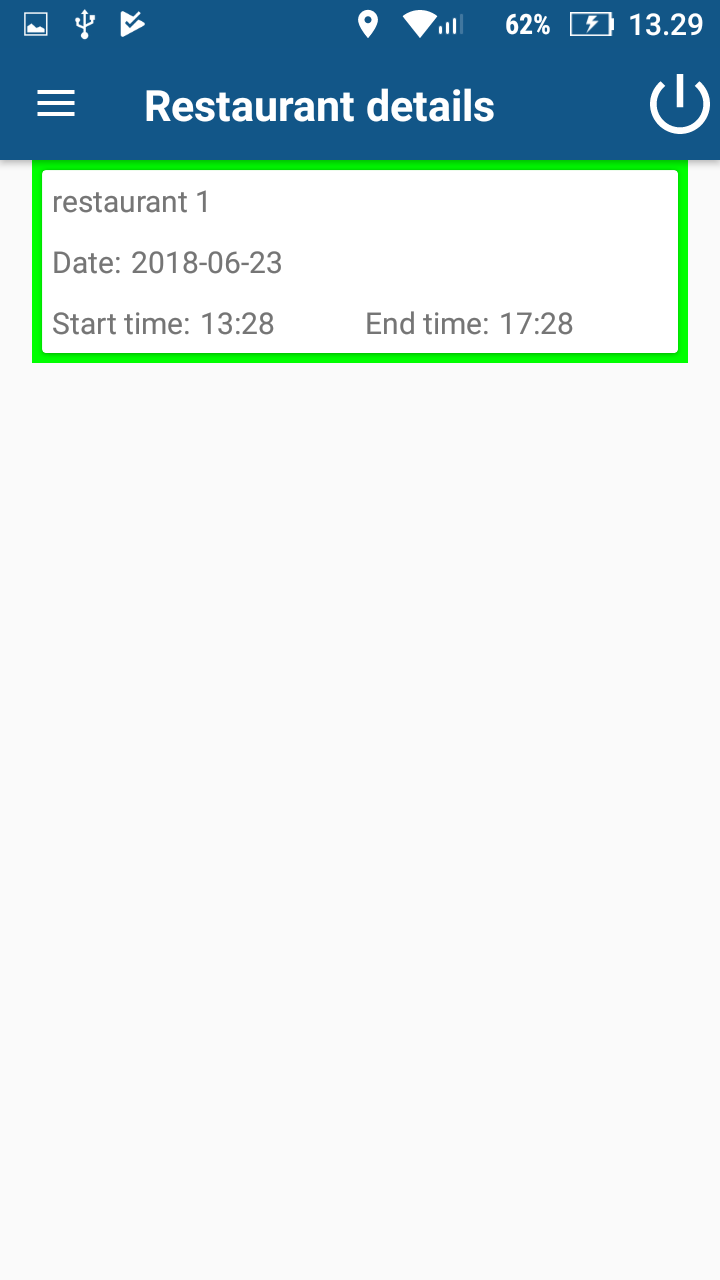
Slika 6.3.6. je forma za detaljan prikaz restorana. Korisnik može da oceni restoran tako što stavi ocenu i pritisne dugme “rate”. Klikom na zeleno dugme sa brojem telefona, korisnik poziva restoran. Pritiskom na plavo dugme ”reservation” korisnik pravi rezervaciju za restoran, slika 6.3.7.

*Slika 6.3.7.-Forma za rezervaciju*

Slika 6.3.7. je forma za rezervaciju restorana. Korisnik treba da popuni sve podatke kako bi napravio rezervaciju i poslao obaveštenje prijateljima. Treba da unese datum, vreme kad počinje, vreme kad se završava i da doda sve svoje prijatelje preko e-mail, tako što unese e-mail i pritisne na dugme “add friend”. Na kraju korisnik pritisne dugme “reservation”, rezervacija se napravi i pošalju se obaveštenja svim prijateljima o rezervaciji.



Slika 6.3.8. je forma za prikaz omiljenih restorana. Korisnik može da izbaci restoran iz omiljenih, tako što klikne na zvezdu i restoran se briše iz omiljeni restorana.

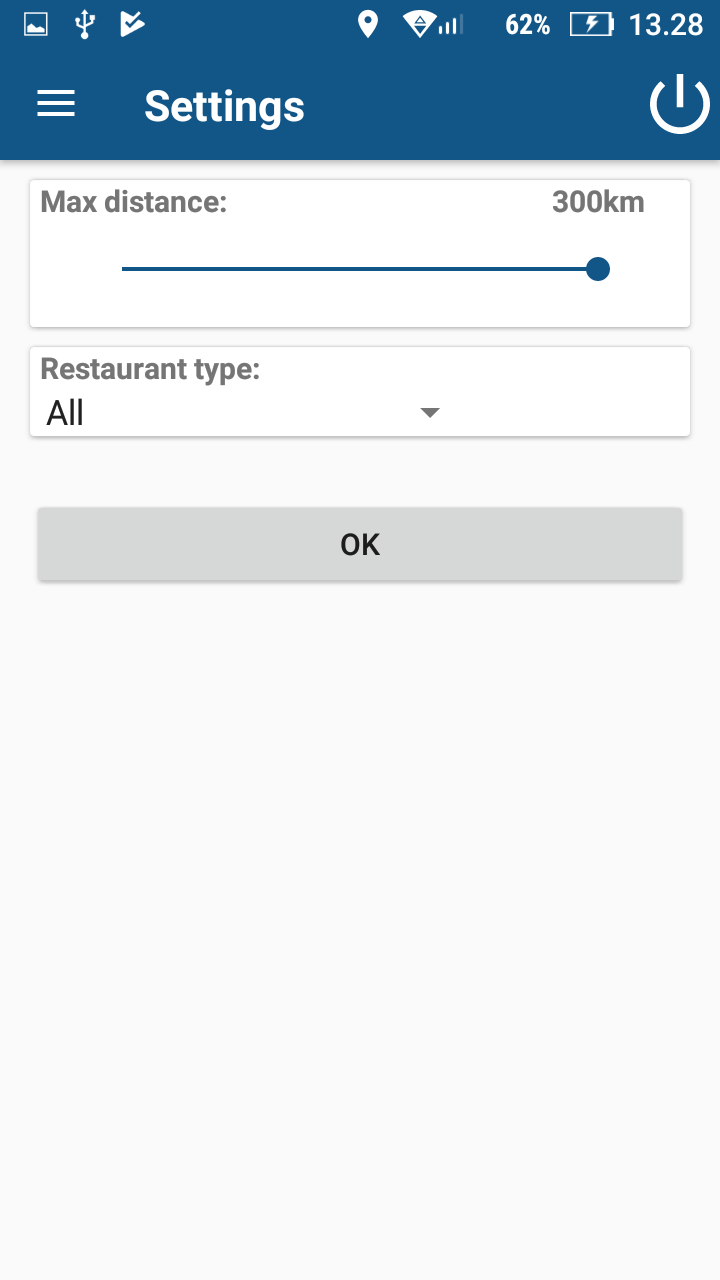
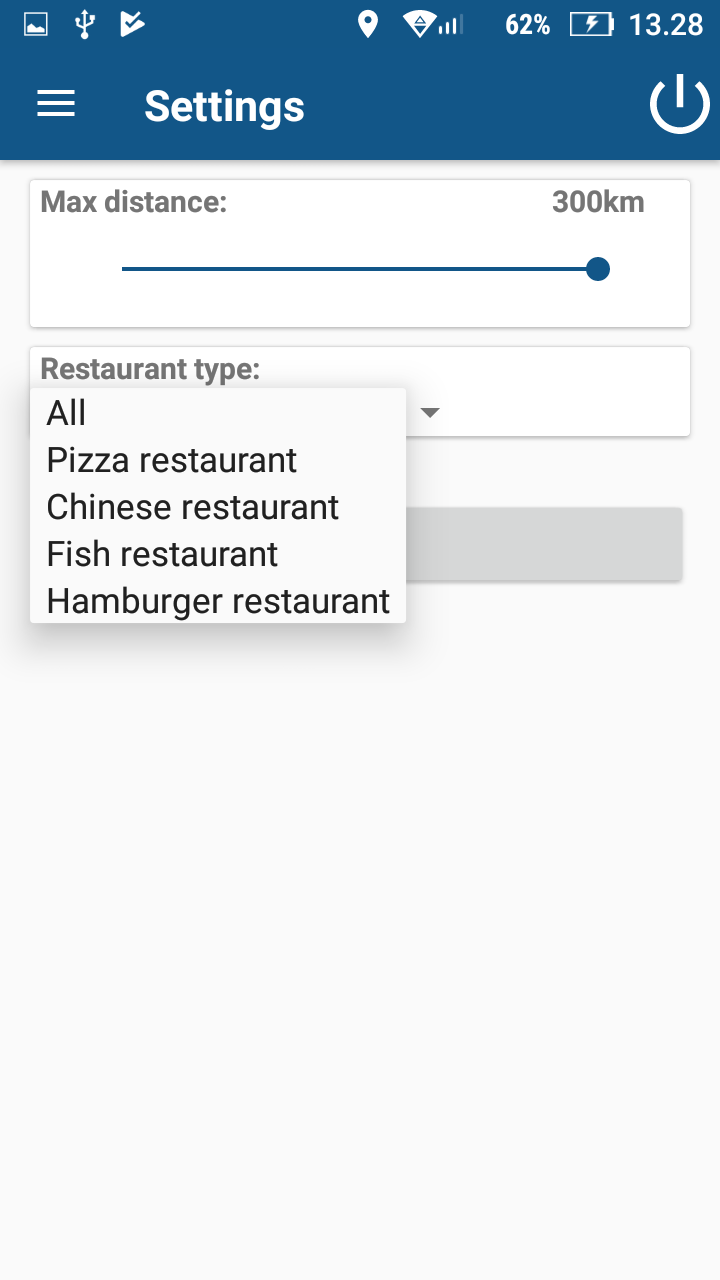


Slika 6.3.8.-Forma za prikaz omiljenih restorana

Slika 6.3.9. je forma za prikaz svih rezervacija. U toj formi su prikazane sve rezervacije koje je korisnik napravio i na koje je korisnik pozvan. Ako je okvir zelene boje znači da je korisnik napravio rezervaciju, a ako je sive onda znači da je pozvan na nečiju rezervaciju. Svaka rezervacija ima ime restorana, datum i vreme od kad do kad traje. Korisnik može da pritisne na rezercaiju, da bi video detelje rezervacije.

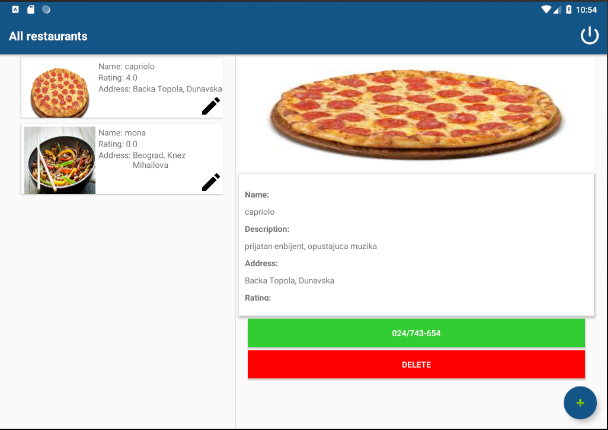
Slika 6.3.9.-Forma za prikaz rezervacija

Slika 6.3.10. je forma “settings” koja prikazuje podešavanje za prikaz restorana na mapi. Korisnik može da odabere koliko kilometara u blizini da mu se prikazuju restorani i može da odabere tip restorana, slika 6.3.11.



Slika 6.3.11.-Forma“settings”-type

***Slika 6.3.10.-Forma “settings”***



Takođe aplikacije je napravljena za sve vrste mobilnih uređaja, na slici 6.3.12 prikazana je forma aplikacije na tabletu.

Slika 6.3.12.-Prikaz rada aplikacije na tabletu

# 7. ZAKLJUČAK

Ideja pravljenja aplikacije za rezervaciju restorana zasniva se na olakšanom, tj. bržem pristupu određenim destinacijama za degustaciju obroka. Da bi se aplikacija što više iskoristila odabrana je Android platforma. Ona se pokazala kao najpopularna platforma među korisnicima. Android je platforma koja služi za razvoj mobilnih aplikacija. Pošto su mobilni uređaji na trenutnom tržištu najzastupljeniji, Android operativni sistem je olakšao pristup svim programerima da uz veliki skup ponuđenih alata razviju aplikacije.

Ovo je prva verzija aplikacije. Aplikacija za rezervaciju restorana imala je za cilj da na jednom mestu sadrži restorane na koje je moguće rezervisati i odabrati društvo za rezervaciju uz pomoć par pritiskanja dugmadi. Sa rezervacijom se poziva broj restorana i sa pozivom je rezervacija napravljena. Pored toga ova aplikacija je imala i kolekciju omiljenih restorana.

Trenutna aplikacija je ostavila prostor za proširenje funkcionalnosti, ali i pojedinim unapređivanjima postojećih funkcionalnost. Prvenstveno bi se zamenom pozivanja broja sa direktnom rezervacijom na pritisak dugmeta uštedelo vreme. Pored toga bi se moglo proširiti da se u sklopu rezervacije prikazuje mapa stolova i da sam korisnik može da izabere sto koji želi.

# LITERATURA

1. <https://www.android.com/>
2. <https://developer.android.com/>
3. <https://developers.google.com/maps/documentation/geolocation/intro>
4. <https://developers.google.com/maps/documentation/geolocation/get-api-key>
5. <https://developers.google.com/maps/documentation/android-sdk/intro>
6. <https://console.firebase.google.com/>
7. <https://console.firebase.google.com/project/restaurants-64179/overview>
8. <https://spring.io/projects/spring-boot>
9. <https://start.spring.io/>
10. <https://moodle.pmf.uns.ac.rs/course/view.php?id=356>
11. <https://moodle.pmf.uns.ac.rs/enrol/index.php?id=169>
12. <https://git-scm.com/>

# BIOGRAFIJA

Pane Odalović, rođen je 26. juna 1995. godine u Bačkoj Topoli, Srbija. Osnovnu školu “Čaki Lajoš” u Bačkoj Topoli zavšio je 2010. godine, iste godine upisuje srednju ekonomsku školu “Dositej Obradović” u Bačkoj Topoli. Srednju školu uspešno završava 2014. godine. Nakon završetka odlučuje da se upiše na akademske studije. 2014. godine upisuje Prirodno-matematički fakultet univerziteta u Novom Sadu, smer informacione tehnologije na departmanu za matematiku i informatiku.



**UNIVERZITET U NOVOM SADU**

**PRIRODNO MATEMATIČKI FAKULTET**

**DEPARTMAN ZA MATEMATIKU I INFORMATIKU**

|  |
| --- |
| **KLJUČNA DOKUMENTACIJA** |

Redni broj:

**RBR**

Indentifikacioni broj:

**IBR**

Tip dokumentacije: Monografska

**TD** dokumentacija

Tip zapisa: Tekstualni

**TZ** štampani materijal

Vrsta rada: Završni rad

**VR**

Autor: Pane Odalović

**AU**

Mentor: dr Danijela Tešendić

**MN**

Naslov rada: Android aplikacija

**NS** za rezervaciju restorana

Jezik publikacije: Srpski/latinica

**JP**

Jezik izvoda: Srpski

**JI**

Zemlja publikacije: Srbija

**ZP**

Uže geografsko područje: Vojvodina

**UGP**

Godina: 2018

**GО**

Izdavač: Autorski reprint

**IZ**

Mesto i adresa: Prirodno – matematički fakultet, Trg

**MA** Dositeja Obradovića 4, Novi Sad

Fizički opis rada: 8 poglavlja, 67 strana,

**FO** 29 slika, 17listinga

Naučna oblast: Informatika

**NO**

Naučna disciplina: Mobilne aplikacije

**ND**

Predmetna odrednica/ Android, Sprig Boot,

Ključne reči: Firebase, Google

**PO**

**UDK:**

Čuva se: Biblioteka Departmana za matematiku

**ČU** i informatiku, Novi Sad

Važna napomena: Nema

Izvod

**IZ**: Opis aplikacije za rezervaciju restorana

Datum prihvatanja teme:

**DТ**

Datum odbrane:

**DO**

Članovi komisiji:

**1.** dr Miloš Racković, redovni profesor PMF-a, predsednik

**2.** dr Danijela Tešendić, vandredni profesor PMF-a, mentor

**3.** dr Danijela Boberić Krstićev, vandredni profesor PMF-a, član

**UNIVERSITY OF NOVI SAD**

**FACULTY OF SCIENCES**

**DEPARTMENT OF MATHEMATICS AND INFORMATICS**

|  |
| --- |
| **KЕY WORDS DOCUMENTATION** |

Accession number:

**ANO**

Identification number:

**INO**

Document type: Monograph type

**DT**

Type of record: Printed text

**TR**

Contents code: Diploma's thesis

**CC**

Autor: Pane Odalović

**AU**

Mentor: PhD Danijela Tešendić

**MN**

Title:Android application for

**XI** restaurant reservation

Language of text: Serbian/Latinica

**LT**

Language of abstract: Serbian

**LA**

Country of publication: Serbia

**CP**

Locality of publication: Vojvodina

**LP**

Publication year: 2018.

**PY**

Publisher: Author's reprint

**PU**

Publication place: PMF, Trg Dositeja

**PP** Obradovića 4, Novi Sad

Physical description: 8 chapters, 67 pages,

**PD** 29 photos, 17 listings

Scientific field: Informatics

**SF**

Scientific discipline: Mobile applications

**SD**

Key words: Android, Spring Boot,

**KW** Firebase, Google

Holding data: Library of Department of mathematics

**SD** and informatics, Novi Sad

Note: None

Abstract: Application for restaurant reservation

**AB** implemented by using Android

and SpringBoot technologies

Accepted by the Scientific Board on:

Defended on:

Thesys defend board:

**1.** PhD Miloš Racković, full professor, Faculty of Sciences, Novi Sad, president

**2.** PhD Danijela Tešendić, associate professor, Faculty of Sciences, Novi Sad, menthor

**3.** PhD Danijela Boberić Krstićev, associate professor, Faculty of Sciences, Novi Sad, member