SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Davorin Horvat

Filip Antonio Jeličić

Nadja Kosojević

Josipa Tadić

VisitMe

projektni rad IZ KOLEGIJA ANALIZA I RAZVOJ PROGRAMA

Link na github: <https://github.com/josipatadic/VisitMe>

Varaždin, 2015.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Davorin Horvat, 44494/15-R

Diplomski studij, smjer: Informacijsko i programsko inženjerstvo

Filip Antonio Jeličić, 44471/15-R

Diplomski studij, smjer: Organizacija poslovnih sustava

Nadja Kosojević, 44416/15-R

Diplomski studij, smjer: Organizacija poslovnih sustava

Josipa Tadić, 44417/15-R

Diplomski studij, smjer: Organizacija poslovnih sustava

VisitMe

PROJEKTNI RAD KOLEGIJA ANALIZA I RAZVOJ PROGRAMA

Link na github: <https://github.com/josipatadic/VisitMe>

Mentor:

Ivan Švogor, mag.inf

Varaždin, prosinac 2015.

# Sadržaj

[Sadržaj I](#_Toc439273536)

[1. Korištene tehnologije i alati 1](#_Toc439273537)

[2. Artefakti 2](#_Toc439273538)

[2.1. Korisnički zahtjevi i Use Case dijagram 2](#_Toc439273539)

[2.2. Arhitektura aplikacije 6](#_Toc439273540)

[2.3. Mockup – idejne skice ekrana 7](#_Toc439273541)

[2.4. ERA model 20](#_Toc439273542)

[2.4.1. Tablica Apartment 21](#_Toc439273543)

[2.4.2. Tablica Images 21](#_Toc439273544)

[2.4.3. Tablica Owner 21](#_Toc439273545)

[2.4.4. Tablica Address 21](#_Toc439273546)

[2.4.5. Tablica Town 21](#_Toc439273547)

[2.4.6. Tablica POI 21](#_Toc439273548)

[2.4.7. Tablica Category 22](#_Toc439273549)

[2.5. Dijagram klasa 23](#_Toc439273550)

[3. Programska dokumentacija 26](#_Toc439273551)

[3.1. Struktura VisitMe aplikacije 26](#_Toc439273552)

[3.2. Google maps 27](#_Toc439273553)

[3.3. Retrofit 27](#_Toc439273554)

[3.3.1. Početne postavke 27](#_Toc439273555)

[3.3.2. Kreiranje Java objekata 27](#_Toc439273556)

[3.3.3.Sučelje ApiMethods 28](#_Toc439273557)

[3.3.4. Asinkroni poziv 28](#_Toc439273558)

[3.3.5. Inicijalizacija Retrofita 28](#_Toc439273559)

[3.3.6.Dohvaćanje podataka 29](#_Toc439273560)

[4. Aplikacija 30](#_Toc439273561)

# 1. Korištene tehnologije i alati

Tehnologije koje koristimo za ovaj projekt su:

* + Git
  + Web: MySQL, PHP
  + Mobilne: Java, SQLite

Aplikacija se izrađuje u Android studiju. Od alata smo još koristili Ninjamock za izradu skica ekrana, zatim Visual Paradigm za izradu Usecase dijagrama, Toad Modeler za izradu ERA modela te koristimo Scrumdesk za planiranje i praćenje projekta. Tijekom izrade projekta koristimo i Genymotion za pregled aplikacije.

Potrebna verzija API-ja je 16.

# 2. Artefakti

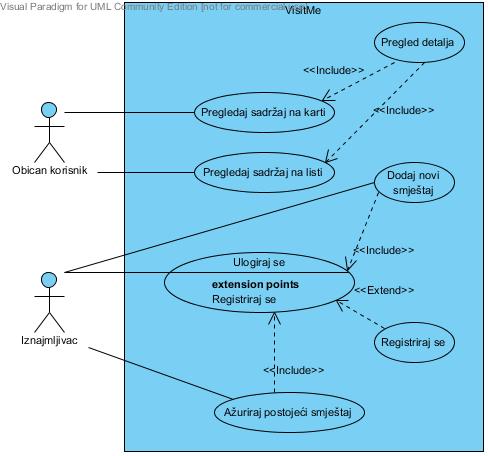
## 2.1. Korisnički zahtjevi i Use Case dijagram

Korisnički zahtjevi su:

* dodavanje novog smještaja
* ažuriranje postojećeg smještaja
* pregled smještaja, restorana, kafića ili znamenitosti na karti
* pregled smještaja, restorana, kafića ili znamenitosti na listi

Dodavanje novog smještaja podrazumijeva da registrirani korisnik može putem aplikacije u bazu pohraniti podatke o smještaju kojeg nudi (cijenu, broj ležaja, lokaciju...), a ažuriranje postojećeg smještaja bi se koristilo kada treba promijeniti podatke, primjerice da je smještaj trenutno nedostupan, to jest iznajmljen, zatim u slučaju promjene cijene ili broja ležaja. Pregled sadržaja (smještaja, restorana, kafića ili znamenitosti) na karti omogućavao bi korisniku da na mapi vidi jednu od navedenih kategorija sadržaja, ovisno o tome koju želi da mu se prikaže na mapi te da prilikom dodira znaka na karti za određeni saržaj korisnik vidi detalje poput naziva, adrese i slično. Pregled sadržaja na listi omogućavao bi isto, samo što bi sadržaj bio prikazan na listi, te bi se dodirom na određenu stavku na listi vidjeli detalji i pokazala bi se mogućnost prikaza samo tog sadržaja na mapi.

Za korištenje VisitMe aplikacije odredili smo dva tipa korisnika, to su korisnici koji se ne moraju registrirati niti ulogiravati u aplikaciju (turisti i ostali koji žele samo gledati sadržaj karte/liste), to jest *tip korsnika 1* (TK1), te registrirani korisnici – iznajmljivači, to jest *tip korisnika 2* (TK2), koji time dobivaju mogućnosti dodavanja i ažuriranja smještaja. Detaljan prikaz može se promotriti na slici 2.1. Use case dijagram.

****

**Slika 2.1. Use Case dijagram *VisitMe***

Dakle, TK1, običan korisnik ima pravo pregledavati sadržaj na karti ili na listi, ovisno o njegovim željama. U oba slučaja bit će mu prikazani isti detalji o sadržaju koji mu je od interesa. S druge strane, iznajmljivač je TK2, koji se prvo mora ulogirati u aplikaciju kako bi mogao unositi ili mijenjati sadržaj, u protivnom se tretira kao običan tip korisnika, a ukoliko nije registriran, zahtjeva se registracija. Nakon ulogiravanja u aplikaciju ima pravo dodavati novi smještaj ili ažurirati podatke o postojećem kojeg je već prije unio.

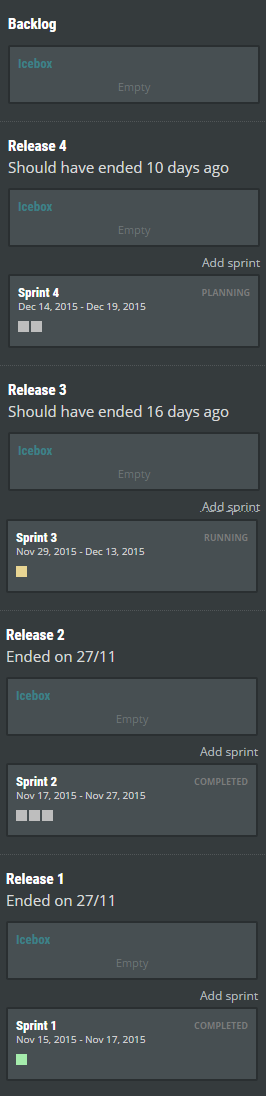
Svi zahtjevi aplikacije, funkcionalni i nefunkcionalni, definirani prilikom prvog planiranja projekta, prikazani su u tablici 2.1. Zahtjevi aplikacije (Product Backlog).

**Tablica 2.1. Zahtjevi aplikacije (Product Backlog)**

|  |  |
| --- | --- |
| **Zahtjevi: ■ - funkcionalni, ■ - nefunkcionalni** | **Važnost (O – obavezno, N – neobavezno)** |
| 1. Baza podataka i sučelje | O |
| 1. Pregled podataka na karti | O |
| 1. Pregled podataka na listi | O |
| 1. Dodavanje sadržaja | O |
| 1. Ažuriranje sadržaja | O |
| 1. Registracija | O (zbog TK2) |
| 1. Login | O (zbog TK2) |

Baza podataka stavljena je na 1. mjesto prema prioritetima zbog toga što bez podataka koji se prikazuju na mapi/listi cijela aplikacija nema smisla, a također, i bez sučelja na kojem će se to prikazivati aplikacija nema smisla. Baza podataka i sučelje pripadaju u nefunkcionalne zahtjeve. Pregled podataka na karti je na drugom mjestu po prioritetima jer smo mi kao Product owner odlučili da zadani prikaz prilikom otvaranja aplikacije bude na mapi. Zatim na trećem mjestu slijedi prikaz istih tih podataka na listi, zbog toga jer se neki korisnici vjerojatno bolje snalaze u takvom načinu pregleda podataka. Dodavanje sadržaja je na 4. mjestu, a ažuriranje sadržaja na 5. mjestu. Registraciju i login, također nefunkcionalne zahtjeve, smo stavili na 6. i 7. mjesto, jer aplikacija može funkcionirati i bez toga, ukoliko omogućimo svim korisnicima unos podataka, ali s registracijom i loginom imamo određenu mogućnost kontrole tko unosi podatke u bazu i tko ažurira te podatke. Sve stavke su obavezne za napraviti.

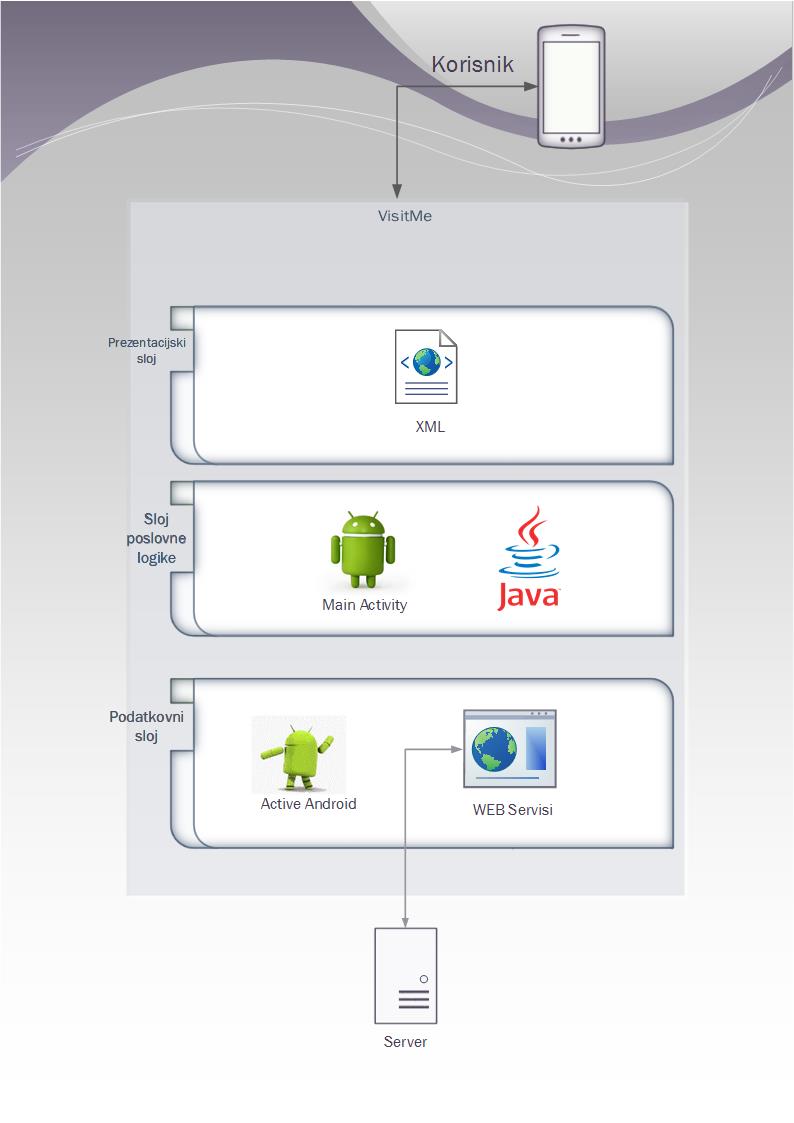
Aplikaciju planiramo realizirati kroz 4 releasea, u kojima je po jedan sprint, što se vidi i na slici 2.2. Backlog (stanje s 30.12.2015). Dakle, backlog se sastoji od 4 releasea, koji se svaki sastoji od jednog sprinta. Release 1 ima pogrešku završnog datuma, treba biti 17.11.2015, kao i završni datum sprinta 1 koji pripada tom releaseu, Scrum Master osoba nije odmah uočila da prilikom završavanja sprinta ne ide automatsko završavanje releasea. Sa slike je vidljivo koji sprintevi su gotovi, koji traju, a koji su u fazi planiranja.



**Slika 2.2. Backlog**

## 2.2. Arhitektura aplikacije

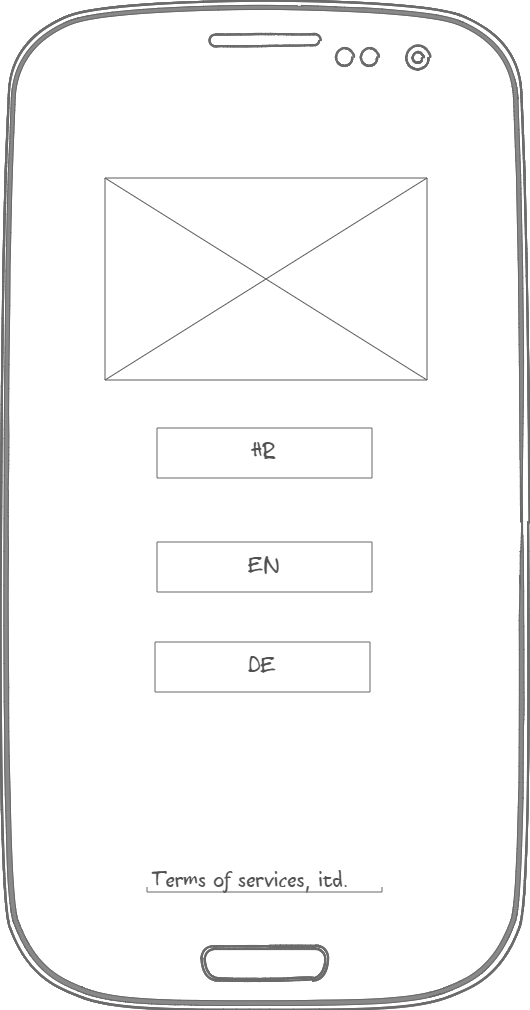
Arhitektura aplikacije VisitMe je prikazana na slici 2.3. Arhitektura aplikacije te vrlo sažeto prikazuje sastavnice aplikacije po slojevima. Prezentacijski sloj izrađen je pomoću XML jezika kojime su definirana sučelja i to je jedini sloj s kojim korisnik dolazi u direktnu interakciju preko odabranih sučelja. Sloj poslovne logike se definira u programskom jeziku Java te izrađuje u Android Studio-u. Funkcijama i događajima omogućit ćemo kontrolirano ponašanje aplikacije, u skladu sa korisničkim zahtjevima. Podatkovni sloj predstavlja povezivanje i komuniciranje aplikacije s bazom, odnosno serverom, preko web servisa, što ćemo ostvariti korištenjem ActiveAndroida i Json objekta.



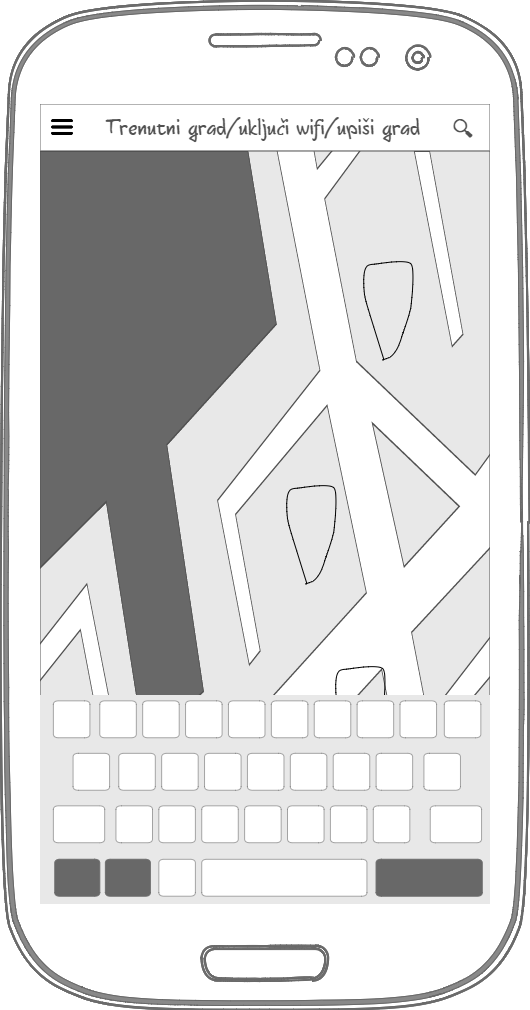
**Slika 2.3. Arhitektura aplikacije**

## 2.3. Mockup – idejne skice ekrana

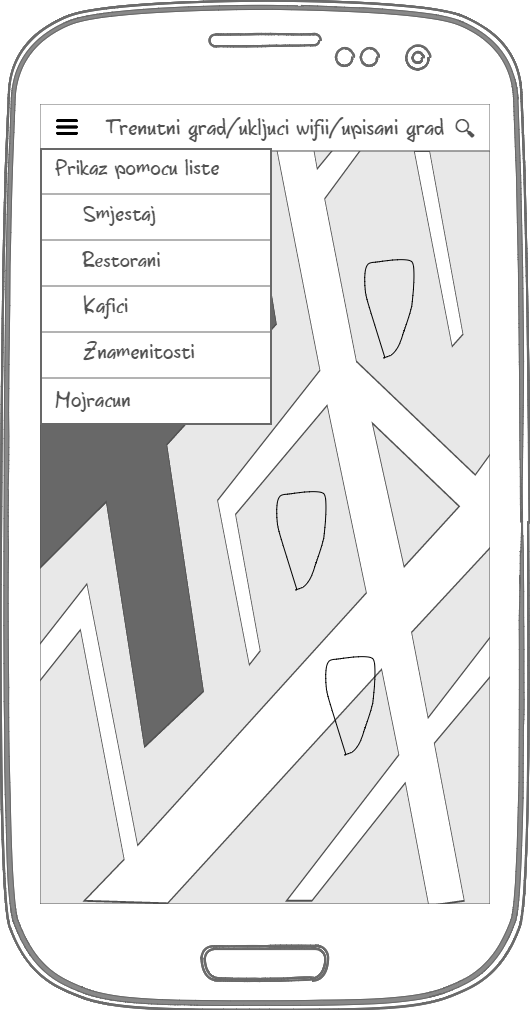
Mockup prikazuje planirani, to jest željeni okvirni izgled sučelja (**skicu**) koji je definiran u pre-game i prvoj fazi (prvom sprintu), a završit će ih se i po potrebi prilagoditi u sljedećim fazama. U nastavku su priloženi mockup-ovi (slike 2.4 – 2.16) prve faze koji bi trebali kasnije prema elementima odgovarati elementima sučelja u aplikaciji, a slike ekrana iz aplikacije bit će stavljene u dokumentaciju kada aplikacija bude gotova u potpunosti. Ukoliko želite vidjeti planiranu navigaciju po aplikaciji, možete je pogledati [ovdje](https://ninjamock.com/s/8C6DS).



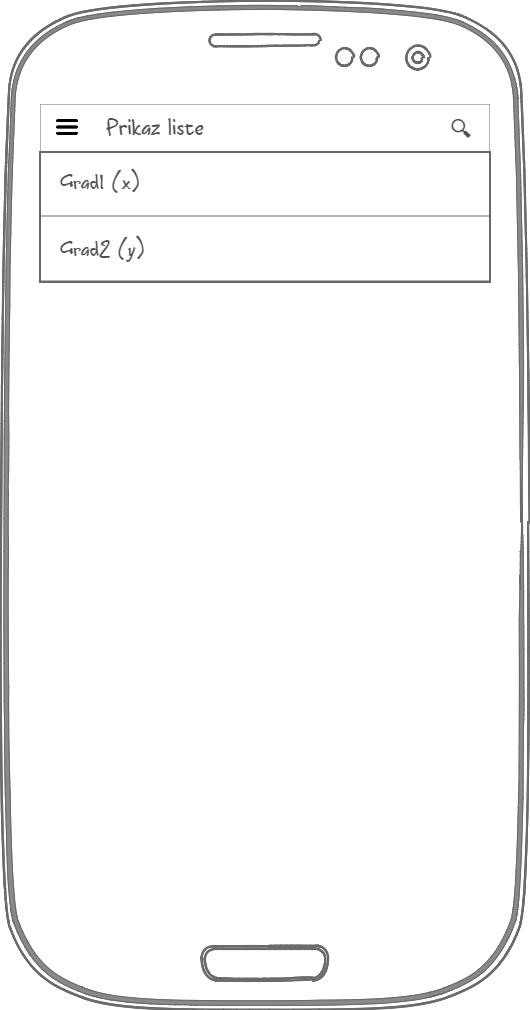
**Slika 2.4. Mockup početnog sučelja**

****

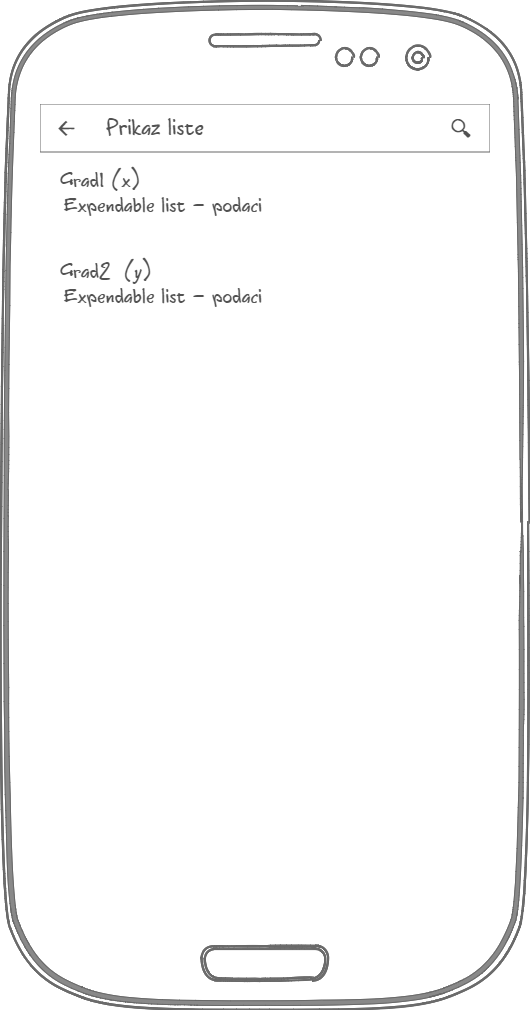
**Slika 2.5. Mockup Default sučelja**

****

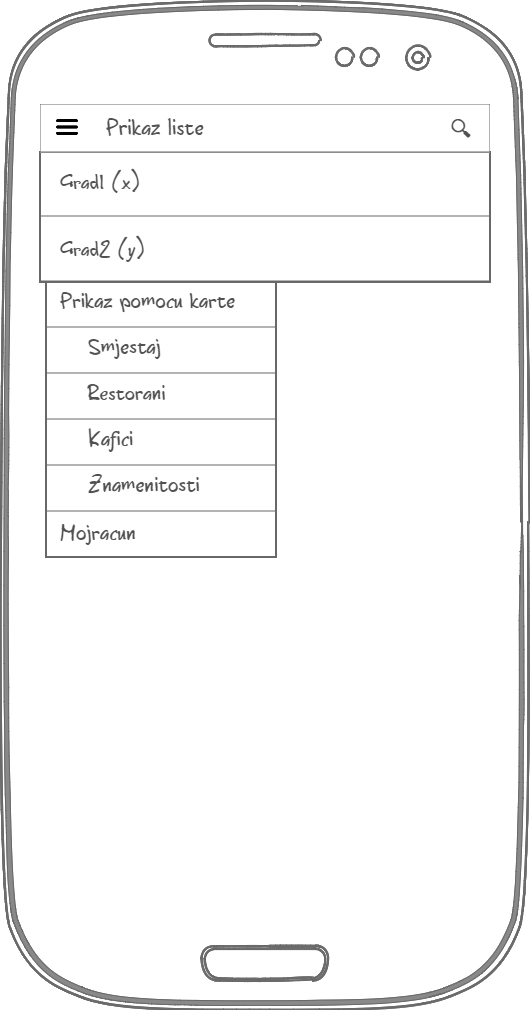
**Slika 2.6. Mockup Menu sučelja**

****

**Slika 2.7. Mockup sučelja s listom gradova**

****

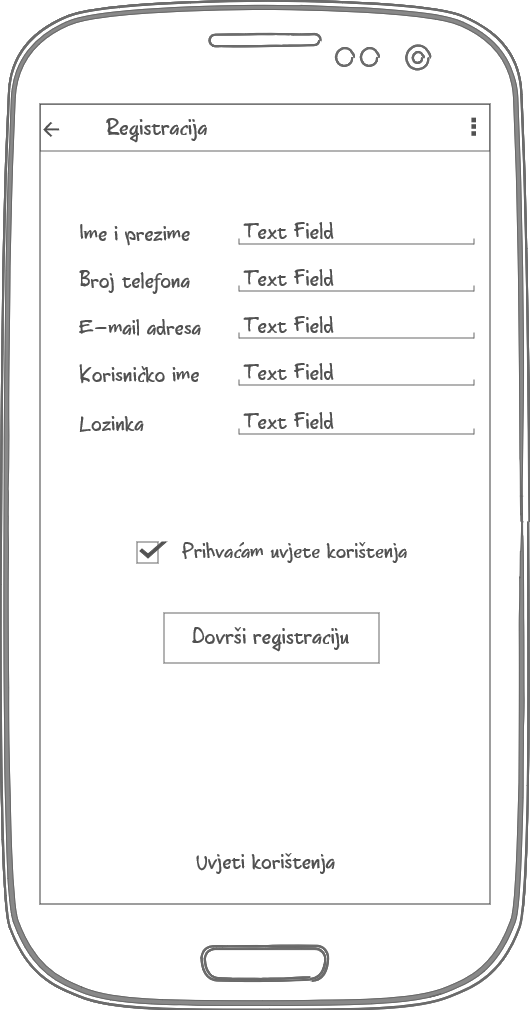
**Slika 2.8. Mockup sučelja s proširenom listom grada i njegovih sadržaja**

****

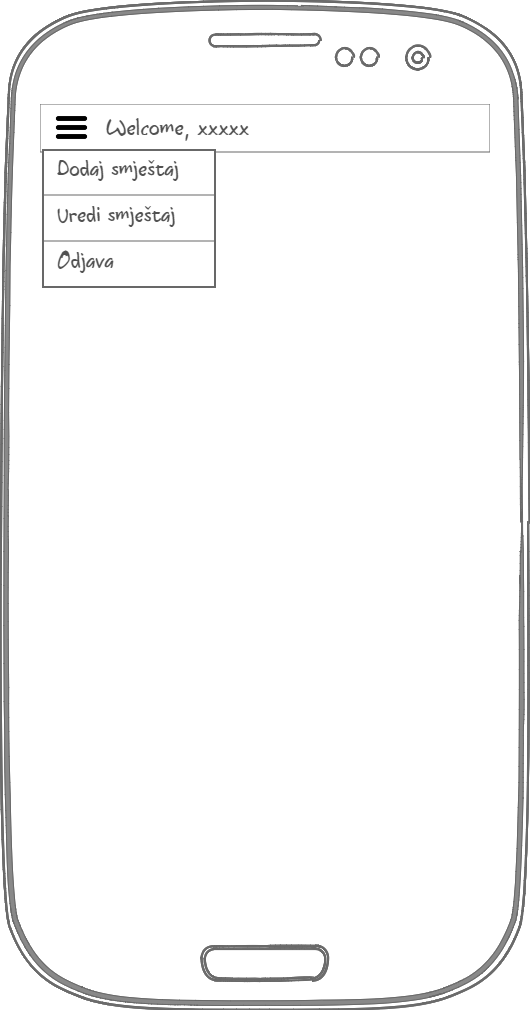
**Slika 2.9. Mockup sučelja liste s Menu-em[[1]](#footnote-1)**

****

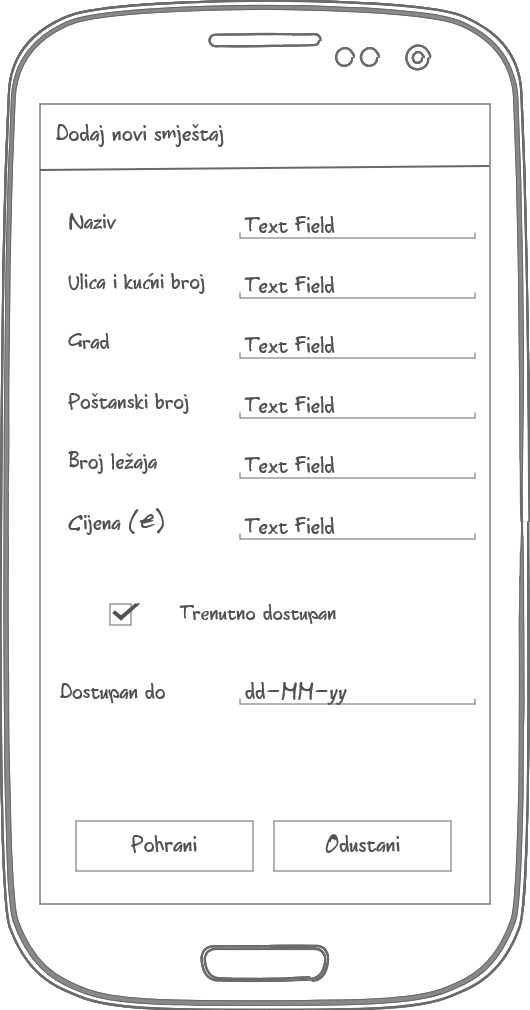
**Slika 2.10. Mockup LogIn sučelja[[2]](#footnote-2)**

****

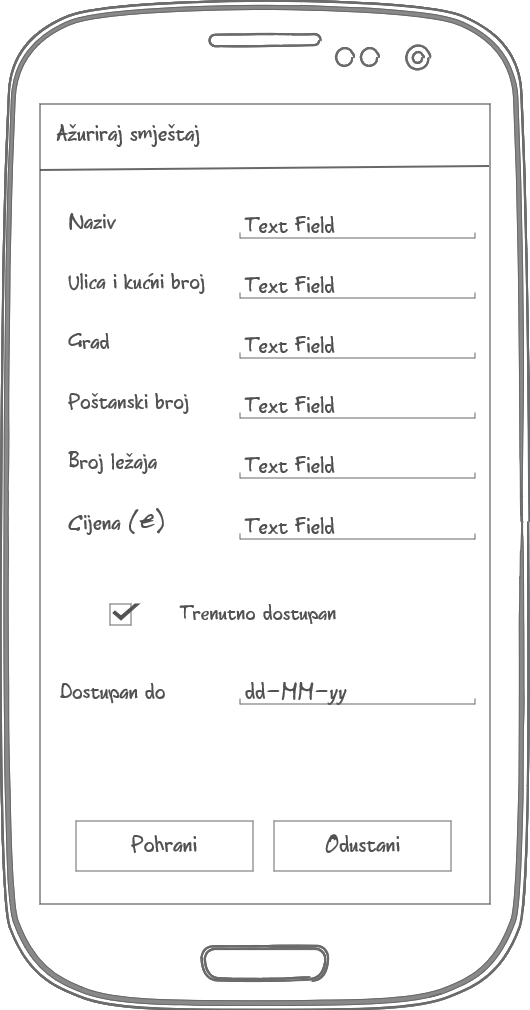
**Slika 2.11. Mockup sučelja Registracija**

****

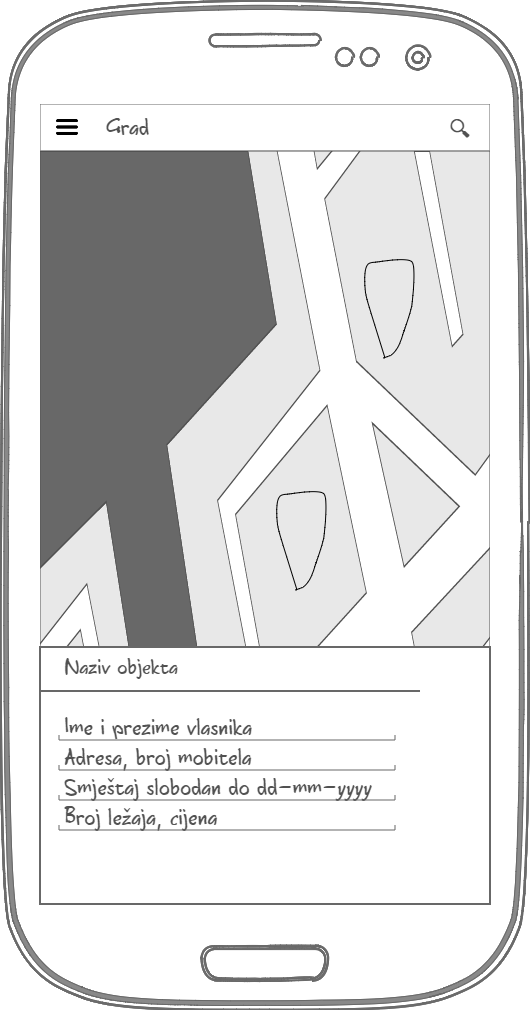
**Slika 2.12. Mockup sučelja za dodavanje novog/ažuriranje postojećeg smještaja**

****

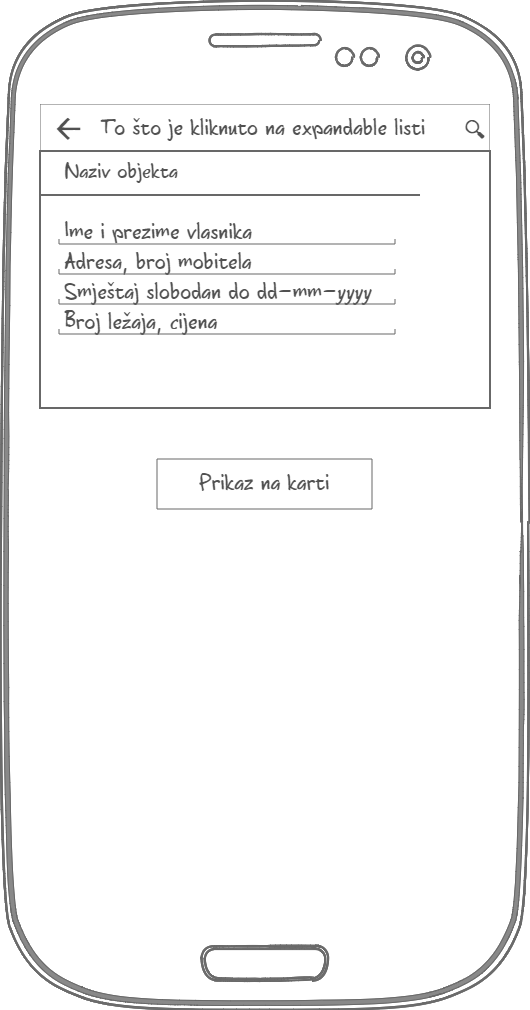
**Slika 2.13. Mockup sučelja za dodavanje novog smještaja**

****

**Slika 2.14. Mockup sučelja za ažuriranje smještaja**

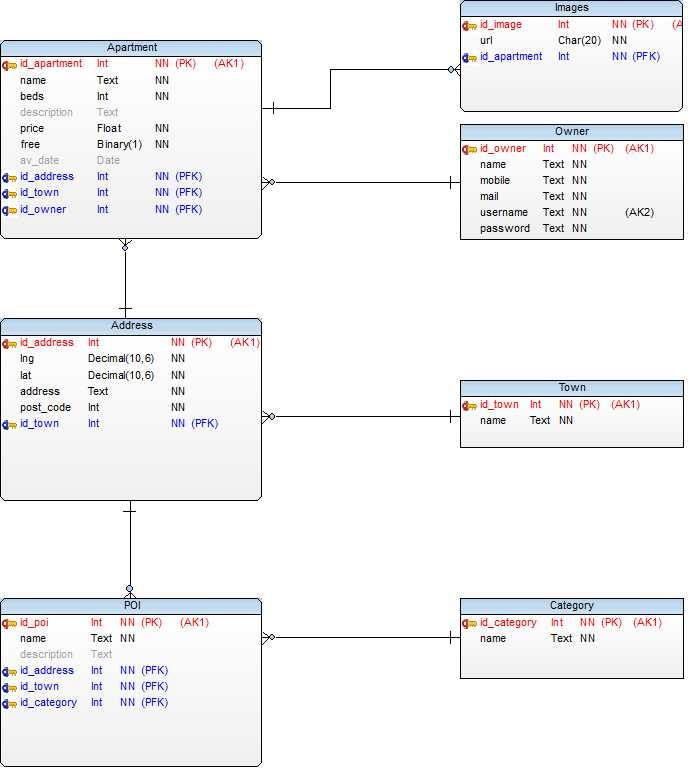
****

**Slika 2.15. Mockup sučelja za odabir grada/objekta na mapi**

****

**Slika 2.16. Mockup sučelja sa listom detalja o odabranom smještaju**

## 2.4. ERA model

**Slika 2.17. ERA model VisitMe aplikacije**

U ERA modelu prikazali smo sve entitete i njihove atribute te međusobne veze koje smo koristili prilikom izrade baze podataka potrebne za realizaciju ovog projekta. Detaljan grafički prikaz nalazi se na slici 2.17.

Na navedenoj slici vidimo da ERA model sastoji od 7 tablica:

* Tablica Apartment
* Tablica Images
* Tablica Owner
* Tablica Address
* Tablica Town
* Tablica POI
* Tablica Category

### 2.4.1. Tablica Apartment

U tablici Apartment se nalaze svi apartmani koje može prijaviti i postaviti registrirani korisnik. U tablici se nalazi naziv apartmana, broj kreveta, cijena, dostupnost, opis, grad i adresa te podaci o iznajmljivaču.

### 2.4.2. Tablica Images

Tablica Images pohranjuje sve adrese slika određenih apartmana koje vlasnik može postaviti (opcionalno).

### 2.4.3. Tablica Owner

U tablicu Owner pohranjuju se svi podaci o iznajmljivačima, to jest, ime i prezime, kontakt broj, e-mail adresa te korisničko ime i lozinka kojom se prijavljuje u aplikaciju VisitMe.

### 2.4.4. Tablica Address

Tablica Address se sastoji od adresa apartmana koji se iznajmljuju zajedno sa njihovim koordinatama te poštanskim brojem grada kojemu pripadaju.

### 2.4.5. Tablica Town

Town je tablica koja služi za pohranjivanje svih gradova u kojima će se moći koristiti VisitMe aplikacija.

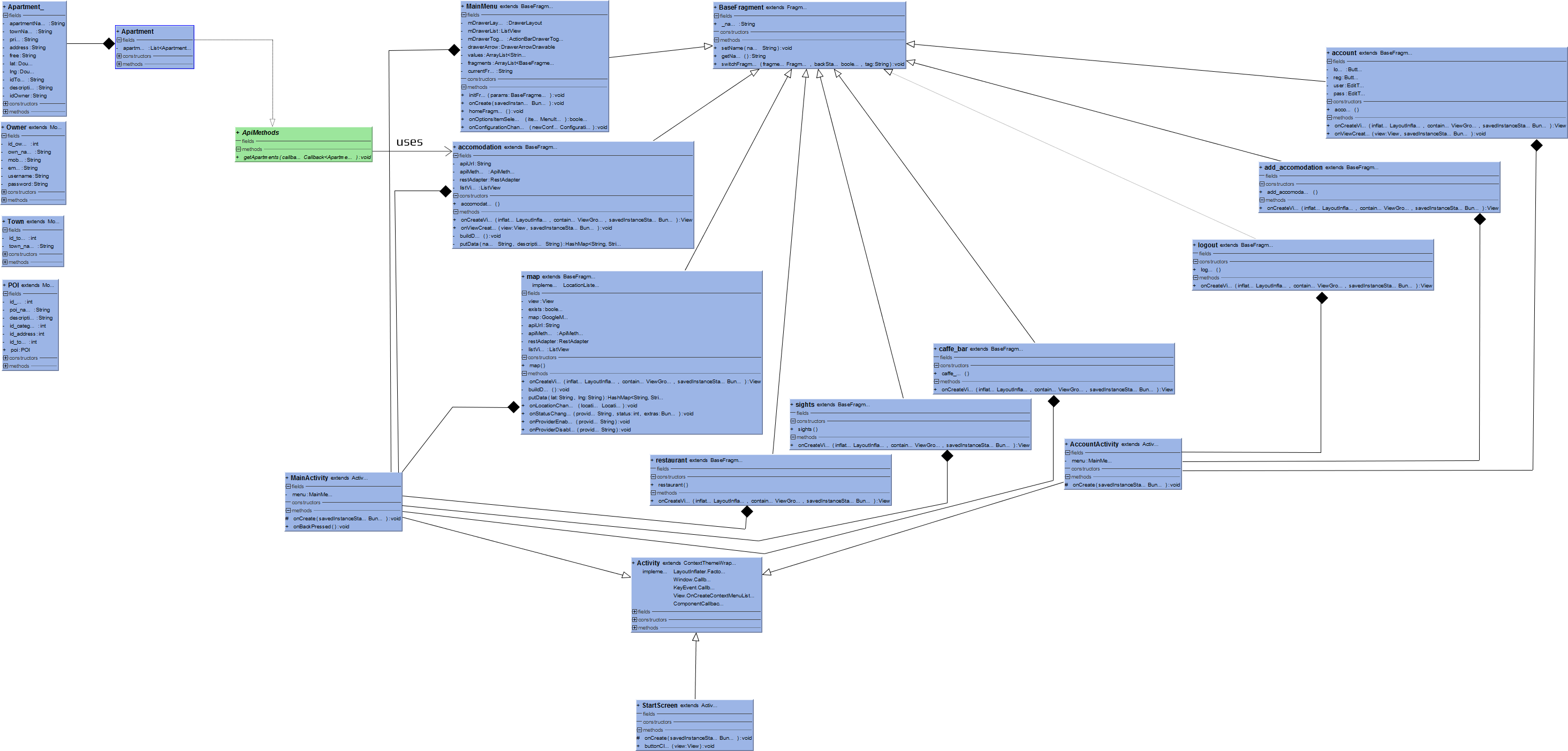
### 2.4.6. Tablica POI

U tablici POI spremaju se podaci o kulturnim znamenitostima, restoranima i caffee barovima, noćnim klubovima i ostalim turističkim mogućnostima koje svaki grad nudi.

### 2.4.7. Tablica Category

Tablica Category predstavlja pomoćnu tablicu tablice POI koja sadrži sve kategorije kulturnih znamenitosti i ostalih kategorija koje su atraktivne turistima te se prikazuju na zemljopisnoj karti u aplikaciji VisitMe.

## 2.5. Dijagram klasa



**Slika 2.18. Dijagram klasa**

Aplikacija VisitMe sastoji se od tri glavne aktivnosti implementirane u klasama StartScreen, MainActivity i AccountAcitvity. Sve tri aktivnosti naslijeđuju klasu Activity. Za upravljanje fragmentima kreirana je klasa BaseFragment koja proširuje klasu Fragment atributom String name i metodama setName(String), String getName() i switchFragment(...).

Svi fragmenti korisničkog proširuju klasu BaseFragment.

Fragmente možemo podijeliti na tri skupine:

**- fragmenti glavnog korisničkog sučelja:**

- accomodation

- map

- restaurant

- sights

- caffe\_bar

**- fragmenti sučelja za upravljanje profilom**

- account

- add\_accomodation

- logout

**- fragment izbornika**

- MainMenu

Za komunikaciju s bazom podataka kreirano je sučelje ApiMethods pomoću kojega se aplikacija spaja na aplikacijsko sučelja poslužitelja i odgovore poslužitelja sprema u odgovarajuće klase.

Klase u koje se sprema odgovor sa poslužitelja kreirane su alatom jsonshema2pojo:

- Apartment instancira listu objekata klase Apartment\_

- Owner

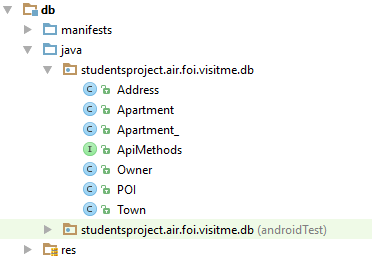
- Town

- POI

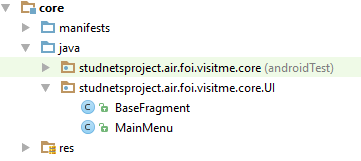
# 3. Programska dokumentacija

## 3.1. Struktura VisitMe aplikacije

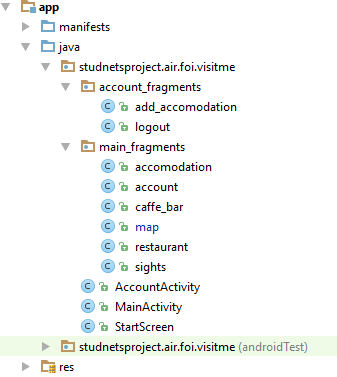
Aplikacija VisitMe se sastoji od tri modula: app, core i db od kojih svaki sadržio dređena programska rješenja. Db modul sadrži klase za lokalnu bazu podataka kreirane pomoću jsonschema2pojo.



Core modul sadrži klase koje definiraju glavni izbornik u aplikaciji.



U app modulu su fragmenti poput prikaza smještaja na mapi ili u listi te opcije za korisnikeaplikacije poput login forme i dodavanja smještaja.



## 3.2. Google maps

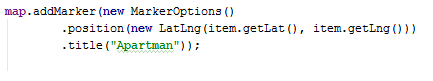
Za prikaz lokacija apartmana na zemljopisnoj karti korišten je Google Maps API. Aplikacija treba API ključ kako bi pristupila Google Maps serverima. API ključ se dobiva registracijom aplikacije na Google Developers Console, koristeći SHA-1 potpis i *package name* aplikacije. Dobiveni API key se Unosi u AndroidManifest.xml.

Capture

Dodavanje mape u fragment se iszvršava sa:

Capture

Dodavanje markera na mapu, u ovom slučaju su to lokacije apartmana:



## 3.3. Retrofit

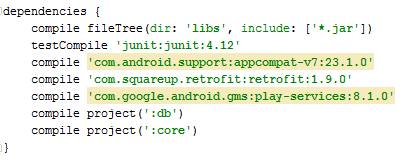
Aplikacija komunicira sa serverom koristeći Retrofit REST klijent za Android i Javu koji olakšava primanje i slanje JSON podataka prema REST baziranom webservisu.

### 3.3.1. Početne postavke

Kako bi se omogućilo korištenje Retrofita u aplikaciji potrebno je u AndroidManifest.xml dodati dopuštenje za pristup internetu.

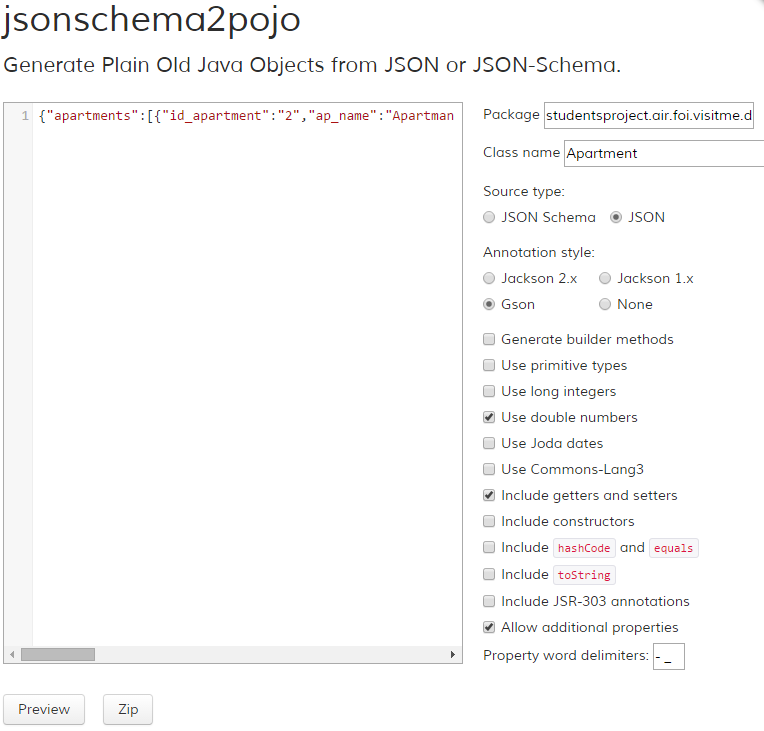
Capture

Potrebno je postaviti Retrofit u app/build.gradle



### 3.3.2. Kreiranje Java objekata

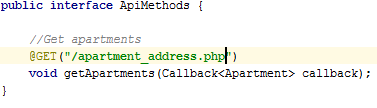
Za kreiranje Java objekata korišten je jsonschema2pojo servis koji generira Java objekte pomoću json odgovora sa servera.



Pritiskom na Preview dobivaju se klase koje pohranjuju odgovor sa servera te se spramaju u DB modul aplikacije.

### 3.3.3.Sučelje ApiMethods

Preko klase sučelja ApiMethods aplikacija komunicira sa php skriptama na serveru. Sučelje sadrži metode za dohvaćanje podataka iz baze podataka i sadržavati će metode za slanje podataka na obradu poslužitelju.



### 3.3.4. Asinkroni poziv

U verziji Retrofit-a 1.9 asinkroni pozivi nemaju povratni tip (metoda void getApartments(Callback<Apartment> callback)). Kod definiranje metode potrebo je kao zadnji parametar uvrstiti tipizirani uzvratni poziv (typed callback) u koji pohranjuje odgovor sa poslužitelja.

### 3.3.5. Inicijalizacija Retrofita

Klasa koja koristi Retrofit mora imati defenirana tri atributa:

* private String apiUrl;
* private ApiMethods apiMethods;
* private RestAdapter restAdapter;

Atribut apiUrl sadržava putanju do vanjskog poslužitelja, apiMethods je sučelje prema vanjskom poslužitelju, a restAdapter je klasa Retrofit-a koja se brine o postavkama samog Retrofit-a. Unutar onCreateView metode postavljaju se vrijednosti definiranih atributa:

|  |
| --- |
| apiUrl = getActivity().getString(R.string.API);  restAdapter = new RestAdapter.Builder()  .setEndpoint(apiUrl)  .build();  apiMethods = restAdapter.create(ApiMethods.class); |

Putanja prema poslužitelju zapisana je u strings.xml datoteci. Kod kreiranja instance ReastAdapter klase potrebno je samo dodati putanju prema poslužitelju i atributu apiMethods kreirati sučelje prema poslužitelju.

### 3.3.6.Dohvaćanje podataka

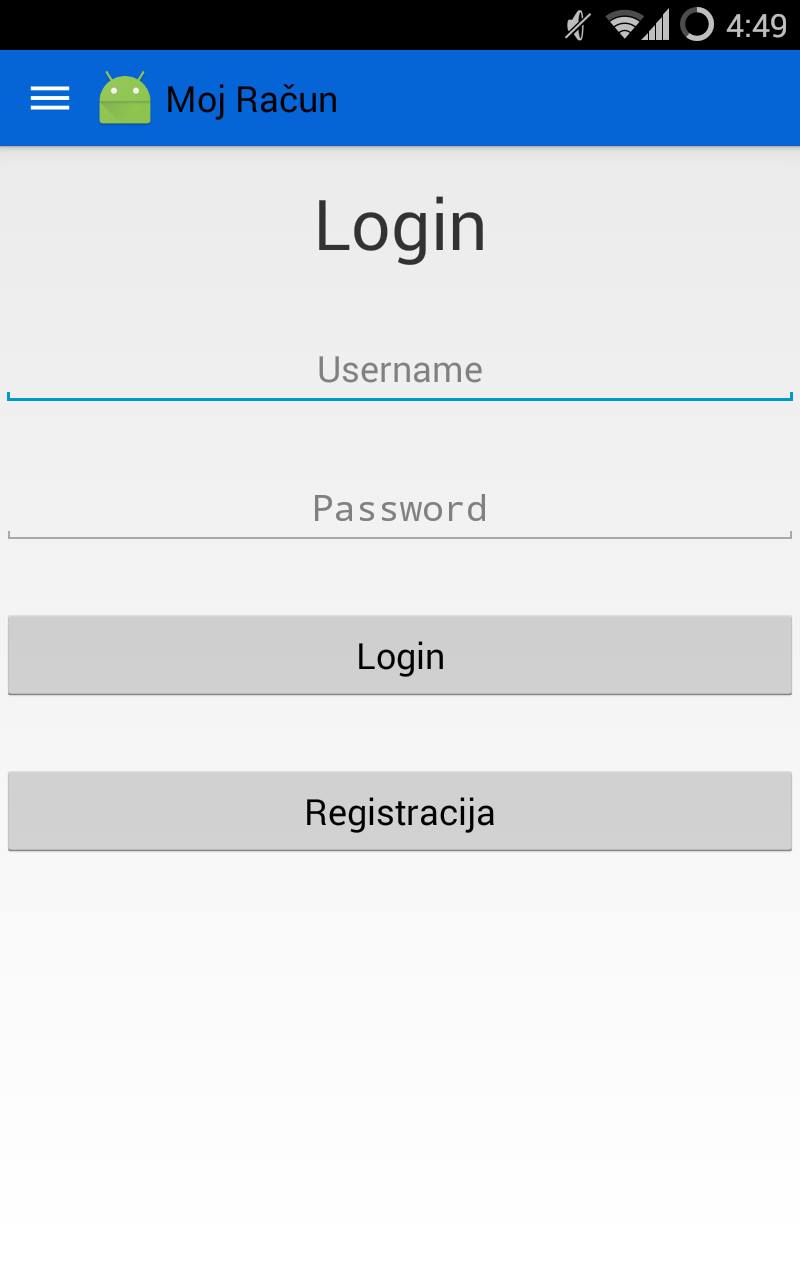
Dohvaćanje podataka sa poslužitelja vrši se privatnom metodom void buildData() koja dohvaća podatke sa poslužitelja i priprema upravlja sa uspješnim i neuspješnim dohvaćanjem:

|  |
| --- |
| private void buildData({  apiMethods.getApartments(new Callback<Apartment>() {  @Override  public void success(Apartment apartment, Respons response) {  //Upravljanje uspješnim dohvaćanjem iz baze  }    @Override  public void failure(RetrofitError error){  //Upravljanje neuspješnim dohvaćanjem iz baze  }  });  } |

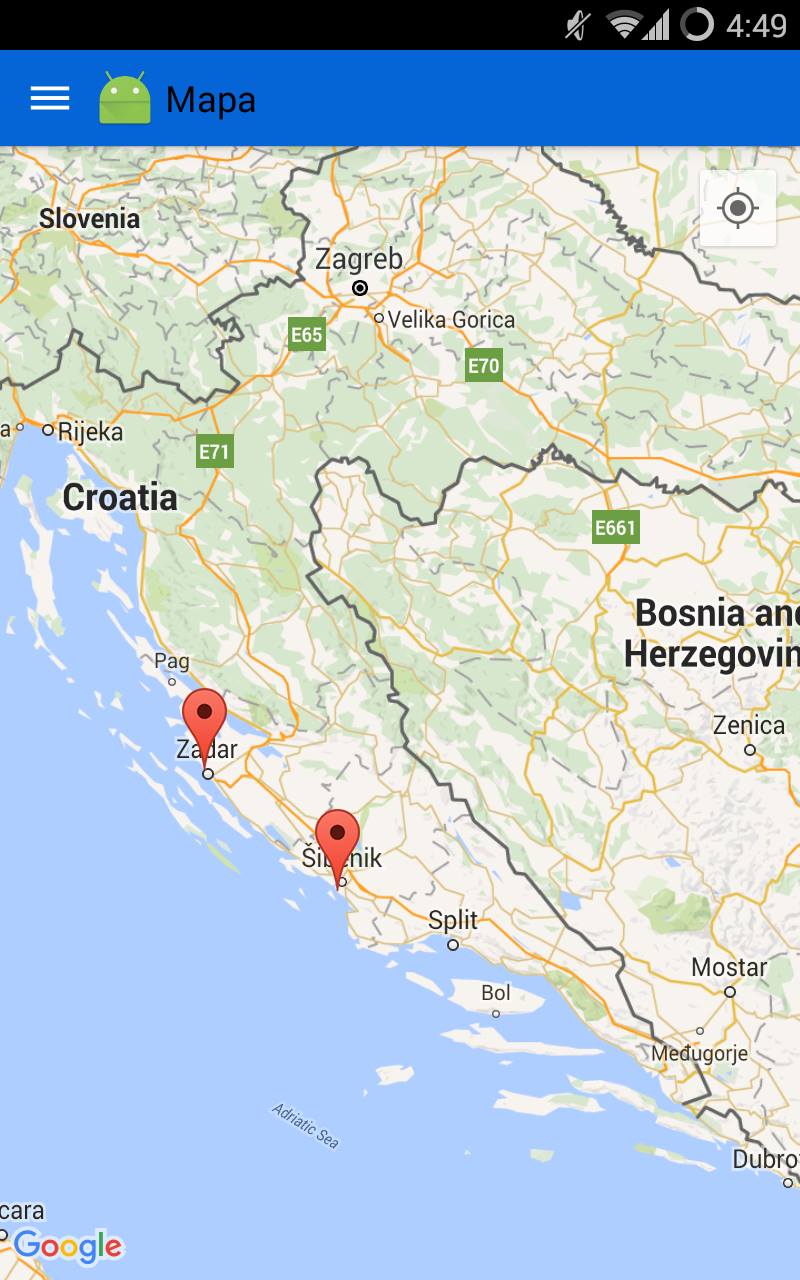
Ako Retrofit uspješno dohvati podatke sa poslužitelja pozvati će se void success(Apartment apartment, Response response) metoda, a dohvaćeni podatci biti će pohranjeni u apartment atributu metode. U suportnom, ako Retrofit neuspješno dohvati podatke, pozvati će se metoda void failure(RetrofitError error).

# 4. Aplikacija

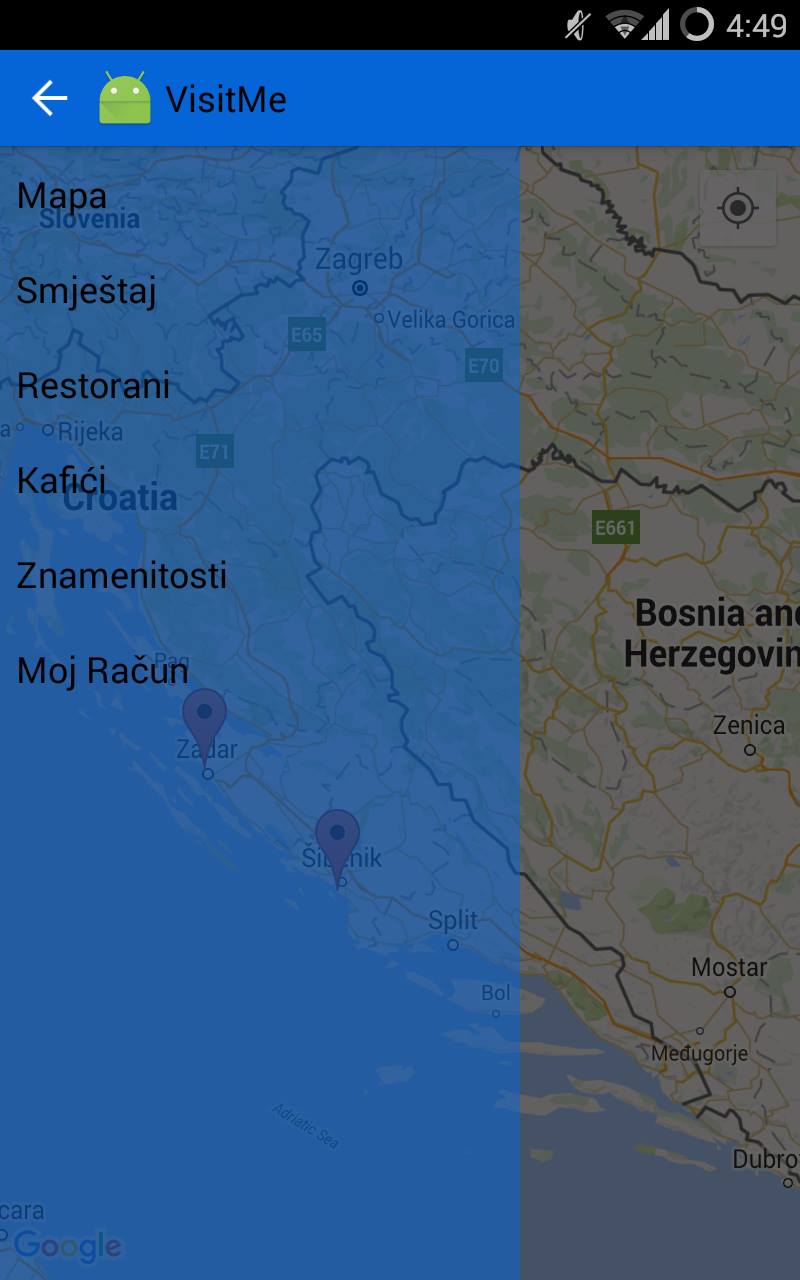
Na sljedećim slikama vide se 4 ekrana iz aplikacije. Svi podaci koji su trenutno uneseni u bazu su testni podaci, to jest izmišljeni. Kod testne prijave u sustav trenutno k.ime, lozinka su tim redoslijedom: admin, admin.



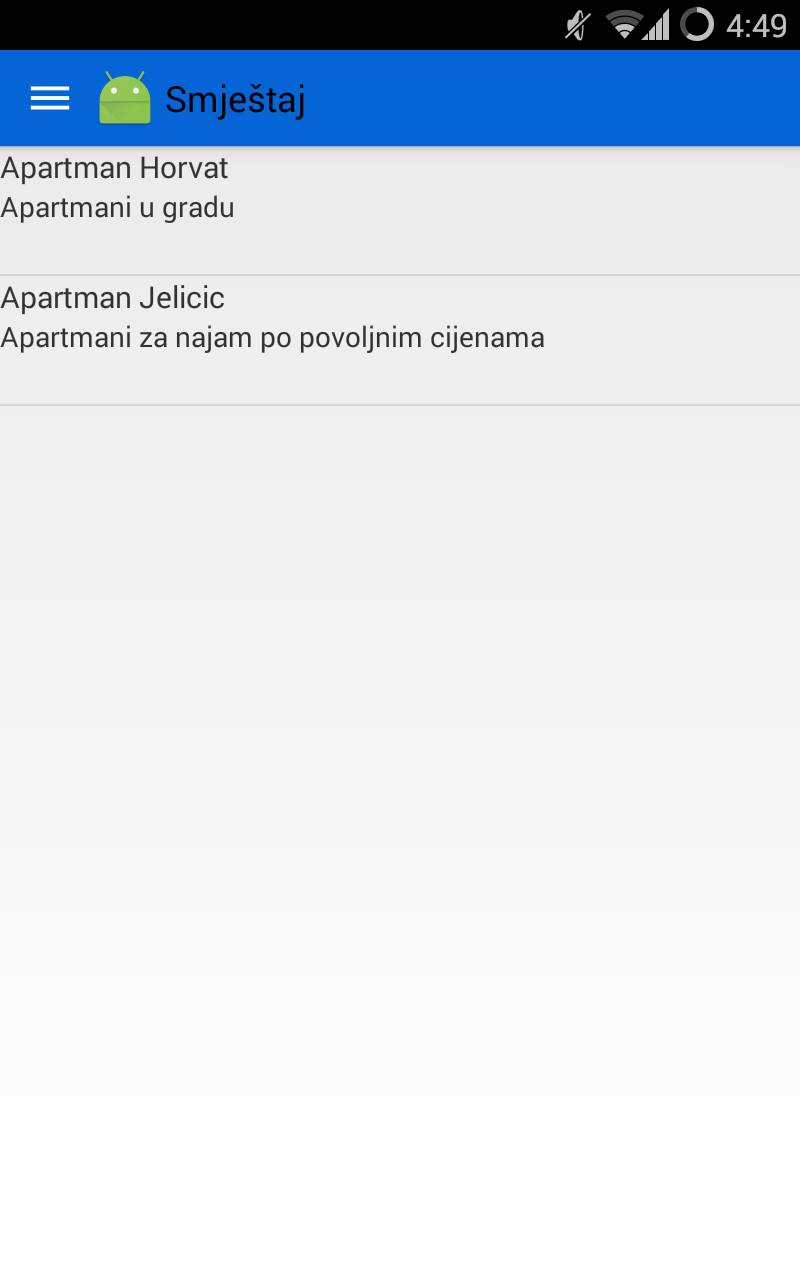
**Slika 4.1. Login ekran**

****

**Slika 4.2. Mapa**

****

**Slika 4.3. Menu**

****

**Slika 4.4. Lista**

1. Prikaz menu-a bi trebao biti preko liste, ali nismo uspjeli složiti u alatu [↑](#footnote-ref-1)
2. Kod prijave u sustav trenutno k.ime, lozinka su tim redoslijedom: admin, admin [↑](#footnote-ref-2)