Mobilna aplikacija za brigu oko kućnih ljubimaca

Papić, Kristijan

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:599879

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-11-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Stručni studij

Mobilna aplikacija za brigu oko kućnih ljubimaca

Završni rad

Kristijan Papić

Osijek, 2022.

SADRŽAJ

1. UVOD

Sadržaj ovog završnog rada sadržavati će detaljan opis zadatka mobilne aplikacije za brigu oko kućnih ljubimaca kao i njezinih zahtijeva te metode koje su se koristile kako bi se zadatak zadovoljio i na kraju pogled u konačni rezultat. Glavni cilj zadatka je bio izraditi mobilnu aplikaciju koja bi omogućila korisniku da pronađe osobu iliti drugog korisnika aplikacije koji će se brinuti za njegovog ljubimca na određeni način u slučajevima kada vlasnik ljubimca nije u mogućnosti. Vrsta brige koja će se pružati se može definirati na više načina. Od isključivo šetanja ljubimca do pune brige u paziteljevom domu.

Aplikacije je morala imati minimalno tri osnovne uloge:

1. Vlasnik
2. Pazitelj
3. Administrator

Uloge vlasnika i pazitelja rade na vrlo sličan način, kroz oglase. Vlasnika stvara oglas u kojem definira opis potrebne brige, ljubimca za kojeg je briga potrebna te ostale informacije vezane za način izvršavanje brige. Pazitelji se uz dogovor sa vlasnikom prijavljuju na oglas nakon čega vlasnik odabire željenog prijavljenog kandidata. Isti postupak je omogućen i za pazitelje. Pazitelj stvara oglas na kojem definira uslugu koju želi pružati i vlasnici ljubimaca se prijavljuju na taj oglas. Cijeli proces je popraćen sa ocjenjivanjem pri završetku oglasa. Vlasnik daje ocjenu usluge koju je pružao pazitelj i pazitelj daje ocjenu njegovog iskustva sa vlasnikom po više kriterija. Prosječne ocjene korisnika su vidljive na njihovim profilima i pri odabiru prijavljenih kandidata što olakšava odabir. Uz ulogu vlasnika i pazitelja tu je još dodatna uloga administratora koja ima mogućnost uklanjana neprimjerenih oglasa i korisnika takvih oglasa. U radu će biti prikazane metode i tehnologije koje su se koristile za ostvarivanje ovih zahtijeva za aplikaciju. Također proći će se kroz nekoliko sličnih aplikacija koje su već na tržištu te njihove razlike u odnosnu na ovaj zadatak. Te naravno na kraju proći će se kroz rezultat ovog rad što je naravno sama aplikacija.

1.1. Zadatak završnog rada

Mobilna aplikacija koja omogućuje korisniku koji je vlasnika kućnog ljubimca da pronađe osobu koje će se moći brinuti za navedenog ljubimca u slučaju da vlasnik nije u mogućnosti. Također pruža mogućnost korisnika da kao pazitelj ljubimaca pronađe vlasnika kojem će pružati uslugu brige za ljubimca.

2. SRODNE APLIKACIJE

2.1. Rover

Jedan od najuspješnijih primjera aplikacija za brigu oko kućnih ljubimaca je američka aplikacija Rover. Osnovana 2011. aplikacija i web stranica Rover povezuju roditelje pasa i mačaka s ljubavnim čuvarima i šetačima pasa u četvrtima diljem SAD-a, Kanade, Ujedinjenog Kraljevstva i Europe [1]. Sa preko milijun instalacija na Google Play trgovini jedna je od najpopularniji. Dostupna je kao android i kao IOS aplikacija i isto kao i ovaj rad zadatak joj je da spoji vlasnike kućnih ljubimaca sa potencijalnim paziteljima. Za razliku od aplikacije ovoga rada to spajanje je u ovom slučaju jednosmjerno. Vlasnici isključivo traže i biraju pazitelje. Dok pazitelji samo mogu čekati ponude ne mogu aktivno tražiti potencijalne vlasnike kojima bi pružali uslugu. Što bi značilo da je aplikacija glavnim dijelom fokusirana na vlasnike dok je aplikacija ovoga rada podjednako dijeli pozornost na vlasnike i pazitelje. No taj nedostatak kompenzira sa mnoštvom dodatnih funkcionalnosti kao izravno slanje poruka i slika unutar aplikacije, GPS praćenje pazitelja i ljubimca tijekom šetanja te integrirani sustav za plaćanje usluga. Neke od sličnosti koje dijeli sa aplikacijom ovog rada su mogućnosti ocjenjivanja pazitelja, definiranje usluga po različitim kategorijama te definiranje vrste ljubimaca kojima je potrebna briga. Iako također prikazuje ocijene korisnika isključivo vrijede za pazitelje dok kod aplikacije ovoga rada vlasnici također mogu dobivati ocijene koje se prikazuju na njihovom profilu. Još jedna razlika je to što pruža uslugu za isključivo pse i mačke dok aplikacija ovog rada dopušta bilo kakvu vrstu kućnog ljubimca.

2.2. Wag

Također američka aplikacija koja je dostupna isključivo u Americi. Sadrži slične funkcionalnosti kao i Rover i aplikacija ovoga rad: Ocjenjivanje korisnika, traženje usluga po kategorijama te definiranje vrste ljubimca. No kao i Rover ograničeno je samo na pse i mačke i također je glavni fokus samo na vlasnike. No posjeduje druge značajke koje ju odvajaju od konkurencije kao mogućnost usluge treniranja ljubimaca te dodate usluge kao šišanje i pranje ljubimaca kao i direktna veza sa veterinarima.[2] Također aplikacija funkcionira kao društvena mreža di korisnici imaju mogućnosti objavljivanja slika svojih ljubimaca na koje drugi korisnici onda mogu reagirati i dati svoja mišljenja komentiranjem.

3. TEHNOLOGIJE

Izvedba rada i tehnologije koje su se koristile se mogu podijeliti u tri glavna dijela:

- Aplikacija
- API
- Baza Podataka

Aplikacija je ono što korisnik vidi i sa čim izvršava interakcije. Baza podataka je mjesto u kojem se pohranjuju svi potrebni podaci o korisnicima i podaci potrebni za rad aplikacije. API je veza između aplikacije i baze podataka.

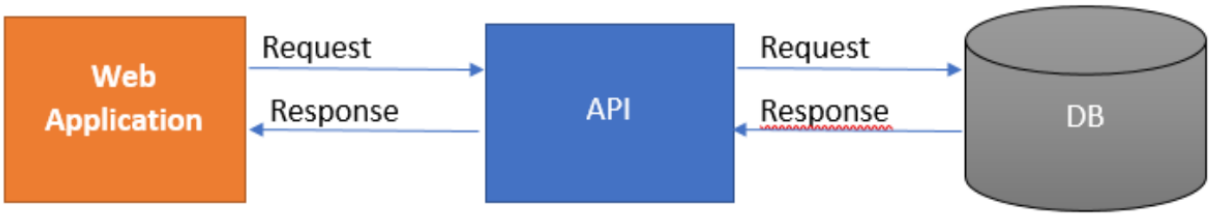
3.1. Aplikacija

Aplikacija je napravljena za android sustav pomoću Kotlin programskog jezika. Kotlin je opće namijenjen, besplatni, *open source*, statički tipkan „pragmatski“ programski jezik koji je prvobitno dizajniran za JVM (*Java Virtual Machine*) i Android koji kombinira objektno orijentirane i funkcionalne programske značajke [3]. Po sintaksi vrlo je sličan ostalim objektno orijentiranim programskim jezicima kao C# i Java sa kojom je međusobno izmjenjiv što bi značilo da se u Kotlin projektu može pisati Java kod. Kotlin je trenutno službeno podržan od strane Androida što i govori statistika da danas šezdeset posto android programera upravo razvija Android aplikacije u Kotlinu [4]. Što je i jedan od glavnih razloga zašto je Kotlin izabran za ostvarivanje aplikacije ovoga rada. Način razvijanja Android aplikacije je uglavnom naučen preko Android Developers portala koji sadrži razne tečajeve i uputstva za razvijanje Android aplikacija koje su većinom demonstrirane u Kotlin jeziku iako ima i u Java programskom jeziku. Kotlin je poprilično moderan po pitanju svoje jednostavnosti što ga čini izvrsnim za nove programere ali i činjenica da je po sintaksi sličan drugim objektno orijentiranim jezicima ga čini vrlo lakim za savladati od strane starijih programera.

Aplikacija je razvijana u službenom Android programskom okruženju Android Studio iz razloga što je također kao i Kotlin bio među glavnim prezentacijskim elementima u tečajevima na Android Developers portalu. Neke od prednosti su mu odličan sustav za sastavljanje pogleda aplikacije, odnosno rasporeda određenih komponenti aplikacije sa kojim korisnik obavlja interakcije, mogućnost stvaranja i pokretanja aplikacija na virtualnom Android uređaju kao i prevođenje Java koda u Kotlin kod.

3.2. API

API je skraćenica od *Application Programming Interface*. To je posredna aplikacija koja omogućuje dvije ili više aplikacija da međusobno komuniciraju.[4] U slučaju ovoga rada to je komunikacije između mobilne aplikacije i baze podataka u kojoj su pohranjeni potrebni podaci. On je tu kako bi ostvario sloj sigurnosti i skalabilnosti mobilnoj aplikaciji na način da odvaja aplikaciju od baze podataka. Aplikacija nikada nema direktan pristup bazi podataka. Svaki zahtjev za prikaz informaciji i spremanje novih informacija prolazi kroz API što znatno povećava razinu sigurnosti. Isto tako sa implementacijom API-a promjena baze podataka ne zahtijeva nikakve izmijene na mobilnoj aplikaciji.



Sl.3.1. Dijagram API rada

Za ovaj rad API je razvijen kao ASP.NET API u programskom okruženju Visual Studio 2022 u C# programskom jeziku. Implementacija API se dijeli u četiri funkcionalne cjeline:

- Kontroler
- Servis
- Repozitorij
- Modeli

Kontroleri su ti koji primaju HTTP zahtjeve od mobilne aplikacije i vračaju tražene podatke.

Servisi služe kao veza između Kontrolera i Repozitorija koji direktno komunicira sa bazom podataka i šalje ih preko servisa kontroleru koji ih čini dostupnim mobilnoj aplikaciji. Modeli služe kako bi API mogao protumačiti primljene podatke.

Ova tehnologija je izabrana iz razloga velikog stečenog iskustva za vrijeme stručne prakse i zbog toga što olakšava sam rad sa bazom podataka.

3.3. Baza Podataka

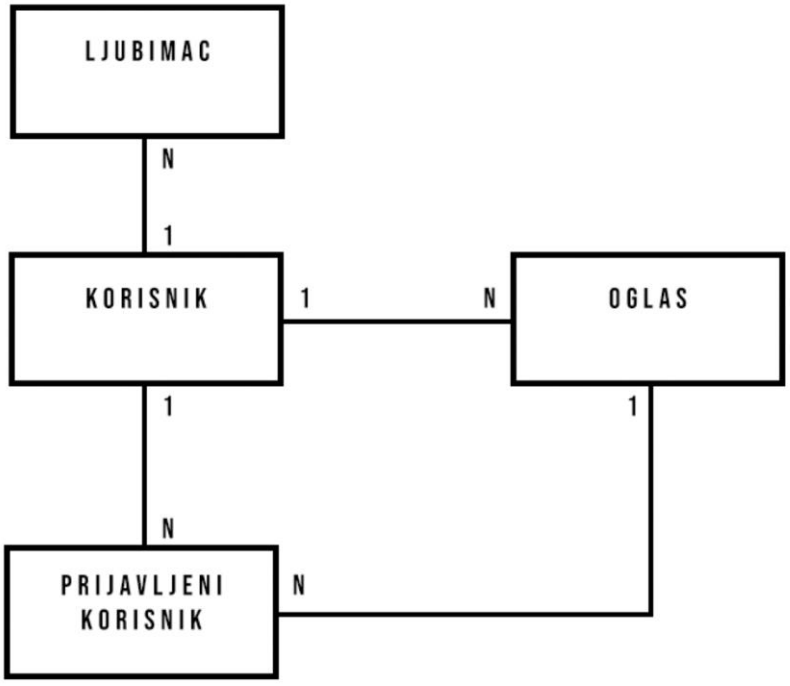
Baza podatak je temeljni dio aplikacije ovoga rada. Sadrži sve podatke potrebne za rad aplikacije i sve podatke koje upravo čine ono što korisnik vidi na svome ekranu kada upali aplikaciju. Korištena je Microsoft SQL server baza podataka iz razloga što je napravljena kao i API od strane Microsofta što osigurava veliku razinu kompatibilnosti. Također podržava klasične SQL naredbe na kojima je stečeno veliko iskustvo za vrijeme studija. Samo stvaranje i administracija baze je postignuta sa alatom Microsoft SQL Server Management Studio koji je također izabran zbog svoje kompatibilnosti sa ostalim tehnologijama.

4. RIJEŠENJA

4.1. Pohrana

Rješenje zadatka ovoga rada krenulo je od baze podataka. Baza je trebala omogućiti pohranjivanje konkretnih korisnika aplikacije te informacije o njihovim kućnim ljubimcima. Uz to trebala je pohranjivati informacije o pojedinim oglasima te prijavama na oglase od strane drugih korisnika. Sukladno navedenim zahtjevima kreirane su četiri tablice:

1. Korisnik
2. Ljubimac
3. Oglas
4. Prijavljeni korisnik



Sl. [4.1.] Dijagram baze podataka

Iz dijagram 4.1. može se vidjeti kako imamo tablicu korisnik koji ima vezu jedan naprema više na tablicu ljubimac. Naknadno tablica korisnik također ima vezu jedan naprema više sa tablicom oglas. Na kraju tu je još tablica prijavljeni korisnik koja ima referencu na korisnika koji se prijavio i oglas na kojem se obavila prijava.

```
1:      CREATE TABLE Pet(
2:      Id UNIQUEIDENTIFIER PRIMARY KEY,
3:      Name varchar(20),
4:      Kind varchar(40),
5:      Breed varchar(20) default '',
6:      Owner UNIQUEIDENTIFIER FOREIGN KEY REFERENCES [User](Id) ON DELETE CASCADE,
7:      );
```

Sl.4.2. Primjer koda za stvaranje SQL tablice

Gore navedeni kod 4.2. prikazuje postupak stvaranja tablice ljubimac. Možemo vidjeti da se tablica sastoji od identifikacijskog broja, imena ljubimca, vrste ljubimca, te veze na vlasnika.

```
1:      SELECT * FROM Pet
2:      WHERE Owner = '47CCF84E-36CD-4699-ADC9-1235DC6AED97';
```

Sl.[4.3.] Primjer koda za dohvaćanje podataka

U ovom primjeru koda 4.3. vidimo naredbu koja se često koristi za vrijeme rada aplikacije a to je dohvaćanje svih ljubimaca od nekog određenog korisnika, najčešće korisnika koji trenutno koristi aplikaciju.

4.2. Komunikacija sa bazom

Kao što je prijašnje objašnjeno glavnu komunikaciju između baze i aplikacije izvršava API. To radi na način da ima predefinirane adrese na koje kad dobije HTTP zahtjev zna značenje toga zahtjeva i kakvu informaciju ili kakvu operaciju mora napraviti za taj zahtjev. To se sve odvija pomoću kontrolera.

```
1:      public async Task<HttpResponseMessage> GetPetsAsync([FromUri]PetFilter filter = null)
2:      {
        List<PetViewModel> pets = mapper.Map<List<PetDomainModel>, List<PetViewModel>>(await
3:      petService.GetAllPetAsync(filter));
4:      return Request.CreateResponse(HttpStatusCode.OK, pets);
5       }
```

Sl.[4.4.] Primjer koda Get kontrolera

Iz primjera koda kontrolera 4.4. možemo vidjeti što točno jedan API zahtjev treba sadržavati. Kao prvi stvar iz imena kontrolera može se vidjeti da adresa na koju će se zahtjev slati mora imat „Pet“ u sebi te da mora biti tipa „Get“ i kao parametre prima filtar po kojemu će filtrirati rezultat.

Imamo više tipova HTTP zahtjeva u primjeru prikazani je „Get “ zahtjev koji se najčešće koristi za dohvaćanje podataka tu još imamo i „Post“ koji se koriti za unos podataka te uz mnoštvo ostalih još ima „Put“ i „Delete“ za uređivanje i brisanje podataka[5].

Primljene parametre filtera uzima i šalje na servis koji to prosljeđuje do repozitorij sloja koji dohvaća podatke primjenjujući filtar te rezultat vraća natrag do kontrolera koji ih onda šalje do aplikacija kao odgovor za zahtjev koji je poslala.

Filter koji je u ovom primjeru ulazni parametar je zapravo Model preko kojeg kontroler interpretira primljene podatke.

```
1:      public class PetDomainModel
2:        {
3:           private Guid id = Guid.NewGuid();
4:           private string name;
5:           private string breed = "";
6:           private string kind;
7:           private Guid ownerId;
8:        }
```

Sl.[4.5.] Primjer modela

Modele možemo koristit za različite namjene. U primjeru konkretni model se koristi za rad sa bazom podataka. Kao što se može primijetit iz koda 4.5. ima sve identične elemente kao i tablica ljubimac u bazi podataka. Modeli se još mogu koristit sa podatke koje će se isključivo prikazivati ili koji su namijenjeni za unos te kao u primjeru kontrolera mogu se koristit kako bi definirali nekakve filtre za podatke.

Komunikacija sa mobilnom aplikacijom funkcionira na način da za svaku potrebnu operaciju koja sa mora odraditi kako bi aplikacija funkcionirala postoji definirana adresa na koju se može poslati HTTP zahtjev sa određenim parametrima te ti traženi podaci ili samo informacija da je zahtjev uspješno izvršen dolazi kroz odgovor od tog zahtijeva.

4.3. Aplikacija

Glavna ideja stvaranja mobilne aplikacije je upravo slaganje različitih komponenti koje imaju određenu zadaći i funkcionalnost na smislen i dizajnerski zadovoljavajući način. Komponente mogu biti stvari kao Dugmadi, Liste, Slike, Meniji, Klizači i puno više. Te komponente se nalaze unutar jednog pogleda. Aplikacija može i najčešće ima više pogleda koji još mogu biti dodatno podijeljeni u fragmente. Aplikacija ovoga rada podijeljena je u tri glavna pogleda koji još dodatno sadržavaju pod poglede i fragmente

- Glavni pogled
- Pogled za objavu oglasa
- Pogled profila

U glavnom pogledu se nalaze liste oglasa. Pogled je podijeljen da dva dijela. Dio za dostupne oglase na koje se korisnik može prijaviti i aktivne oglase koji sadrži vlastite oglase od korisnika ili oglase na koje se prijavio.

Pogled za obaju oglasa radi upravo ono što mu je u imenu, objavljuje nove oglase. Bilo to oglas pazitelja ili vlasnika

Pogled profila služi kako bi prikazao sve informacije o trenutnom ili nekom drugom odabranom korisniku. Prikazuje stvari kao kontaktne informacije, ljubimce, opis te prosječne ocjene korisnika.

Definiranje pogleda se u programu Android studio može odvijati na dva način kroz interaktivno sučelje i kroz pisanje koda, konkretnije Xml-a

```
1:  <com.google.android.material.button.MaterialButton
2:      android:id="@+id/owner_post_button"
3:      android:layout_width="wrap_content"
4:      android:layout_height="wrap_content"
5:      android:layout_gravity="center|bottom"
6:      android:layout_marginVertical="20dp"
7:      android:text="Post"
```

Sl.[4.6.] Primjer definicije dugmeta

U primjeru 4.6. vidimo konkretno dugme koje se klikne kada se želi nekakav oglas objaviti. Glavne vrijednosti su mu identifikacijska oznaka sa kojim se onda dugme može programirati, širina i visina te tekst koji će pisati na dugmetu u ovome slučaju to je „Post“.

Koristeći identifikacijsku oznaku koju smo dugmetu odredili u definiciji možemo izvršavati željene funkcije na njemu. U primjeru ta funkcija je postavljanje slušača da dugme, tako da kada korisnik pritisne na dugme izvršiti će se metoda „postAd“.

```
1:    binding.ownerPostButton.setOnClickListener {
2:        postAd()
3:    }
```

Sl.[4.7.] Primjer programiranja dugmete

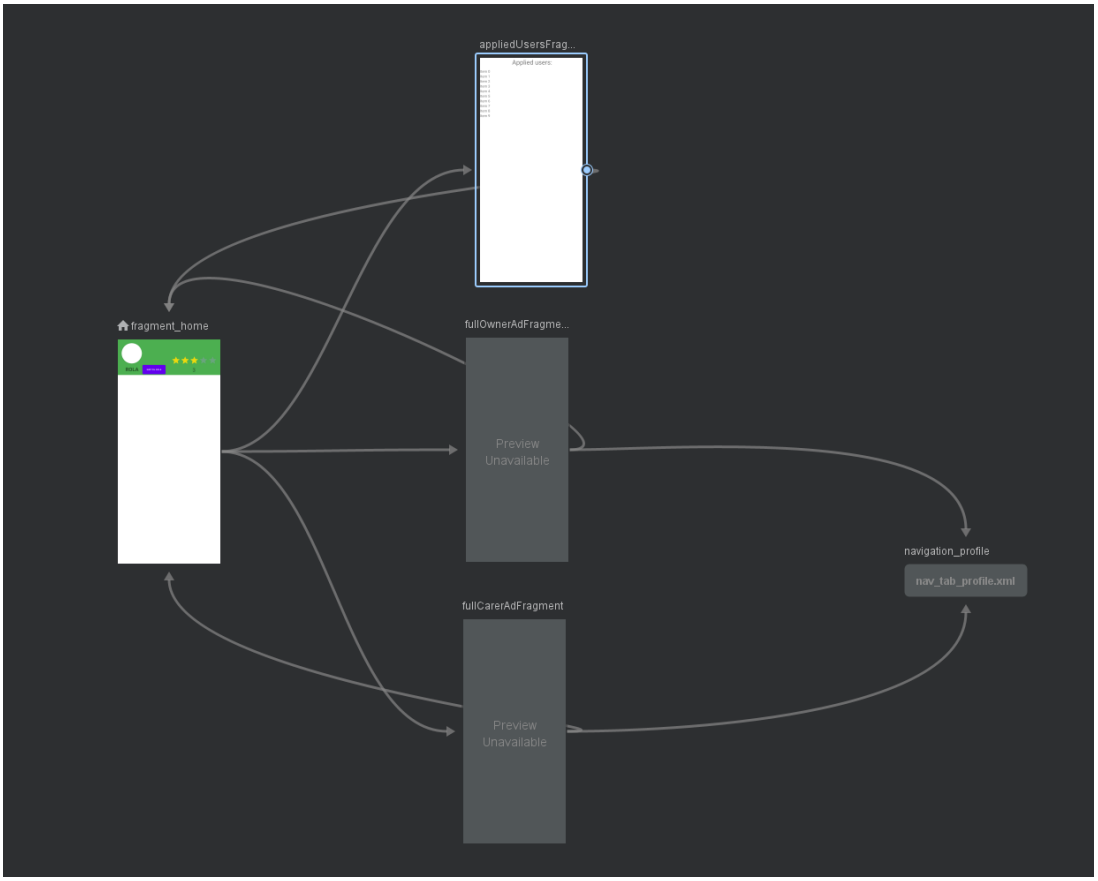
Iz primjera metode „postAd“ 4.8. možemo vidjeti jedu od uobičajenih radnji koje se izvršavaju za vrijeme korištenja aplikacije. Nakon što korisnik klikne dugme slušač se okida i poziva metodu iz primjera.

```
1:  private  fun postAd(){
2:      val pet = currentUser.user.value!!.Pets.filter
3:      { it.Name.equals(binding.petDropdown.text.toString()) }
4:      val ad = AdInput(1,binding.descInput.text.toString(),
5:      binding.jobSelectDropdown.text.toString(),pet[0].Id!!,currentUser.user.value!!.Id)
6:      CoroutineScope(Dispatchers.Main).launch { currentUser.postAd(ad)}
7:      val action = PostFragmentDirections.actionFragmentPostToHomeFragment()
8:      findNavController().navigate(action)
```

Sl.[4.8.] Primjer metode postAd

Prva stvar koja se odvija je sakupljaju se podaci koji će se poslati na API za dodavanje u bazu. Stvari kao ljubimac iz oglasa, opis oglasa, identifikacijska oznaka korisnika koji objavljuje oglas i tako dalje. To se odvija na isti način kao i postupak programiranja dugmeta koristeći identifikacijske oznake pozivaju se komponente iz pogleda koje su u ovome slučaju nekakvi izbornici i površine za unos teksta te se u pozivu izvlače vrijednosti koje su izabrane i upisane nakon čega se stvara zahtjev na API kako bi se ti podaci obradili i spremili u bazu podataka.

Ono što se na kraju metode odvija je također bitna stavka aplikacije a to je navigacija između različitih pogleda. U ovom slučaju to je navigacija između pogleda za objavljivanje oglasu u početni pogled di su prikazani svi oglasi.



Sl.[4.9.] Primjer navigacijskog grafa

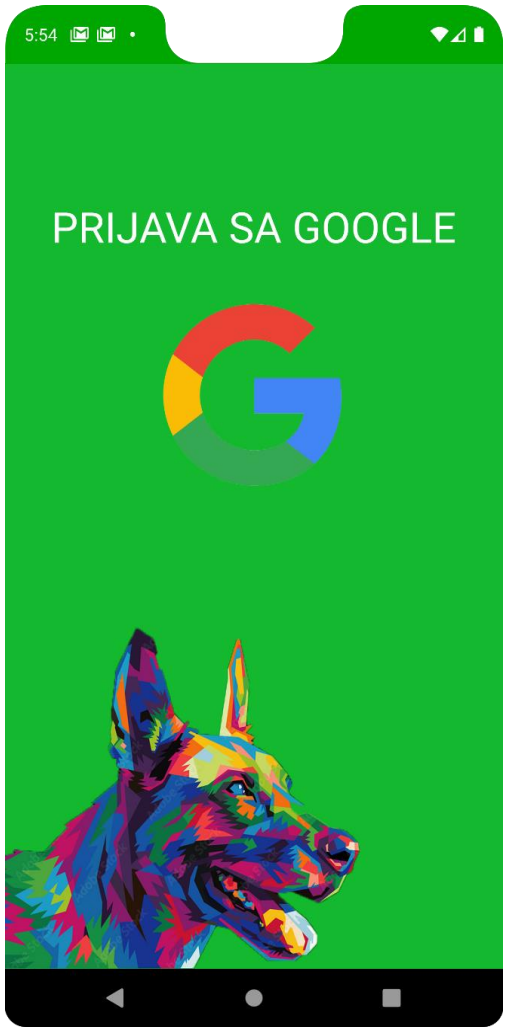
Najlakši način ostvarivanja navigacije je pomoću navigacijskih grafova koji se sa interaktivnim sučeljem mogu definirati unutar Android Studija. Navigacijski graf definira sve putanje od pogleda do pogleda kroz koje aplikacija može prolaziti za vrijeme njezinog rada. Koristeći definirane putanje korisnik može navigirati između različitih pogleda. Osim navigacije u putanjama se definiraju i podaci koju se mogu poslati u krajnji pogled. Na primjer kada se želi otvoriti profilni pogled nekog korisnika za vrijeme navigacije mora se poslati identifikacijska oznaka od korisnika za kojeg se želi otvoriti profil kako bi se mogli dohvatiti podaci i popuniti komponente pogleda dobivenim podacima.

Ostatak aplikacije radi na sličan način. Učitaju se komponente pogleda za korisnika nakon čega se čeka da korisnik pritisne određenu komponentu i da se odradi određena radnja, bilo to slanje podataka na API ili navigacija na neki drugi dio aplikacije.

5. REZULTAT

Nakon implementacije svih potrebnih tehnologija rezultat je mobilna aplikacija koja zadovoljava sve zahtjeve definirane na samom početku rada.

Pri prvom ulasku u aplikacijom prikazuje nam se uvodni pogled koji navodi da se prijavimo Google računom. Prijava se isključivo vrši sa Google računom iz razloge izvrsne razine sigurnosti takve metode prijave i olakšava korisniku početak rada sa aplikacijom jer ne mora otvarati novi račun jer je velika šansa da već posjeduje Google račun.

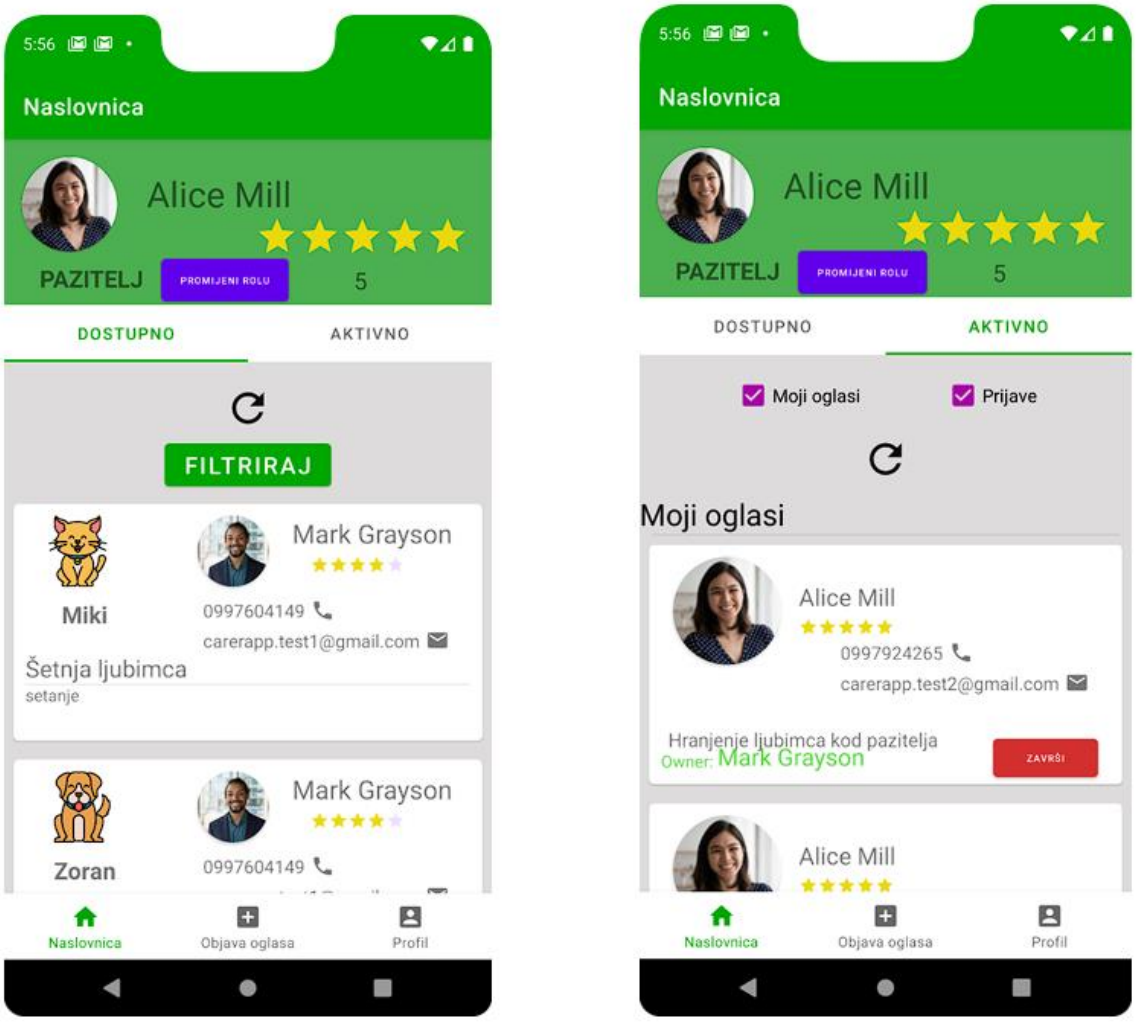


Sl.[5.1.] Uvodni pogled

Kao što je u uvodu definirano, aplikacija se može podijeliti u tri glavna djela. U ovome dijelu rada proći ćemo kroz te dijelove i načine kako su međusobno povezani

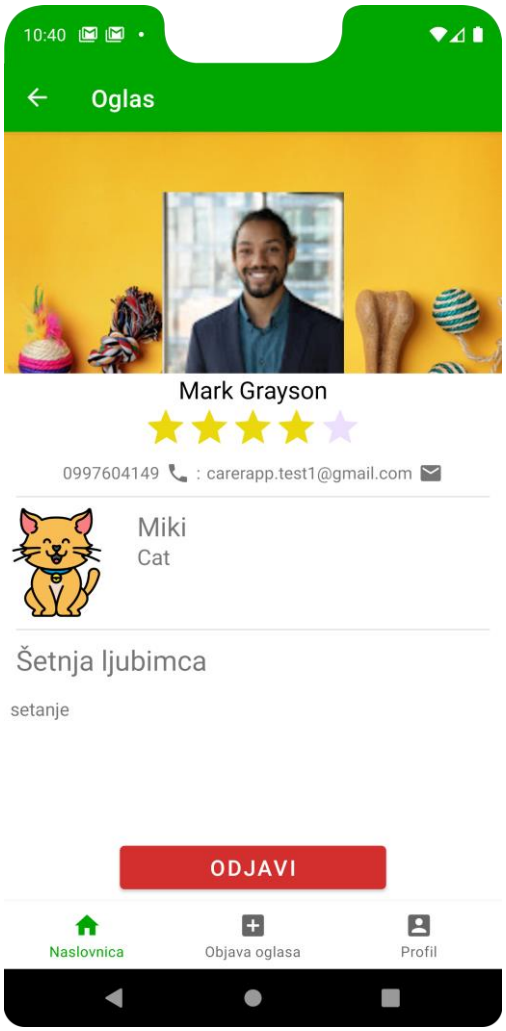
5.1. Glavni pogled

Glavni pogled služi za prikaz oglasa. Dijeli se na dva dijeli. Dostupni oglasni, odnosno oglasni od drugih korisnik na koje je moguće prijavit se. Aktivni oglasni odnosno oglasni pod vlasništvom trenutnog korisnika te oglasni na koje se trenutni korisnik prijavio.



Sl.[5.2.] Glavni pogled

Na vrhu glavnog pogleda imamo dio koji prikazuje informacije o trenutnome korisniku. Pokazuje ime, prosječnu ocjenu te trenutnu korisnika ulogu pored čega je dugme koje prebacuje iz jedne uloge u drugu. Ispod toga nalaze se lista oglasa. U slučaju dostupnih oglasa jednostavno su prikazani dostupni oglasi od drugih korisnika koji se pri pritiskom na njih otvaraju i prikazuje se mogućnost prijave na taj oglas. Dostupni oglasi se mogu filtrirati prema više kategorija kao vrsta posla, minimalna ocjena korisnika te u slučaju da se radi o oglasu vlasnika, vrsta ljubimca.



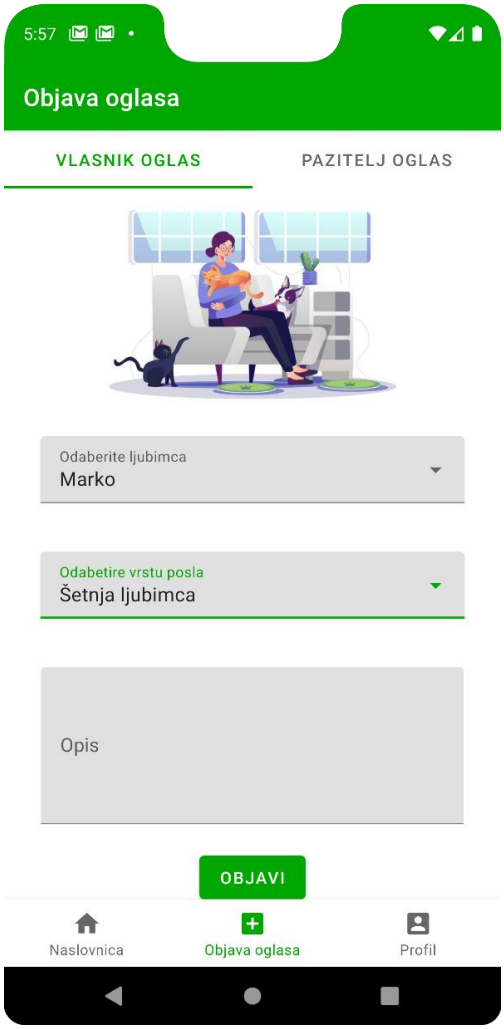
Pritiskom na oglas otvara se njegov puni pogled koji detaljnije prikazuje informacije o oglasu te pruža više mogućnosti za razliku od njegovog osnovanog pogleda u listu. Ovisno o korisniku kojemu se oglas prikazuje omogućuje prijavu ili odjavu sa oglasa te u slučaju da se radi o vlasniku oglasa pruža mogućnost uređivanja oglasa. Omogućuje još odlazak na profil od korisnika koji je napravio oglas kako bi se detaljnije prikazale informacije o njemu.

U slučaju aktivnih oglasa podijeljeni su u dvije kategorije. Vlastiti oglasi na kojima se dodatno prikazuje broj prijavljenih korisnika te mogućnost da se izabere jedan od prijavljenih korisnika kao i mogućnost da se obriše oglas te prijavljeni oglasi koji prikazuju oglas i status prijave te mogućnost da se poništi prijava.

5.2. Pogled objavljivanja oglasa

Izvor svih oglasa je upravo pogled za objavljivanje oglasa u kojemu korisnici imaju mogućnost stvoriti i objaviti svoj oglas koji će se prikazivati drugim korisnicima.

Pri stvaranju novog oglasa ovisno o vrsti oglasa korisnik određuje ljubimca za kojega se odnosi oglas, vrstu posla i upisuje opis sa potrebnim informacijama. Ako stvara oglas kao pazitelj upisuje samo vrstu posla i opis. Nakon što upiše sve podatke pritiskuje tipku post i objavljuje oglas nakon čega je vraćen nazad u glavni pogled

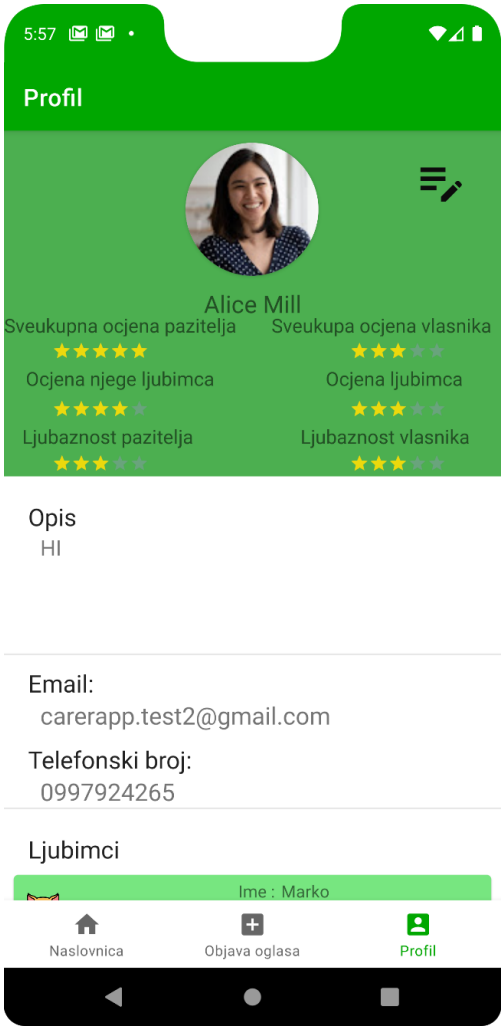


Sl.[5.3.] Pogled za objavljivanje oglasa

5.3. Pogled profila

Mjesto gdje korisnik može vidjeti informacija o svome računu ili informacije o nekom drugom korisniku je upravo pogled profila. Na svoj profil dolazi putem glavne navigacije na dnu ekrana a na neki drugi profil dolazi kada pritisne na korisnika u nekom od oglasa iz glavnog pogleda.

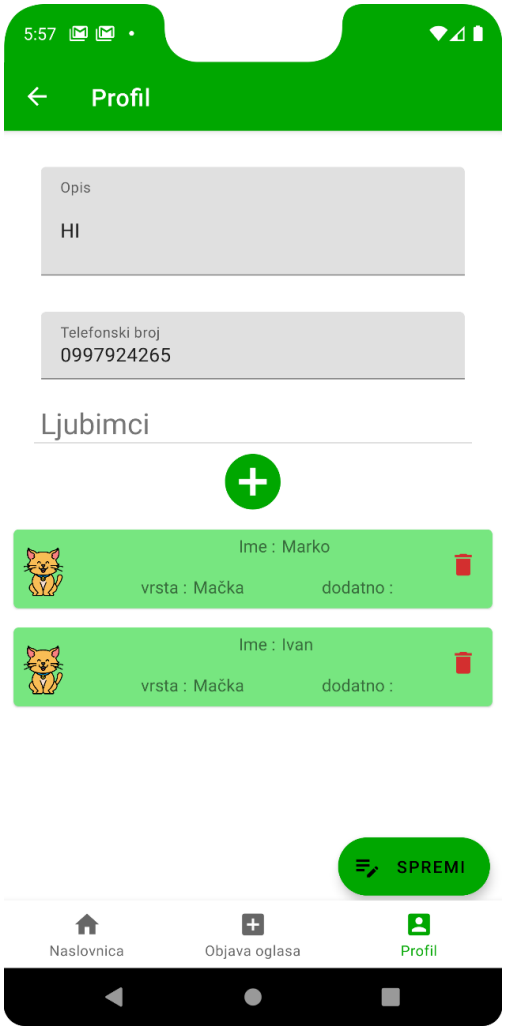
U ovome pogledu možemo vidjeti da za obje uloge imamo tri kategorije ocjena i sve su prikazane u ovom pogledu za obje uloge. Osim toga prikazuje se još opis, kontaktne informacije, te ljubimci od toga korisnika.



Sl.[5.4.] Pogled profila

Osim samo pregleda informacija korisnik koji se nalazi na svome profilu ima mogućnost uređivanja svojih informacija. Pritiskom na ikonu u gornjem desnom kutu pogleda otvara se pogled uređivanja profila.

U tom pogledu korisnik ima mogućnost uređivati opis i kontaktne informacije kao i dodavati nove ljubimce na profile te uklanjati stare. Pri završetku pritisne se dugme „spremi“ i korisnik je vraćen na svoj pogled profila.



Sl.[5.5.] Pogled uređivanja profila

6. ZAKLJUČAK

Iz rezultata mobilne aplikacije vidljivo je da su zadovoljeni svi zahtjevi za nju. Korisnik kroz svoj vlastiti oglas ili oglas na koji se prijavi može pronaći osobu koja će se brinuti za njegovog ljubimca. Dodatno pazitelj na isti način može pronaći ljubimca za kojeg će se brinuti. Pri kraju svakog oglasa korisnici si mogu davati ocjene po više kategorija koje su onda prikazan na njihovim profilima. Također korisnici koju dobiju uloge administratora mogu brisati oglase i korisnike.

Prednosti ovakvog načina funkcionalnosti sa ulogama je to što aplikacija nije fokusirana ni na jednu ulogu već različite uloge dijele istu količinu pozornosti aplikacije. Aplikacija nije tu da isključivo pruža uslugu vlasnicima ljubimaca već podjednako pruža uslugu i paziteljima. Nedostatak tome je što korisnici koji će aplikaciju isključivo koristiti kao vlasnik ili isključivo kao pazitelj će možda smatrati da taj cijeli drugi dio aplikacije koji nikada neće koristit da je ne potreban ili možda čak i zbunjujući za ukupni rad aplikacije.

Uz sve to vjerujem da uz daljnje proširenje funkcionalnosti aplikacije i njene optimizacije za različite uređaje da može postati konkurentni proizvod na ogromnom tržištu mobilnih aplikacija.

7. LITERATURA

[1] Rover https://www.rover.com/about-us/

[2] Wag https://wagwalking.com/

[3] Kotlin overview, Android Developers https://developer.android.com/kotlin/overview

[4] Develop Android apps with Kotlin https://developer.android.com/kotlin

[5] What is an API?, Mulesoft https://www.mulesoft.com/resources/api/what-is-an-api

[6] HTTP request methods https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

8. SAŽETAK

Glavni zadatak rada je bio napraviti mobilnu aplikaciju koja omogućuje korisnicima da kao vlasnik ljubimca pronađe osobu koja će se brinuti za ljubimca u slučaju da vlasnik nije u mogućnosti. Isto tako omogućuje korisnicima da kao pazitelj pronađu vlasnika kojem će pružati uslugu brige oko ljubimca. To je postignuto krom mobilnu aplikaciju koja implementira oglase koje korisnici mogu stvarati i na koje se mogu prijavljivati. Vlasnici oglasa odabiru prijavljenog korisnika i usluga započinje. Korisnici si na kraju usluge daju međusobno ocijene po više kategorija u obliku zvjezdica od kojih je prosjek vidljiv na njihovom profilu te je vidljiv pri odabiru prijavljenih kandidata što olakšava odluku. Korisnik stvara oglas na temelju ljubimca kojeg dodaje i sprema na svoj profil, te po vrsti posla koji se određuje po više kategorija. Korisnik pristupa aplikaciji putem Google računa što olakšava sam pristup aplikaciji jer ne mora izrađivati novi račun koji je isključivo namijenjen za aplikaciju. Sve to zajedno čini mobilnu aplikaciju koja zadovoljava sve zahtijeva ovoga rada.

Ključne riječi: Mobilna aplikacija, Android, API, Ljubimci, Briga

„Mobile application for pet care“

Summary

The main task of this paper was to create a mobile application that allows users that are pet owners to find someone who will take care of their pet in case the owner is unable. It also allows users to find an owner as a caregiver to whom they will provide a pet care service. It was accomplished through a mobile application that implements ads that users can create and sign up for. Ad owners select an applied user and the service begins. At the end of the service, users also give each other ratings in several categories in the form of stars, the average of which is visible on their profile and is visible when selecting applied candidates, which makes the decision easier. The user creates an ad based on the pet that he adds and saves to his profile, and by the type of job that is determined by several categories. The user accesses the application via a Google account, which makes it easier to access the application because he does not have to create a new account that is exclusively intended for the application. All this together makes a mobile application that meets all the requirements of this paper.

Keywords: Mobile application, Android, API, Pets, Care

9. ŽIVOTOPIS

Autor ovog rada je student stručnog studija Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek i prijašnji učenik Elektrotehničke i prometne škole Osijek, Kristijan Papić. Ljubitelj informatike i elektrotehnike. Kroz studiranje i kroz iskustvo rada kao software developer razvio je vještine za izradu ovog rada.

_____

Potpis autora