Usluga za prijavljivanje i udomljavanje ljubimaca

Duraković, Jelena

Master's thesis / Diplomski rad

2016

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:200:777378

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Diplomski studij

USLUGA ZA PRIJAVLJIVANJE I UDOMLJAVANJE LJUBIMACA

Diplomski rad

Jelena Duraković

Osijek, 2016.

Sadržaj

1. UVOD 1
1.1 Zadatak rada 1
2. PREGLED KORIŠTENIH TEHNOLOGIJA 2
2.1 Internetska stranica 2
2.1.1 HTML5 2
2.1.2 CSS3 3
2.1.3 JavaScript, jQuery i AJAX 4
2.1.4 PHP 6
2.1.5 MYSQL 7
2.2 Android aplikacija 8
2.2.1 Operativni sustav Android i Android aplikacije 8
2.2.2 Java 10
3. INTERNETSKA STRANICA 12
3.1 Realizacija internetske stranice 12
3.2 Rad sa Google kartama 18
4. ANDROID APLIKACIJA 22
4.1 Realizacija Android aplikacije 22
4.2 Komunikacija sa poslužiteljem 27
4.3 Rad sa Google kartama 30
5. ZAKLJUČAK 33
LITERATURA 34
SAŽETAK 36
ABSTRACT 36
ŽIVOTOPIS 37

1. UVOD

Posljednjih par godina svjedoci smo porasta broja volontera koji spašavaju izgubljene i napuštene kućne ljubimce, pružaju im privremene i pronalaze stalne domove. Volonteri su trenutno primorani koristiti društvene mreže za objavljivanje oglasa. Nakon pronalaska, problem često predstavlja način na koji predočiti točnu lokaciju životinje. Ideja za izradu web stranice i Android aplikacije nastala je iz razloga što trenutno ne postoji usluga koja bi pružila aktivistima jednostavniji pristup i način objavljivanja oglasa. Korisnici prilikom objavljivanja oglasa ispunjavaju obrazac u kojem opisuju pronađenu životinju. Osim unošenja naziva i opisa, omogućena je i klasifikacija oglasa po već definiranim kriterijima, te unošenje trenutne lokacije pomoću Google karti. Izradom usluge ostvaren je univerzalni način objavljivanja oglasa čime je olakšana njihova pretraga i skraćeno vrijeme pronalaska i udomljavanja životinja.

U drugom poglavlju dan je pregled tehnologija korištenih prilikom razvoja web stranice i Android aplikacije. Pružen je kratak opis pojedine tehnologije, te su objašnjeni osnovni pojmovi. Nakon pojašnjavanja tehnologija, u trećem poglavlju analizira se pristup dizajnu i realizaciji web stranice, dok se u četvrtom poglavlju analizira pristup Android aplikaciji. Oba poglavlja detaljnije objašnjavaju funkcionalnosti kojima korisnik ima pristup i načini na koji su ostvarene. Kako se kod korištenja Google karti radi o posebnom aplikacijskom programskom sučelju, u trećem i četvrtom poglavlju poseban značaj se daje baš toj tehnologiji.

1.1 Zadatak rada

Izraditi aplikaciju za Android platformu pomoću koje korisnik može slikati napuštenu životinju (ljubimca) te poslati sliku i lokaciju poslužitelju. Omogućiti dohvaćanje podataka na mobilni uređaj i prikaz korisniku. Za poslužitelj napraviti aplikaciju koja prihvaća podatke sa mobilnih uređaja i pohranjuje u bazu podataka.

2. PREGLED KORIŠTENIH TEHNOLOGIJA

2.1 Internetska stranica

2.1.1 HTML5

HTML, punog naziva *Hyper Text Markup Language*, je prezentacijski jezik za izradu internetskih stranica. Najnovija inačica službeno je izdana 2014. godine pod nazivom HTML5. Prema [1], jedan od glavnih temelja pete inačice je podrška već postojećeg sadržaja, što znači da je sadržana većina specifikacija i implementacija njegovog prethodnika. Na početku svakog HTML dokumenta definirana je korištena inačica. Izdavanjem pete inačice stvorena je jedinstvena deklaracija koja ne zahtjeva pisanje poziva na dokumente izdane od strane W3C-a (engl. *World Wide Web Consortium*) (Sl. 2.1).

```
<!DOCTYPE html>
```

***Sl. 2.1** HTML5 deklaracija*

Unutar takve deklaracije sadržana je podrška za sve atribute i elemente koji su sadržani u prethodnim inačicama. Prema [2], također je sadržana i podrška za nove semantičke, strukturne i multimedijske elemente. Semantički i strukturni elementi, kao što su primjerice *nav, header* ili *footer*, omogućuju jasno definiranje sadržaja internetske stranice. Takvi elementi najčešće služe kao zamjena za korištenje generičkih blokova, primjerice <*div*>. Kako s vremenom HTML5 postaje standard, semantički, strukturni i multimedijski elementi sve su više sadržani u okvirima otvorenog tipa kao što je Bootstrap. Osim prethodno navedenih elemenata, trenutna verzija omogućava korištenje elemenata za glazbu, video, te grafičke elemente. Neki od novo dodanih elemenata prikazani su u tablici Tab 2.1.

***Tab 2.1** HTML5 elementi*

<article>	Definira članak
<audio>	Definira standard za sviranje audio datoteka
<canvas>	Definira područje unutar koje je moguće crtati grafike
<details>	Definira dodatne podatke koje korisnik može prikazati ili skriti
<dialog>	Definira blok ili prozor sa dijalogom
<footer>	Definira donji dio dokumenta ili dijela stranice
<header>	Definira gornji dio dokumenta ili dijela stranice
<nav>	Definira poveznice navigacije
<progress>	Predstavlja napredak zadatka
<section>	Definira dio dokumenta
<video>	Definira standard za prikazivanje video datoteka

Prema [3], HTML5 je dizajniran za stvaranje bogatog sadržaja bez potrebe za korištenjem dodatnih alata. Kombinirajući ga s funkcionalnostima CSS3, moguće je stvarati takozvane odzivne (engl. *responsive*) stranice koje se prilagođavaju veličini ekrana uređaja preko kojeg se pristupa stranici. HTML-ova podrška odzivnog dizajna najviše dolazi do izražaja kada se uzme u obzir da je u posljednjih desetak godina primjetan porast broja korisnika koji pristupaju internetu preko mobilnih uređaja. Na [4] se kao jedna od prednosti HTML5 navodi smanjenje vremena potrebnog za razvoj stranica. Programerima je omogućeno korištenje najnovijih tehnologija i fokusiranje na novije Internet preglednike, pritom ne gubeći funkcionalnosti stranice na starijim verzijama preglednika. Uz sve veću podršku na internet preglednicima na mobilnim uređajima i računalima, HTML5 samo potvrđuje svoj status najboljeg rješenja pri stvaranju univerzalnih internetskih stranica.

2.1.2 CSS3

CSS, punog naziva *Cascading Style Sheets*, je jezik korišten za formatiranje i opisivanje internetskih stranica pisanih u prezentacijskom jeziku kao što je HTML. Prema [5], CSS je prvenstveno dizajniran kako bi se omogućilo odvajanje strukture dokumenta od njegove prezentacije, uključujući boje, raspored na samoj stranici, izgled teksta, itd. Odvajanjem dviju komponenti programerima je olakšano održavanje i nadograđivanje stranica. CSS3 je najnovija inačica CSS-a. Prema [6], naziv „CSS3“ ne predstavlja samo treću inačicu, već i treću razinu razvoja CSS specifikacija. Za razliku od svojih prethodnika, gdje se set značajki nalazio u jednom dokumentu, u trećoj inačici taj set je podijeljen u zasebne dokumente poznate kao moduli. Takvim se pristupom, prema [7], ostvaruje puno lakše i efikasnije nadograđivanje individualnih komponenti. Modulima je moguće dodavati nove funkcionalnosti i svojstva, pritom zadržavajući kompatibilnost sa prethodnim inačicama. Razine razvoja modula i CSS-a kao cjeline su međusobno nezavisne što znači da je već sada moguće pronaći module u četvrtoj razini razvoja. Prema [8], modularnost pogoduje razvojnim programerima jer im je omogućena implementacija pojedinih dijelova čim postanu dostupni i prije nego se CSS3 kao cjelina smatra završenim. Pregled dostupnih modula dan je na [9]. Neki od važnijih modula su medijski upiti (engl. m*edia query*), odabirnici (engl. *selectors*), animacije (engl. *animations*), pozadine i obrubi (engl. *background and borders*) itd.

Medijski upit je modul koji omogućuje stvaranje odzivnih stranica spomenutih u prethodnom poglavlju. Korištenjem medijskih upita uvjetuje se primjena određenih stilova u ovisnosti o razlučivosti zaslona uređaja preko kojeg se pristupa stranici bez potrebe za smanjivanjem sadržaja.

Ovaj modul se pokazao posebno korisnim proteklih par godina upravo zbog porasta broja pristupa internetu preko uređaja s manjim razlučivostima ekrana, kao što su mobiteli i tableti. Programeri sve više stavljaju prioritet na stvaranje stranica koje dobro izgledaju na većem broju uređaja, umjesto na stvaranje stranica posebno namijenjenih mobilnim uređajima. Primjena jednostavnog medijskog upita prikazana je slikom 2.2.

```
@media screen and (max-width: 480px){
background: blue;
}
```

Sl. 2.2 *Jednostavni medijski upit*

Ovim medijskim upitom definirana su dva uvjeta: zaslon (engl. *screen)* i maksimalna širina (engl. *max-width*). Promjena boje pozadine na plavu primijenit će se ukoliko je širina ekrana manja od 480px.

CSS3 omogućava dodavanje dinamičnih i dekorativnih svojstava običnoj HTML stranici. Razvojem novih modula olakšano je dodavanje svojstava koja prije nisu bila ostvariva bez kompliciranih kodova. Efekti, kao što su zaobljeni uglovi, gradijenti ili animacije, koji su prije zahtijevali primjenu više tehnologija odjednom, sada su ukomponirani u CSS3. CSS3 postaje sve popularniji jer čini uređivanje stranica puno lakšim, te uklanja potrebu za korištenjem drugih jezika, primjerice JavaScript, ili drugih programa, kao što je Flash.

2.1.3 JavaScript, jQuery i AJAX

JavaScript je jezik za skriptiranje i interpretiranje internetskih stranica. Implementiran unutar drugog jezika, veže se uz objekte svoje okoline i omogućava kontrolu nad njima. Trenutno je jedini jezik za skriptiranje koji je podržan na svim većim internetskim preglednicima. Korišten je za stvaranje interakcije sa grafičkim sučeljem stranice, odnosno ostvarivanje dinamičnih i interaktivnih internetskih stranica. Kod pisan JavaScript jezikom naziva se skripta. Tako napisane skripte mogu biti dva tipa: prevedene od strane preglednika (engl. *Client-side*) i prevedene od strane poslužitelja (engl. *Server-side*). Skripte prevedene od strane preglednika proširuju temeljni skup objekata sa objektima za kontrolu preglednika i njegovim objektnim modelom dokumenta. Tim proširenjem omogućeno je dodavanje elemenata koji reagiraju na korisničke radnje kao što su pritisak tipke miša, navigacija stranicom itd. Skripte prevedene od strane poslužitelja proširuju skup objekata sa objektima za pokretanje skripte na poslužitelju omogućavajući tako aplikaciji komunikaciju sa bazom podataka, manipuliranje datotekama na poslužitelju i slično. Sve veći

zahtjev za dinamičnijim i ljepšim stranicama potaknuli su razvoj brojnih JavaScript okvira (pr. Angular.js) i JavaScript biblioteka (pr. JQuery). Kako je prilikom izrade rješenja diplomskog rada korišten jQuery, nadalje će se u tekstu više pažnje posvetiti baš toj biblioteci.

jQuery je mala i brza JavaScript biblioteka dizajnirana s ciljem pojednostavljivanja pretrage unutar HTML dokumenta, rukovanja događajima, animiranja i Ajax interakcija za brzu izradu internetskih stranica. Prema [10], prije nastanka jQuery-a programeri su morali sami pisati vlastite okvire i biblioteke radi zaobilaženja mogućih pogrešaka. Suradnjom programera koji su svoje okvire objavljivali pod licencom otvorenog koda, nastaje jQuery zaklada. jQuery zaklada je neprofitno trgovačko društvo za Internet programere. Kao cilj postavljaju poboljšavanje otvorenog Interneta kroz razvoj i podršku programske podrške otvorenog koda, čineći ga tako dostupnim svima. Najveća razlika između JavaScript-a i jQuery-a je činjenica da je jQuery optimiziran za rad s različitim preglednicima što čini implementaciju puno sigurnijom. Jedna od najvećih prednosti korištenja ove biblioteke je brzina kojom kod može biti pisan.

```
a) $('body').css('background', '#000000');

b) function changeBackground(color) {
     document.body.style.background = color;
   }
   onload="changeBackground('black');"
```

Sl. 2.3 *Usporedba jQuery (a) i JavaScript (b) koda*

Na slici 2.3 prikazani su primjeri jQuery i JavaScript kodova. Iako izvršavaju istu radnju, JavaScript kod pod točkom b) puno je duži od jQuery koda pod točkom a). Osim dužine koda, nedostatak JavaScript koda je i potrebno vrijeme provjere mogućih pogrešaka koje se mogu pojaviti na različitim preglednicima kao što je Internet Explorer.

AJAX je skraćenica za *Asynchronous JavaScript and XML*, odnosno asinkroni JavaScript i XML. AJAX je novija tehnika za stvaranje bržih, boljih i interaktivnijih internetskih stranica i aplikacija. Prema [11], koristi XHTML za sadržaj, CSS za prezentaciju, zajedno sa objektnim modelom dokumenta (engl. *DOM - Document Object Model*) i JavaScriptom za dinamički prikaz sadržaja. Standardne internetske stranice i aplikacije koriste sinkrone zahtjeve prema poslužitelju za prenošenje i prihvaćanje informacija. Koristeći AJAX, prilikom predaje podataka, JavaScript šalje zahtjev poslužitelju, interpretira vraćene rezultate i ažurira trenutni zaslon. Korisnik tako neće

dobit uvid u činjenicu da je uopće uspostavljena komunikacija sa poslužiteljem.

```
var data = $('#login-form').serialize();
  $.ajax({
            type: 'POST',
            url: '../php/ajaxLogin.php',
            data: data,
            success: function (response) {
                alert("Uspješna prijava.");
            },
            error: function (response) {
                alert("Dogodila se pogreška.");
            }})
```

***Sl. 2.4** Primjena AJAX i jQuery koda*

Slikom 2.4 prikazana je jednostavna primjena jQuery *ajax()* metode. Prije slanja asinkronog zahtjeva na server, dohvaćeni su podaci iz forme koja sadrži podatke za prijavu. Dohvaćeni podaci su tada predani *ajax()* metodi. Definiran je tip slanja, te adresa na koju će podaci biti poslani. U slučaju uspješne prijave ili pojave greške, korisnik je obavješten prikazom odgovarajuće poruke.

Kombiniranjem ovih triju tehnika programeri stvaraju dinamične, brze stranice koje pružaju izvrsno korisničko iskustvo.

2.1.4 PHP

PHP (engl. *Hypertext Preprocessor*) je jezik otvorenog izvornog koda koji služi za skriptiranje i posebno je pogodan za razvoj internetskih stranica. Prema [12], ono što razlikuje PHP od drugih jezika za skriptiranje, kao primjerice JavaScript čiji kod se prevodi od strane preglednika, je činjenica da se PHP kod izvodi na poslužitelju, te se tako generira HTML kod koji se prenosi prema klijentu. Izvršavanjem koda kao izlaz ne mora uvijek biti HTML, podržane su slike, PDF datoteke, čak i Flash filmovi. Umjesto ispisivanja, takve datoteke mogu biti automatski generirane i spremljene u datotečni sustav. Osim izvršavanja koda na strani poslužitelja, PHP skripte se mogu koristiti unutar komandnih linija ili za pisanje računalnih aplikacija. Korištenjem komandne linije moguće je pokretati skriptu bez korištenja poslužitelja ili preglednika. Takva vrsta upotrebe idealna je za skripte koje je potrebno često izvršavati pomoću *cron-a* na Linuxu ili *Task schedulera* na Windowsima. Kod može biti ugrađen u HTML ili se može koristiti u kombinaciji s različitim sustavima za upravljanje sadržajem internetskih stranica ili postojećim okvirima i tehnologijama, kao što su već spomenuti jQuery i AJAX. PHP podržava funkcionalno i objektno orijentirano programiranje. Jedna od najvećih prednosti je podrška velikog broja baza podataka. Stvaranje internetske stranice s pristupom bazi podataka izuzetno je lagano koristeći neko od specifičnih

proširenja (primjerice za MySQL) ili pomoću apstrakcijskog sloja kao što je PDO. PHP kod potrebno je pisati unutar oznaka <?PHP i ?>. Slika 2.5 prikazuje jednostavnu PHP skriptu ugrađenu u HTML kod. Skripta će u ovom slučaju kao izlaz predati tekst koji će tada biti ispisan unutar tijela stranice.

```
<html>
      <head></head>
      <body>
            <?php echo "Hello World"; ?>
      </body>
 </html>
```

Sl. 2.5 *Jednostavna PHP skripta ugrađena u HTML kod*

2.1.5 MYSQL

U današnje vrijeme za pohranu i upravljanje velikom količinom podataka koriste se relacijski sustavi za upravljanje bazom podataka (engl. *RDBMS – Relational Database Management Systems*). Naziv relativna baza proizlazi iz činjenice da su svi podaci pohranjeni u različite tablice, te su im odnosi određeni korištenjem primarnih i stranih ključeva. MySQL je jedan od najraširenijih i najpoznatijih relacijskih sustava na svijetu. Skraćenica MySQL kombinacija je imena kćeri jednog od osnivača My, te skraćenice za strukturirani jezik upita (engl. *Structured Query Langugage*). Prema [13], baza podataka je posebna aplikacija koja pohranjuje zbirku podataka. Svaka baza podataka ima jedno ili više aplikacijskih programskih sučelja za stvaranje, pristupanje, pretragu i repliciranje podataka koje sadrži. MySQL je svoju popularnost stekao zbog brojnih prednosti kao što su činjenica da je izdan pod licencom otvorenog izvornog koda, sposobnosti brzog i sigurnog upravljanja velikim bazama podataka, podršci na velikom broju operativnih sustava i jezika (pr. PHP, C, Java, itd) i dopuštanje programerima uređivanje izvornog koda.

```
CREATE TABLE IF NOT EXISTS products (
          productID     INT UNSIGNED  NOT NULL AUTO_INCREMENT,
          name          VARCHAR(30)   NOT NULL DEFAULT '',
          price         DECIMAL(7,2)  NOT NULL DEFAULT 99999.99,
          PRIMARY KEY   (productID)
        );
```

Sl. 2.6 *Primjer koda za stvaranje tablice*

Slika 2.6 prikazuje kod izvršavanjem kojeg se stvara tablica unutar baza podataka. Definirani su nazivi stupaca unutar tablice, te primarni ključ.

2.2 Android aplikacija

2.2.1 Operativni sustav Android i Android aplikacije

Android je mobilni operativni sustav temeljen na jezgri operativnog sustava Linux. U [14] je definiran kao platforma otvorenog izvornog koda projektirana za mobilne uređaje. Android razdvaja sklopovlje mobilnog uređaja od programske podrške koja se na njemu izvršava i time omogućava da mnogo veći broj uređaja izvršava iste aplikacije. Prvi put na tržištu se pojavljuje 2008. godine na mobilnom uređaju T-Mobile G1, te je od tada izdano šest verzija operativnog sustava. Prema [15], operativni sustav se sastoji od brojnih Java aplikacija i Java biblioteka koje se izvode u objektno orijentiranom okruženju. Podržava 2D i 3D grafički prikaz i uobičajene audio i video formate, te omogućava pristup dijelovima uređaja kao što su kamera i senzori. Prema [16], mogućnost stvaranja aplikacija koje izgledaju dobro na različitim uređajima i koriste puni potencijal sklopovlja uređaja, lako dostupno integrirano razvojno okruženje, te otvoreno tržište za objavljivanje aplikacija samo su neke od prednosti za korištenje Android operativnog sustava.

Android aplikacije (engl. *application*) pisane su u Java programskom jeziku, o kojem će biti više govora u idućem potpoglavlju. Kôd se pomoću Android razvojnog okruženja (engl. *SDK – Software Development Kit*) prevodi u Android paket (engl. *APK – Android package*). Paket u sebi sadrži sve dijelove aplikacije i služi za instalaciju aplikacije na uređaj. Prema [17], aplikacija se sastoji od četiri komponente: aktivnosti (engl. *activity*), servisi (engl. *service*), pružatelji usluga (engl. *content providers*) i primatelji emitiranja (engl. *broadcast recievers*). Aktivnost predstavlja jedan prikaz zaslona sa korisničkim sučeljem. Iako funkcioniraju zajedno za stvaranje cjelovite aplikacije, aktivnosti su međusobno nezavisne.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Sl. 2.7 *Primjer kôda za aktivnost*

Na slici 2.7 prikazan je osnovni kôd pomoću kojeg je stvorena aktivnost. Ovakva aktivnost nema nikakvih specifičnih funkcionalnosti, već joj je samo definirano korisničko sučelje pozivanjem funkcije *setContentView()*. Funkciji se predaje identifikator resursa za izgled korisničkog sučelja.

Servisi su komponente koje u pozadini izvode dugačke operacije ili operacije za udaljene procese. Za razliku od aktivnosti, servis nema korisničko sučelje. Pružatelji usluga upravljaju dijeljenim skupom podataka. Podaci spremljeni u trajna mjesta pohrane (sustav datoteka, baza, itd.) dostupni su drugim aplikacijama preko pružatelja usluga. Primatelji emitiranja su komponente koje odgovaraju na emitiranje poruka od strane drugih aplikacija ili sustava. Sve komponente aplikacije, izuzev pružatelja usluga, se pozivaju preko asinkrone poruke zvane *intent*. *Intent* definira što je potrebno izvesti, te se dijeli na eksplicitni i implicitni. Pozivanjem eksplicitnog *intent*-a definirana je komponenta koju sustav treba pozvati, dok je pozivanjem implicitnog definirana radnja koju je potrebno izvršiti.

```
Intent activityIntent = new Intent (MainActivity.this,
OtherActivity.class);
startActivity(activityIntent);

Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(cameraIntent, REQUEST_CODE);
```

***Sl. 2.8** Primjeri intent-a*

Slikom 2.8 prikazana su dva načina definiranja i pozivanja *intenta*, gdje *activityIntent* predstavlja eksplicitni, a *cameraIntent* implicitni *intent*. Prvim načinom definiran je prijelaz iz jedne aktivnosti u drugu, dok je drugim definirana akcija pozivanja kamere. Kao što je vidljivo iz slike, u drugom načinu aktivnost je pokretana pomoću funkcije *startActivityForResult()* kojoj su predani *intent* i kôd koji služi kao identifikator zahtjeva. Takvim pozivom su po završetku akcije dobiveni podaci, u ovom slučaju fotografija.

Osim osnovne četiri komponente, aplikacija sadrži manifest i resurse. Resursi mogu biti animacije, crteži (engl. *drawable*), izgled korisničkog sučelja unutar aktivnosti (engl. *layout*), nizovi (engl. *string*), itd. Manifest sadrži osnovne informacije o aplikaciji kao što su popis komponenti i dopuštenja (engl. *Permission*).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hr.etfos.durakovic.testapp">

<uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        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>
```

***Sl. 2.9** Primjer manifesta*

Na slici 2.9 prikazan je jednostavni manifest aplikacije koja se sastoji od samo jedne aktivnosti. Osim definiranja svojstava aktivnosti, kao što su tema i naziv, unutar manifesta definirano je i dopuštenje za pristup internetu.

2.2.2 Java

Java je prema [18] definirana kao objektno orijentirani programski jezik opće namjene koji se koristi u svim industrijama i za gotovo sve vrste aplikacija. Dizajnirana je kako bi se omogućilo razvijanje prenosivih aplikacija visokih performansi. Čineći aplikacije dostupnima preko različitih okruženja, prema [19], tvrtke mogu uz minimalne troškove pružiti više usluga, time povećavajući produktivnost krajnjih korisnika i olakšavajući suradnju i komunikaciju.

Java je objektno orijentirani jezik, što znači da ima konstrukcije za predstavljanje objekata iz stvarnog svijeta. Svaki Java program ima najmanje jednu klasu koja sadrži metode i polja.

```
public class Image{
	String path;

	public Image(String path){
			this.path = path;
			}

	public String getPath(){
		return path;
	}
}
```

Sl. 2.10 *Primjer Java kôda i klase*

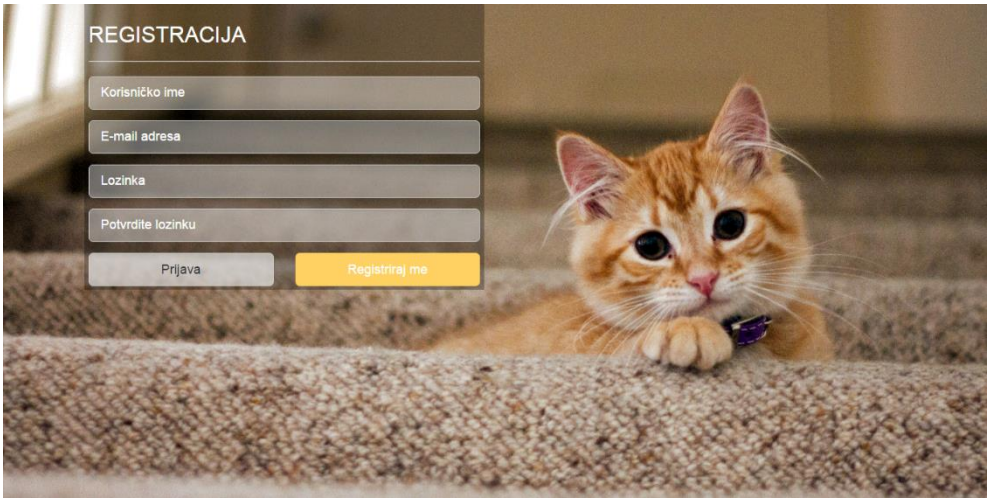
Na slici 2.10 prikazan je jednostavan Java kôd koji se sastoji od klase, njenog konstruktora, te metode za dohvaćanje vrijednosti polja. Kako bi Java kôd mogao biti izvršen preko Java tumača, izvorni kôd, odnosno datoteke sa ekstenzijom *.java*, prevode se u format zvan *bytecode* (datoteke sa ekstenzijom *.class*). Prema [20], prevedeni Java kôd moguće je pokrenuti na većini računala zbog toga što su Java tumači i izvršna okruženja, odnosno Java Virtualni Strojevi (engl. *VM – Virtual Machine*), sadržani u većini operativnih sustava, uključujući UNIX, Windows i Macintosh OS. Unutar Android sustava virtualni stroj za tumačenje kôda dugo vremena je bio Dalvik, no nedavno je tu ulogu preuzeo ART, punog naziva Android Runtime. Iako oba stroja koriste isti *bytecode*, korištenjem ART-a poboljšavaju se performanse aplikacija. Java kôd moguće je pisati u bilo kojem uredniku, no za prevođenje su potrebni alati i biblioteke uključene u Java razvojnom okruženju (engl. *JDK – Java Development Kit*). Prema [18], neovisnost Jave o platformi proizlazi iz činjenice da Java program nema uvid pod kojim se operativnim sustavom izvodi. Program se izvršava unutar ranije instaliranog okruženja zvanog Java izvršno okruženje (engl. *JRE – Java Runtime Enviroment*). Uzevši u obzir sve prednosti Java programskog jezika, kao najveća prednost ističe se činjenica da je Android operativni sustav baziran baš na tom jeziku. Sve većim razvojem visokih tehnologija koje koriste Android, kao što su pametni televizori ili satovi, Java postaje jedan od najtraženijih programskih jezika.

3. INTERNETSKA STRANICA

U ovome poglavlju dan je pregled korištenih kodova za stvaranje internetske stranice uz koje su priložene slike krajnjeg izgleda stranice. Poseban značaj u drugom potpoglavlju daje se radu s Google kartama preko koji korisnik unosi trenutnu lokaciju pronađene ili posljednju poznatu lokaciju izgubljene životinje.

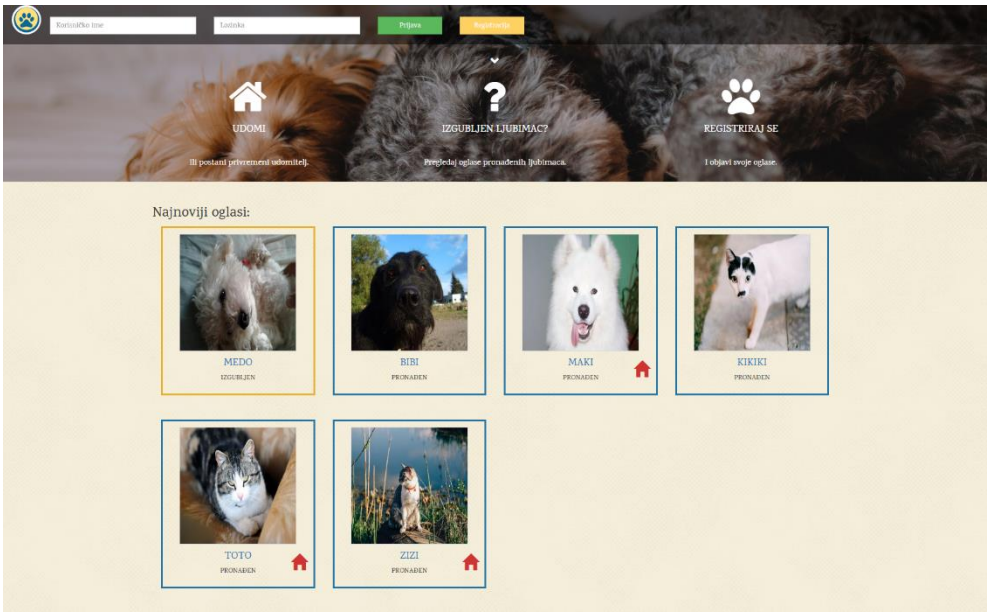
3.1 Realizacija internetske stranice

Korisnicima stranice omogućeno je objavljivanje i brisanje oglasa, pretraživanje do sada objavljenih oglasa te kontaktiranje korisnika koji su objavili oglas. Kako bi korisnik dobio pristup objavljivanju oglasa, potreban je korisnički račun.



Sl. 3.1 *Prikaz formulara za registraciju*

Slikom 3.1 prikazan je obrazac koji korisnici ispunjavaju pri stvaranju računa. Od korisnika se zahtjeva unos jedinstvenog korisničkog imena, adrese električne pošte, te lozinke. Nakon uspješne prijave, korisnicima na unesenu adresu pristiže e-pošta s poveznicom za potvrđivanje prijave, te im je sada omogućena prijava. Prijava je moguća koristeći zasebnu stranicu sličnu onoj za registraciju, ili pomoću obrasca koji se uvijek nalazi unutar navigacije pri vrhu stranice. Slikom 3.2 prikazana je početna stranica. Na početnoj stranici prikazana je nekolicina najnovijih oglasa, te navigacija. Unutar navigacije smještene su poveznice za profil korisnika, objavu novog oglasa, pristup pretrazi oglasa, te poveznica za odjavu. Kako u ovome slučaju nijedan korisnik nije prijavljen, unutar navigacije smješten je obrazac za prijavu.



Sl. 3.2 *Prikaz glavne stranice*

Iako se nalaze ispod službene navigacije, njenim dijelom mogu se smatrati i tri točke koje novom korisniku daju uvid u usluge pružene korištenjem stranice. Dinamičnosti stranice pridonosi nasumično dodjeljivanje pozadine navigaciji. Svakim ponovnim otvaranjem stranice korisniku se prikazuje jedna od šest mogućih fotografija. Ovakva navigacija se ponavlja i na ostatku stranice, a ostvarena je kombinacijom objašnjenih tehnologija u prethodnom poglavlju. Najbitniji dio korisničkog iskustva navigacije je mogućnost prijave na račun neovisno o trenutnoj stranici. Takva prijava, čiji se kod nalazi na slici 3.3, ostvarena je korištenjem jQuery metode za AJAX, te PHP-a za komunikaciju sa poslužiteljem.

```
$(document).on('click', '#btnLogin', function (e) {
    e.preventDefault();
    var data = $('#login-form').serialize();
    $.ajax({
        type: 'POST',
        url: '../php/ajaxLogin.php',
        data: data,
        success: function (response) {
            loginResponse(response);

        },
        error: function (response) {
            console.log('Greska ' + JSON.stringify(response)); }})});
```

Sl. 3.3 *jQuery kod za prijavu*

Pritiskom na gumb za prijavu pokreće se funkcija unutar koje se pristupa podacima. Pozivanjem funkcije *serialize()*, podaci unutar obrasca pretvoreni su u tekstualni niz zapisan u standardnoj

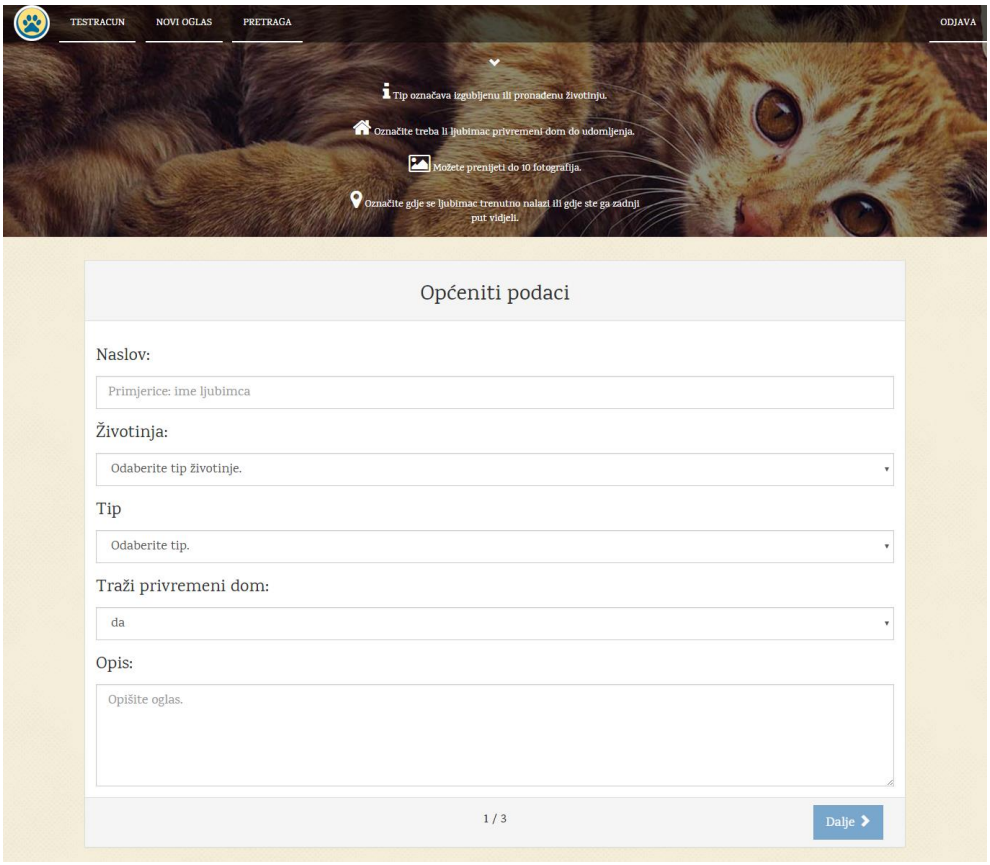
URL notaciji. Tako zapisani podaci su pomoću AJAX metode predani PHP skripti koja tada provjerava postoji li korisnik s takvim korisničkim imenom, te poklapaju li se uneseni podaci s onima unutar baze podataka. U slučaju uspješno izvedene skripte, pozvana je funkcija koja obavještava korisnika o poslužiteljevu odzivu.

Nakon uspješne prijave, korisnicima je omogućen pristup stranici za objavljivanje oglasa. Objava oglasa podijeljena je na tri dijela: unos općenitih podataka, odabir fotografija i unos lokacije. Podjela oglasa na tri dijela ostvarena je raspoređivanjem pojedinih dijelova obrasca unutar blokova * *, koji su zatim skrivani i prikazivani u ovisnosti o pritisnutim gumbima. Slikom 3.4 prikazan je kod pomoću kojeg je ostvareno takvo korisničko sučelje. U ovom slučaju, prikazan je prijelaz sa dijela s općenitim podacima na dio s fotografijama.

```
$("#next-images").click(function () {
    $(".postform").hide("slow");
    $("#images").show("slow");
});
```

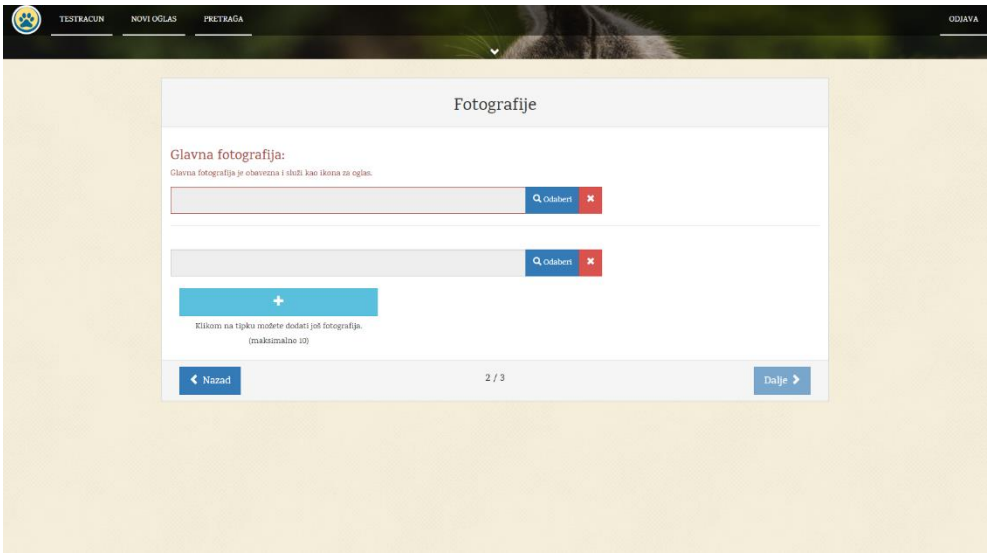
Sl. 3.4 *Kod prijelaza sa prvog dijela oglasa na drugi*

U prvom dijelu od korisnika se traži unos imena i vrste ljubimca, traži li privremeni dom do udomljenja, te je li životinja trenutno izgubljena ili joj je poznata lokacija. Kao što je prikazano slikom 3.5, korisnik ima i mogućnost unosa opisa. Omogućavanjem unosa opisa, korisnicima je dana sloboda dočaravanja stanja i osobnosti ljubimca kako bi se ubrzalo udomljavanje. Unutar navigacije su nadodani kratki naputci za ispunjavanje oglasa.



Sl. 3.5 *Prvi dio objavljivanja oglasa*

U drugom dijelu korisnicima je dopušten odabir fotografija. Kako ne bi došlo do prijenosa prevelike veličine podataka, te kako bi se definirao standard izgleda oglasa, broj fotografija je ograničen na deset. Radi povećanja korisničkog iskustva i izbjegavanja praznih oglasa, od korisnika je zahtjevan prijenos minimalno jedne fotografije koja je kasnije korištena kao ikona. Prilikom pristupanja tom dijelu objave oglasa, korisnicima je odmah dostupan odabir glavne, te gumb pomoću kojeg se na stranicu pridodaje još izbornika za slike. Ukoliko korisnik odluči da je dodano previše dodatnih fotografija, pritiskom na crvenu tipku uklanja se određeni red, odnosno izbornik slike. Opisana funkcionalnost prikazana je slikom 3.6.



Sl. 3.6 *Drugi dio objave oglasa*

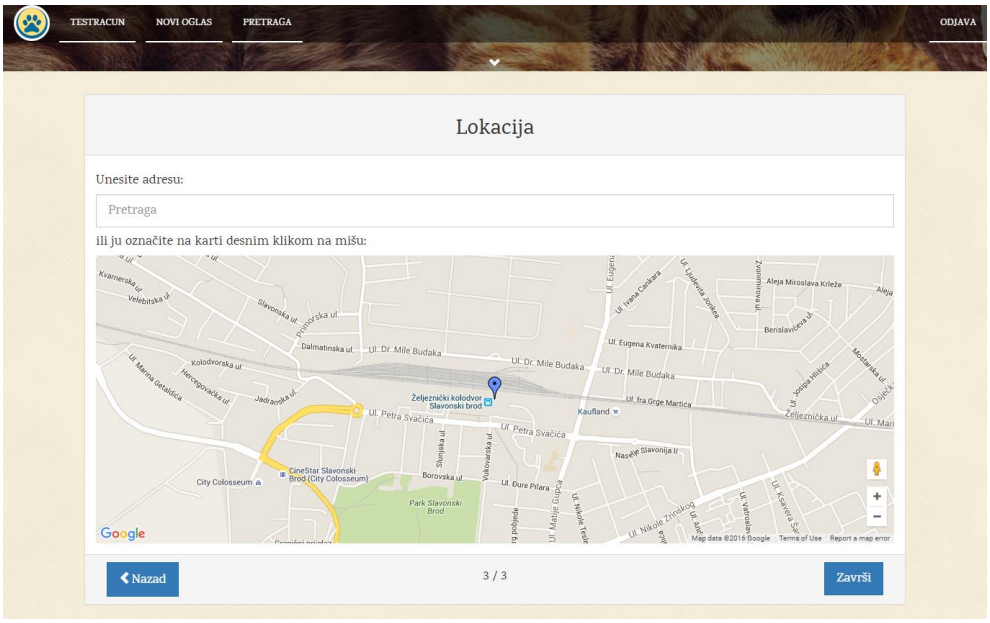
Pridodavanje i uklanjanje izbornika sa slike ostvareno je pisanjem jQuery koda. Pridodavanje je ostvareno spremanjem HTML koda potrebnog za prikaz izbornika u varijablu. Tako pripremljeni kod je zatim nadodan na blok predviđen za izbornike. Kod uklanjanja izbornika potrebno je odrediti točan izbornik na koji se uklanjanje odnosi.

```
$(document).on('click', '#remove', function (e) {
    e.preventDefault();
    $(this).closest('.maindiv').remove();
    if ($('#add-more').is(':disabled')) {
        $('#add-more').prop('disabled', false);
    }
    maxappend--;
    return false;
});
```

Sl. 3.7 *Uklanjanje dodatnog izbornika za slike*

Kodom prikazanim na slici 3.7 ostvarena je funkcionalnost uklanjanja izbornika. Pomoću *$(this)* automatski se dohvaća element kojim je pokrenuta funkcija, te se pronalazi blok najbliži pritisnutom gumbu. Uklanjanjem pronađenog bloka sa stranice se uklanjaju se i svi dijelovi izbornika. Identifikatorom *#add-more* definiran je gumb za dodavanje dodatnih izbornika. Kako je broj fotografija ograničen, definiran je brojač *maxappend*. Dostizanjem maksimalnog broja, gumb za dodavanje dodatnih izbornika postaje onesposobljen te je zbog toga potrebna provjera i ponovna aktivacija gumba prilikom smanjivanja iznosa brojača.

U trećem dijelu korisnik unosi trenutnu ili zadnju poznatu lokaciju. Korisniku je omogućena pretraga adresa, te direktan odabir lokacije pritiskom desne tipke miša na karti. Kako stranica na kojoj je objavljen rad nema dostupan protokol za sigurnu komunikaciju, odnosno HTTPS, onemogućen je pristup trenutnoj lokaciji na kojoj se korisnik nalazi. Tako će prilikom otvaranja trećeg dijela, korisniku na karti biti prikazana najbliža lokacija. Ukoliko korisniku odgovara prikazana lokacija, pritiskom na prikazani marker ju potvrđuje. O radu s Google kartama bit će više govora u idućem poglavlju.



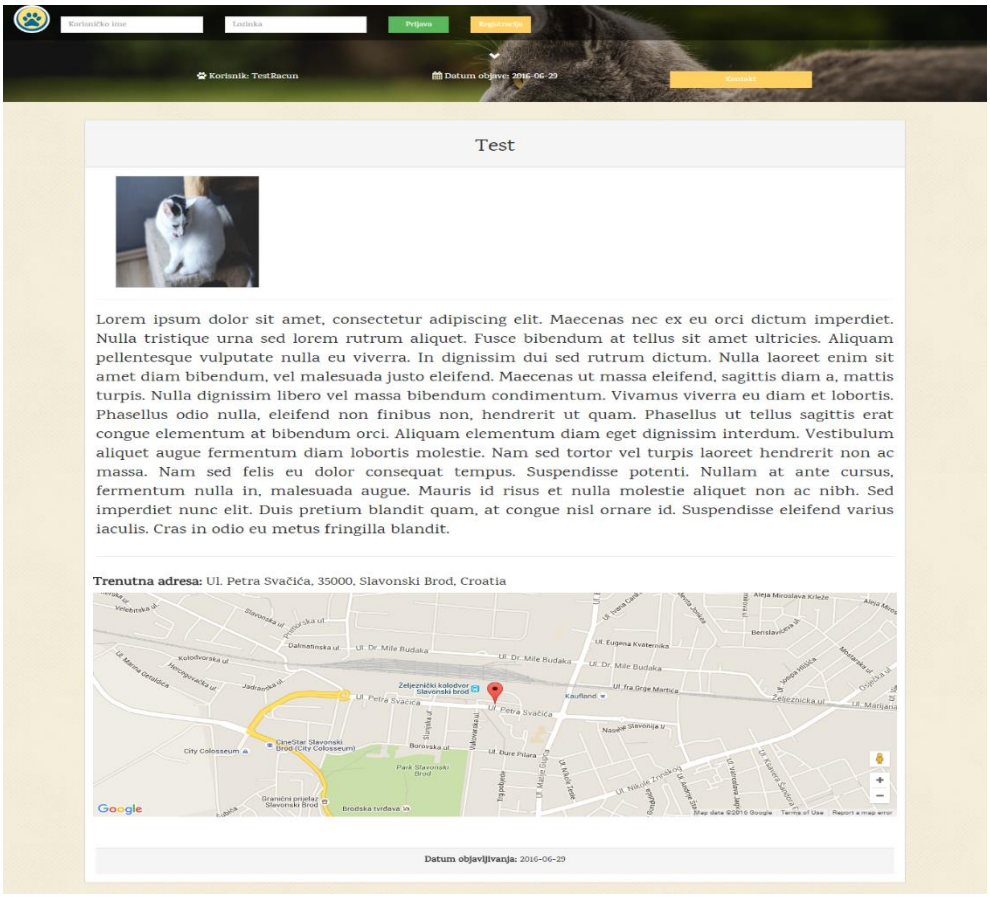
Sl. 3.8 *Treći dio objave oglasa*

Slikom 3.9 prikazan je poziv biblioteke potreban za aktivaciju Google karte. Prije aktivacije karte, preko [21] dobiven je ključ aplikacijskog programskog sučelja.

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=API_KLJUC&libraries=places
">
```

Sl. 3.9 *Pozivanje biblioteke za aktivaciju Google karte*

Nakon uspješnog ispunjavanja obrasca korisnik je preusmjeren na stranicu na kojoj izabire sljedeću akciju: pregled upravo objavljenog oglasa, uređivanje oglasa ili objavljivanje novog oglasa. Odabirom pregleda oglasa, korisnik je preusmjeren na stranicu koja sadrži kompletne podatke o oglasu. Primjer pregleda oglasa dan je slikom 3.10.



Sl. 3.10 *Pregled jednog od objavljenih oglasa*

Osim objavljivanja oglasa, korisnik ima mogućnost iste uređivati. Uređivanje oglasa funkcionira na sličnom principu kao i objava.

3.2 Rad sa Google kartama

Kako bi Google karte bile osposobljene na stranici, potreban je ključ aplikacijskog programskog sučelja. Koristeći Google-ovu konzolu za programere registriran je projekt, te je omogućen ključ za preglednike. Osim omogućivanja ključa potrebno je aktivirati aplikacijska programska sučelja koja će se koristiti, kao što su geolokacija i geokodiranje.

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=API_KLJUC&libraries=places
&callback=initMap">
</script>
```

Sl. 3.11 *Pozivanje biblioteke Google karti*

Slika 3.11 prikazuje pozivanje biblioteke Google karti. Radi boljeg korisničkog iskustva na kartama je prvobitno bila prikazana točna lokacija korisnika. No, kako više nije dopušteno dohvaćanje točne lokacije korisnika na stranicama koje nemaju dostupan protokol za sigurnu komunikaciju, odnosno HTTPS, na kartama je prikazana najbliža moguća lokacija koju je moguće saznati preko faktora kao što je IP adresa. Slikom 3.12 prikazano je dohvaćanje obližnje lokacije.

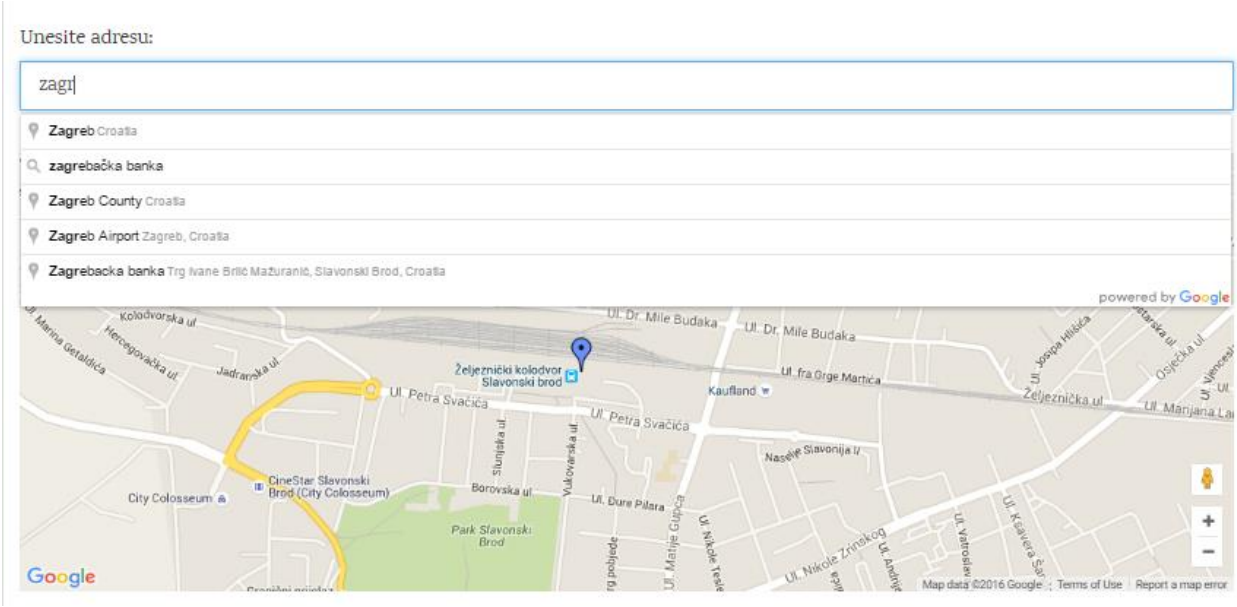
```
var jsonData = {
    "homeMobileCountryCode": 219,
    "considerIp": "true"
};
$.ajax({
    type: 'POST',
    url:
'https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyCmctjY9IgrY9
D9Qkodjf6EeiRAd0x1VF4',
    contentType: "application/json",
    dataType: "json",
    data: JSON.stringify(jsonData),
    success: function (response) {
        success(response);
    },
    error: function (reponse) {
        console.log(jsonData);
        console.log('location error: ' + JSON.stringify(reponse));
    }

});
```

Sl. 3.12 *Dohvaćanje približne lokacije*

Na internetsku adresu predviđenu za geolokaciju poslani su podaci koji sadrže kod države, te uvjet za uzimanje IP adrese u obzir pri pretraživanju lokacije. Kao odziv dobiven je objekt koji sadrži sve podatke o pronađenoj poziciji. Uspješnim izvršavanjem AJAX zahtjeva, pokretana je funkcija koja tada postavlja marker na dobivenu poziciju, te izvršava kod za inicijalizaciju karte.

Unutar koda prikazanog slikom 3.11, dodatno je definirano korištenje biblioteke *places*, te funkcije za inicijalizaciju karte. Biblioteka *places* koristi se za primjenu pretraživača mjesta i adresa.



Sl. 3.13 *Primjena biblioteke „places“*

Za omogućavanje korištenja pretrage mjesta, potrebno je stvoriti blok ulaza (engl. *input*), te mu unutar skripte za inicijalizaciju karte pridodati Google-ov *SearchBox*. Omogućavanje *SearchBox*-a prikazano je slikom 3.13.

```
var input = document.getElementById('pac-input');
var searchBox = new google.maps.places.SearchBox(input);
```

Sl. 3.14 *Omogućavanje Google SearchBox*

Pretraživaču mjesta pridodan je slušač (engl. *listener*) koji reagira na promjenu odabranog mjesta. Promjenom odabranog mjesta preuzimaju se podaci iz pretraživača, centrira se karta na odabrane koordinate i postavlja se novi marker. Osim kod pretraživača, primjena slušača potrebna je i kod karti. Kako bi se korisniku omogućilo odabiranje mjesta preko prikazane karte, karti se postavlja slušač koji reagira na pritisak desne tipke miša. Pritiskom desne tipke miša na kartu, dohvaćane su koordinate označenog mjesta. Primjena slušača prikazana je slikom 3.14.

```
google.maps.event.addListener(map, 'rightclick', function (event) {
    var geocoder = new google.maps.Geocoder;
    geocoder.geocode({'location': event.latLng}, function (results,
status){
        if (status == google.maps.GeocoderStatus.OK) {
            if (marker) {
                marker.setMap(null);
            }
            //ako ima dobra adresa
            if (results[1]) {
                marker = new google.maps.Marker({
                    position: event.latLng,
                    map: map
                });
                $('#pac-input').val(results[0].formatted_address);
                $('#lat').val(event.latLng.lat());
                $('#lon').val(event.latLng.lng());

            }
            else {
                marker = new google.maps.Marker({
                    position: event.latLng,
                    map: map
                });
                $('#pac-input').val(event.latLng.lat() + "," +
event.latLng.lng());

            }
        }
    })

})
```

Sl. 3.15 *Primjena slušača na karti*

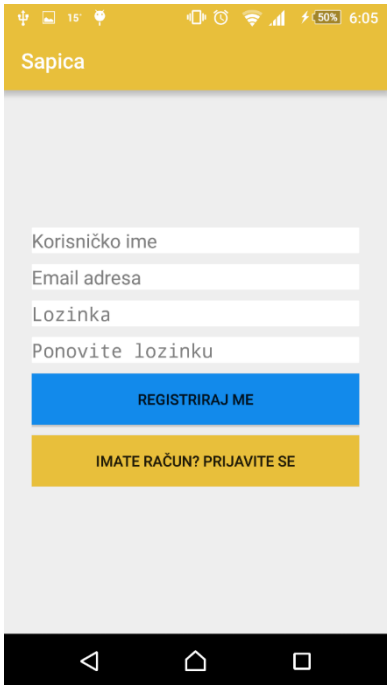
Kako bi se iz dohvaćenih koordinata dobila adresa primijenjeno je geokodiranje. Geokoderu su predane koordinate, te su kao odziv dobiveni rezultati sa podacima o adresi i status geokodera. Ukoliko dobiveni rezultati sadrže adresu, ona se postavlja kao vrijednost pretraživača. Radi lakše pohrane podataka u bazu podataka unutar obrasca su skrivena 2 ulaza kojima su predane vrijednosti zemljopisne duljine i širine. Ukoliko rezultati ne sadrže adresu, pretražitelju se kao vrijednost postavljene koordinate. U oba slučaja marker je postavljen na dobivene koordinate.

4. ANDROID APLIKACIJA

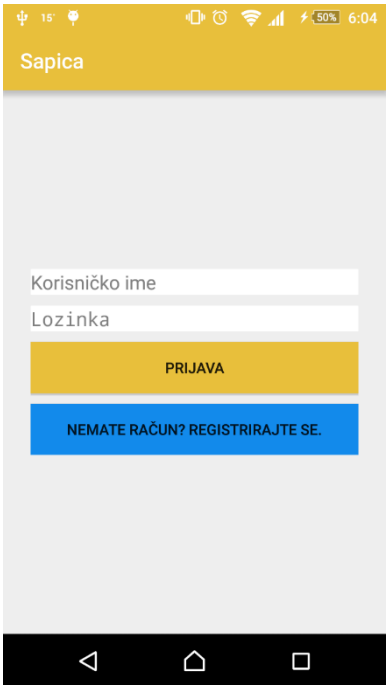
U ovome poglavlju dan je pregled korištenih kodova za stvaranje android aplikacije uz koje su priložene slike krajnjeg izgleda aplikacije. Poseban značaj u drugom potpoglavlju daje se radu s Google kartama preko koji korisnik unosi trenutnu lokaciju pronađene ili posljednju poznatu lokaciju izgubljene životinje.

4.1 Realizacija Android aplikacije

Pristup realizaciji aplikacije sličan je pristupu realizacije internetske stranice. Korisnici moraju imati iste mogućnosti na aplikaciji kao i na stranici. Tako je na aplikaciji omogućeno objavljivanje i uređivanje oglasa, te pretraživanje istih. Kako bi korisnik dobio pristup objavljivanju oglasa, potreban mu je račun koji je već napravio na internetskoj stranici, ili koji će napraviti preko aplikacije.



Sl. 4.1 *Prikaz registracije preko Android aplikacije*



Sl. 4.2 *Prikaz prijave preko Android aplikacije*

Slikama 4.1 i 4.2 prikazani su izgledi aktivnosti za registraciju i prijavu korisnika na Android aplikaciji. Aktivnosti su jednostavnog izgleda, te se sastoje od nekolicine prikaza teksta i gumbova. Pritiskom na gumb ostvaruje se veza sa poslužiteljem, te se prima odgovarajući odziv. O ostvarivanju veze sa poslužiteljem više pažnje će se posvetiti u idućem potpoglavlju. Kako Android ne sadrži nikakve metode upravljanja sesijama (engl. *session*), one su u aplikaciji ostvarene preko dijeljenih postavki (engl. *Shared Preferences*) postavljenih za rad u privatnom načinu. Stvorena je klasa *SessionHelper* koja služi za upravljanje sesijama, odnosno za zapisivanje i brisanje podataka iz dijeljenih postavki. Kod konstruktora *SessionHelper*-a prikazan je slikom 4.3.

```
public SessionHelper(Context context) {
        this._context = context;
        preferences = _context.getSharedPreferences(PREF_NAME,
PRIVATE_MODE);
        editor = preferences.edit();
    }

    public void sessionStart(boolean loggedIn, String username, String
email, String token) {

        editor.putBoolean(KEY_LOGGEDIN, loggedIn);
        editor.putString(KEY_USERNAME, username);
        editor.putString(KEY_EMAIL, email);
        editor.putString(KEY_TOKEN, token);
        Log.e("token in", token);

        editor.commit();
    }
```

***Sl. 4.3** Kod konstruktora SessionHelper-a i funkcije za spremanje sesije*

U slučaju uspješne prijave na račun, aplikacija sa poslužitelja prihvaća podatke o korisniku, kao što su adresa elektroničke pošte i takozvani žeton (engl. *token*). Žeton je niz znakova stvoren na poslužitelju prilikom prijave korisnika na račun. Pohranjen je posebnoj tablici u bazi podataka, te je nadalje korišten kao način autorizacije korisnika na poslužitelju. Kako bi sesija bila pokrenuta, potrebno je pozvati funkciju *sessionStart()*. Funkciji se predaju podaci prihvaćeni sa poslužitelja, kao što su indikator prijave, korisničko ime, adresa elektroničke pošte i žeton.

Nakon uspješne prijave, korisnik je preusmjeren na glavnu aktivnost. U glavnoj aktivnosti prikazani su do sada objavljeni oglasi. Prikaz popisa oglasa ostvaren je korištenjem *RecyclerView*-a. *RecyclerView* je, prema [22], unaprijeđena verzija *ListView*-a. Služi kao spremnik velikih

skupova podataka koje je moguće efikasno prelistavati. Pogodan je za korištenje kod skupova podataka čiji elementi ovise o akcijama korisnika ili o mrežnim događajima. Kako bi izgled samih podataka bilo moguće uređivati, stvoren je poseban adapter. Adapteru se predaje popis podataka preuzetih sa poslužitelja, te resurs izgleda pojedinog elementa liste. U ovom slučaju elementi su stvoreni od kartice (engl. *CardView*) koja sadrži dva prikaza teksta i jedan prikaz slike. Pošto su podaci preuzeti sa poslužitelja, za prikaz slike korišten je *NetworkImageView*. Adapter iterira kroz primljenu listu podataka, stvara izgled (engl. *View*) pojedinog elementa, te dijelovima unutar njega pridodaje odgovarajuće vrijednosti. Slikom 4.4 prikazan je dio koda adaptera zaslužan za stvaranje izgleda elemenata.

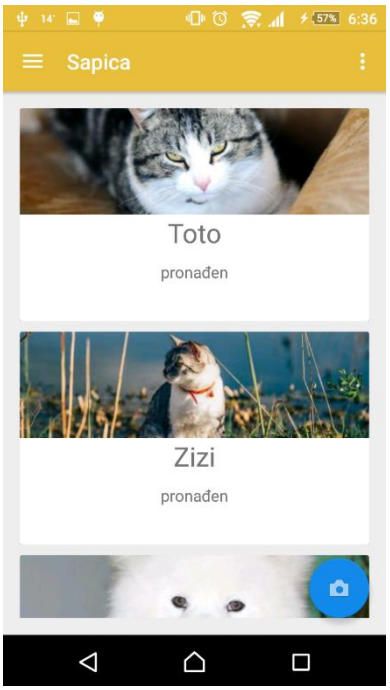
```
  @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType){
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.post_list_content
,parent,false);
        ViewHolder viewHolder = new ViewHolder(v);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        Post post = postList.get(position);
        holder.tvPostName.setText(post.getName());
        holder.tvPostType.setText(post.getType());
        Log.e("type",post.getType());
        imageLoader = VolleyHelper.getInstance(context).getImageLoader();
        holder.nivIcon.setImageUrl(post.getImgpath(),imageLoader);

    }
```

Sl. 4.4 *Postavljanje vrijednosti elemenata RecyclerView-a*

Definiranjem i pozivanjem adaptera unutar glavne aktivnosti, nastaje izgled prikazan na slici 4.5.



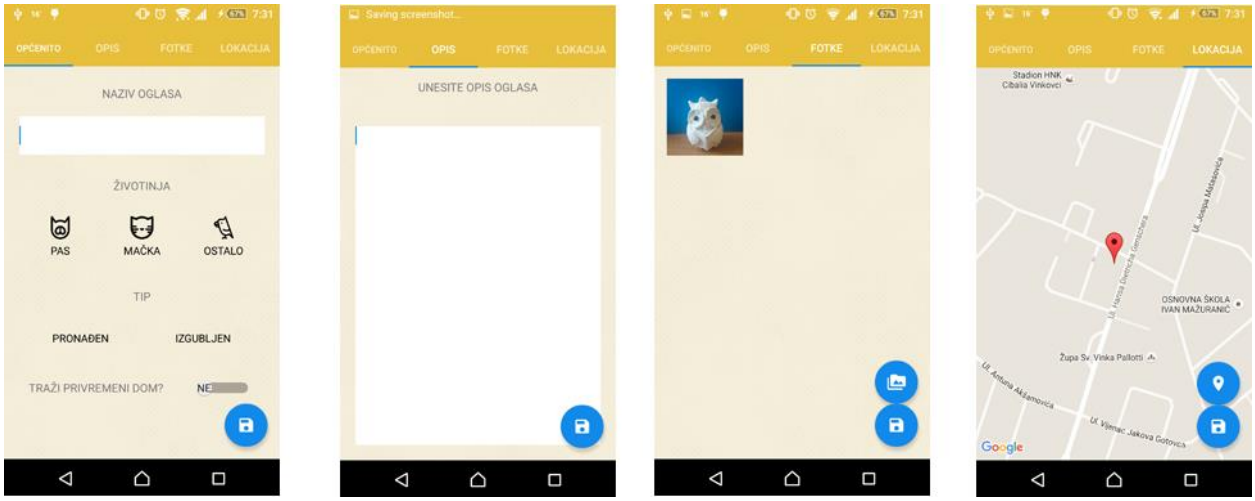
Sl. 4.5 *Prikaz glavne aktivnosti*

Kao što je prikazano slikom 4.5, u donjem desnom kutu glavne aktivnosti, nalazi se *FloatingActionButton*. Pritiskom na *FloatingActionButton* korisniku je omogućen pristup kameri. Nakon fotografiranja, prikazan je izbornik sa prikazom fotografije. Korisnik tada može nastaviti sa fotografiranjem ili nastaviti s ispunjavanjem obrasca za objavljivanje oglasa (Sl. 4.6).



Sl. 4.6 *Pregled slike nakon fotografiranja*

Pritiskom na tipku „ZAVRŠI“, korisnika se preusmjerava na aktivnosti stvaranja novog oglasa. Kao što je već spomenuto u prethodnom poglavlju, obrazac za stvaranje novog oglasa se sastoji od općenitih podataka, kao što su naziv, tip životinje i opis, odabira fotografija, te lokacije. Slika 4.7 prikazuje postupak ispunjavanja obrasca. Prethodno snimljena fotografija, prikazana je u odjeljku namijenjenom fotografijama. Radi ostvarivanja boljeg korisničkog iskustva, korisniku je omogućeno dodavanje fotografija iz albuma. Za razliku od Google karti korištenih na internetskoj stranici, Google karte korištene u aplikaciji imaju pristup trenutnoj lokaciji korisnika. Više govora o radu s Google kartama bit će u daljnjim potpoglavljima. Ukoliko se ljubimac više ne nalazi na lokaciji zaprimljenoj za vrijeme fotografiranja, korisniku je omogućeno pomicanje markera i pretraživanje mjesta pritiskom na gornju desnu tipku.



Sl. 4.7 *Postupak ispunjavanja obrasca novog oglasa*

Aktivnost stvaranja novog oglasa stvorena je koristeći *fragmente*. Kako bi komunikacija između *fragmenta* i aktivnosti bila što lakša, odnosno kako bi se olakšalo prikupljanje podataka, stvorena su sučelja (eng. *Interface*).

```
a) public interface TitleCallback {
        void onTitleTextChanged(String title);
    }

b) titleCallback.onTitleTextChanged(title);

c) @Override
    public void onTitleTextChanged(String fTitle) {
        title = fTitle;
    }
```

Sl. 4.8 *Primjena sučelja: a) definiranje sučelja, b) implementacija u fragmentu, c) implementacija u aktivnosti*

Slikom 4.8 prikazana je primjena sučelja za prikupljanje naziva oglasa. Kod pod točkom b) poziva se na svaku promjenu naziva. Pozivanjem b), postavlja se vrijednost varijable *title* u aktivnosti. Ovakvim rješenjem, aktivnost ima konstantan uvid u iznos varijabli potrebnih za slanje na poslužitelj.

4.2 Komunikacija sa poslužiteljem

Kako bi se ostvarila komunikacija sa poslužiteljem korišten je Volley. Prema [23], Volley je HTTP biblioteka koja čini komunikaciju sa poslužiteljem veoma jednostavnom i brzom. Omogućava automatsko zakazivanje zahtjeva prema poslužitelju, više istovremenih mrežnih povezivanja, podršku za stvaranje prvenstva zahtjeva itd. Volley je izvrstan za izvođenje operacije koje popunjavaju korisničko sučelje, kao što je dohvaćanje strukturiranih podataka sa poslužitelja. Stvorena je pomoćna klasa *VolleyHelper* za rad sa kompliciranijim zahtjevima kao što su dohvaćanje većeg broja oglasa. Komunikacija sa poslužiteljem bit će objašnjena na primjeru autorizacije korisnika.

Prije nego je uspostavljena komunikacija sa poslužiteljem, stvorene su PHP skripte na koje će zahtjevi biti slani.

```
if (mysqli_num_rows($res) == 0) {
        $feedback['error'] = TRUE;
        $feedback['msg'] = 'Nepostojeće korisničko ime.';
        echo json_encode($feedback);
    } else {
        if ($row['password'] == $pass) {
            $user_id = $row['_id'];
            $token = md5(rand(100000,999999999));
            $res = mysqli_query($connect,"INSERT INTO tokens(user_id,token)
VALUES ('$user_id','$token')");
            $feedback['error'] = FALSE;
            $feedback['user']['username'] = $row['username'];
            $feedback['user']['email'] = $row['email'];
            $feedback['user']['loggedin'] = TRUE;
            $feedback['user']['token'] = $token;
            echo json_encode($feedback);

        } else {
            $feedback['error'] = TRUE;
            $feedback['msg'] = 'Pogrešna lozinka.';
            echo json_encode($feedback);
        }
    }
} else {
    $feedback['error'] = TRUE;
    $feedback['msg'] = 'Niste ispunili sva polja.';

    echo json_encode($feedback);
```

Sl. 4.9 *Dio PHP skripte za autorizaciju korisnika*

Na slici 4.9 prikazan je dio skripte za autorizaciju korisnika. Pogreška, poruka i podaci se spremaju u red (engl. *array*) pod nazivom *feedback*. Provjereno je postojanje korisničkog imena u bazi podataka, te ispravnost unesene lozinke. U slučaju nepostojećeg korisničkog imena ili neispravne lozinke, pogreška se postavlja na istinu (engl. *true*) te se postavlja i odgovarajuća poruka koja će biti prikazana korisniku. Ako se prilikom autorizacije ne pojavi greška, podaci o korisniku se spremaju u red i pretvaraju JSON oblik. Takvi podaci se tada šalju kao odziv koji će prihvatiti Volley slušač odziva.

```
StringRequest stringRequest = new
StringRequest(Request.Method.POST,Config.API_LOGIN, new Response.Listener<String>()
{
            @Override
            public void onResponse(String response) {
                Log.i("JSON THING", "["+response+"]");
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    boolean error = jsonObject.getBoolean("error");
                    if (!error ) {
                        JSONObject user = jsonObject.getJSONObject("user");
                        String username = user.getString("username");
                        String token = user.getString("token");
                        String email = user.getString("email");
                        Log.e("login response", username + " " + token);
                        session.sessionStart(true, username, email, token);
                        Intent i = new Intent(LoginActivity.this,
MainActivity.class);
                        startActivity(i);
                        finish();
                    }
                    else {
                        String msgError = jsonObject.getString("msg");
                        Snackbar.make(getCurrentFocus(), msgError,
Snackbar.LENGTH_LONG)
                                .setAction("Action", null).show();
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                    Toast.makeText(getApplicationContext(), "Json error: " +
e.getMessage(), Toast.LENGTH_LONG).show();
                }
            }

        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e("login error", error.getMessage());
                Toast.makeText(getApplicationContext(),
                        error.getMessage(), Toast.LENGTH_LONG).show();
            }}) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> param = new HashMap<String, String>();
                param.put("username", username);
                param.put("password", password);
                return param;
            }
        };
        stringRequest.setTag(LoginActivity.class.getSimpleName());
        requestQueue = Volley.newRequestQueue(getApplicationContext());
        requestQueue.add(stringRequest);
```

Sl. 4.10 *Komunikacija sa poslužiteljem prilikom prijave korisnika*

Slikom 4.10 prikazan je jednostavniji Volley kod korišten za implementaciju komunikacije sa poslužiteljem. Za razliku od ostalih zahtjeva, kao što je dohvaćanje oglasa gdje je obrada dobivenih podataka puno duža, prilikom prijave korisnika potrebno je samo iščitati podatke iz primljenog JSON formata, te pokrenuti prije spomenutu sesiju. U kodu prikazanom slikom 4.9 definiran je

zahtjev za nizom, odnosno *StringRequest*. Pri stvaranju zahtjeva, definira se metoda zahtjeva, adresa na koju se šalju podaci, te slušači odziva. Unutar slušača prepisane (engl. *override*) su funkcije odgovorne za rad s pravilnim odzivom i odzivom pogreški. Parametri za slanje spremljeni su u mapu. Mapa je objekt koji preslikava objekte određenim ključevima. U ovom slučaju mapa se sastoji od dva parametra, korisničkog imena i lozinke. U slučaju ispravnog odgovora poslužitelja, poziva se funkcija *onResponse(String response)*. Pisanjem koda unutar prepisane funkcije *onResponse* dobiven je pristup i manipulacija odziva sa poslužitelja. Niz preuzet sa poslužitelja pretvoren je u JSON objekt radi lakšeg čitanja. Ukoliko je poslužitelj uspješno autorizirao korisnika, unutar JSON objekta pogreška će biti postavljena na neistinu (engl. *false*). Na drugu stranu, ukoliko pogreška postoji, biti će pročitana iz objekta i prezentirana korisniku.

4.3 Rad sa Google kartama

Za osposobljavanje karti na Android uređajima, potrebno je definirati ključ aplikacijskog programskog sučelja u manifestu. Također potrebno je zatražiti dopuštenja za korištenjem lokacijskih usluga kao što su pristup finoj i gruboj lokaciji (Sl. 4.11).

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
<meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="AIzaSyB_co9TxpVImHHPEdhSaGrDCdEYYi8r79w" />
```

Sl. 4.11 *Dio koda iz manifesta*

U aplikaciji su karte implementirane unutar *fragmenta* koji proširuje *SupportMapFragment*. Tako definirani fragment namijenjen je isključivo za rad s kartama, što znači da će cijelu površinu *fragmenta* zauzeti karta. Za dohvaćanje trenutne lokacije korisnika primijenjen je klijent aplikacijskog programskog sučelja Google karti, odnosno engl. *GoogleApiClient*. Stvaranje novog klijenta i pristup trenutnoj lokaciji izvršava se u aktivnosti, a ne u *fragmentu*. (Sl. 4.12).

```
if (googleApiClient == null) {
      googleApiClient = new GoogleApiClient.Builder(this)
                  .addConnectionCallbacks(this)
                  .addOnConnectionFailedListener(this)
                  .addApi(LocationServices.API)
                  .build();
}
googleApiClient.connect();

@Override
public void onConnected(@Nullable Bundle bundle) {
      currentLocation =
LocationServices.FusedLocationApi.getLastLocation(googleApiClient);
}
```

Sl. 4.12 *Dohvaćanje trenutne lokacije*

Trenutna lokacija postaje dostupna tek nakon što se klijent uspije povezati. Nakon što je lokacija pronađena, preko implementiranog sučelja dohvaćena je i unutar *fragmenta*. Za inicijalizaciju karte unutar *fragmenta* korišten je kod prikazan slikom 4.13.

```
getMapAsync(new OnMapReadyCallback() {
      @Override
      public void onMapReady(GoogleMap googleMap) {
            map = googleMap;
            map.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 16));
            marker = map.addMarker(new
MarkerOptions().position(latLng).draggable(true).icon(BitmapDescriptorFacto
ry.defaultMarker(BitmapDescriptorFactory.HUE_RED)));
      }
});
```

Sl. 4.13 *Inicijalizacija karte*

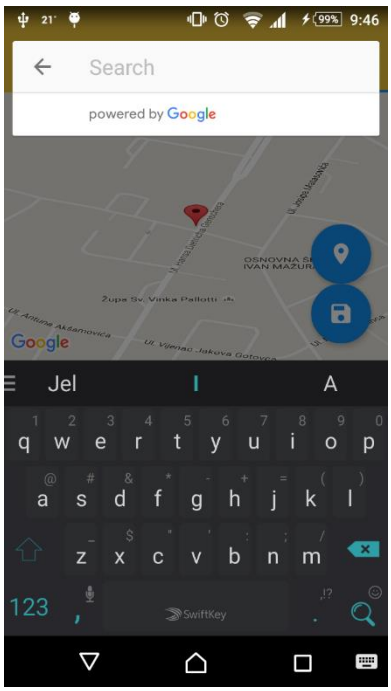
Prema [24], funkcija *getMapAsync* postavlja povratni poziv koji će biti pokrenut kada karta postane spremna za korištenje. Nakon inicijalizacije karte, postavljen je marker na koordinate preuzete iz aktivnost, te mu je postavljena mogućnost povlačenja.

Kao i na internetskoj stranici, u aplikaciji je omogućen pretraživač mjesta. Pritiskom na tipku sa oznakom za lokaciju pokreće se *intent* za pozivanje pretraživača (Sl. 4.14.)

```
Intent i = new
PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_OVERLAY).build(activ
ity);
startActivityForResult(i, PLACE AUTOCOMPLETE REQUEST CODE);
```

Sl. 4.14 *Pozivanje pretraživača*

Slika 4.15 prikazuje pokrenuti pretraživač mjesta. Završetkom pretrage pokretana je funkcija *onActivityResult* koja kao odziv predaje podatke o odabranom mjestu. U podacima su sadržane zemljopisna duljina i širina, te adresa.



Sl. 4.15 *Pretraživač mjesta*

Primjena koda za dohvaćanje podataka o lokaciji, kako trenutnoj, tako onoj traženoj, izuzetno je lakša na Android aplikaciji nego na internetskoj stranici.

5. ZAKLJUČAK

Glavni cilj ovog rada bilo je ostvarivanje usluge za brže i praktičnije pronalaženje i udomljavanje ljubimaca. Usluga je ostvarena u dva dijela: internetska stranica i Android aplikacija. U oba dijela realizirane su iste funkcionalnosti, s dodatkom mogućnosti fotografiranja u sklopu Android aplikacije.

Prije analize praktičnog rada, kao uvod u stvaranje stranice i aplikacije, dan je teorijski uvid u korištene tehnologije. Tako su za stvaranje internetske stranice korišteni HTML, CSS, JavaScript i njegove biblioteke, PHP, te MySQL. Popis tehnologija potrebnih za ostvarivanje rješenja aplikacije puno je kraći; korišten je Java programski jezik na Android platformi. Ostvarivanjem usluge i na internetskoj stranici i pomoću aplikacije, povećana je njena učinkovitost, dostupnost i privlačnost korisnicima. Prilikom realizacije u obzir je uzeto korisničko iskustvo. Korisniku se pokušalo što je više moguće olakšati određivanje i označavanje lokacije. Prilikom završavanja praktičnog dijela rada, došlo je do promjene načina korištenja aplikacijskog programskog sučelja za Google karte. Stranicama koje nemaju dostupan protokol za sigurnu komunikaciju, odnosno HTTPS, onemogućen je pristup zahtjevima za trenutnom lokacijom korisnika. Zbog te prepreke, primijenjen je drugačiji pristup: lokacija je određivana preko IP adrese. Iako takav pristup nije precizan i prikazuje lokacije udaljene preko desetaka kilometara od korisnika, ipak malo olakšava korisnicima korištenje Google karte.

Iako je usluga realizirana u dva dijela, ona funkcionira kao cjelina; omogućavajući korisnicima da vide oglase objavljene preko stranice i aplikacije, te uređuju vlastite objavljene oglase na različitim uređajima.

LITERATURA

[1] J. Keith, HTML for Web Designers, New York: A book apart, 2011.

[2] M. Pilgrim, Dive into HTML5 , http://diveintohtml5.info/. [Pokušaj pristupa 20 05 2016].

[3] Iconnect, Why use HTML5 to design a website or an app?, http://www.inspiresmart.com/why-use-html5-to-design-a-website-or-an-app. [Pokušaj pristupa 20 05 2016].

[4] G. Wolejko, 5 reasons why HTML5 matters, https://www.cognifide.com/our-blogs/ux/5-reasons-why-html5-matters/. [Pokušaj pristupa 20 05 2016].

[5] W3C, HTML & CSS, World Wide Web Consortium, https://www.w3.org/standards/webdesign/htmlcss#whatcss. [Pokušaj pristupa 20 05 2016].

[6] A. Goldstein, L. Lazaris i E. Weyl, HTML5 & CSS3 for the real world, Collingwood: SitePoint, 2011.

[7] W3C, Introduction to CSS3, World Wide Web Consortium, https://www.w3.org/TR/2001/WD-css3-roadmap-20010119/. [Pokušaj pristupa 20 05 2016].

[8] D. Cederholm, CSS3 for Web Designers, New York: A book apart, 2010.

[9] CSS3.info, CSS3 Previews, http://www.css3.info/preview/. [Pokušaj pristupa 20 05 2016].

[10] Purencool, JavaScript and JQuery, what is the difference?, http://purencool.com/javascript-and-jquery-what-is-the-difference. [Pokušaj pristupa 21 05 2016].

[11] tutorialspoint, What is AJAX?, http://www.tutorialspoint.com/ajax/what_is_ajax.htm. [Pokušaj pristupa 22 05 2016].

[12] Php Group, PHP: What is Php?, http://php.net/manual/en/intro-whatis.php. [Pokušaj pristupa 22 05 2016].

[13] tutorialspoint, MySQL Introduction, http://www.tutorialspoint.com/mysql/mysql-introduction.htm. [Pokušaj pristupa 24 05 2016].

[14] M. Gargenta, Naučite Android, Zagreb: Dobar Plan, 2013.

[15] Technopedia, What is Android?, https://www.techopedia.com/definition/14873/android-os. [Pokušaj pristupa 20 05 2016].

[16] Google, Android, the world's most popular mobile platform | Android Developers, https://developer.android.com/about/index.html. [Pokušaj pristupa 12 2015].

[17] Google, Introduction to Android, https://developer.android.com/guide/index.html. [Pokušaj pristupa 20 05 2016].

[18] Y. Fain, Programiranje Java, Zagreb: Dobar plan, 2013.

[19] Oracle, Learn about Java, https://java.com/en/about/. [Pokušaj pristupa 20 05 2016].

[20] V. Beal, What is Java?, http://www.webopedia.com/TERM/J/Java.html. [Pokušaj pristupa 20 05 2016].

[21] Google, Google Api Console, https://console.developers.google.com. [Pokušaj pristupa 07 02 2016].

[22] Google, Creating Lists and Cards, https://developer.android.com/training/material/lists-cards.html. [Pokušaj pristupa 27 05 2016].

[23] Google, Transmitting Network Data Using Volley, https://developer.android.com/training/volley/index.html. [Pokušaj pristupa 28 01 2016].

[24] Google,SupportMapFragment, https://developers.google.com/android/reference/com/google/android/gms/maps/SupportMapFragment.html. [Pokušaj pristupa 13 02 2016].

SAŽETAK

Ovaj diplomski rad ima za cilj stvoriti uslugu za prijavljivanje i udomljavanje ljubimaca. Usluga je stvorena u dva dijela: internetske stranice i Android aplikacije. Prije analize praktičnog rada navedene su i teorijski obrađene korištene tehnologije. Prilikom analize realizacije pojedinog dijela usluge, izdvojeni su kodovi specifični za ostvarivanje usluge. Kodovi su pritom objašnjeni, te su dane slike opisane spomenutim kodovima. Poseban značaj se pridodao radu s Google kartama. Korištenjem usluge korisnik stvara korisnički račun čime dobiva pristup objavljivanju oglasa. U oglasu je moguće postaviti trenutnu lokaciju ljubimca, te dodatne informacije kao što su naziv, potreba za privremenim domom itd.

Ključne riječi: Android aplikacija, internetska stranica, usluga, oglasi, ljubimci

ABSTRACT

Service for registering and adopting pets

This thesis aims to create a service that would allow users to register and adopt pets. The service consists of two parts: a website and an Android application. Theoretical analysis of the used technologies was given before the analysis of the practical part of the thesis. Codes specific for the service were given during the analysis. Codes were then explained, accompanied with web-page and app screenshots they describe. Special emphasis was given to handling Google maps. By using the service, the users creates a user account and with doing that gains access post submitting and publishing. Within the post, it is possible to set the location of the pet, along with additional information, such as the title or the need for a temporary home.

Key words: Android application, web-page, service, posts, pets

ŽIVOTOPIS

Jelena Duraković rođena je 07. srpnja 1991. godine u Vinkovcima, Republika Hrvatska. Nakon završetka osnovne škole „Ivan Mažuranić“ u Vinkovcima, upisala je i završila matematičko-prirodoslovni smjer gimnazije „Matija Antun Reljković“ u Vinkovcima. Po završetku srednje škole, upisala je preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. 2013. godine završava preddiplomski studij računarstva, te upisuje diplomski studij računarstva, smjer procesno računarstvo na Elektrotehničkom fakultetu u Osijeku.