Android aplikacija - turistički vodič za grad Osijek

Kiš, Marijo

Undergraduate thesis / Završni rad

2019

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:200:109512

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

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

Sveučilišni studij

ANDROID APLIKACIJA -

TURISTIČKI VODIČ ZA GRAD OSIJEK

Završni rad

Marijo Kiš

Osijek, 2019.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. NOVE TEHNOLOGIJE U PROMOCIJI TURISTIČKIH SADRŽAJA 2

3. KORIŠTENE TEHNOLOGIJE 3

3.1. Operacijski sustav Android 3

3.2. Android Studio 5

3.3. Java programski jezik 6

3.4. Firebase 7

3.5. GIMP 7

4. PROCES IZRADE APLIKACIJE 9

4.1. Fragmenti 9

4.2. Google Maps 11

4.3. RecyclerView 14

4.4. Cloud Firestore 19

4.5. Firebase autentifikacija 22

5. KORIŠTENJE APLIKACIJE 26

6. POSTOJEĆA RJEŠENJA 31

7. ZAKLJUČAK 32

LITERATURA 33

SAŽETAK 35

ABSTRACT 36

ŽIVOTOPIS 37

PRILOZI 38

1. UVOD

Zadatak ovog završnog rada je izraditi android aplikaciju koja će pomoći turistima koji dolaze u grad Osijek pri snalaženju i odabiru relevantnih mjesta koja bi svaki turist trebao posjetiti. Aplikacija će turistima pružiti najvažnije informacije o turističkim sadržajima u gradu.

U razvoju aplikacije za potrebe ovoga završnog rada bit će korišteni: integrirano razvojno okruženje Android Studio, programski jezik Java, Google karte, baza podataka Firestore te Firebase autentifikacija.

Funkcionalnosti koje će aplikacija omogućiti korisnicima su: registracija korisnika i prijava korištenjem korisnikove elektroničke pošte i zaporke, praćenje lokacije korisnika u stvarnome vremenu, pružanje informacija o turistički zanimljivim lokacijama i događajima u gradu te pregled doživljaja drugih korisnika i dodavanje vlastitog doživljaja grada.

U drugom poglavlju dane su osnovne informacije o ustrojstvu Hrvatske turističke zajednice te su na primjeru Turističke zajednice grada Osijeka opisane nove tehnologije koje su danas korištene u promociji turističkih sadržaja. Treće poglavlje donosi opis korištenih tehnologija pri izradi aplikacije. Nakon toga slijedi poglavlje u kojemu se nalazi slijed procesa pri izradi uz objašnjenja najvažnijih metoda i komponenata, zatim uputstva za korištenje uz slikovne prikaze svih pogleda aplikacije te usporedba sa sličnim rješenjima dostupnima na trgovini Google Play. U zadnjem, sedmom poglavlju, nalazi se zaključak.

1.1. Zadatak završnog rada

Kratko opisati mogućnost upotrebe novih tehnologija u promociji turističkih sadržaja grada Osijeka. Opisati specifičnosti izrade aplikacije za Android platformu. Izraditi aplikaciju za Android platformu pomoću koje će turisti u gradu Osijeku dobiti osnovne informacije o turističkim sadržajima. Opisati proces izrade aplikacije i njene funkcionalnosti. Usporediti izrađenu aplikaciju s postojećim rješenjima.

2. NOVE TEHNOLOGIJE U PROMOCIJI TURISTIČKIH SADRŽAJA

Nove tehnologije postaju sve prisutnije u svim segmentima života, pa tako i u samom promicanju turizma. Jedna od glavnih prednosti novih tehnologija je ta što pružaju informacije u stvarnome vremenu. Turisti mogu vidjeti ponudu iz udobnosti vlastitog doma, a dolaskom na odabranu destinaciju dobit će sve relevantne informacije o mjestu koje su posjetili, te o trenutnim događanjima u njemu.

Jedan od pozitivnih primjera promicanja hrvatskog turizma je projekt Hrvatske turističke zajednice *croatia.hr* (uz slogan „*Croatia – Full of life*“[1]).

Hrvatska turistička zajednica je nacionalna organizacija koja za cilj ima stvaranje i promicanje hrvatskog turizma, te podizanje kompletne turističke ponude Hrvatske. Tijela Hrvatske turističke zajednice su Sabor, Turističko vijeće, Nadzorni odbor i predsjednik čiju dužnost vrši ministar turizma. Zajednica ima urede diljem Hrvatske, ali isto tako ima i predstavništva u inozemstvu.

Turistička zajednica grada Osijeka djeluje uz slogan „*Posjeti Osijek*“ i za zadaću ima promicanje i obogaćivanje turističke ponude grada. Na njihovoj mrežnoj stranici mogu se pronaći podatci kao što su osnovne informacije o Osijeku, informacije o Hrvatskom Podunavlju (širem području oko Osijeka), gastronomiji, smještaju u gradu, rekreacijskom sadržaju, trenutnim događanjima u gradu, te o mnogim drugim turističkim sadržajima. Također se mogu pronaći i poveznice na moderne oblike komuniciranja kao što su Facebook (korisnički račun: *Visit Osijek – Turistička zajednica Grada Osijeka*), Instagram (korisnički račun: *Visit Osijek – Turizam Osijek*), te YouTube (korisnički račun: *TZ Osijek*).

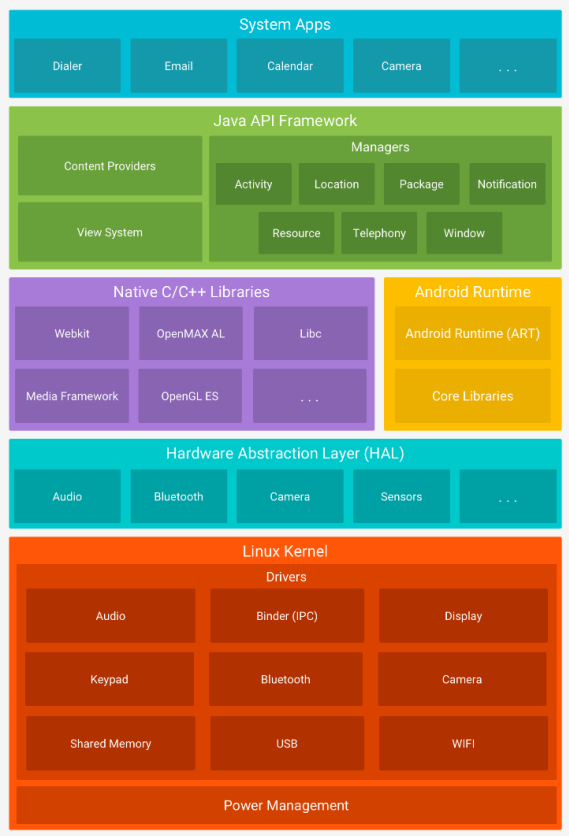
[1] Hrvatska – Puna života

3. KORIŠTENE TEHNOLOGIJE

U ovome poglavlju opisuju se tehnologije korištene pri izradi aplikacije za potrebe ovoga završnoga rada. Pobliže će se objasniti Android operacijski sustav, Android Studio, Java programski jezik, Firebase i na koncu GIMP.

3.1. Operacijski sustav Android

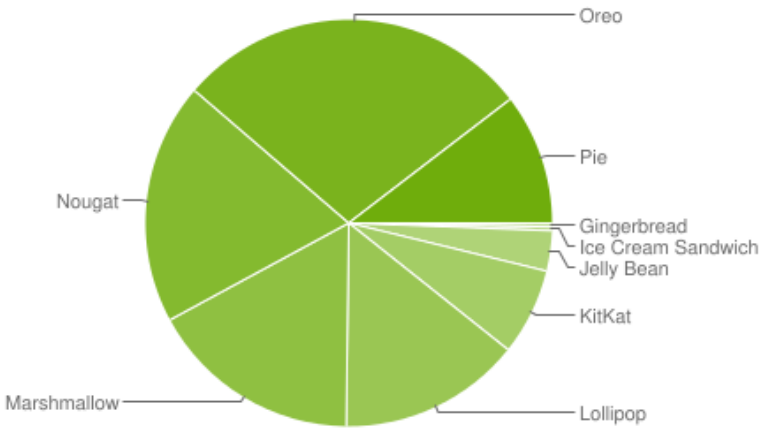
Razvoj operacijskog sustava Android počeo je 2003. godine kada su Andy Rubin, Rich Miner, Nick Sears i Chris White osnovali tvrtku Android Inc. Prve nakane tvrtke bile su razvitak naprednih operacijskih sustava za digitalne kamere, no ubrzo su ocijenili da je to tržište premaleno za njihove apetite, te su ubrzo najavili svoj ulazak u utrku osvajanja dominacije na tržištu mobilnih uređaja i najavili konkuriranje tadašnjim predvodnicima Symbianu[2] i Windows Mobileu [3]. 2005. godine Google kupuje tvrtku Android Inc. 2008. godine Android postaje sustav otvorenog koda i krajem iste izbacuje svoju prvu verziju, Android 1.0 radnog naziva Apple Pie.



Slika 3.1. *Građa Android sustava*[4]

[2] Mobilni operacijski sustav koji više nije na tržištu, a razvila ga je tvrtka Symbian Ltd.
[3] Mobilni operacijski sustav koji više nije na tržištu, a razvila ga je tvrtka Microsof
[4] Preuzeto: https://developer.android.com/guide/platform (26.6.2019.)

Android je sustav otvorenoga koda baziran na Linuxu. Prema slici 3.1. vidi se kako je na dnu građe sustav zadužen za upravljanje potrošnjom. Linux jezgra sadrži pogonske programe (engl. *driver*) kao što je *IPC*[5] čija je dužnost međuprocesna komunikacija, odnosno služi za razmjenu podataka između procesa i niti. *HAL*[6] sloj sastoji se od raznih biblioteka od kojih svaka ugrađuje sučelje za određeni dio hardverske sastavnice. *Android Runtime (ART)* služi za pokretanje aplikacija. Sastoji se od ključnih biblioteka, te *DVM-a*[7] koji omogućuje pokretanje aplikacija kao zasebnih instanci virtualnog stroja, što se očituje kao prednost na uređajima s manje memorije jer omogućava ugodniji i fluidniji rad. Android omogućuje aplikacijama pisanim u C ili C++ programskom jeziku pristup nativnim C/C++ bibliotekama, a dijelovi Android građe koji ih koriste su *ART* i *HAL*. Java *API*[8] sučelje sastoji se od mehanizama koji pojednostavljuju proces izrade aplikacije. Razvojnim programerima dostupna su sva *API* sučelja koja koristi Android sistemska aplikacija. Neki od mehanizama pomoći su: nadziranje aktivnosti aplikacije, omogućavanje dijeljenja informacija između aplikacija, te prikaz obavijesti na statusnoj traci. Na vrhu arhitekture Android sustava nalaze se sistemske aplikacije kao što su aplikacije za čitanje i slanje mailova, SMS komunikaciju, pretraživanje interneta i mnoge druge. Sistemske aplikacije dostupne su korisniku, ali isto tako i programerima koji ih mogu koristiti pri izradi svojih vlastitih aplikacija. Naprimjer, ukoliko se želi napraviti aplikacija za slanje fotografija, ne mora se posebno omogućavati sučelje kamere, već se može koristiti sistemska aplikacija kamere.

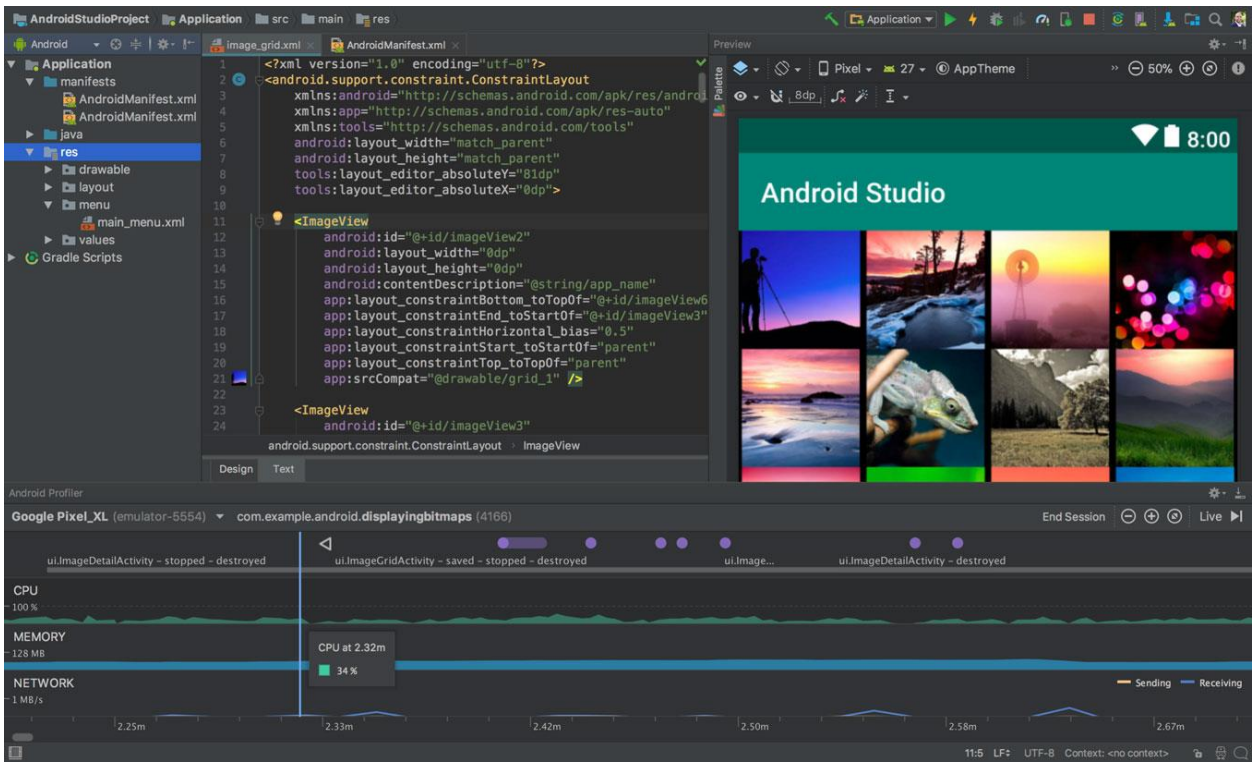


Slika 3.2. *Zastupljenost pojedinih verzija Androida (verzije s manje od 0.1% zastupljenosti nisu prikazane)*[9]

[5] Inter Process Communication
[6] Hardware Abstraction Layer
[7] Dalvik Virtual Machine
[8] Application Programming Interface
[9] Preuzeto: https://developer.android.com/about/dashboards (26.6.2019.)

3.2. Android Studio

Android Studio je razvojno okruženje za pisanje Android aplikacija. Neke od mogućnosti koje pruža su: fleksibilni gradivni sustav baziran na *Gradleu*[10], brz i opcijama bogat emulator, te jedinstveno okruženje za izgradnju aplikacija za sve Android uređaje.



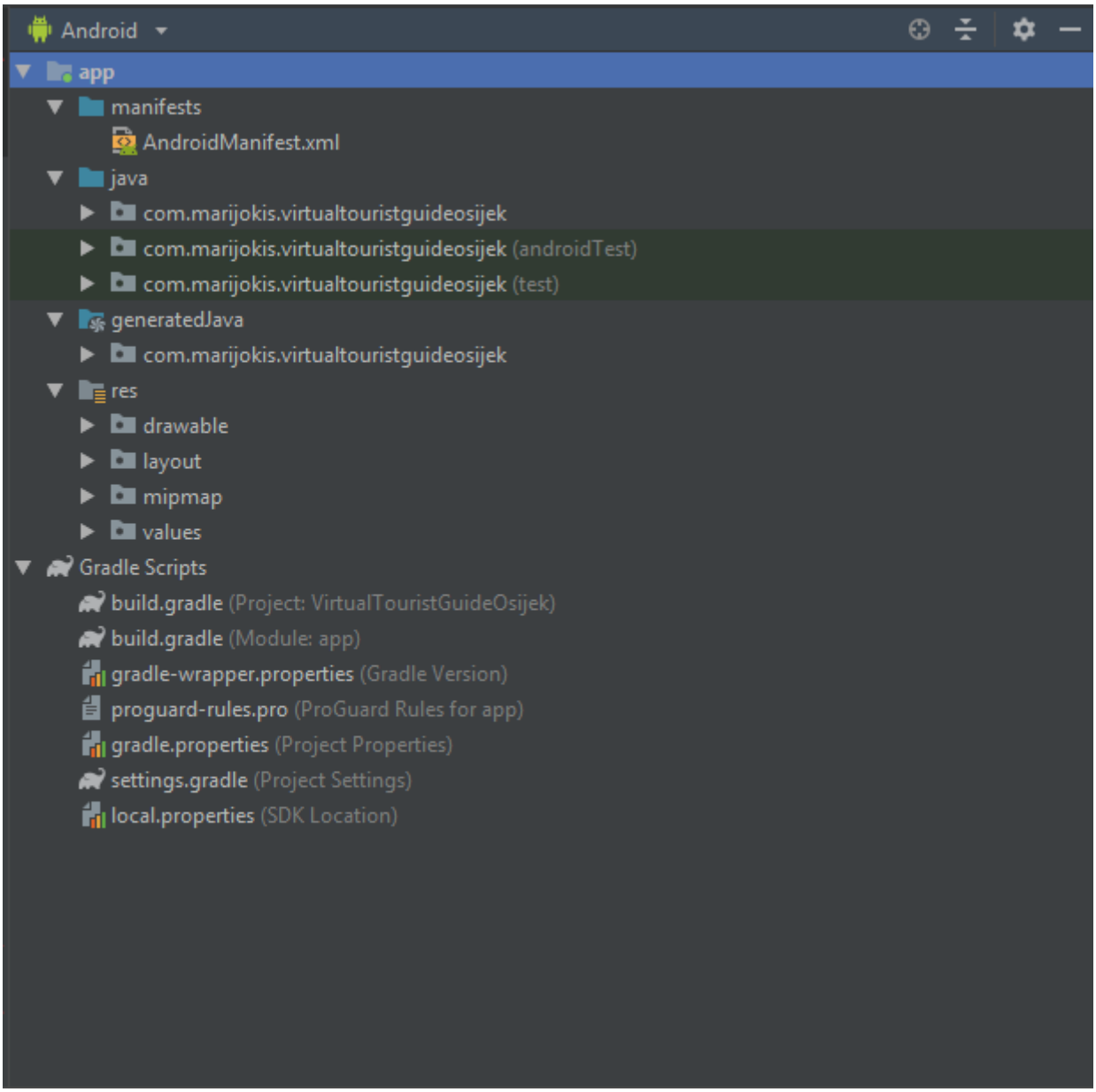
Slika 3.3. *Izgled korisničkog sučelja Android Studija*[11]

Projekti sadrže jedan ili više modula s izvornim kodom i datotekama resursa. Tipovi uključenih modula su: aplikacijski moduli, moduli biblioteka, te *Google App Engine* moduli. Slijedi objašnjenje aplikacijskog modula. On sadrži tri direktorija, a to su: *manifests* u kojemu se nalazi AndroidManifest.xml datoteka, java direktorij s datotekama java izvornog koda i *JUnit* test kodom, te *res* u kojemu su smješteni resursi kao što su XML[12] datoteke i slikovni resursi.

[10] Sustav otvorenog koda koji je načinjen tako da je dovoljno fleksibilan kako bi bio u mogućnosti graditi skoro svaku vrstu softvera

[11] Preuzeto: https://developer.android.com/studio (26.6..2019.)

[12] Extensible Markup Language



Slika 3.4. *Struktura projekta u Android pogledu*

3.3. Java programski jezik

James Gosling pokrenuo je projekt razvoja Jave 1991. godine. Prvi naziv ovog programskog jezika bio je Oak prema hrastu koji je rastao ispred Goslingovog ureda, no na kraju ipak dobiva naziv Java prema vrsti kave. Objavljen je u studenom 1995. godine.

Java je dizajnirana prema C/C++ sintaksi. Objektno je orijentiran programski jezik, te struktura kreće s paketima, nakon kojih slijede klase, zatim metode, varijable, konstante, te tako dalje. Velika prednost Jave pri objavljivanju u odnosu na druge tadašnje programske jezike bila je ta što

je Java u mogućnosti izvoditi se na svakom operacijskom sustavu bez ikakvih preinaka ukoliko sustav sadrži *Java Virtual Machine*. Ona pripada skupini viših programskih jezika, a danas se uz Kotlin i C# ubraja u najpopularnije jezike za razvoj Android aplikacija.

3.4. Firebase

Firebase je razvojna platforma namijenjena mobilnim i web aplikacijama koja nudi široku paletu alata i servisa, a sve to kako bi se razvila što kvalitetnija aplikacija. Firebaseovi servisi mogu se razdijeliti na dvije glavne grane, a to su servisi za razvoj i testiranje, te servisi za rast i privlačenje korisnika.

Razvoj ove popularne platforme započeo je 2011. godine kao razvojna tvrtka pod imenom Envolve. Osnivači su James Tamplin i Andrew Lee. U najranijim početcima nudio je *API* koji se mogao dodati u mrežne aplikacije kako bi se omogućila funkcionalnost chata. U travnju 2012. godine Firebase nastaje kao zasebna tvrtka koja je nudila pozadinski servis kao uslugu (engl. *Backend-as-a-Service*) s razmjenom podataka u stvarnom vremenu. Nakon što je 2014. Google preuzeo Firebase, on postepeno postaje ono što je danas.

U ovome projektu bit će korišteni servis za autentifikaciju korisnika, te servis za pohranu i dohvaćanje podataka.

3.5. GIMP

GIMP (engl. *GNU Image Manipulation Program*[13]) je vrlo popularan softver za obavljanje operacija nad fotografijama. Besplatan je, te je otvorenog koda.

Sami početci GIMP-a kreću 1995. godine na Sveučilištu Kalifornije kao dio studentskog projekta, a razvili su ga Spencer Kimball i Peter Mattis. Danas softver razvijaju skupine volontera, a dostupan je za brojne operacijske sustave.

Za potrebe ovog završnog rada GIMP je korišten kako bi se stvorio logo aplikacije (slika 3.5.). Logo pokazuje poznatu osječku vizuru koja se ovako poravnata može vidjeti šetajući lijevom obalom rijeke Drave, a to su pješački most, konkatedrala Sv. Petra i Pavla, te hotel Osijek.

[13] GNU program za manipulaciju fotografijama

Slika 3.5. *Izrađeni logo aplikacije*

4. PROCES IZRADE APLIKACIJE

Ovo poglavlje donosi opis procesa izrade aplikacije kroz pojašnjenja njezinih najvažnijih elemenata. Prvo će biti objašnjeni fragmenti koji čine same temelje aplikacije, a zatim slijede Google karte, *RecyclerView,* Firestore, te Firebase autentifikacija.

4.1. Fragmenti

Aplikacija se temelji na fragmentima. Fragmenti omogućuju modularan dizajn aplikacije. Imaju vlastiti životni ciklus, no blisko su vezani uz *activity* u kojemu se nalaze; ukoliko se *activity* zaustavi, svi fragmenti unutar njega se istovremeno zaustavljaju. Interakcija s fragmentima odvija se pomoću *FragmentManagera*. Svakim fragmentom se može rukovati neovisno o drugima. Kako bi se fragmenti implementirali u projekt, prvi korak je dodati *FrameLayout* komponentu pomoću koje se fragmenti prikazuju, a dodaje se na glavni zaslon, odnosno u activity_main.xml datoteku koja reprezentira izgled glavnog zaslona. Visina i širina se postavlja tako da komponenata zauzme cijeli zaslona, a taj će element predstavljati kontejner za fragmente na što upućuje i sami naziv identifikacijske oznake.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/bottom_navigation" />

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/bottom_navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:background="@color/appThemeColor"
        app:itemIconTint="@color/colorWhite"
        app:itemTextColor="@color/colorWhite"
        app:menu="@menu/bottom_navigation" />

</RelativeLayout>
```

Slika 4.1. *Layout resurs activity_main.xml*

Na slici 4.1. može se vidjeti još jedna komponenta smještena unutar activity_main.xml datoteke, a to je navigacijski izbornik na dnu ekrana koji služi za odabir željenog fragmenta, odnosno, na konkretnom primjeru izrađene aplikacije, služi za kretanje između pogleda karte, iskustava, vijesti, te korisničkog profila. Navigacija će biti vidljiva na svim fragmentima, osim onih za prijavu i registraciju korisnika, a ta će funkcionalnost biti omogućena programski zasebnom funkcijom za postavljanje vidljivosti navigacijske trake.

Svaki fragment mora biti povezan sa XML resursom. Na slici 4.2. prikazan je XML resurs fragmenta na kojemu se prikazuju vijesti.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/colorWhite">

   <android.support.v7.widget.RecyclerView
      android:id="@+id/recyclerViewNews"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@color/colorWhite"
      android:padding="4dp"
      android:scrollbars="vertical" />

</RelativeLayout>
```

Slika 4.2. *Layout resurs fragment_news.xml*

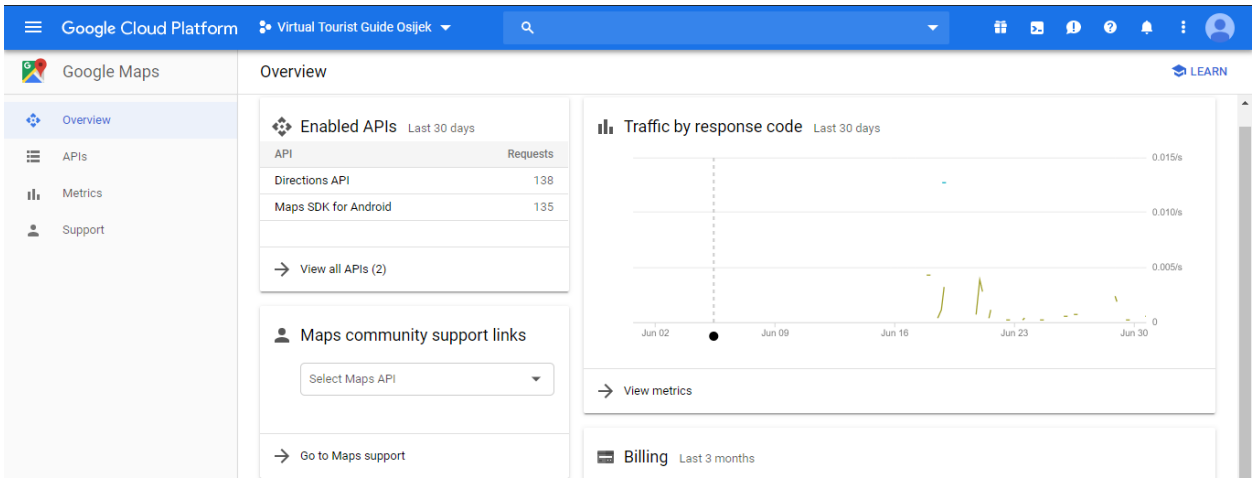
Kako bi se fragment ispravno kreirao, klasa mora naslijediti klasu *Fragment*. Pomoću poziva metode *onCreateView* definira se XML resurs fragmenta, a fragment se instancira na korisniku trenutno vidljivom pogledu. Neophodno je ponoviti postupak kreiranja XML resursa i pozivanje *onCreateView* metode na svakom fragmentu kako bi proces kreiranja bio finaliziran i kako bi tranzicija među pogledima bila funkcionalna.

```
public class NewsFragment extends Fragment {
   @Nullable
   @Override
   public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
            Bundle savedInstanceState) {
                  return inflater.inflate(R.layout.fragment_news, container, false);
   }
}
```

Slika 4.3. *onCreateView metoda unutar klase NewsFragment.java*

4.2. Google Maps

Prvi korak u implementaciji Google karte jest izraditi Google korisnički račun, te pribaviti *API* ključ za pristup *Google Maps* serveru. To je moguće učiniti u *Google Cloud Platform* konzoli. Nakon kreiranja *API* ključa za ovaj projekt bilo je važno omogućiti *Directions API* i *Maps SDK for Android* opcije kako bi dohvaćanje trenutne lokacije korisnika bilo moguće.



Slika 4.4. *Izgled Google Cloud Platform konzole*

Idući korak je dodati *API* ključ u AndroidManifest.xml datoteku na način prikazan na slici 4.5. Zvjezdice predstavljaju postavljenu masku pojedinom znaku *API* ključa u ovoj tekstualnoj verziji rada što je učinjeno radi osiguravanja privatnosti ključa. Također na istoj slici mogu se vidjeti dozvole koje su morale biti omogućene kako bi Google karta radila ispravno. Unutar ove datoteke postavljena je i personalizirana ikona same aplikacije.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:toolsNs="http://schemas.android.com/tools"
    package="com.marijokis.virtualtouristguideosijek">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/myicon_round"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="**************************************"/>
        <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>
```

Slika 4.5. *Prikaz sadržaja AndroidManifest.xml datoteke*

Na slici 4.6. prikazan je izgled XML datoteke fragmenta na kojemu se nalazi karta. Google karta će biti prikazana uz pomoć komponente *MapView*.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_maps_frame"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="false">

    <com.google.android.gms.maps.MapView
        android:id="@+id/fragment_map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</FrameLayout>
```

Slika 4.6. *Layout resurs fragment_map.xml*

Kako bi se omogućio rad s kartom unutar aplikacije, potrebno je implementirati *OnMapReadyCallback* sučelje. Potrebno je postaviti povratni poziv (engl. *callback*) na fragment

uz pomoć *getMapAync* metode koja vraća instancu GoogleMap klase spremnu za korištenje. Metoda *onMapReady* omogućuje rukovanje GoogleMap objektom. Metoda se okida kada je karta spremna za korištenje. Unutar te metode, postavljeno je inicijalno zumiranje karte, dodani su markeri na kartu za koje su podatci dohvaćeni iz baze podataka, uključeno je praćenje lokacije putem GPS-a[14] u stvarnom vremenu, te je omogućen odlazak na singlicu[15] lokacije na način da se klikom na informacijski prozor markera prema imenu lokacije iz baze dohvate informacije o odabranoj lokaciji, te se one pomoću *bundlea*[16] prenesu na fragment singlice.

```
public class MapFragment extends Fragment implements OnMapReadyCallback {
    private String locationTitle;
    private String locationInfo;
    private String locationImage;
    //…
    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        MapView mapView = (MapView) view.findViewById(R.id.fragment_map);
        mapView.onCreate(savedInstanceState);
        mapView.onResume();
        try {
            MapsInitializer.initialize(getActivity().getApplicationContext());
        } catch (Exception e) {
            e.printStackTrace();
        }
        mapView.getMapAsync(this);
    }
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    @Override
    public void onMapReady(final GoogleMap googleMap) {
        googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(45.5553897, 18.6926184),
                12));
        //…
        db.collection("places")
                .get()
                .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document : task.getResult()) {
                                googleMap.addMarker(new MarkerOptions().title(document.getString("name")).position(new
                                        LatLng(document.getDouble("lat"), document.getDouble("lng"))));
                            }
                        } else {
                            Toast.makeText(getActivity(), "Something went wrong", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
        googleMap.setMyLocationEnabled(true);
        //…
}
```

Slika 4.7. *Prikaz klase MapFragment.java*

[14] Global Positioning System
[15] Naziv prikaza informacija jednog objekta
[16] Bundle daje mogućnost razmjene podataka između activitija

4.3. RecyclerView

RecyclerView je komponenta koja ima velike prednosti za korištenje pri prikazivanju velikog broja podataka. Podatci se prikazuju u obliku liste. Moć ove komponente je ta što reciklira svoje elemente i time ubrzava rad sebe same, a time i fluidniji i ugodniji korisnički doživljaj cijele aplikacije. Sličnu funkcionalnost omogućuje i *ListView*, no rjeđe se upotrebljava jer funkciju recikliranja mora implementirati sam korisnik.

Kako bi korištenje *RecyclerViewa* bilo moguće, potrebno ga je dodati u projekt pomoću *Gradle* skripte (slika 4.8.). Na slici su također prikazane i ostale implementacije dodane za potrebe izrade ove aplikacije.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support:design:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    //firebase
    implementation 'com.google.firebase:firebase-core:16.0.9'
    implementation 'com.google.firebase:firebase-firestore:18.0.0'
    implementation 'com.google.firebase:firebase-auth:16.2.1'
    //recyclerView
    implementation 'com.android.support:recyclerview-v7:28.0.0'
    //cardView
    implementation 'com.android.support:cardview-v7:28.0.0'
    //circleImageView
    implementation 'de.hdodenhof:circleimageview:3.0.0'
    //circleButton
    implementation 'com.github.markushi:circlebutton:1.1'
    //maps&locations
    implementation 'com.google.android.gms:play-services-maps:16.0.0'
    implementation "com.google.android.gms:play-services-location:16.0.0"
    //picasso
    implementation 'com.squareup.picasso:picasso:2.71828'
}
```

Slika 4.8. *Gradle skripta*

Na slici 4.9. nalazi se primjer *RecyclerViewa* koji će se koristiti za prikaz korisničkih iskustava. Može se primijetiti kako je na ovoj komponenti klizanje (engl. *scrolling*) omogućeno vertikalno.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorWhite">

    <at.markushi.ui.CircleButton
        android:id="@+id/cb_openInput"
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="10dp"
        android:elevation="5dp"
        android:onClick="openReviewInput"
        android:src="@drawable/ic_add_circle_black_24dp"
        app:cb_color="#BF275AA9"
        app:cb_pressedRingWidth="10dp" />

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerViewReviews"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorWhite"
        android:padding="4dp"
        android:scrollbars="vertical" />

</RelativeLayout>
```

Slika 4.9. *Layout resurs fragment_reviews.xml*

Korisnička iskustva trebaju svoju reprezentaciju u vidu vizualne komponente na zaslonu, stoga je potrebno izraditi XML datoteku koja će se koristiti za prikazivanje svakog pojedinog korisničkog iskustva unutar liste *RecyclerViewa*. Bazna reprezentacijska komponenta koja je korištena u ovom slučaju je *CardView*. Unutar nje, postavljen je *RelativeLayout* pomoću kojega su razvrstani još jedan *RelativeLayout* s *TextViewovima* za korisničko ime i tekst, te profilna slika.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/cv_review_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    app:cardCornerRadius="5dp">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <de.hdodenhof.circleimageview.CircleImageView xmlns:app="http://schemas.android.com/apk/res-auto"
            android:id="@+id/profile_image"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_margin="5dp"
            android:padding="5dp"
            android:src="@drawable/ic_person_white_24dp"
            app:civ_border_color="@color/colorWhite"
            app:civ_border_width="1dp"
            app:civ_circle_background_color="@color/appThemeColor" />

        <RelativeLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@+id/profile_image">

            <TextView
                android:id="@+id/tv_profile_nickname"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="15dp"
                android:layout_marginTop="10dp"
                android:layout_marginRight="10dp"
                android:textColor="@color/colorBlack"
                android:textSize="15sp"
                android:textStyle="bold" />

            <TextView
                android:id="@+id/tv_profile_review"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/tv_profile_nickname"
                android:layout_marginLeft="15dp"
                android:layout_marginTop="5dp"
                android:layout_marginRight="10dp"
                android:layout_marginBottom="15dp"
                android:textColor="#777777" />

        </RelativeLayout>
    </RelativeLayout>
</android.support.v7.widget.CardView>
```

4.10. *Layout resurs review_item.xml*

Adapter je klasa koja za zadatak ima kreiranje *ViewHoldera* i brigu o podacima koji će se prikazivati unutar *RecyclerViewa*. Tri su metode koje je potrebno iskoristiti kako bi adapter ispravno funkcionirao. Prva od njih je *onCreateViewHolder* čiji je zadatak kreiranje i recikliranje *ViewHoldera*. Nadalje, metoda *onBindViewHolder* povezuje podatke koji su u adapteru s *ViewHolderom*. Posljednja od tri metode je *getItemCount*, te ona vraća ukupni broj podataka koji se nalaze u *RecyclerViewu*.

ViewHolder je klasa koja reprezentira jedan element *RecyclerViewa*, a kreiranje mu se vrši unutar adaptera kojemu i pripada. *RecyclerView* stvara onoliko instanci *ViewHoldera* koliko je potrebno da se zaslon ispuni podatcima. Za podatke koji se nalaze izvan trenutnog prikaza zaslona reciklira se postojeći *ViewHolder*, te se popunjava novim podatcima, odakle i samo ime za *RecyclerView* prema njegovoj glavnoj funkcionalnosti.

```
public class ReviewsAdapter extends RecyclerView.Adapter<ReviewsAdapter.ReviewsViewHolder> {
    private ArrayList<ReviewItem> mReviewList;

    public class ReviewsViewHolder extends RecyclerView.ViewHolder {
        CardView mCardView;
        CircleImageView mCircleImageView;
        TextView mNickname;
        TextView mReview;

        public ReviewsViewHolder(View itemView) {
            super(itemView);
            this.mCardView = itemView.findViewById(R.id.cv_review_item);
            this.mCircleImageView = itemView.findViewById(R.id.profile_image);
            this.mNickname = itemView.findViewById(R.id.tv_profile_nickname);
            this.mReview = itemView.findViewById(R.id.tv_profile_review);
        }
    }

    public ReviewsAdapter(ArrayList<ReviewItem> reviewList) {
        mReviewList = reviewList;
    }
    @Override
    public ReviewsAdapter.ReviewsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.review_item, parent, false);
        ReviewsAdapter.ReviewsViewHolder holder = new ReviewsAdapter.ReviewsViewHolder(v);
        return holder;
    }
    @Override
    public void onBindViewHolder(@NonNull ReviewsViewHolder reviewViewHolder, int i) {
        ReviewItem currentItem = mReviewList.get(i);
        reviewViewHolder.mNickname.setText(currentItem.getNickname());
        reviewViewHolder.mReview.setText(currentItem.getText());
    }
    @Override
    public int getItemCount() {
        return mReviewList.size();
    }
}
```

Slika 4.11. *Prikaz klase ReviewsAdapter.java*

Unutar metode *buildRecyclerView* ostvareno je povezivanje *RecyclerViewa* i adaptera metodom *setAdapter*. Podatci koje drži adapter spremljeni su na Firestoreu[17]. Na slici 4.12. unutar *onViewCreated* metode nalazi se kod za dohvaćanje podataka iz baze. Prvi korak je instancirati bazu, te zatim na instanci, uz ime kolekcije koju se želi dohvatiti, pozvati *get* metodu. Svako pojedino dohvaćeno korisničko iskustvo pretvara se u objekt prema nacrtu klase *ReviewItem* u kojoj se nalaze konstruktori, te metode za postavljanje i dohvaćanje atributa (engl. *getteri* i *setteri*). Nakon toga se objekte dodaje u listu koja drži sva korisnička iskustva metodom *add*. Pozivom metode *notifyDataSetChanged* obavještava se adapter da je došlo do promjene, te se tada postavlja *RecyclerView*. Važno je napomenuti da se svakim pozivom ove metode *recyclerView* postavlja ponovno.

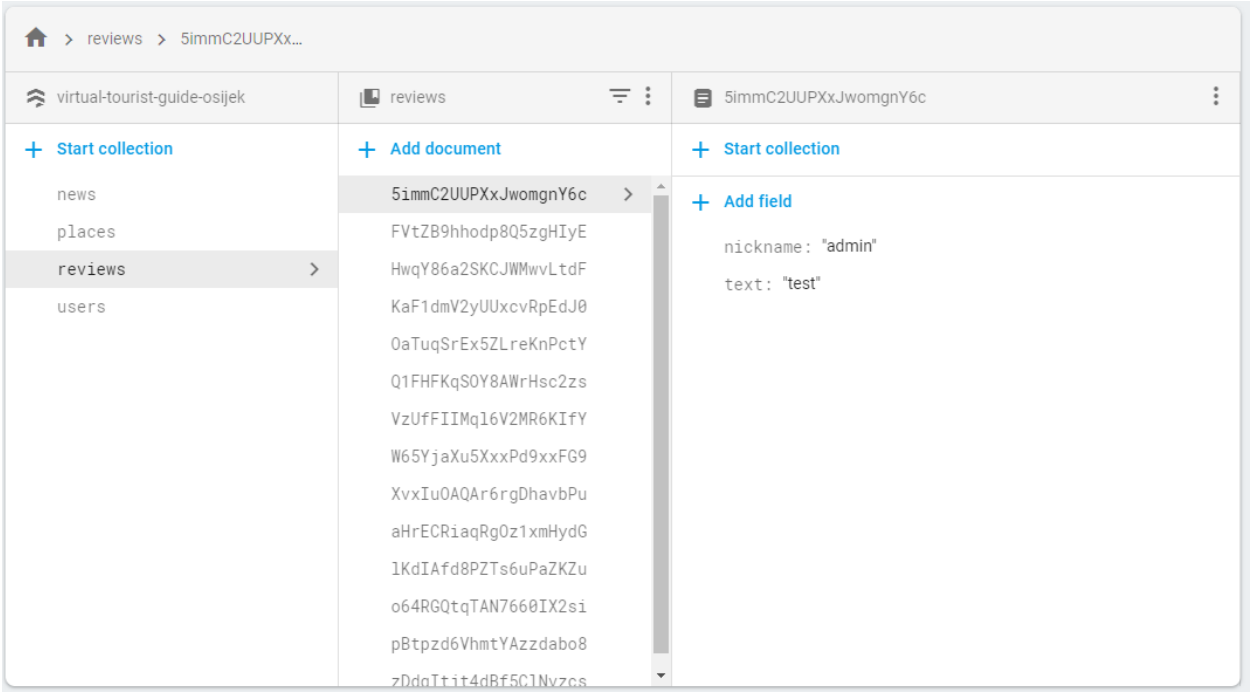
```
public class ReviewsFragment extends Fragment {
    private RecyclerView mRecyclerView;
    private ReviewsAdapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;
    private CircleButton cb;
    ArrayList<ReviewItem> mUserReviews = new ArrayList<ReviewItem>();

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable
            Bundle savedInstanceState) {
                return inflater.inflate(R.layout.fragment_reviews, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        buildRecyclerView(view);

        cb = view.findViewById(R.id.cb_openInput);
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection("reviews").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        ReviewItem item = document.toObject(ReviewItem.class);
                        mUserReviews.add(item);
                        mAdapter.notifyDataSetChanged();
                    }
                } else {
                    Toast.makeText(getActivity(), "Something went wrong", Toast.LENGTH_SHORT).show();
                }
            }
        });
        mAdapter.notifyDataSetChanged();
    }
    public void buildRecyclerView(View view) {
        mRecyclerView = view.findViewById(R.id.recyclerViewReviews);
        mLayoutManager = new LinearLayoutManager(getContext());
        mAdapter = new ReviewsAdapter(mUserReviews);
        mRecyclerView.setLayoutManager(mLayoutManager);
        mRecyclerView.setAdapter(mAdapter);
    }
}
```

Slika 4.12. *Prikaz klase ReviewsFragment.java*

[17] Fleksibilna, skalabilna NoSQL baza podataka u oblaku koja se koristi za pohranu i sinkronizaciju podataka

4.4. Cloud Firestore

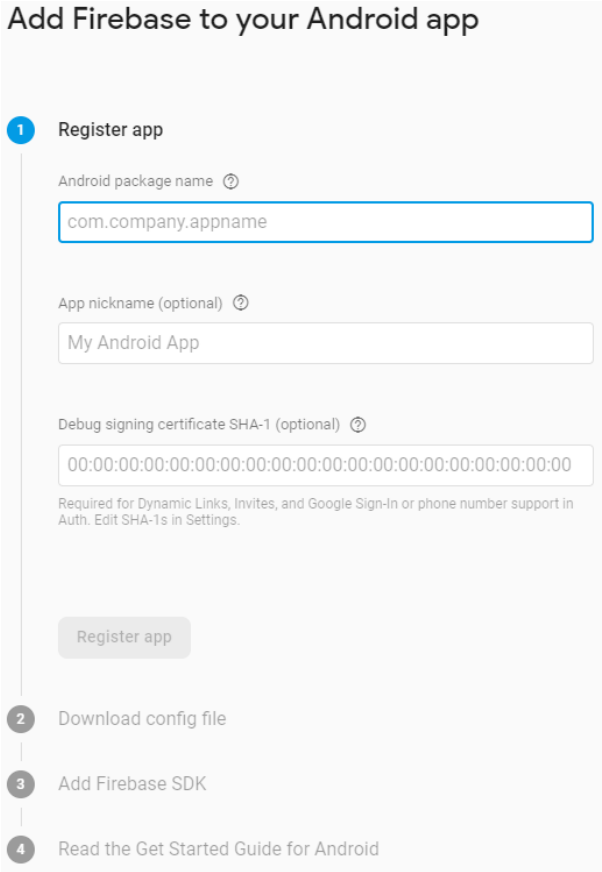
Firestore je fleksibilna, skalabilna NoSQL[18] baza podataka u oblaku za pohranu i sinkronizaciju podataka na klijentskoj i serverskoj strani razvoja. Moguće je povezivanje s Android, iOS i web projektima. Podatke pohranjuje kao kolekcije dokumenata.



Slika 4.13. *Struktura baze podataka koju aplikacija koristi*

U bazu se pohranjuju podatci o vijestima, lokacijama, iskustvima, te korisnicima. Svaki od spomenutih cjelina naziva se kolekcija. Unutar kolekcije pohranjuju se dokumenti s jedinstvenom identifikacijskom oznakom. Dokumenti se sastoje od polja koja sadrže podatke. Na slici 4.13. prikazan je primjer kolekcije iskustava koja sadrži dokumente s automatski generiranom identifikacijskom oznakom unutar kojih su pohranjena polja s nadimkom korisnika koji je objavio iskustvo, te tekst iskustva.

[18] Baza podataka koja nije zasnovana na relacijskim sustavima za upravljanje bazom podataka



Slika 4.14. *Dodavanje Firebasea Android aplikaciji*

Prvenstveno, kako bi aplikacija mogla komunicirati s Firebaseom, potrebno ju je povezati putem jednostavnog i intuitivnog sučelja u Firebase konzoli. Zatim, kako bi se aplikaciji omogućio pristup čitanju i pisanju u bazu, važno je to omogućiti u postavkama pravila Firestore baze. Naposljetku, baza se instancira unutar programskog koda prije nego što se pozove neka od njezinih metoda.

Na slici 4.15. prikazana je metoda za upisivanje u bazu. Metoda se poziva pritiskom na gumb za objavu iskustva na fragmentu za dodavanje iskustava korisnika. Zapisuju se korisnikov nadimak, te tekst koji korisnik unese. Prije svega, provjerava se je li korisnik unio tražene podatke. Ukoliko tekst nije unesen, korisnik se putem *toast* poruke obavještava da unese tekst kako bi nastavio s objavljivanjem, a ukoliko je sve uredu, metodom *add* vrši se zapis u kolekciju iskustava, te se automatski generira ID za dodani dokument.

```
public void submitPressed(View view) {
    mReview = findViewById(R.id.et_review);

    String review = mReview.getText().toString();

    Map<String, Object> data = new HashMap<>();
    data.put("nickname", mReviewNickname);
    data.put("text", review);

    if (!TextUtils.isEmpty(review)) {
        db.collection("reviews")
                .add(data)
                .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        Toast.makeText(getApplicationContext(), "Your review is added", Toast.LENGTH_SHORT).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(getApplicationContext(), "Something went wrong",
                                Toast.LENGTH_SHORT).show();
                    }
                });
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                new ReviewsFragment()).commit();
    } else {
        Toast.makeText(getApplicationContext(), "Please enter your review", Toast.LENGTH_SHORT).show();
    }
}
```

Slika 4.15. *Metoda za upisivanje u bazu podataka*

S druge strane, baza podataka također ima i mogućnost čitanja. Specificiranjem kolekcije iz koje se žele dohvatiti podatci, te pozivom metode *get* ta funkcija je i omogućena. Na slici 4.16. vidi se dohvaćanje iskustava i spremanje istih. Iterira se kroz cijelu kolekciju kako bi se pribavili svi dokumenti spremljeni u bazu.

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    buildRecyclerView(view);

    cb = view.findViewById(R.id.cb_openInput);

    FirebaseFirestore db = FirebaseFirestore.getInstance();

    db.collection("reviews").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    ReviewItem item = document.toObject(ReviewItem.class);
                    mUserReviews.add(item);
                    mAdapter.notifyDataSetChanged();
                }
            } else {
                Toast.makeText(getActivity(), "Something went wrong", Toast.LENGTH_SHORT).show();
            }
        }
    });
    mAdapter.notifyDataSetChanged();
}
```

Slika 4.16. *Metoda za čitanje iz baze podataka*

4.5. Firebase autentifikacija

Firebase autentifikacija nudi servise na serverskoj strani, jednostavne alate za razvijanje softvera (engl. *SDK*), te unaprijed spremne biblioteke korisničkih sučelja za prijavu u aplikaciju. Podržava prijavu i registraciju koristeći lozinku, broj mobitela, te popularne davatelje usluga kao što su Google, Facebook, Twitter i mnogi drugi. Podržane su funkcionalnosti kao što su slanje elektroničke pošte pri registraciji kako bi korisnik dovršio proces registriranja i potvrdio da je unio postojeću i vlastitu adresu e-pošte, izmjena zaporke, SMS potvrda korisničkog računa te brojne druge.

Prije korištenja, potrebno je uključiti željenu metodu prijave u Firebase konzoli. U ovom projektu korištena je metoda prijave putem elektroničke pošte i zaporke.

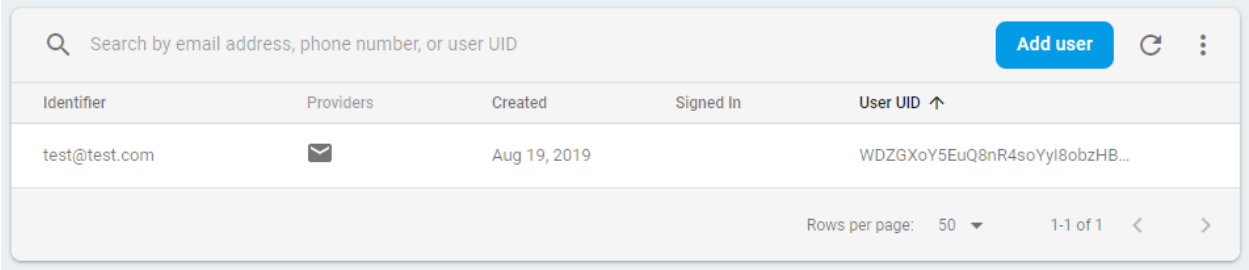
Slika 4.17. prikazuje metodu za registraciju korisnika. Korisnik unosi svoje korisničko ime, zatim adresu elektroničke pošte na koju je postavljena provjera ispravnosti, te željenu lozinku dvaput kako bi se izbjegao eventualni krivi unos. Niti jedno od spomenutih polja ne smije biti prazno, a također provjerava se i je li korisnik prihvatio uvjete korištenja. Metodom

createUserWithEmailAndPassword kreira se novi korisnik te se svi podatci osim njegove zaporke upisuju u bazu podataka kako bi bili dostupni za dohvaćanje na fragmentu profila. Šifra se ne zapisuje niti u bazu niti u tablicu kreiranih korisnika, ona je privatna i poznata samo korisniku. Ukoliko je kreiranje korisnika uspješno, automatski se vrši prijava u aplikaciju.

```
public void registerUser(View view) {
    String username, email, password, passwordConfirm;
    Boolean agreed;

    mUsername = findViewById(R.id.et_registration_username);
    mEmail = findViewById(R.id.et_registration_email);
    mPassword = findViewById(R.id.et_registration_password);
    mPasswordConfirm = findViewById(R.id.et_registration_password_confirm);
    mCb = findViewById(R.id.cb_registration);
    username = mUsername.getText().toString();
    email = mEmail.getText().toString();
    password = mPassword.getText().toString();
    passwordConfirm = mPasswordConfirm.getText().toString();
    agreed = mCb.isChecked();

    final Map<String, Object> userInfo = new HashMap<>();
    userInfo.put("username", username);
    userInfo.put("email", email);

    final FirebaseFirestore db = FirebaseFirestore.getInstance();
    if (username.isEmpty() || email.isEmpty() || password.isEmpty() || passwordConfirm.isEmpty() || !agreed) {
        Toast.makeText(getApplicationContext(), "Please enter all data and agree to the terms and conditions",
                Toast.LENGTH_SHORT).show();
    } else {
        if (!isEmailValid(email)) {
            Toast.makeText(getApplicationContext(), "Please enter valid email address",
                    Toast.LENGTH_SHORT).show();
        } else if (!password.equals(passwordConfirm)) {
            Toast.makeText(getApplicationContext(), "Please enter matching passwords",
                    Toast.LENGTH_SHORT).show();
        } else {
            mAuth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                Toast.makeText(getApplicationContext(), "Account created",
                                        Toast.LENGTH_SHORT).show();
                                getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                                        new MapFragment()).commit();
                                setBottomNavigationVisibility(true);
                                db.collection("users").document(mAuth.getCurrentUser().getUid()).set(userInfo);
                            } else {
                                Toast.makeText(getApplicationContext(), task.getException().getMessage(),
                                        Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
        }
    }
}
```

Slika 4.17. *Metoda za registraciju korisnika*

Tablica registriranih korisnika (slika 4.18.) sadrži informacije o svakom pojedinom korisniku. U njoj se može pronaći više o tome koji identifikacijski podatak korisnik koristi pri prijavi, koji je pružatelj usluge (npr. prijava metodom e-pošta/zaporka, Google, Microsoft…), datum kreiranja korisničkog računa, datum zadnje prijave u aplikaciju, te naposljetku jedinstvena oznaka prema kojoj se svaki korisnik unutar Firebase projekta razlikuje.



Slika 4.18. *Izgled tablice registriranih korisnika*

Slika 4.19. prikazuje metodu za prijavu u aplikaciju. Pri prijavi se prvotno provjerava jesu li unesena sva obavezna polja, odnosno je li korisnik unio adresu elektroničke pošte i zaporku. Ukoliko je ta provjera uspješna, poziva se metoda *signInWithEmailAndPassword* te se u slučaju uspješne autentifikacije korisniku pogled prebacuje na fragment karte, a donja traka navigacije postaje vidljiva. Od ovog trenutka na instanci *FirebaseAutha* može se pozvati metoda *getCurrentUser* kojom se dohvaća trenutno prijavljeni korisnik, a metoda vraća objekt koji predstavlja korisnika.

```
public void logIn(View view) {
    String email, password;

    mEmailLogin = findViewById(R.id.et_login_email);
    mPasswordLogin = findViewById(R.id.et_login_password);

    email = mEmailLogin.getText().toString();
    password = mPasswordLogin.getText().toString();

    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Please enter all data",
                Toast.LENGTH_SHORT).show();
    } else {
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
                                    new MapFragment()).commit();
                            setBottomNavigationVisibility(true);
                            Toast.makeText(getApplicationContext(), "Welcome",
                                    Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(getApplicationContext(), task.getException().getMessage(),
                                    Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
}
```

Slika 4.19. *Metoda za prijavu korisnika*

Odjava korisnika vrši se jednostavno metodom prikazanom na slici 4.20. na način da se pozove metoda *signOut*, a korisnikov pogled se prebacuje na fragment za prijavu.

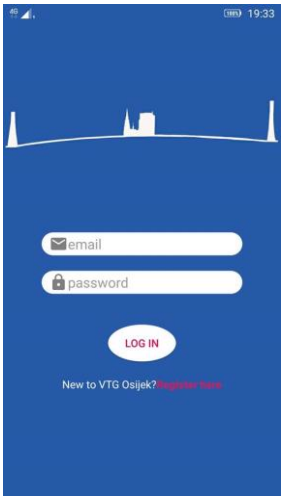
```
public void logOut(View view) {
    mAuth.signOut();
    Toast.makeText(getApplicationContext(), "Logged out", Toast.LENGTH_SHORT).show();
    getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, new
            LoginFragment()).commit();
}
```

Slika 4.20. *Metoda za odjavu korisnika*

5. KORIŠTENJE APLIKACIJE

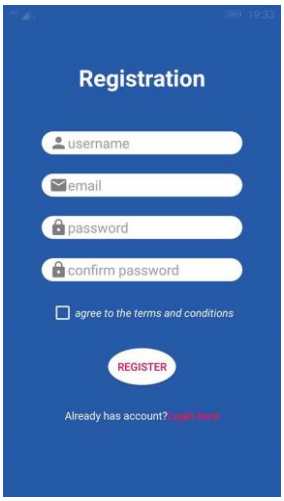
Prije samoga pokretanja aplikacije, kako bi se osigurao ispravan rad, praćenje lokacije korisnika u stvarnome vremenu i dohvaćanje zahtijevanih informacija, važno je da korisnik uključi GPS i mobilne podatke.

Na slici 5.1. prikazan je izgled fragmenta za prijavu i to je inicijalni zaslon koji korisnik vidi pri pokretanju. Unosom e-pošte i zaporke, korisnik se prijavljuje i pogled mu se prebacuje na fragment na kojemu se nalazi karta. Ukoliko korisnik nema kreiran račun, može se prebaciti na registraciju pritiskom na tekst ispod gumba za prijavu.



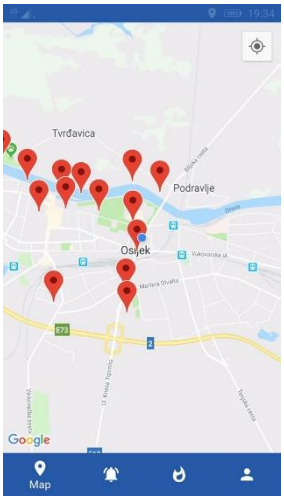
Slika 5.1. *Pogled na prijavu u aplikaciju*

Obrazac za registraciju (slika 5.2.) od korisnika zahtijeva unos korisničkog imena, elektroničke pošte, lozinke i prihvaćanje uvjeta korištenja. Registracija nije moguća ukoliko korisnik nije unio sva polja ili nije prihvatio uvjete korištenja, ukoliko je unio adresu elektroničke pošte neispravnog formata ili ukoliko se unesene lozinke ne podudaraju. Prilikom uspješne registracije korisnika se automatski prijavljuje u aplikaciju. Ukoliko se netko nepotrebno nađe na ovom prozoru, povratak na prijavu mu je omogućen putem pritiska na tekst ispod gumba za registraciju.

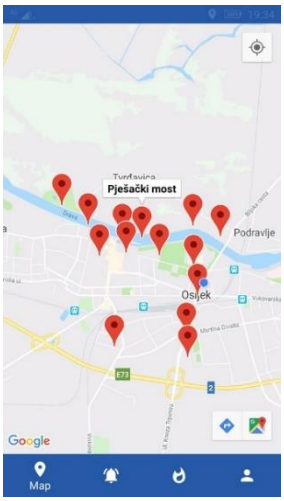


Slika 5.2. *Obrazac za registraciju*

Pogled prikazan na slici 5.3. korisnici će vidjeti automatski nakon uspješne prijave ili registracije ili u slučaju da su pri pokretanju aplikacije već bili prijavljeni. Plava točkica prikazuje trenutnu lokaciju korisnika. Karti se pri pokretanju pogled postavlja na grad Osijek, no klizanjem se može otići i dalje. Pritiskom na gumb u desnom gornjem kutu, pogled se centrira na trenutnu lokaciju korisnika. Ukoliko se korisnik pomiče, pomicat će se i njegova oznaka na zaslonu u stvarnome vremenu. Svaki marker predstavlja jednu od turistički atraktivnih lokacija, a pritiskom na marker otvara se informacijski prozor koji tada pokazuje ime odabrane lokacije (slika 5.4.). Daljnjim pritiskom na sami informacijski prozor, omogućeno je odlaženje na singlicu lokacije te dohvaćanje i prikaz informacija iz baze podataka (slika 5.5.).



Slika 5.3. *Fragment na kojemu se nalazi karta*



Slika 5.4. *Informacijski prozor markera lokacije*

Slika 5.5. *Singlica lokacije*

Na singlici lokacije korisnik može vidjeti naziv lokacije, fotografiju s te lokacije kako bi ju lakše pronašao/uočio, te najbitnije informacije.

Dolaskom na fragment novosti (slika 5.6.) dobivaju se informacije o prošlim, trenutnim i budućim događanjima unutar grada.

Slika 5.6. *Fragment novosti*

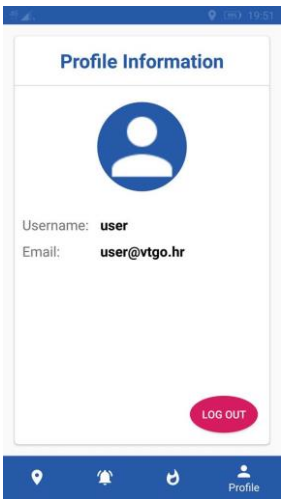
Otvaranjem fragmenta iskustava (slika 5.7.) korisnik može vidjeti mišljenja ostalih korisnika o lokacijama, te preporuke koje bi lokacije posebno trebao posjetiti. Korisnik također može upisati i svoje iskustvo pritiskom na gumb u donjem desnom kutu pri čemu mu se pogled prebacuje na obrazac za unos (slika 5.8.).

Slika 5.7. *Fragment korisničkih iskustava*

Korisnik upisuje željeni tekst koji je obavezan (nije moguće izvršiti praznu objavu), a korisničko ime se postavlja automatski dohvaćanjem korisničkog imena trenutno prijavljenog korisnika. Nakon objave (ili odustajanjem od objave pritiskom gumba za povratak), pogled se prebacuje na fragment iskustava svih korisnika.

Slika 5.8. *Obrazac za unos iskustva*

Fragment korisničkog profila (slika 5.9.) daje informacije o korisničkom imenu i adresi e-pošte trenutno prijavljenog korisnika. Također, u donjem desnom kutu nalazi se gumb na čiji se pritisak izvršava odjava iz aplikacije.



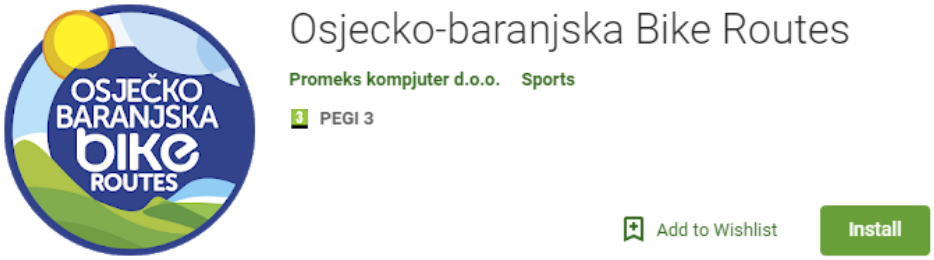
Slika 5.9. *Fragment korisničkog profila*

6. POSTOJEĆA RJEŠENJA

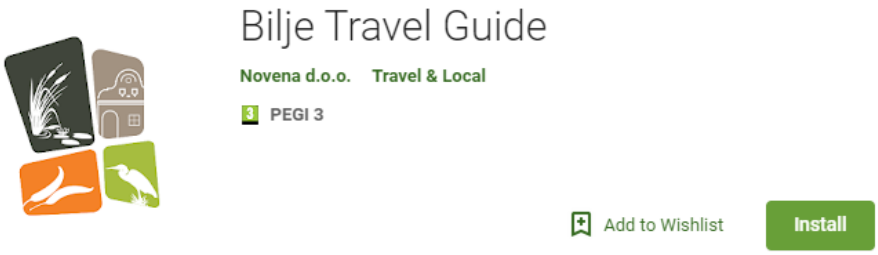
Dva su slična postojeća aplikacijska rješenja na Google Playu, a to su: Osjecko-baranjska Bike Routes (slika 6.1.) i Bilje Travel Guide (slika 6.2.).

Aplikacija Osjecko-baranjska Bike Routes specijalizirana je za cikloturiste. Koristi Google karte za prikaz i odabir ruta, a uz to također omogućuje i informiranje korisnika o pojedinostima rute kao što su duljina, razlika nadmorske visine, te vrsta podloge po kojoj se vozi. Nadalje, aplikacija korisnicima dostavlja informacije o turističkim sadržajima na koji putem nailaze.

Bilje Travel Guide aplikacija je koja turistima nudi informacije o turističkoj ponudi Bilja, mjesta nedaleko od Osijeka. Informacije koje se nude su: prirodne znamenitosti, gastronomska ponuda, smještaj, događaji u okruženju, te multimedijski sadržaj.



Slika 6.1. *Aplikacija Osjecko-baranjska Bike Routes*[19]



Slika 6.2. *Aplikacija Bilje Travel Guide*[20]

[19] Preuzeto: https://play.google.com/store?hl=en (28.8.2019.)
[20] Preuzeto: https://play.google.com/store?hl=en (28.8.2019.)

7. ZAKLJUČAK

Pametni telefoni i aplikacije koje dolaze s njima pustili su duboko korijenje u sve pore modernog društva i postali važan dio života za nebrojeno mnogo korisnika diljem svijeta. Cilj ovog završnog rada bio je izraditi funkcionalnu i društveno korisnu aplikaciju koja će turistima koji posjećuju grad Osijek taj posjet učiniti edukativnijim i što bolje iskorištenim, ali isto tako i jednostavnijim, te uspješnijim.

Aplikacija sadrži sve važne funkcionalnosti koje su potrebne turistu u nepoznatom gradu. Korisniku je omogućeno kreiranje vlastitog korisničkog računa koji kasnije koristi za prijavu, te za identifikaciju pri objavljivanju iskustava u aplikaciji. Uz praćenje trenutne lokacije olakšano je dolaženje do željene znamenitosti u gradu jer korisnik u svakom trenutku može vidjeti kreće li se u pravome smjeru. Dolaskom do same znamenitosti može provjeriti je li na željenom mjestu uspoređujući fotografiju lokacije s onime što on vidi s mjesta na kojemu se nalazi. Također, ponuđene su mu najvažnije informacije o znamenitostima. Putem aplikacije doznaje vijesti o svim događanjima u gradu koja bi turistima mogla biti privlačna, ali i mnoge druge turistički relevantne vijesti.

Dodatne funkcionalnosti koje bi se u daljnjem radu mogle omogućiti kako bi aplikacija bila još pristupačnija turistima jest crtanje putanje od trenutne lokacije korisnika pa do odabranog markera na karti, odnosno odabrane lokacije. Radi veće razine personalizacije korisničkog profila, moglo bi se omogućiti postavljanje profilne slike, ali također i mijenjanje zaporke te oporavak korisničkog računa u slučaju zaboravljene zaporke. Trenutna velika slabost aplikacije jest što su sve informacije koje se dohvaćaju iz baze podataka na hrvatskom jeziku, što bi bilo poželjno ispraviti daljnjim radom, odnosno za početak dodati barem engleski jezik. Naposljetku, poželjno bi bilo dodati i fragment na kojima će turisti dobivati informacije o gastro ponudi, te smještaju u gradu Osijeku i bližoj okolici.

LITERATURA

[1] Hrvatska – Puna života (24.6.2019.)
https://croatia.hr

[2] Turistička zajednica grada Osijeka (24.6.2019.)
https://www.tzosijek.hr

[3] Hrvatska turistička zajednica (24.6.2019.)
https://htz.hr

[4] Službena Android dokumentacija (26.6.2019.)
https://developer.android.com

[5] Operacijski sustav Android (26.6.2019.)
https://en.wikipedia.org/wiki/Android_(operating_system)

[6] Windows Mobile operacijski sustav (26.6.2019.)
https://en.wikipedia.org/wiki/Windows_Mobile

[7] Symbian operacijski sustav (26.6.2019.)
https://en.wikipedia.org/wiki/Symbian

[8] Verzije Androida (26.6.2019.)
https://developer.android.com/about/dashboards

[9] Java programski jezik (27.6.2019.)
https://en.wikipedia.org/wiki/Java_(programming_language)

[10] Fragmenti (27.6.2019.)
https://developer.android.com/reference/android/app/Fragment

[11] Google Maps dokumentacija (27.6.2019.)
https://developers.google.com/maps/documentation/android-sdk/intro

[12] Recycler View (28.6.2019.)
https://developer.android.com/guide/topics/ui/layout/recyclerview

[13] Java programski jezik (28.6.2019.)
https://developer.ibm.com/tutorials/j-introtojava1/

[14] Firestore (28.6.2019.)
https://firebase.google.com/docs/firestore

[15] Firebase autentifikacija (4.8.2019.)
https://firebase.google.com/docs/auth

[16] GIMP službena stranica (10.8.2019.)
https://www.gimp.org/

[17] GIMP (10.8.2019.)
https://hr.wikipedia.org/wiki/GIMP

[18] Introduction to Firebase (20.8.2019.)
https://hackernoon.com/introduction-to-firebase-218a23186cd7

[19] Google Play (28.8.2019.)
https://play.google.com/store?hl=en

[20] Stack Overflow (28.8.2019.)
https://stackoverflow.com/

SAŽETAK

Cilj ovoga završnog rada bio je izraditi Android aplikaciju koja će voditi turiste do najvažnijih lokacija u Osijeku i dati im važne i poučne podatke o gradu. U početku ovoga seminara nalazi se kratki uvod u moderne tehnologije koje se danas koriste u turizmu. Nadalje, objašnjeni su Android operacijski sustav, integrirano razvojno okruženje Android Studio, Java programski jezik, razvojna platforma Firebase i GIMP softver. Glavni dio pruža kronološki pregled implementacije ovoga softvera s pojašnjenjima svih najznačajnijih funkcionalnosti. Dan je cjeloviti pregled konačnog proizvoda s korisničkim uputstvima. Na kraju seminara nalaze se prijedlozi za nadogradnju, te plan za daljnji razvoj ovoga projekta.

Ključne riječi: Android, Java, Firebase, Google Maps, Turizam

ABSTRACT

Android application – Virtual tourist guide for city of Osijek

The main goal of this final paper was to create an Android application that will guide tourists to the most important locations in Osijek and give them relevant and educational information about the city. Firstly, there is a brief introduction about modern technologies used in tourism today. Furthermore, Android operating system, Android Studio integrated development environment, Java programming language, Firebase development platform and GIMP software are explained. The main part provides a chronological order of implementation of this software solution with clarifications for all fundamental functionalities. There is also a full preview of final product with a step-by-step user guide. At the end of this paper, there are some suggestions for upgrades and a plan for future development of this project.

Keywords: Android, Java, Firebase, Google Maps, Tourism

ŽIVOTOPIS

Marijo Kiš rođen je 2. svibnja 1997. godine u Virovitici. Pohađao je Osnovnu školu Vladimira Nazora u Virovitici gdje biva nagrađen kao izuzetno uspješan učenik (završio je svih 8 razreda s prosjekom 5,00). 2012. godine upisuje Gimnaziju Petra Preradovića u Virovitici, opći smjer. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja bio je sudionik raznih natjecanja od kojih je značajnije rezultate postizao na županijskoj razini iz predmeta biologija i geografija. 2016. godine polaganjem državne mature završava srednju školu, te iste godine upisuje preddiplomski sveučilišni studij na Fakultetu za elektrotehniku, računarstvo i informacijske tehnologije Osijek, smjer Računarstvo. Postaje stipendist općine Lukač na temelju uspješnosti tokom srednjoškolskog obrazovanja. U vrijeme pisanja ovog završnog rada zaposlen je kao student developer u Gauss Development-u. Vrlo se dobro služi engleskim jezikom, a posjeduje osnovno znanje njemačkog jezika, te vozačku dozvolu B kategorije.

PRILOZI

CD

-izvorni kod aplikacije

-.docx i .pdf verzija završnog rada