Mobilna aplikacija za promociju turizma u Daruvaru

Đuranović, Tomislav

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

TOMISLAV ĐURANOVIĆ

Mobilna aplikacija za promociju turizma u Daruvaru

Diplomski rad

Pula, rujan, 2019. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

TOMISLAV ĐURANOVIĆ

Mobilna aplikacija za promociju turizma u Daruvaru

Diplomski rad

JMBAG: 0303054493, redoviti student
Studijski smjer: Informatika

Predmet: Mobilne aplikacije
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Siniša Sovilj

Pula, rujan, 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Tomislav Đuranović, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, 2019. godine

IZJAVA

o korištenju autorskog djela

Ja, Tomislav Đuranović dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom *Mobilna aplikacija za promociju turizma u Daruvaru* koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 20. ožujka 2019.

DIPLOMSKI ZADATAK

Pristupnik: **Đuranović Tomislav (0303054493)**
Studij: Sveučilišni diplomski studij Informatike

Naslov (hrv.): **Mobilna aplikacija za promociju turizma u Daruvaru**
Naslov (eng.): Mobile application for tourism promotion in Daruvar

Opis zadatka: Zadatak je izraditi mobilnu Android aplikaciju kojoj bi primarna funkcija bila promoviranje turističkog sadržaja grada Daruvara. Aplikacija treba većim dijelom biti u obliku interaktivne igre čiji je cilj obilazak najatraktivnijih lokacija u Gradu, te rješavanje kviza vezanog za posjećene lokacije. Uspješno riješenim kvizom, korisnici bi imali mogućnost ostvariti razne pogodnosti u samom Gradu. Također, aplikacija treba sadržavati informacije vezane uz iznajmljivanje smještaja, usluge ugostiteljskih objekata, sportsko-rekreacijskih sadržaja i slično. Korištenje mape i lokacije realizirat će se Google API servisima, a razvoj aplikacije pomoću Android Studio razvojnog okruženja.

Zadatak uručen pristupniku: 20. ožujka 2019.
Rok za predaju rada: 01. veljače 2020.

Mentor:

_____
doc.dr.sc. Siniša Sovilj

SADRŽAJ

1. UVOD

Vrlo važna grana gospodarstva, kako u Hrvatskoj, tako i na globalnoj razini, je turizam. U mnogim zemalja on predstavlja vrlo važan dio ekonomije te njegova važnost svakim danom sve više raste. Zasigurno, kao jedan od bitnih čimbenika odgovornih za razvoj turizma je informacijsko-komunikacijska tehnologija (eng. *ICT*). Danas postoje mnoge mobilne i web aplikacije koje korisnicima omogućuju pronalazak i rezerviranje smještaja, dostupnost informacija o mjestu koje posjećuju, upute, odnosno informacije o putovanju itd. Uz navedeni primjer spoja tehnologije i turizma, postoje sustavi za interno korištenje namijenjeni poslovnim objektima (npr. hotelima) koji omogućuju upravljanje cjelokupnim poslovanjem.

Ovaj diplomski rad opisuje potrebu za stvaranjem aplikacije za promoviranje turističkog sadržaja grada Daruvara, kao i razvoj, odnosno implementaciju iste. Daruvar, poznat kao najuređeniji mali grad kontinentalne Hrvatske, sadrži bogatu ponudu turističkih sadržaja i aktivnosti. Velik dio ponude Grada jesu Daruvarske toplice koje su dio zdravstvenog turizma te, kao takve, upotpunjuju turistički sadržaj, odnosno promoviraju Grad. Motiv razvoja aplikacije *Smart Daruvar* jest upravo promocija i upotpunjavanje turističke ponude. Mobilna aplikacija *Smart Daruvar* nudi korisnicima upoznavanje Grada na zanimljiv i zdrav način, obilaženjem najatraktivnijih lokacija te igranjem kviza vezanog uz posjećene lokacije. Aplikacija nudi pregled mjesta u blizini korisnika, informacije o Daruvaru itd. Ista je dostupna na dva jezika (hrvatski i engleski) što povećava broj potencijalnih korisnika te podržava višejezičnost.

Rad se sastoji od šest poglavlja. Prvo poglavlje opisuje strukturu rada, dok drugo opisuje tehnologije i turizam općenito te se dijeli na dva potpoglavlja: *Turističke aplikacije* (navodi se primjer dvije aplikacije te njihova usporedba) i *Turizam u Daruvaru* (opisuje stanje turizma u Daruvaru kao i njegovu trenutnu ponudu). Treće potpoglavlje naziva *Operacijski sustav Android*, grafički prikazuje zastupljenost mobilnih operacijskih sustava i dosadašnje verzije sustava *Android*. Također, prikazuje i opisuje slojeve arhitekture sustava. Poglavlje *Korištene tehnologije* navodi tehnologije koje su korištene za izradu *Smart Daruvar* mobilne i web aplikacije. Dijeli se na tri potpoglavlja koja

opisuju navedene tehnologije (*Firebase i API's*, *Android Studio* i *PHP Storm*). Peto poglavlje *Implementacija Smart Daruvar aplikacije* opisuje razvoj aplikacije pomoću prethodno navedenih tehnologija. U potpoglavlju *Baza podataka* definirana je struktura baze, dok se dva sljedeća potpoglavlja (*Mobilni dio aplikacije* i *Web dio aplikacije*) bave implementacijom funkcionalnosti, odnosno prikazom korisničkih scenarija. Nakon toga se, u poglavlju *Zaključak*, sažima navedeno i opisuju postignuti rezultati.

2. TEHNOLOGIJE I TURIZAM

Turizam je društveni, kulturni i ekonomski fenomen koji označava kretanje ljudi izvan mjesta boravka zbog osobnih ili poslovnih razloga. Osobe koje to čine nazivaju se posjetiteljima, odnosno turistima ili izletnicima. Posjetitelj je osoba koja provede manje od jedne godine izvan mjesta stanovanja i koja ne ostvaruje prihod u posjećenim destinacijama. Posjetitelj je izletnik ako boravi u destinaciji do 24 sata, u suprotnom postaje turist.

Turizam, kao takav, utječe na gospodarstvo, prirodu i izgrađeni okoliš, lokalno stanovništvo u destinaciji i na same turiste (UNWTO, 2008:1-10). Prema Gržinić (2019:198) „Turizam u čitavome svijetu trenutno proživljava revoluciju u globalnom kontekstu, koriste se nove tehnologije, mijenjaju se navike turista tj. potrošača. Nove tehnologije promijenile su turizam u posljednja dva desetljeća te će se isto razvijati i u budućem razdoblju.“. Iz navedenog se može zaključiti da tehnologije imaju sve veći utjecaj na turizam koji postaje ovisan o istima.

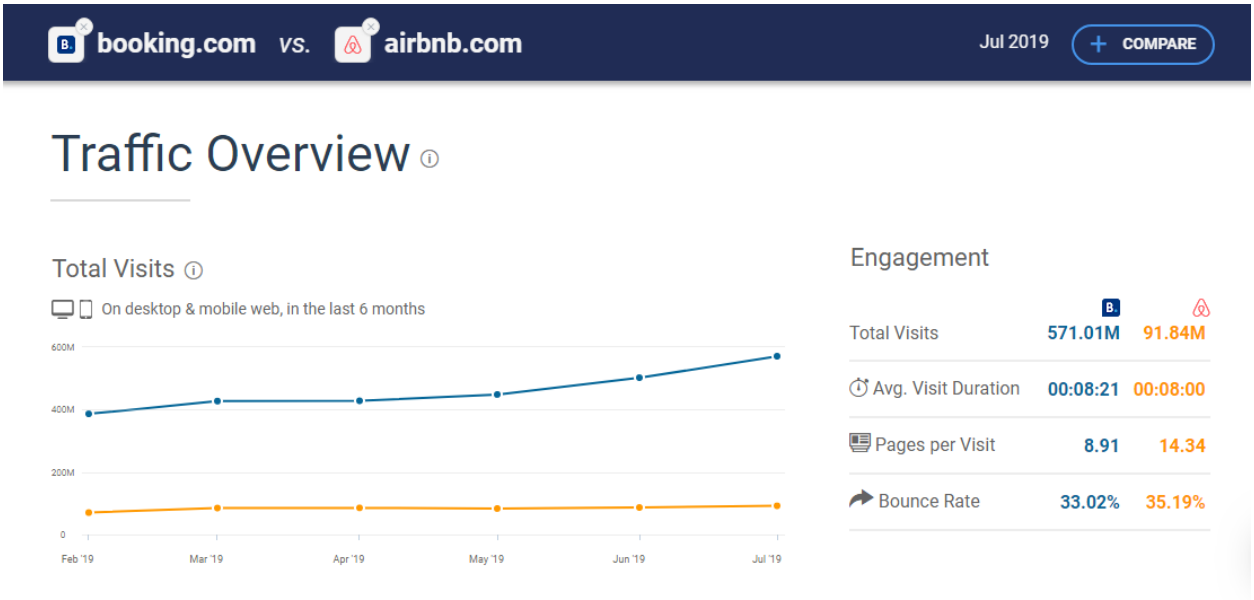
Takozvani *ICT* je spoj informacijske i komunikacijske tehnologije i, kao takav, pruža korisne informacije potrošačima turističkih proizvoda i usluga (Gržinić i Floričić, 2015:96). Stoga će se, u sljedećem potpoglavlju prikazati međusobna ovisnost tehnologije i turizma na konkretnim primjerima aplikacija.

2.1. TURISTIČKE APLIKACIJE

Booking.com je jedna od najvećih putničkih e-trgovina na svijetu koja je osnovana 1996. godine u Amsterdamu kao *startup*. Misija im je potaknuti ljude da istraže svijet, tako što ulažu u digitalnu tehnologiju te čine putovanja jednostavnijima. *Booking.com* povezuje putnike s najvećim izborom zadivljujućih mjesta za boravak, što uključuje sve, od apartmana, kuća za odmor, obiteljskih pansiona, pa do luksuznih odmarališta s pet zvjezdica, kuća na drveću, čak i iglua. *Booking.com* sadrži web i mobilnu aplikaciju na više od 40 jezika, nudeći više od 29 milijuna smještaja na preko 154 tisuće lokacija unutar 227 zemalja diljem svijeta (Booking.com, 2019).

Airbnb je osnovan 2008. godine kao platforma koja služi za turistička putovanja. Za cilj

ima što bolje iskoristiti tehnologiju u svrhu ostvarivanja ekonomske dobiti kroz iznajmljivanje prostora. *Airbnb* pruža pristup u više od šest milijuna jedinstvenih mjesta za boravak u više od 100 tisuća gradova, odnosno u 191 zemlji. Naglasak stavlja na upoznavanje turista s lokalnim stanovništvom, njihovim načinom života i kulturom (Airbnb, 2019). Na sljedećoj slici (Slika 1) se nalazi usporedba prethodno opisanih turističkih platformi.



Slika 1. Posjećenost web stranica (*Booking.com* i *Airbnb*)

Izvor: *SimilarWeb*, URL: https://www.similarweb.com/website/booking.com?competitors=airbnb.com (19.8.2019)

Slika prikazuje usporedbu posjećenosti web stranica *Booking.com* i *Airbnb.com* u posljednjih šest mjeseci putem računala i mobilnih uređaja. Web stranica *Booking.com* bilježi više od 570 milijuna posjeta, dok *Airbnb* nešto više od 91 milijun. Zadržavanje na web stranicama je u prosjeku oko osam minuta, a broj otvorenih podstranica pri jednom posjetu iznosi 8.91 za *Booking.com*, odnosno 14.34 za *Airbnb*. Iz prethodno navedenog, može se zaključiti da su *Booking.com* i *Airbnb* vrlo utjecajne platforme za pretragu i rezervaciju smještaja u svijetu, što ostvaruju putem *ICT* tehnologija. Višemilijunski pregledi web stranica potvrđuju zainteresiranost za putovanja i upoznavanje svjetskih atrakcija.

2.2. TURIZAM U DARUVARU

Daruvar je najuređeniji mali grad kontinentalne Hrvatske, što potvrđuju čak četiri zlatna, dva srebrna i jedno brončano priznanje Hrvatske turističke zajednice. S tradicijom duljom više od dva tisućljeća, Daruvar je poznati lječilišni grad. Osim toga, u Daruvaru se nalazi Savez Čeha Republike Hrvatske, stoga je središte svih društvenih i kulturnih zbivanja češke nacionalne manjine. Neke od poznatijih manifestacija koje se održavaju u Gradu su: Vincekovo, Vinodar, Darfest, Maska, Martinje, Božićni sajam itd. (Visitdaruvar, 2019). Područje Daruvar-Papuk raspolaže raznovrsnim prirodnim resursima pogodnima za korištenje u turističke i izletničke svrhe. Neki od resursa koji se danas koriste su:

- Ribnjaci u Daruvaru i Končanici (ribolov)
- Izvori termalne vode i ljekovito blato (kupanje i rehabilitacija)
- Vina (ugostiteljstvo i degustacija)
- Pčele (proizvodnja suvenira od meda)
- Park-šume (šetnja)
- Vranjevina i Petrov vrh (planinarenje) itd.

U odnosu na kulturne resurse, prirodni resursi i atrakcije se u većoj mjeri koriste te su uglavnom regionalnog, odnosno nacionalnog karaktera (MarCon, 2011:16).

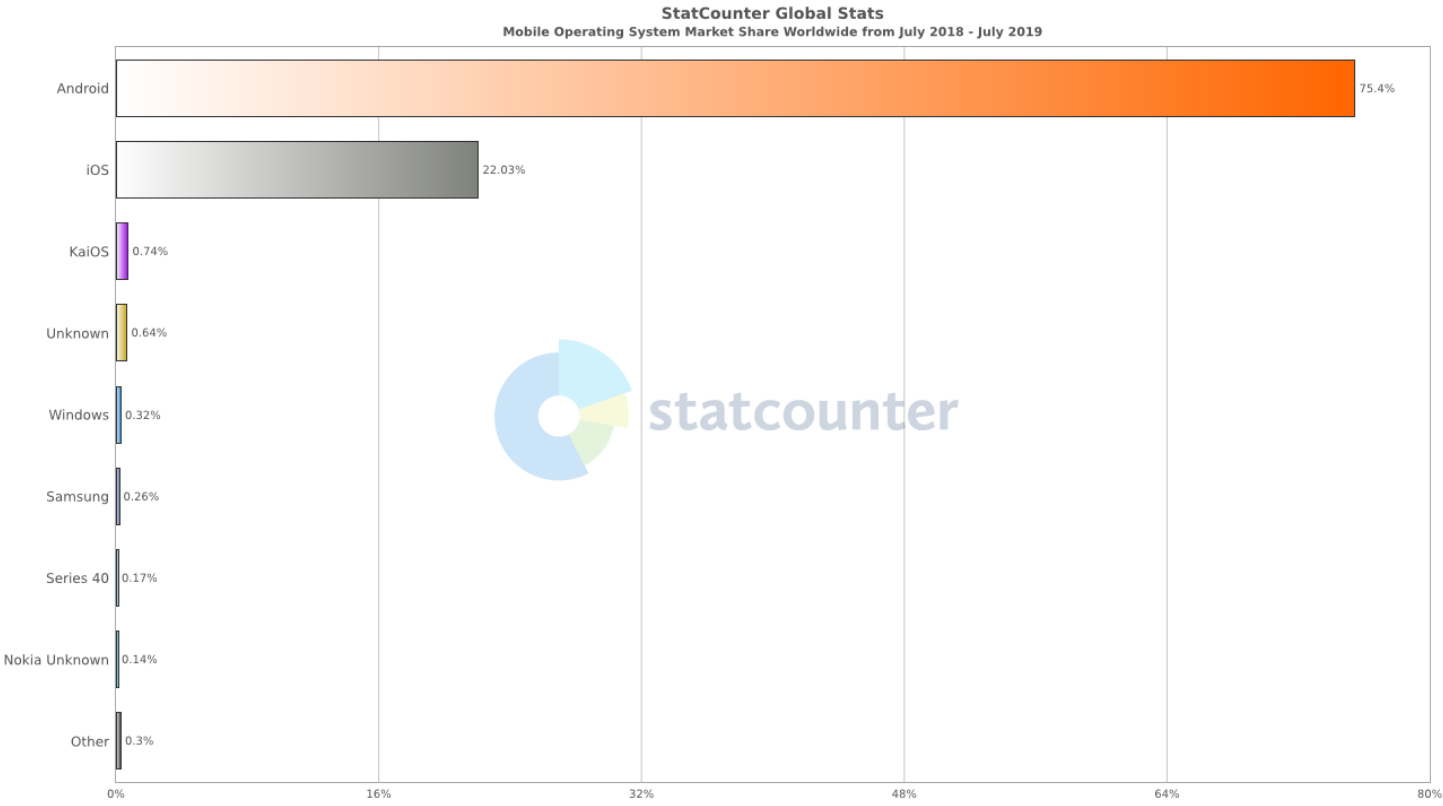
Što se tiče zdravstvenog turizma, u Daruvaru se nalaze Daruvarske toplice koje su poznato lječilište još od rimskog doba. Daruvarske toplice sadrže nekoliko izvora tople vode, čija se temperatura kreće od 39 do 47°C, a koji se uz mineralno blato koriste za liječenje pretežno reumatskih i ženskih bolesti. Također, u toplicama se provode i medicinske rehabilitacije i rekreacija. Uz široku ponudu zdravstvenih usluga, toplice nude i korištenje otvorenog i zatvorenog bazena, nogometnog igrališta, terena za tenis, gimnastičke dvorane i mnogih drugih sadržaja (TZ Daruvar, 2019).

Bačić i Medak (2012:212) navode sljedeće: „U županijama gdje postoje i djeluju toplice i lječilišta, toplice nisu nositelji samo zdravstvenog turizma, već i nositelji turizma uopće te županije. To je vidljivo i na primjeru Bjelovarsko-bilogorske županije za 2011. godinu.

U Županiji su ukupno 823 postelje za turiste (hoteli i moteli). Te godine je ostvareno nešto preko 36.000 noćenja. Od toga Daruvarske toplice same ostvaruju preko 30 % noćenja, iako su zdravstvena ustanova.“. Stoga, zdravstveni je turizam u Daruvaru neophodan za razvoj cjelokupnog turizma kao i druge, prethodno spomenute mogućnosti koje nudi sam Grad.

3. OPERACIJSKI SUSTAV *ANDROID*

Računalo je spoj *hardwarea* i *softwarea*, gdje *hardware* predstavlja komponente koje čine računalo uključujući zaslon, tipkovnicu, miš, pisač i ostale komponente, dok je *software* skup računalnih programa uz operacijski sustav. Operacijski sustav se pokreće na računalu te je odgovoran za upravljanje programa i računalnih resursa. Za računala, najpopularniji operacijski sustav je *Windows*, dok je za mobilne uređaje *Android* (C#Corner, 2019.). Na sljedećoj slici (Slika 2) prikazan je dijagram zastupljenosti mobilnih operacijskih sustava u razdoblju od srpnja 2018. do srpnja 2019. godine.



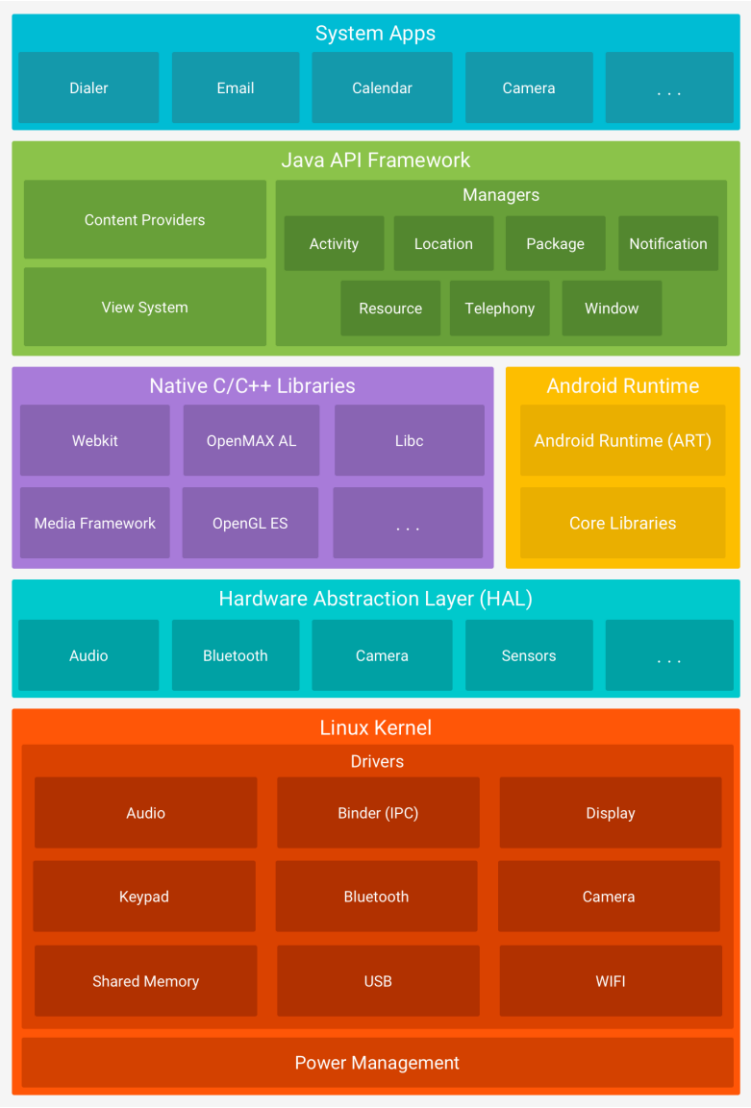
Slika 2. Zastupljenost mobilnih operacijskih sustava

Izvor: *Statcounter*, URL: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201807-201907-bar (9.8.2019)

Na slici je vidljiva izrazita dominantnost *Android* operacijskog sustava (više od 75%) u odnosu na druge. Uz *Android*, ističe se još i *iOS* sustav (više od 22%) namijenjen *Apple* uređajima.

Android je osnovao *Android Inc.* u Kaliforniji, SAD, 2003. godine. Prvenstvena namjera tvrtke bila je razviti napredni sustav za digitalne kamere, no, u to vrijeme, tržište za takve uređaje nije bilo dovoljno veliko što je promijenilo ciljeve tvrtke u razvoj operacijskog sustava za mobilne uređaje. *Google* je 2005. godine kupio *Apple Inc.* te dvije godine nakon toga predstavio prvu distribuciju sustava (Ipwatchdog, 2014).

Android je *stack software* otvorenog koda (eng. *open source*), baziran na *Linux Kernel* operacijskom sustavu. *Android* arhitektura se sastoji od slojeva koji su prikazani na sljedećoj slici (Slika 3).



Slika 3. *Android* arhitektura

Izvor: *Android*, URL: https://developer.android.com/guide/platform (9.8.2019)

Slika prikazuje *Android* arhitekturu sastavljenu od šest slojeva. Donji sloj arhitekture je *Linux Kernel* koji je temelj *Androida* te služi za osnovne funkcionalnosti kao što je upravljanje dretvama (eng. *threading*). Iznad se nalazi *Hardware Abstraction Layer* (*HAL*) koji sadrži razne biblioteke (eng. *libraries*) za npr. kameru ili *bluetooth*. Primjer korištenja *HAL* biblioteke je kada *API* okvir (eng. *framework*) zatraži pristup nekoj od *hardware* komponenti, tada *Android* sustav učitava biblioteku upravo za tu komponentu.

Sljedeći sloj je *Android Runtime* (*ART*). Za uređaje koji koriste *API* verziju 21 ili više, svaka aplikacija pokreće vlastiti proces i instancu *ARTA*. *ART* je implementiran tako da pokreće više virtualnih strojeva (eng. *virtual machines*) na uređajima s malom količinom memorije. Navedeno radi pokretanjem *DEX* datoteka (eng. *files*) koje su *byte* formata te je isti napravljen isključivo za *Android*. Ovaj sloj također sadrži i skup osnovnih biblioteka koje pružaju većinu funkcionalnosti *Java* programskog jezika. Biblioteke koje se nalaze u slojevima *ART* ili *HAL* zahtijevaju dodatne izvorne biblioteke napisane u jezicima *C* i *C++* koje se nalaze u sloju *Native C*/*C++* biblioteke.

Sljedeći sloj je *Java API Framework* koji sadrži cijeli skup funkcionalnosti napisan u programskom jeziku *Java*. Dostupan je preko *API*-ja, te je potreban za razvoj aplikacija. Sadrži, primjerice, sustav za razvoj korisničkog grafičkog sučelja (eng. *graphic user interface*), *Activity Manager* koji upravlja ciklusom aplikacije i mnoge druge mogućnosti.

Zadnji sloj u *Android* arhitekturi je *System Apps* koji sadrži osnovne aplikacije za npr. slanje SMS poruka, kalendar, internet preglednik, kontakte itd. Osnovne aplikacije korisnik ne mora nužno koristiti, već je slobodan instalirati one aplikacije koje mu odgovaraju i postaviti ih kao zadane (Android, 2019).

Operacijski sustav *Android* je objavio mnogo verzija sustava, a svakoj novoj su implementirana poboljšanja u odnosu na prethodne sustave. Slika 4 prikazuje listu sustava od *API* verzije 10 do trenutne 28 verzije.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Slika 4. *Android* verzije

Izvor: *Android*, URL: https://developer.android.com/about/dashboards (20.8.2019)

Sa slike je vidljivo da uglavnom upotrebljavaju verzije od sustava *Lollipop API* 22 pa nadalje, a kao vodeći je *Marshmallow*. Posljednji sustav nazvan je *Pie* te se prema prethodnim nazivima može zaključiti da će naziv sljedećeg sustava započeti sa slovom *Q*.

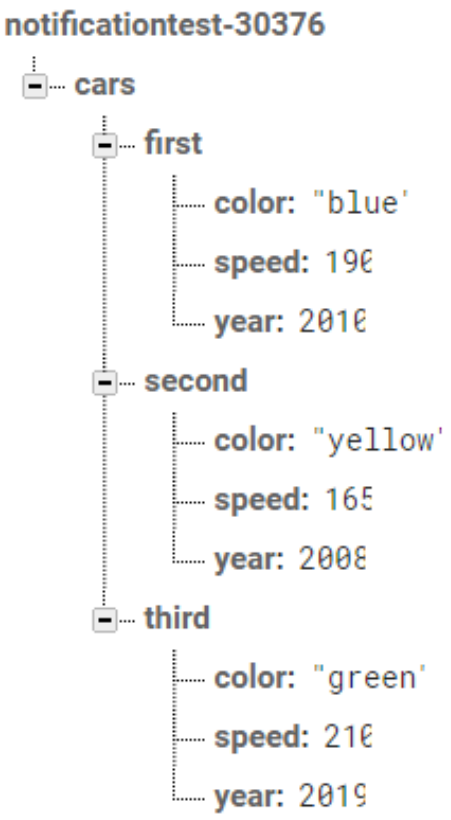
4. KORIŠTENE TEHNOLOGIJE

Smart Daruvar aplikacija se sastoji od mobilne i web aplikacije. Mobilna aplikacija je izrađena u razvojnom okruženju *Android Studio* korištenjem programskog jezika *Java* te *XML*. Također, pri izradi mobilne aplikacije, korišteni su *Google API's* (*Google Maps*, *Google Places* i *Google Directions*). Web aplikacija je razvijena u *JetBrains PhpStorm* okruženju korištenjem više programskih jezika: *JavaScript*, *HTML*, *CSS* i *PHP*. Obje aplikacije, kao bazu podataka (eng. *database*), koriste *Firebase Realtime Database*, odnosno *Firebase Storaga* za pohranu multimedije. Svaka od korištenih tehnologija bit će opisana u sljedećim potpoglavljima.

4.1. *FIREBASE* I *GOOGLE API's*

Firebase Realtime Database je baza podataka koja se nalazi u oblaku, a podaci u bazi su u *JSON* formatu te su sinkronizirani u realnom vremenu. Navedeno znači da svaku promjenu u bazi vide svi korisnici u isto vrijeme, odnosno, podaci se ažuriraju u nekoliko milisekundi na svim uređajima povezanim s bazom podataka. Baza podataka je dostupna izravno s mobilnog uređaja ili web stranice te ne zahtijeva server za pristup istoj. Što se tiče sigurnosti, korisnicima je dostupan fleksibilan jezik nazvan *Firebase Realtime Database Security Rules* kojim je moguće definirati strukturu podataka te pristup čitanju i pisanju istih. *Realtime Database API* je dizajniran tako da dopušta korištenje samo onih operacija koje se mogu izvršiti u kratkom vremenu, što omogućuje izgradnju sustava s višemilijunskim brojem korisnika (Firebase, 2019).

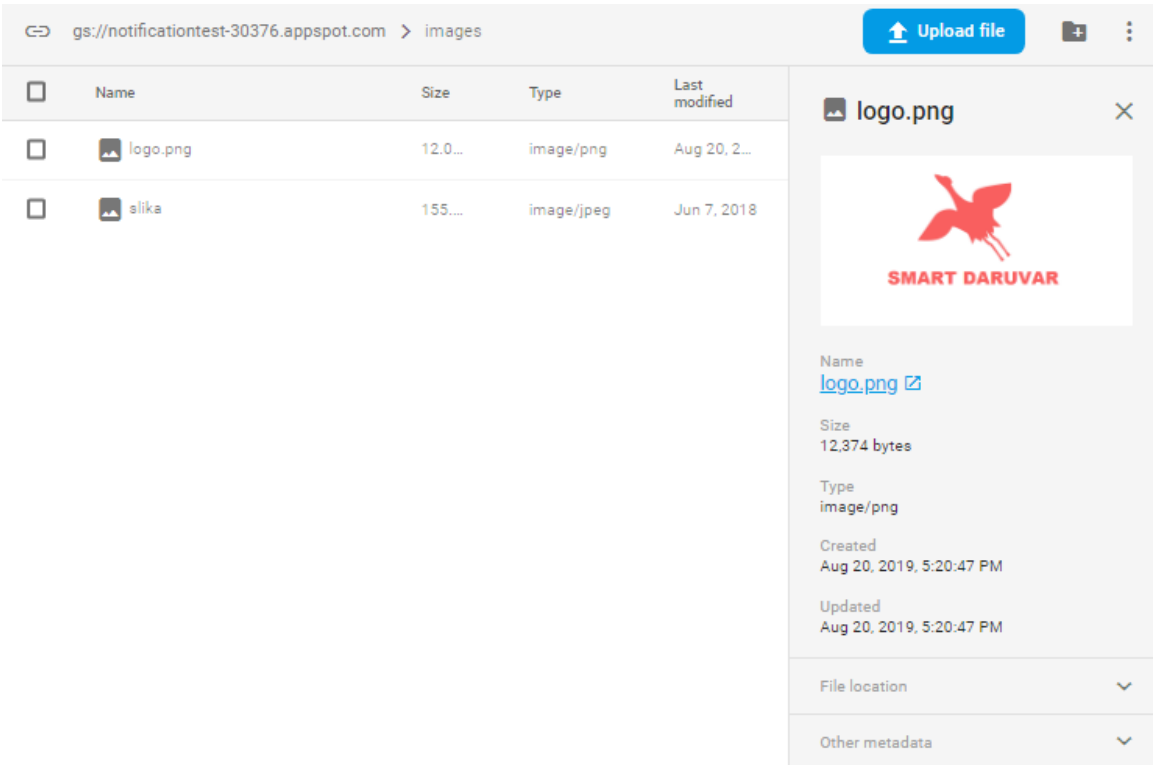
JSON format kojeg koristi i *Firebase Realtime Database* je, prema Tutorials Point (2017:5-9), format za razmjenu podataka punog naziva *JavaScript Object Notation*. Dizajniran tako da bude čitljiv čovjeku, proširenje je programskog jezika *JavaScript*. Koristi se za prijenos strukturiranih podataka internetom, prijenos podataka između servera i web aplikacija te u ostale svrhe. Karakteristično za *JSON* format je laka čitljivost i pisanje te neovisnost o drugim jezicima. Podržava razne tipove podataka (brojevne, tekstualne, logičke, polja, objekte - kolekcije parova oblika ključ:vrijednost itd.). Primjer *JSON* baze podataka je prikazan na sljedećoj slici (Slika 5).



Slika 5. Primjer *Firebase Realtime* baze podataka

Na slici se nalazi jednostavni primjer baze podataka naziva *notificationtest-30376*, izrađene u *Firebase Realtime Database*. Prema (Firebase, 2019), podaci u bazi su pohranjeni kao *JSON* objekti i, kako sadrže više razina na kojima se također nalaze objekti, cijela struktura podsjeća na stablo. Za razliku od uobičajene *SQL* baze podataka, ovakva baza ne sadrži tablice ili zapise, već svaki sljedeći pohranjeni podatak postaje novi čvor u stablu.

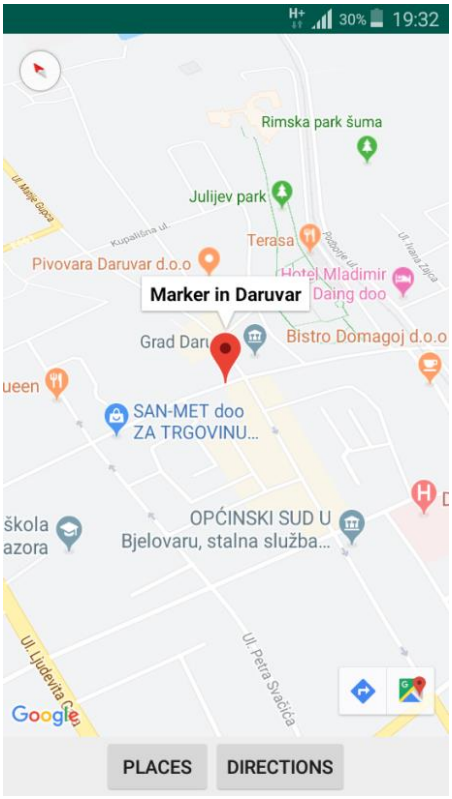
Firebase Storage je servis za pohranu i preuzimanje sadržaja poput slika i videa. Radi tako da korisnici uz pomoć *Firebase SDK* pohranjuju i preuzimaju sadržaj izravno s klijentske strane. U slučaju nedostupnosti internetske veze prilikom izvođenja neke od operacija, korisnik naknadno može nastaviti gdje je proces zastao (Ibid.). Sljedeća slika (Slika 6) prikazuje *Firebase Storage* sučelje.



Slika 6. Primjer *Firebase Storage* sučelja

U *Storageu* se nalazi mapa naziva *images* koja sadrži dvije slike: *logo.png* i *slika*. Podaci o slici se nalaze s desne strane sučelja te prikazuju veličinu slike, datum kreiranja i datum pohrane.

Google Maps SDK je razvojni alat kojim je moguće dodati *Google* kartu u aplikaciju. *API* automatski pruža pristup *Google Maps* serverima, preuzimanju podataka i prikazu karte. Moguće je, također, dodati na kartu i markere (oznake na određenim mjestima), poligone (ograđene segmente), *polyline* (skup linija), mijenjati poglede na određeno područje karte itd. Korištenje *Google* karte za *Android* moguće je samo uz *API* ključ koji je jedinstveni identifikator te služi za validaciju zahtjeva povezanog s korisnikovim projektom u svrhu pružanja usluge i naplaćivanja (Google, 2019). Slika 7 prikazuje *Google* kartu u *Android* aplikaciji.

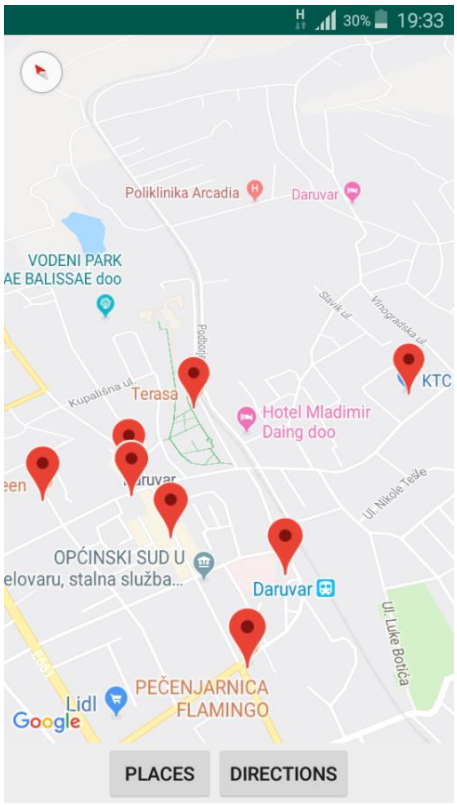


Slika 7. Primjer *Google* karte

Na slici se nalazi primjer korisničkog sučelja koji prikazuje *Google* kartu te dva gumba (eng. *button*) ispod karte. Karta sadrži marker (crvena oznaka) koji služi za označavanje lokacija te, dodirom na marker, pojavljuju se njegov naziv i dodatne informacije ako postoje.

Google Places omogućuje izradu aplikacija koje prepoznaju trenutnu lokaciju korisnika, kao i mjesta u blizini. Mjesto (eng. *place*) je definirano kao fizički prostor koji ima ime, odnosno sve ono što se može pronaći na karti. *Places API* pruža pristup *Google* bazi podataka koja sadrži informacije o lokalnim mjestima i poslovnim informacijama. Jedna od mogućnosti je pretraživanje mjesta na temelju korisničkog pretraživanja, odnosno *Autocomplete*. Rezultat pretraživanja su mjesta koja sadrže informacije, kao npr. ime, adresa, geografska lokacija, *ID*, broj telefona, vrsta mjesta, web stranica itd. Uz *Autocomplete*, druge mogućnosti su *Place Details* (vraća detaljnije informacije o mjestu), *Place Photos* (vraća visokokvalitetne slike mjesta) itd. Pri izradi aplikacije *Smart Daruvar* korištena je opcija *Nearby Search Request*. *Nearby Search Request*

pruža mogućnost pretrage mjesta unutar određenog područja. Zahtjev za mjesta je oblika *URL*, a odgovor je u *JSON* ili *XML* formatu. *URL* zahtjev mora sadržavati *API* ključ, geografsku lokaciju i radijus, dok su opcionalni jezik, minimalna ili maksimalna cijena mjesta, radno vrijeme, ocjene, tip mjesta itd. (Ibid.). Sljedeća slika (Slika 8) prikazuje *Google* kartu s mjestima koja su rezultat *Nearby Search Request* pretrage.

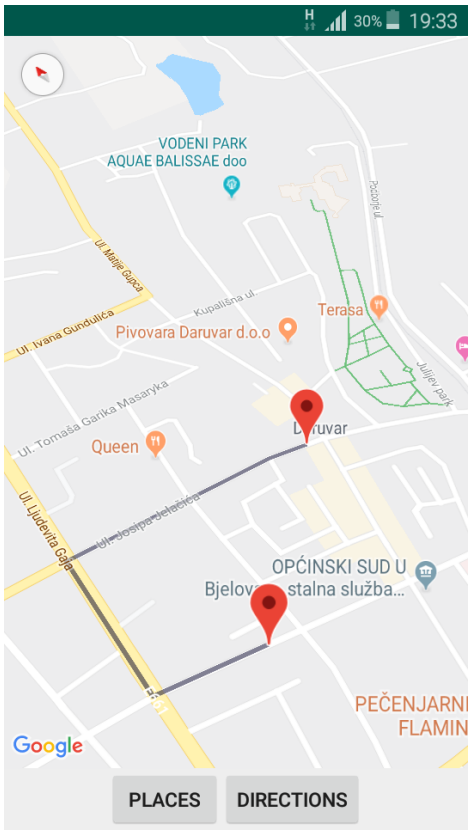


Slika 8. Primjer *Google* mjesta

Slika prikazuje korisničko sučelje kao i na prethodnoj slici (Slika 7), no na ovom prikazu karte, nalaze se markeri koji su rezultat pretrage mjesta u blizini u odnosu na proslijeđenu geografsku lokaciju zahtjevu.

Google Directions je servis koji izračunava upute između lokacije koristeći *HTTP* zahtjev (eng. *request*). Servis pruža upute u nekoliko različitih modova (hodanje, vožnja vozilom ili biciklom), a, također, pruža više mogućih ruta do istog mjesta. Rezultat zahtjeva su najučinkovitije rute u *JSON* ili *XML* formatu, a pri izračunu, primarni faktor je vrijeme putovanja. *HTTP* zahtjev mora sadržavati geografsku lokaciju početnog mjesta i krajnjeg mjesta te *API* ključ. Neobavezni parametri su mod (zadan je vožnja), putne

točke (lokacije koje su obavezne pri izračunu rute), izbjegavanje (vrste cesta koje ne ulaze u rutu) itd. (Ibid.). Na sljedećoj slici (Slika 9) je primjer rute od točke A do točke B.



Slika 9. Primjer *Google* uputa

Na karti se nalaze dva markera, jedan predstavlja početnu točku, a drugi krajnju točku rute. Rezultat upita je u *JSON* formatu te je iskorišten za iscrtavanje *Polyline* upute (crna linija koja povezuju točke).

4.2. *ANDROID STUDIO*

Android Studio je službeno integrirano razvojno okruženje za razvoj *Android* aplikacija, a temeljeno na *IntelliJ IDEA* razvojnom okruženju. Osim uređivača koda (eng. *code editor*) i alata za razvoj (eng. *development tools*) koje nudi *IntelliJ*, *Android Studio* pruža više mogućnosti pri razvoju. Neke od njih su:

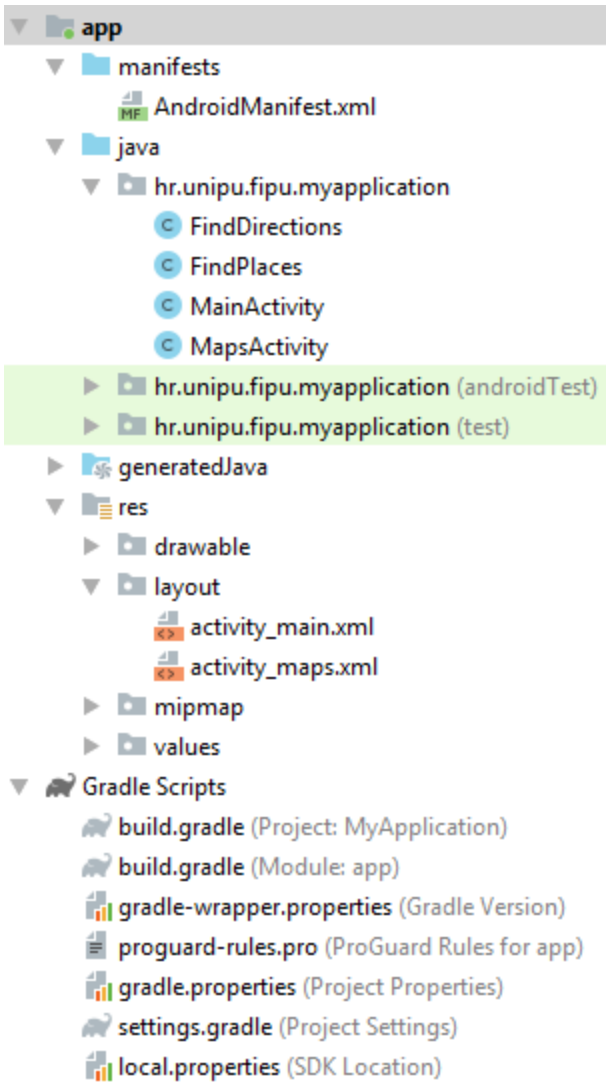
- fleksibilan *Gradle* sustav
- brz emulator

- mogućnost razvoja za sve *Android* uređaje
- predlošci koda te integracija s *GitHubom* itd.

Struktura projekta u *Android Studiou* se sastoji od jednog ili više modula koji sadrže datoteke izvornog koda (eng. *source code*) i datoteke resursa. Datoteke projekta se nalaze u *Android project view*-u kako bi pretraživanje datoteka bilo olakšano. Iste su podijeljene u mape:

- manifest - sadrži *AndroidManifest.xml* datoteku koja opisuje osnovne informacije o aplikaciji
- java - sadrži *Java* izvorni kod
- res - sadrži sve datoteke koje ne uključuju programski kod, npr. *XML* datoteke, bitmap slike itd. (Android Developers, 2019)

Na sljedećoj slici (Slika 10) prikazana je struktura projekta *My Application* (projekt za demonstraciju izrade mobilne aplikacije). Mapa *app* odgovara jednom modulu koji sadrži datoteke izvornog koda unutar mape *java* i *generatedJava* te datoteke resursa unutar mape *res*. Projekt *My Application* sadrži četiri *Java* klase, a datoteke resursa su podijeljene u mape *drawable* (grafičke datoteke), *layout* (datoteke za prikaz grafičkog sučelja), *mipmap* (ikone za pokretanje aplikacije) i *values* (datoteke za definiranje boja, naziva, stilova itd.). *Gradle Scripts* sadrži datoteke potrebne za izgradnju sustava. Prema Androidu (2019), *Gradle* sustav je jedan od integriranih alata *Android Studia* koji koristi sve datoteke projekta za izgradnju *APK* datoteke koja služi za testiranje ili distribuciju sustava. Također, *Gradle* je zadužen za preuzimanje i konfiguriranje biblioteka s ostalih repozitorija korištenih u projektu.



Slika 10. Primjer strukture projekta u *Android Studiou*

Primjer izrade jednostavne mobilne *Android* aplikacije bit će prikazan na sljedećim slikama. Aplikacija sadrži jedan *TextView* koji prikazuje, odnosno skriva tekst dodirom na jedan od dva gumba. Također, sadrži i gumb za otvaranje nove aktivnosti za prikaz karte, mjesta i uputa kao što je prikazano na slikama 7, 8 i 9. Slika 11 prikazuje izradu korisničkog sučelja u jeziku *XML*.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnShow"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Show text" />

        <Button
            android:id="@+id/btnClear"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Clear text" />

    </LinearLayout>

        <Button
            android:id="@+id/btnMap"
            android:layout_marginTop="50dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Map" />

</LinearLayout>
```

Slika 11. Primjer *XML* datoteke

Korisničko sučelje se sastoji od glavnog *LinearLayout*-a čiji su atributi navedeni u tijelu tog elementa. Isti sadrži i druge elemente: *TextView*, *LinearLayout* unutar kojeg se nalaze dva gumba te još jedan gumb izvan *LinearLayout*-a. *TextView* i gumbi sadrže *ID* koji služi za povezivanje grafičkog elementa s programskim kodom aplikacije. Tekst gumba nije definiran unutar *xml* datoteke, nego se isti nalazi u zasebnoj datoteci *strings.xml* koja je prikazana na sljedećoj slici (Slika 12).

```xml
<resources>
    <string name="app_name">My Application</string>

    <string name="btnShowText">Show text</string>
    <string name="btnClearText">Clear text</string>
    <string name="btnMap">Map</string>
    <string name="btnPlaces">Places</string>
    <string name="btnDirections">Directions</string>

    <string name="txtShow">Hello World!</string>
    <string name="title_activity_maps">Map</string>

</resources>
```

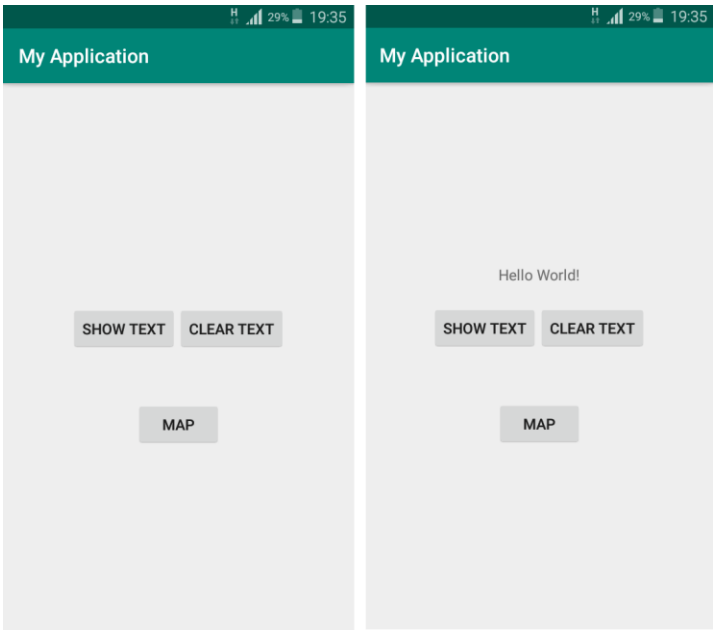
Slika 12. Primjer *strings.xml* datoteke

Datoteka *strings.xml* sadrži osam *string*-ova za definiranje teksta aplikacije. Svaki *string* ima ime i vrijednost. Ime mora biti jedinstveno jer se prema imenu dodjeljuje tekst određenom elementu. Vrijednost je tekst koji će element poprimiti. Povezivanje korisničkog sučelja i programskog koda se nalazi na 13. slici.

```java
public class MainActivity extends AppCompatActivity {

    private TextView mText;
    private Button mBtnShowText;
    private Button mBtnClearText;
    private Button mBtnMap;

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

        mText = (TextView) findViewById(R.id.txt);
        mBtnShowText = (Button) findViewById(R.id.btnShow);
        mBtnClearText = (Button) findViewById(R.id.btnClear);
        mBtnMap = (Button) findViewById(R.id.btnMap);

        mBtnShowText.setOnClickListener((view) -> {
                mText.setText("Hello World!");
        });

        mBtnClearText.setOnClickListener((view) -> { mText.setText(""); });

        mBtnMap.setOnClickListener((view) -> {
                Intent intent = new Intent( packageContext: MainActivity.this, MapsActivity.class);
                startActivity(intent);
        });

    }
}
```

Slika 13. Primjer *MainActivity* datoteke

Aplikacija se sastoji od klase (eng. *class*) *MainActivity* koju proširuje *AppCompatActivity* (vrsta aktivnosti koja dopušta upravljanje navigacijskom trakom). Klasa se sastoji od četiri atributa (*mText*, *mBtnShowText*, *mBtnClearText i mBtnMap*) te metode *onCreate* koja povezuje *activity_main* sučelje s klasom *MainActivity*. Također, povezuje navedene atribute s grafičkim elementima unutar korisničkog sučelja. Unutar metode, nalaze se i tri slušača na događaje (eng. *event listener*) gumb atributa. Na dodir atributa *mBtnShowText*, *mText* prikazuje tekst *Hello World!*, dok na dodir *mBtnClearText*, *mText* ne prikazuje ništa. Dodir na *mBtnMap* poziva se *Intent*, klasa koja otvara novu aktivnost. Na slici 14 je prikazan izgled aplikacije.



Slika 14. Primjer *Android* aplikacije

Lijevi prikaz aplikacije odgovara izgledu nakon dodira na gumb *show text*, desni prikaz aplikacije je nakon dodira gumba *clear text*. gumb *map* otvara novu aktivnost za prikaz karte.

4.3. *PHP STORM*

PhpStorm je razvojno okruženje namijenjeno razvoju *PHP* aplikacija. *PhpStorm* pruža prevenciju grešaka tijekom pisanja koda, automatsko dopunjavanje koda (eng. *autocomplete*) itd. Također, dozvoljava razvoj aplikacija pomoću *HTML*, *CSS* i

JavaScript programskih jezika. Bogat i inteligentan uređivač koda ističe sintaksu, provjerava greške prilikom pisanja koda te dovršava ključne riječi koda (ComponentSource, 2019). Primjer izrade jednostavne web aplikacije u *PhpStorm* razvojnom okruženju prikazan je na slici 15.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Example</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
</head>

<style>
    #title{
        font-size: 30px;
        margin-bottom: 20px;
        margin-top: 20px;
        text-decoration: none;
    }

    #frame{
        text-align: center;
    }

    #data{
        margin-top: 20px;
    }
</style>

<body>
    <div class="container-fluid padding">
        <div class="row welcome text-center">
            <div class="col-12">
                <div id="title">Example</div>
            </div>
        </div>
    </div>

    <div id="frame">
        <button type="button" class="btn btn-primary" onclick="showData()">Click me</button>
        <div id="data"></div>
    </div>
</body>

</html>

<script>
    function showData() {
        var div = document.getElementById("data");
        div.innerHTML = "This function will print 'Hello World!' 5 times.<br><br>";

        <?php
        for($i = 0; $i<5; $i++){?>
        div.innerHTML += "Hello World!<br>";
        <?php
        }?>
    }
</script>
```

Slika 15. Primjer izrade web aplikacije

Cijeli kod aplikacije se nalazi u datoteci .php ekstenzije koja sadrži implementaciju više programskih jezika. Na početku datoteke se nalaze oznake (eng. *tags*) tipične pri izradi *HTML* web stranice. *Head* element sadrži podatke o podacima te se isti ne prikazuju na web stranici. Element *meta* definira skup znakova koji se koristi i naziv dokumenta te sadrži poveznicu na eksternu *BootstrapCDN* biblioteku za razvoj web aplikacija. Zatim slijedi definiranje stila unutar *style* elementa.

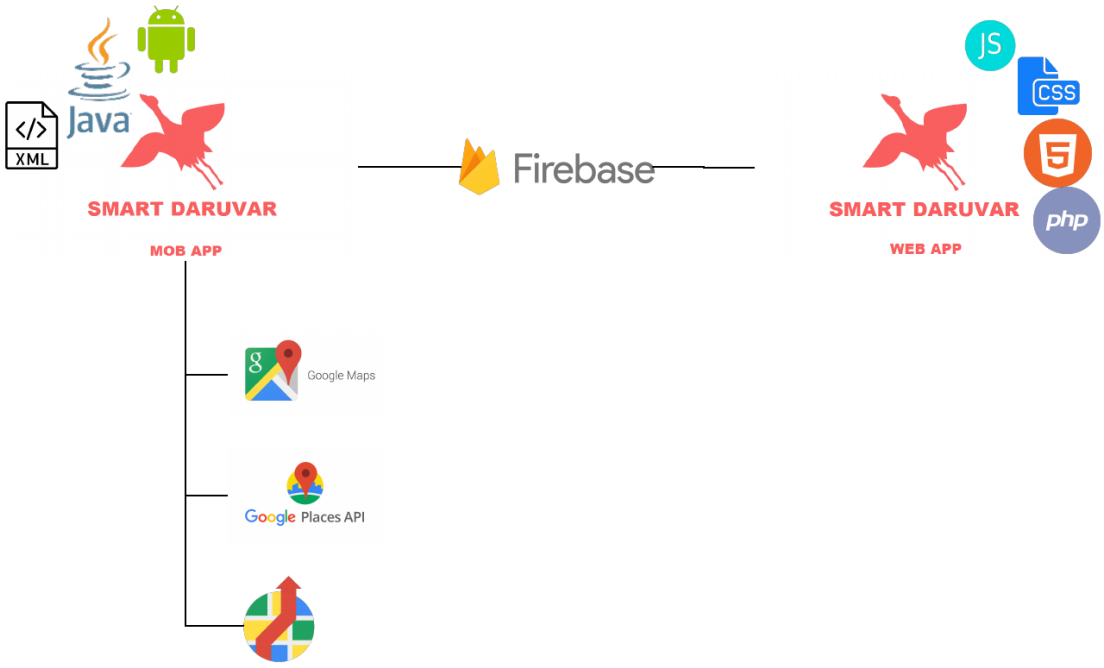
HTML elementu se definirani stil pridružuje putem njegove klase ili vrijednosti *ID*-a. Kod unutar elementa *style* odgovara programskom jeziku *CSS*. Unutar elementa *body* nalaze se elementi koji će biti prikazani na stranici. Prvi od elemenata koji se nalazi unutar elementa *body* je element *div* kojemu je dodijeljena klasa *container-fluid padding* preuzeta iz biblioteke *BootstrapCDN*. Unutar njega se nalaze još tri elementa *div*, a posljednjem je dodijeljen *ID title* te sadrži naziv, odnosno vrijednost koja će biti ispisana na web stranici.

Zatim, definirana su još dva *div*-a te jedan gumb koji je povezan s *JavaScript* funkcijom *showData*. Ista se poziva klikom na gumb. Funkcija je definirana unutar oznaka *script* te se sastoji od ključne riječi *function*, naziva funkcije, oblih zagrada unutar koji se mogu definirati atributi te vitičastih zagrada unutar koji se nalazi tijelo funkcije. Funkcija *showData* definira varijablu *div* koja je povezana s *HTML* elementom *div* preko njegovog *ID*-a. Pozivanjem svojstva *innerHTML* s varijablom postavlja se tekst *HTML* elementu. Zatim, unutar oznaka *<?php* i *?>* nalazi se *PHP* kod koji definira *for* petlju. Petlja se pokreće pet puta te svaki put doda novi tekst *HTML* elementu. Primjer demonstrira izradu jednostavne web stranice upotrebnom više programskih jezika unutar jedne datoteke. Slika 16 prikazuje izgled opisane web stranice.

Slika 16. Primjer web aplikacije

5. IMPLEMENTACIJA *SMART DARUVAR* APLIKACIJE

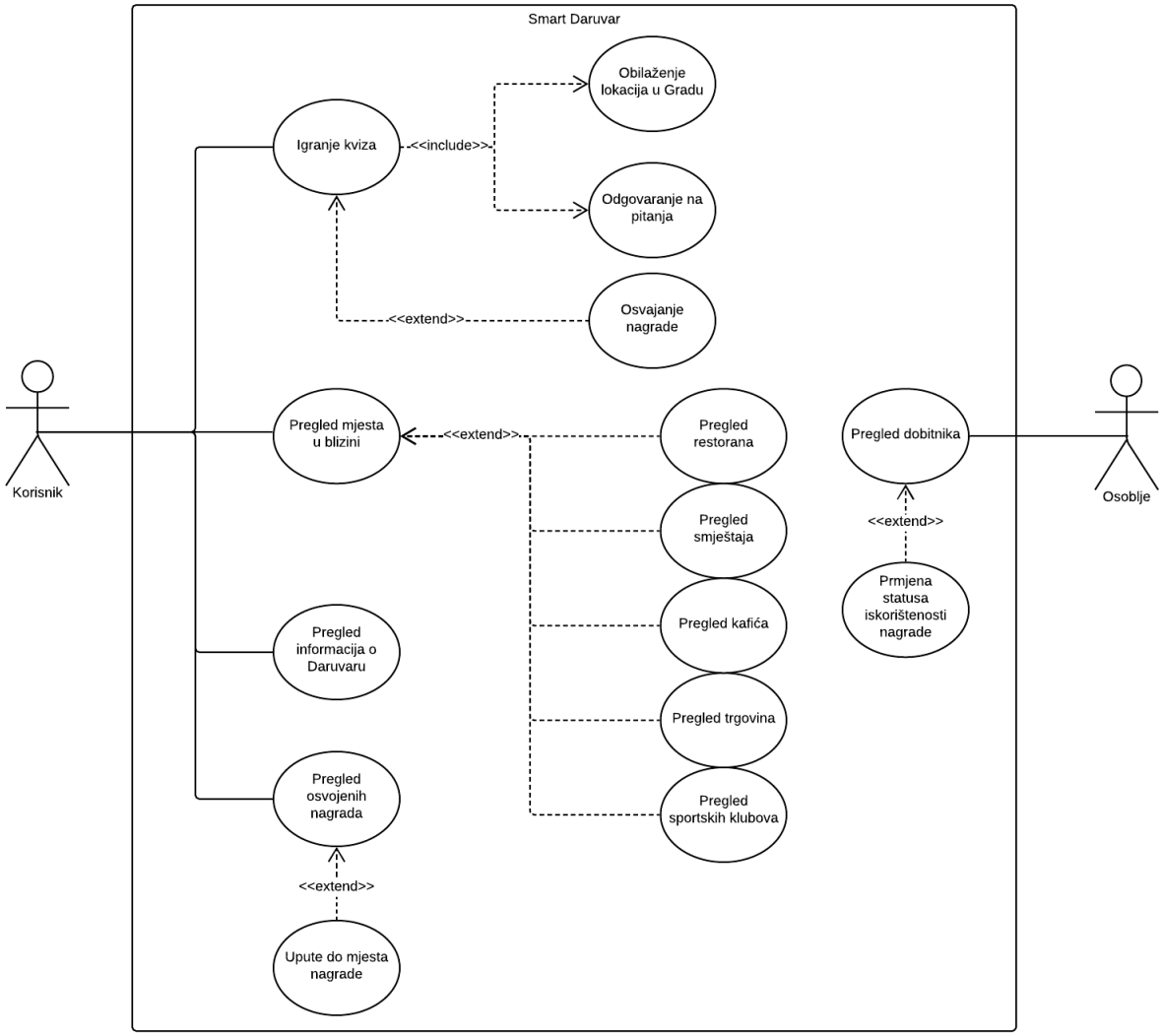
Smart Daruvar aplikacija implementirana je pomoću prethodno navedenih tehnologija. Spoj istih rezultira sustavom za promociju turizma koji spaja web i mobilne tehnologije. Poveznica između tih dviju tehnologija je *Firebase API* koji, u ovom slučaju, služi za pohranu podataka u bazu. Prikaz spomenutih tehnologija se nalazi na sljedećoj slici (Slika 17).



Slika 17. Grafički prikaz korištenih tehnologija

Prikaz *UML use case* dijagrama cijelog sustava koji obuhvaća i mobilnu i web aplikaciju se nalazi na sljedećoj slici (Slika 18). Granice sustava su označene pravokutnikom izvan kojega se nalaze dva sudionika: korisnik (predstavlja osobu koja koristi mobilnu aplikaciju) i osoblje (osoba koji koristi web aplikaciju, djelatnik poslovnog objekta u kojemu je moguće iskoristiti nagradu). Korisnik ima nekoliko slučajeva korištenja: igranje kviza (uključuje obilaženje lokacija u gradu i odgovaranje na pitanja te isti proširuje osvajanjem nagrade), pregled mjesta u blizini (moguć pregled restorana, smještaja, kafića, trgovina i sportskih sadržaja), pregled informacija o Daruvaru i pregled osvojenih nagrada (slučaj korištenja proširen uputama do mjesta korištenja

nagrade). Osoblje ima mogućnost pregleda dobitnika te ima opciju promjene stanja iskorištenosti nagrade.



Slika 18. *Use case* dijagram

5.1. BAZA PODATAKA

Korištena *Firebase* baza podataka je izrađena u obliku *JSON* objekata gdje svaki objekt sadrži podatke za određenu funkcionalnost mobilne ili web aplikacije. *JSON* objekti u bazi su: *accommodations* i *restaurants* (podaci o hotelima i restoranima u kojima je moguće ostvariti popust igranjem kviza), *daruvar_images* (poveznice na fotografije u *Firebase Storageu*), *daruvar_text_en* i *daruvar_text_hr* (podaci o Daruvaru

na hrvatskom i engleskom jeziku), *dvorac_en*, *dvorac_hr, gola_maja_en*, *gola_maja_hr*, *julisbrum_en*, *julisbrum_hr*, *zdral_en* i *zdral_hr* (pitanja i odgovori o lokaciji koju je potrebno posjetiti igrajući kviz), *hotel_1*, *hotel_2*, *hotel_3*, *restoran_1*, *restoran_2*, *restoran_3* (podaci o korisnicima koji su ostvarili nagradu u tom objektu), *locations* (podaci o lokacijama koje je potrebno posjetiti), *sports* (podaci o sportskim klubovima u Daruvaru) i *users* (korisnička imena i lozinke za prijavu u web aplikaciju). Prikaz baze podataka se nalazi na sljedećoj slici (Slika 19).

Slika 19. Baza podataka

Fotografije Daruvara i ikone restorana i hotela se nalaze u *Firebase Storageu*. Poveznice tih slika se nalaze u *JSON* objektima baze podataka te se iste preuzimaju u trenutku kada je potrebno.

5.2. MOBILNI DIO APLIKACIJE

Mobilna aplikacija se sastoji od tri aktivnosti, dva *InfoWindow* adaptera, jednog *PageAdaptera*, jednog *RecycleView* adaptera, dva *AsyncTaska* te 11 fragmenata, dakle, ukupno 20 klasa. Otvaranjem aplikacije, poziva se klasa *SplashScreenActivity* koja priprema podatke za prikaz glavne aktivnosti. Prikaz klase se nalazi na sljedećoj slici (Slika 20).

Slika 20. Mobilna aplikacija - *splash screen*

SplashScreen prikazuje logo aplikacije te *progress bar* na dnu ekrana. Tijekom čekanja na otvaranje glavne aktivnosti, aplikacija provjerava vezu s internetom. Ako je veza uspostavljena, proziva se metoda za preuzimanje poveznica slika s *Firebase* baze podataka, poveznice slika se lokalno pohranjuju u *JSON* datoteku te se otvara glavna aktivnost. Ako internetska veza nije uspostavljena, aplikacija provjerava da li postoji lokalna *JSON* datoteka. U slučaju da postoji, poveznice slika iz *JSON* datoteke se prosljeđuju glavnoj aktivnosti, u suprotnom, otvaranje aplikacije nije moguće. Dakle, prilikom prvog pokretanja aplikacije potrebna je internetska veza, dok pri sljedećim pokretanjima ista nije nužna. Dio programskog koda klase *SplashScreenActivity* se nalazi na sljedećoj slici (Slika 21).

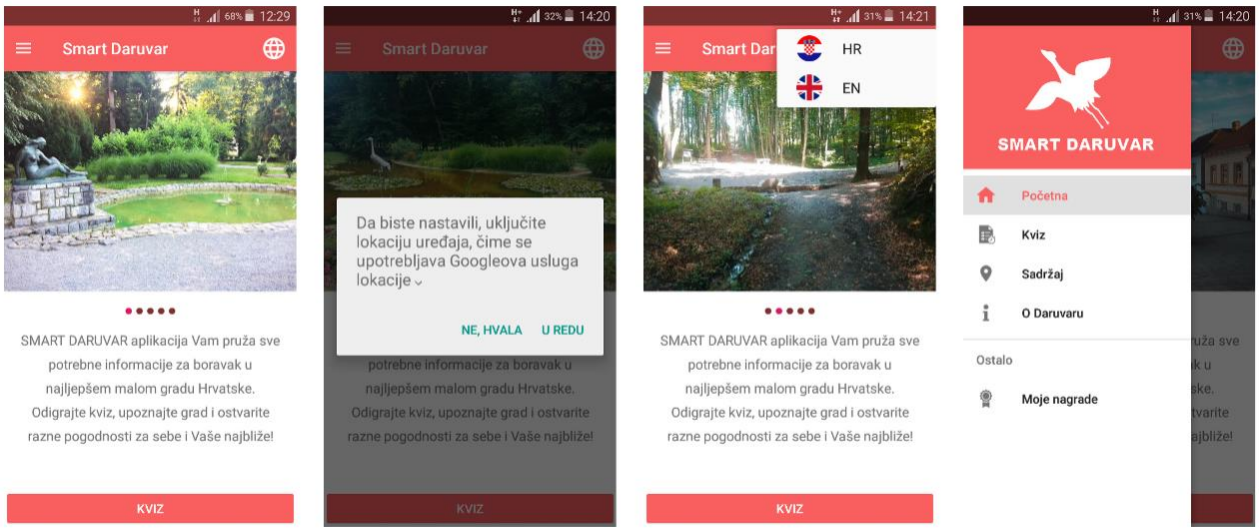
```
if (!connected){
    JSONArray jsonArray;
    jsonArray = getSavedJSONArrayPrizes();
    if (jsonArray != null) {
        for (int i = 0; i < jsonArray.length(); i++) {
            try {
                images.add(i, jsonArray.getJSONObject(i).getString( name: "img" + i));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                progressBar.setVisibility(View.INVISIBLE);
                Intent intent = new Intent( packageContext: SplashScreenActivity.this, MainActivity.class);
                startActivity(intent);
            }
        }, delayMillis: 3000);

    } else {
        Toast.makeText( context: this, "Prilikom prvog otvaranja aplikacije, potrebna je intern...", Toast.LENGTH_LONG).show();
    }
} else {
    mAuth = FirebaseAuth.getInstance();
    FirebaseUser user = mAuth.getCurrentUser();
    if (user != null){
        downloadImageUrls();
    } else {
        signInAnonymously();
    }
}
```

Slika 21. *Splash screen* - programski kod

Na slici se nalazi programski kod za prethodno opisani scenarij *SplashScreenActivity* klase za preuzimanje poveznica slika koje se prosljeđuju *MainActivity* klasi.

5.2.1. *POČETNA*

Glavna aktivnost je proširena s *NavigationView*, odnosno izbornikom koji sadrži poveznice na fragmente aplikacije. Nakon otvaranja glavne aktivnosti, aplikacija poziva metodu za izgradnju *Google API* klijenta, što znači da poziva neku od biblioteka unutar *Google Play Services*. Točnije, poziva biblioteku za dohvaćanje korisnikove lokacije. Stoga, nakon upućenog zahtjeva za korisnikovu lokaciju, otvara se prozor sa zahtjevnom za pristup lokaciji. Korisnik ima opcije potvrditi zahtjev ili odbiti isti. U slučaju da odbije zahtjev, aplikacija neće moći koristiti korisnikovu lokaciju, a korisnik neće moći koristiti sve funkcionalnosti koje aplikacija nudi. Glavna aktivnost sadrži grafičko sučelje *main_content.xml* koji služi kao okvir na koji se postavljaju fragmenti. Prikaz početne aktivnosti se nalazi na sljedećoj slici (Slika 22).



Slika 22. Mobilna aplikacija - početni ekrani

Prvi prozor s lijeva na desno prikazuje početni fragment koji sadrži *ViewPager* te on predstavlja *slideshow* galeriju slika. Ispod njega se nalazi *LinearLayout* koji predstavlja indikatore za aktivnu i neaktivne slike unutar *slideshowa*. Ispod njega prikazan je tekst unutar *TextViewa* te na dnu ekrana se nalazi gumb za pokretanje kviza. Sljedeći prozor je prikaz zahtjeva za pristup korisnikovoj lokaciji. Isti se pokreće na razini aktivnosti, a ne na fragmentu koji se nalazi na aktivnosti. Na trećem prozoru je otvoren izbornik s dvije opcije odabira jezika. Aplikacija uz hrvatski podržava i engleski jezik. Zadnji prozor sadrži prikaz glavnog izbornika koji služi kao navigacija unutar aplikacije. Sastoji se od loga na vrhu izbornika te pet *itema* za otvaranje ostalih fragmenata, odnosno aktivnosti. Trenutno prikazani fragment unutar glavne aktivnosti je na izborniku prikazan kao *item* sa sivom pozadinom. Primjer koda za promjenu fragmenta dodirom na jednu od ponuđenih opcija glavnog izbornika se nalazi na slici 23.

```java
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.nav_home) {
        HomeFragment homeFragment = new HomeFragment();
        Bundle bundle = new Bundle();
        bundle.putStringArrayList("images", (ArrayList<String>) images);
        homeFragment.setArguments(bundle);
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, homeFragment,  tag: "HomeFragment").commitAllowingStateLoss();
    } else if (id == R.id.nav_quiz) {
        QuizMainFragment quizMainFragment = new QuizMainFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, quizMainFragment,  tag: "QuizMainFragment").commitAllowingStateLoss();
    } else if (id == R.id.nav_content) {
        PlacesFragment placesFragment = new PlacesFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, placesFragment,  tag: "PlacesFragment").commitAllowingStateLoss();
    } else if (id == R.id.nav_about) {
        AboutFragment aboutFragment = new AboutFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, aboutFragment,  tag: "AboutFragment").commitAllowingStateLoss();
    } else if (id == R.id.nav_prize){
        MyPrizesFragment myPrizesFragment = new MyPrizesFragment();
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container, myPrizesFragment,  tag: "MyPrizeFragment").commitAllowingStateLoss();
    }

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

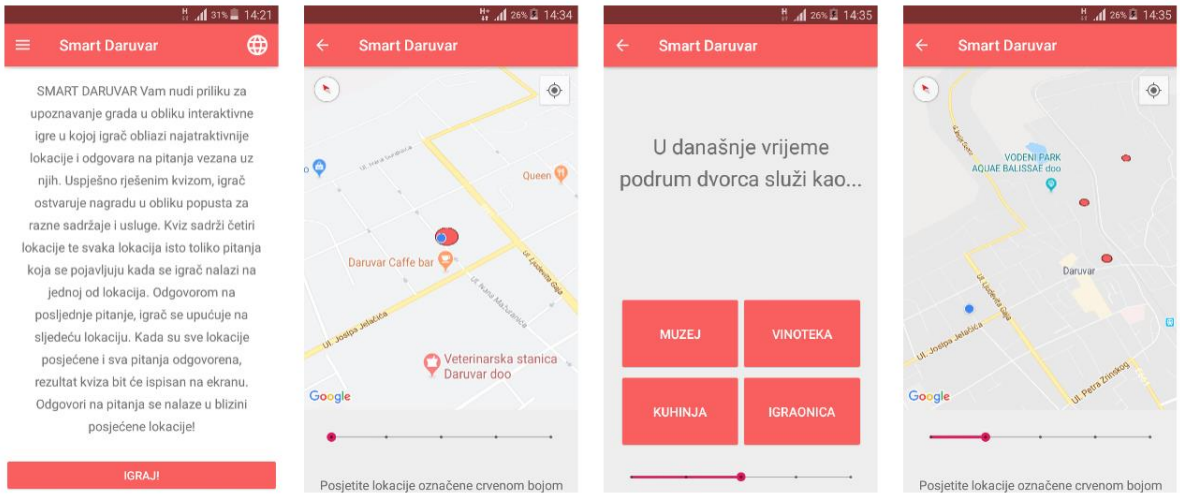
Slika 23. Promjena fragmenta - programski kod

Na slici se nalazi nadjačana metoda *onNavigationItemSelected* koja sadrži argument *item* (*item* predstavlja element glavnog izbornika). Metoda se poziva kada korisnik odabere jedan od ponuđenih elemenata u glavnom izborniku. Unutar tijela metode se nalazi varijabla *id* kojoj se dodjeljuje cjelobrojna jedinstvena vrijednost elementa, nakon čega se ta vrijednost uspoređuje s jedinstvenim vrijednostima elemenata izbornika. Nakon pronalaska odabranog elementa, instancira se novi objekt tog fragmenta te se u okvir glavne aktivnosti za prikaz fragmenta postavlja upravo onaj koji je odabran. Također, fragmentu se dodjeljuje oznaka zbog kasnije mogućnosti pronalaska istog.

5.2.2. *KVIZ*

Glavna funkcionalnost ove aplikacije je kviz. Korisnik aplikacije ima mogućnost igranja kviza te osvajanja nagrade u obliku popusta za neke usluge i sadržaje u Daruvaru. Igra je osmišljena tako da korisnik, odnosno, igrač obilazi najatraktivnije lokacije u Daruvaru te rješava kviz u obliku pitanja s četiri ponuđena odgovora. Pitanja su vezana uz posjećenu lokaciju, a odgovori se nalaze na istim mjestima. Igra sadrži četiri lokacije i četiri pitanja za svaku od lokacija. Odgovorom na posljednje pitanje na prvoj, drugoj ili trećoj lokaciji, korisnik obilazi preostale lokacije. Kada obiđe sve lokacije

i odgovori na sva pitanja, aplikacija izračunava postotak točno odgovorenih pitanja te, ako je rezultat veći ili jednak 70%, korisnik odabire nagradu. Odabir nagrade se odvija tako da korisnik, u trenutku kada on hoće, dodiruje gumb *Uzmi nagradu!* te se *slideshow* s ikonama nagrada zaustavlja i na ekranu se ispisuje nagrada koju je osvojio. Nakon toga, potrebno je ostaviti osobne podatke za dokazivanje uspješno riješenog kviza. Sljedeća slika (Slika 24) prikazuje kviz ekrane.



Slika 24. Mobilna aplikacija - kviz ekrani

Na prvom lijevom prozoru je prikazan fragment *kviz*. Sadrži *TextView* koji prikazuje upute za igranje kviza te gumb za pokretanje koji otvara novu aktivnosti *QuizActivity*. Na istoj se izmjenjuju fragmenti s pitanjima i kartom. Sljedeći ekran je fragment na kojemu se nalazi karta s lokacijama koje korisnik treba posjetiti u obliku krugova crvene boje. Ekran prikazuje korisnikovu lokaciju (plavi krug) unutar jedne od lokacija (crveni krug), što znači da je korisnik stigao na jednu od lokacija. Nakon toga se otvara prozor s pitanjima i odgovorima. Pitanje je smješteno unutar *TextViewa*, a odgovori unutar gumba. Trenutni broj odgovorenih pitanja može se vidjeti na *SeekBaru* ispod gumba za odgovore. Odgovorom na posljednje pitanje ponovno se otvara fragment s kartom na kojemu je ažuriran *SeekBar* s brojem posjećenih lokacija. Dio koda za provjeru je li korisnik stigao do jedne od lokacija prikazan je na slici 25.

```
float[] distance = new float[3];
for (int i = 0; i < circles.size(); i++) {
    Location.distanceBetween(location.getLatitude(), location.getLongitude(),
            circles.get(i).getCenter().latitude, circles.get(i).getCenter().longitude, distance);
    if (distance[0] < circleRadius) {
        numberOfVisitedLocations++;

        String locationName = circles.get(i).getTag().toString() + "_" + Locale.getDefault().getLanguage();
        SharedPreferences.Editor editor = preferences.edit();
        editor.putInt( s: "numberOfVisitedLocations", numberOfVisitedLocations);
        editor.putString( s: "location", locationName);
        editor.putString( s: "stopDownloadData",  s1: "true");
        editor.commit();
        // ...
```

Slika 25. Izračun udaljenost korisnika do lokacije - programski kod

Nakon svake promjene korisnikove lokacije, poziva se dio koda sa slike. *For* petlja služi za izračun udaljenost korisnikove lokacije i svih lokacija koje treba posjetiti. Udaljenost se sprema u polje *distance.* Ako je udaljenost manja nego što je promjer lokacije na karti koje treba posjetiti, dohvaća se oznaka posjećene lokacije i oznaka jezika na koji je postavljena aplikacija te se taj *string* sprema u *SharedPreferences* (lokalna memorija uređaja) i prosljeđuje fragmentu koji preuzima podatke te lokacije za prikaz pitanja i odgovora. Programski kod za preuzimanje kviz podataka prikazan je na sljedećoj slici (Slika 26).

```
private void downloadQuizData(){
    ref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(final DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot: dataSnapshot.getChildren()){
                for (DataSnapshot snapshot1: snapshot.getChildren()){
                    if (snapshot1.getKey().equals("tocno")){
                        correctAnswers.add(snapshot1.getValue().toString());
                        allAnswers.add(snapshot1.getValue().toString());
                    } else if (snapshot1.getKey().equals("pitanje_tekst")){
                        questions.add(snapshot1.getValue().toString());
                    } else {
                        allAnswers.add(snapshot1.getValue().toString());
                    }
                }
            }

            for (int i=3; i<allAnswers.size(); i+=4){
                int tmp = i-3;
                Collections.shuffle(allAnswers.subList(tmp,  i1: i+1));
            }
            nextQuestion();
        }
        //...
```

Slika 26. Preuzimanje podataka za kviz - programski kod

Na slici je prikazan dio koda metode *downloadQuizData*. *Ref* je objekt klase *DatabaseRefernce* kojemu je dodijeljena prethodno spomenuta vrijednost preuzeta iz *SharedPreferences*. Nad tim objektom se poziva slušač događaja te metoda *onDataChange*. Unutar te metode se u *for* petlji preuzimaju podaci iz baze podataka. Podaci su *JSON* objekti, oblika ključ:vrijednost. Za svaki čvor *JSON stabla* se uspoređuje njegov ključ. Ako je ključ *tocno*, u listu predviđenu za spremanje točnih odgovora se sprema vrijednost tog čvora, ako je ključ *pitanje_tekst*, vrijednost se dodaje u listu pitanja. Vrijednost ostalih čvorova kao i onih s ključem *tocno* se sprema u listu svih odgovora. Zatim, u sljedećoj *for* petlji se lista svih odgovora nasumično *promiješa* kako redoslijed odgovora na pitanja ne bi svaki puta bio isti te se poziva metoda *nextQuestion* čiji se programski kod nalazi na sljedećoj slici (Slika 27).

```
private void nextQuestion(){
    txtQuestion.setText(questions.get(questionCounter));
    questionCounter++;

    answerA.setText(allAnswers.get(answerCounter));
    answerCounter++;

    answerB.setText(allAnswers.get(answerCounter));
    answerCounter++;

    answerC.setText(allAnswers.get(answerCounter));
    answerCounter++;

    answerD.setText(allAnswers.get(answerCounter));
    answerCounter++;
}
```

Slika 27. Promjena pitanja kviza - programski kod

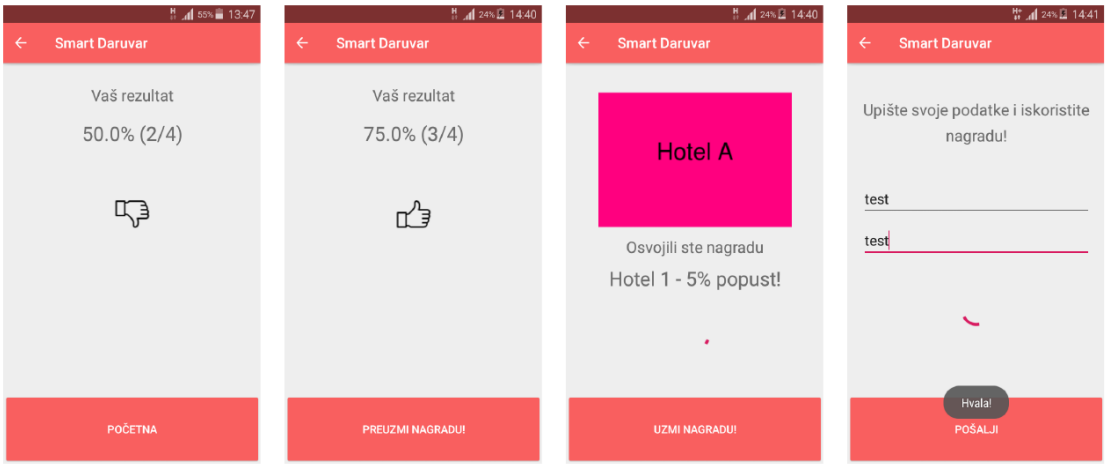
Metoda se poziva nakon preuzimanja podataka iz baze ili nakon odgovora na pitanje. Ista postavlja tekst pitanja iz liste svih pitanja u *TextView* i tekst odgovora gumbima korisničkog sučelja. Nakon svakog poziva metode, brojač *questionCounter* se poveća za jedan, a brojač *answerCounter* za četiri, kako se ne bi sljedećim pozivom metode prikazala ista pitanja i odgovori. Nakon dodira na jedan od odgovora poziva se programski kod sa slike 28.

```
answerA.setOnClickListener((view) → {
        seekUp();
        if (answerA.getText().toString().equals(correctAnswers.get(questionCounter-1))){
            answerA.startAnimation(AnimationUtils.loadAnimation(mContext, R.drawable.transform));
        } else {
            answerA.startAnimation(AnimationUtils.loadAnimation(mContext, R.drawable.shake));
        }

        userAnswers.add(answerA.getText().toString());
        int numberOfVisitedLocations = preferences.getInt( s: "numberOfVisitedLocations",  i: 0);

        if (numberOfVisitedLocations == numberOfLocations && questionCounter == 4){
            int numberOfCorrectAnswers = 0;
            for (int i=0; i<userAnswers.size(); i++){
                if (userAnswers.get(i).equals(correctAnswers.get(i))){
                    numberOfCorrectAnswers++;
                }
            }

            int numberOfPreviousCorrectAnswers = preferences.getInt( s: "numberOfCorrectAnswers",  i: 0);
            numberOfCorrectAnswers = numberOfCorrectAnswers + numberOfPreviousCorrectAnswers;
            final double result = (numberOfCorrectAnswers/(double)numberOfQuestions)*(double) 100;
            final int correctAnswers = numberOfCorrectAnswers;
            final int question = numberOfQuestions;
            // ...
```

Slika 28. Odgovor na pitanje kviza - programski kod

Nakon dodira na jedan od gumba za odgovor na pitanje poziva se metoda za ažuriranje stanja *SeekBara* te se provjerava je li odgovor točan. Ako je točan pokreće se animacija gumba za točan odgovor, suprotno tome, pokreće se animacija za netočan odgovor. U listu korisnikovih odgovora se dodaje odgovor koji je odabran. Zatim, iz lokalne memorije se dohvaća broj posjećenih lokacija te, ako je taj broj jednak ukupnom broju lokacija i broj odgovorenih pitanja na trenutnoj lokaciji iznosi četiri, izračunava se broj točnih odgovora te kviz završava. Izračunava se tako da se uspoređuju odgovori iz liste korisnikovih odgovora s onima iz liste točnih odgovora. Taj broj se dodaje broju točno odgovorenih pitanja s prethodnih lokacija te se isti dijeli s ukupnim brojem svih pitanja na svim lokacijama kako bi se dobio postotak uspješnosti rješavanja kviza. Kod koji se nalazi na slici definira scenarij ako su sve lokacije posjećene, odnosno ako su sva pitanja za trenutnu lokaciju odgovorena.

Nakon obilaska svih lokacija i odgovorom na posljednje pitanje, otvara se novi fragment čiji sadržaj ovisi o postignutom rezultatu. Fragmenti aplikacije nakon rješavanja kviza su prikazani sljedećom slikom (Slika 29).



Slika 29. Mobilna aplikacija - nagrada ekrani

Prvi prozor prikazuje izgled aplikacije u slučaju da je rezultat kviza manji od 70%. Korisnikova jedina opcija je napuštanje kviza, odnosno, dodir na gumb na dnu ekrana. Suprotno tome, korisnik je zadovoljavajuće riješio kviz te gumb na dnu ekrana ga ne vodi na početni fragment, već na fragment za odabir nagrade. Treći prozor sadrži *slideshow* nagrada kao što je već spomenuto. Dodirom na gumb *uzmi nagradu* u bilo kojem trenutku, *slideshow* se zaustavlja i nagrada je *izvučena*. Sljedeći fragment za ostavljanje osobnih podataka se automatski otvara. Ondje korisnik upisuje ime i prezime te potvrđuje podatke dodirom na gumb *pošalji*. Scenarij odgovara posljednjem ekranu na slici. Programski kod koji odgovara scenarijima s prva dva ekrana se nalazi na sljedećoj slici (Slika 30).

```
if (result >= 70){
    imageView.setImageDrawable(getResources().getDrawable(R.drawable.like));
    imageView.setAnimation(animationWin);
    resultBtn.setText("Preuzmi nagradu!");
    resultBtn.setOnClickListener((view) → {
            QuizPrizeFragment quizPrize = new QuizPrizeFragment();
            getFragmentManager().beginTransaction().replace(R.id.fragment_container_quiz, quizPrize).commit();
    });
} else {
    imageView.setImageDrawable(getResources().getDrawable(R.drawable.unlike));
    imageView.setAnimation(animationLost);
    resultBtn.setText("Početna");
    resultBtn.setOnClickListener((view) → {
            Intent intent = new Intent(getActivity(), MainActivity.class);
            startActivity(intent);
    });
}
```

Slika 30. Rezultat kviza - programski kod

Nakon preuzimanja proslijeđene vrijednosti koja označava postotak uspješnosti rješavanja kviza, programski kod sa slike provjerava da li je vrijednost veća odnosno manja od 70. Ako je veća ili jednaka 70, grafičkom sučelju se dodaje animacija te gumb na dnu ekrana usmjerava korisnika na fragment za preuzimanje nagrade. U suprotnom, dodaje se animacija, no gumb usmjerava korisnika na početni fragment. Nakon odabira nagrade korisnik se upućuje na posljednji fragment kviza, onaj gdje ostavlja svoje podatke kao dokaz o uspješno riješenom kvizu. Programski kod za taj scenarij se nalazi na sljedećoj slici (Slika 31).

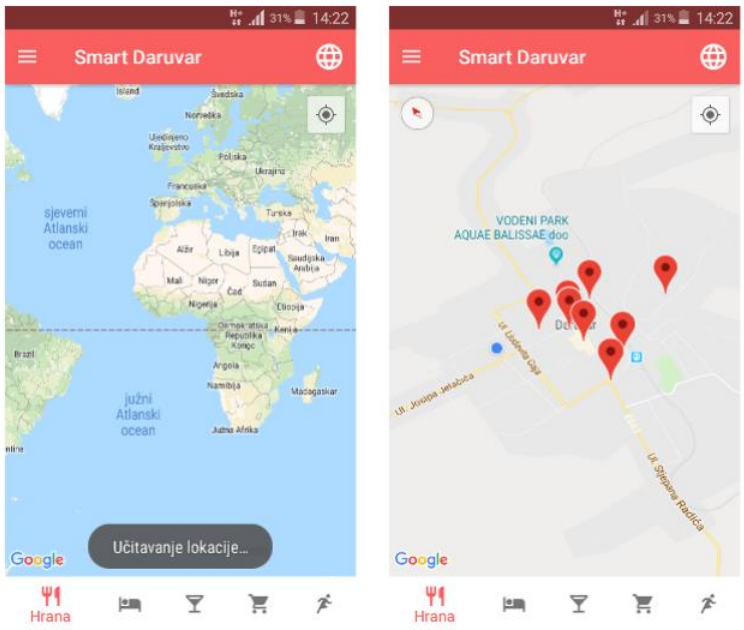
```
private void sendData(){
    ref = database.child(prizeCode);
    DatabaseReference pushedPostRef = ref.push();
    String postId = pushedPostRef.getKey();

    database.child(prizeCode).child(postId).child("first_name").setValue(firstName.getText().toString());
    database.child(prizeCode).child(postId).child("last_name").setValue(lastName.getText().toString());
    database.child(prizeCode).child(postId).child("used").setValue("ne");
    // ...
```

Slika 31. Slanje podataka u bazu - programski kod

Na slici se nalazi metoda *sendData* koja povezuje instancu klase *DatabaseReference* s *JSON* objektom baze podataka (*prizeCode)*. *JSON* objekt u bazi sadrži podatke o dobitnicima nagrade te se istom dodaje novi čvor s podacima koje korisnik upiše u elemente grafičkog sučelja. Uz ime i prezime se dodaje i podatak o iskorištenosti nagrade. Zadana vrijednost je *ne*, sve dok korisnik ne iskoristi svoju nagradu.

5.2.3. *SADRŽAJ*

Jedna od funkcionalnosti koju aplikacija nudi je prikaz sadržaja u blizini korisnika. Za ovu opciju, potreban je pristup lokaciji. Prikaz karte realiziran je pomoću *Google Maps API*-ja, dok je za sadržaje u blizini korišten *Google Places API*. Od sadržaja korisnik može pretraživati restorane, smještaj, kafiće, trgovine i sportske sadržaje. Podaci za sportske sadržaje se nalaze u *Firebase* bazi podataka te se formatiraju i prikazuju kao markeri na karti. Ostali sadržaj je rezultat pretrage *Google*-ovog *Nearby Search* zahtjeva. Ekrani na sljedećoj slici (Slika 32) prikazuju fragment *Sadržaj*.



Slika 32. Mobilna aplikacija - sadržaj ekrani

Prvi prozor je prikaz karte tijekom učitavanja lokacije. U podnožju prozora se nalazi horizontalni izbornik za odabir vrste sadržaja. Početno zadan sadržaj pri otvaranju karte je *Hrana*. Rezultati zahtjeva za obližnjim restoranima su prikazani markerima na karti, kao i za sve ostale pretrage. Programski kod za pozivanje klase dohvaćanja mjesta se nalazi na sljedećoj slici (Slika 33).

```
final BottomNavigationView bottomNavigationView = (BottomNavigationView) view.findViewById(R.id.bottomMenu);
bottomNavigationView.setOnNavigationItemSelectedListener((menuItem) → {
        switch (menuItem.getItemId()) {
            case R.id.restaurants:
                if (userLocation != null){
                    map.clear();

                    String url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json?location="+
                            userLocation.getLatitude()+"," + userLocation.getLongitude()+
                            "&radius=1500&type=restaurant&key=api_key";
                    Object data[] = new Object[5];
                    data[0] = url;
                    data[1] = map;
                    data[2] = userLocation.getLatitude();
                    data[3] = userLocation.getLongitude();
                    data[4] = mContext;
                    findNearbyPlaces = new FindNearbyPlaces();
                    findNearbyPlaces.execute(data);
                } else {
                    Toast.makeText(mContext, "Učitavanje lokacije…", Toast.LENGTH_SHORT).show();
                }
                break;
                // ...
```

Slika 33. Odabir vrste mjesta - programski kod

Na slici se nalazi programski kod nadjačane metode koja se poziva odabirom jednog elementa na donjoj navigacijskog traci. Konkretan primjer na slike odgovara odabiru elementa za prikaz restorana u blizini korisnika. Prvi uvjet je lokacija korisnika, ako ista nije dostupna, korisnik se obavještava o tome *Toast* porukom. Ako je lokacija dostupna, s karte se brišu svi elementi koji se nalaze na istoj. Zatim se definira zahtjev za mjestima u obliku poveznice koja sadrži geografsku lokaciju korisnika, radijus pretrage, vrstu mjesta te *API* ključ. Nakon toga, definira se polje tipa objekt koje sadrži prethodno definirani *URL*, kartu na kojoj će se prikazati mjesta, korisnikovu lokaciju te kontekst aktivnosti. Instancira se novi objekt klase *FindNearbyPlaces* kojemu se predaju navedeni argumenti te se poziva njegovo izvođenje. Slika 34 prikazuje programski kod klase *FindNearbyPlaces*.

```
@Override
protected String doInBackground(Object...params) {
    urlTmp = (String)params[0];
    map = (GoogleMap)params[1];
    lat = (double)params[2];
    lng = (double)params[3];
    mContext = (Context)params[4];

    try {
        URL url = new URL(urlTmp);
        HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
        httpURLConnection.connect();
        inputStream = httpURLConnection.getInputStream();
        bufferedReader = new BufferedReader(new InputStreamReader(inputStream));

        String line = "";
        stringBuilder = new StringBuilder();

        while ((line = bufferedReader.readLine()) != null){
            stringBuilder.append(line);
        }
        data = stringBuilder.toString();

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return data;
}
```

Slika 34. Metoda *doInBackground* - programski kod

Metoda *doInBackground* je dio *AsyncTask* klase *FindNearByPlaces*. Kako se korisnikovo sučelje ne bi blokiralo tijekom dohvaćanja mjesta, taj proces se izvodi u pozadinskoj dretvi, što znači da korisnik u glavnoj dretvi može nastaviti rad bez obzira na pozadinsko izvođenje procesa. Kada proces završi, njegovo sučelje će biti ažurirano novim stanjem na karti. Metoda sa slike u pozadini izvodi zahtjev za dohvaćanje lokacije. Na početku, proslijeđeni argumenti instanci klase *FindNearByPlaces* se pohranjuju u varijable. Instancira se varijabla *url* s kojom se poziva metoda za otvaranje konekcije *HTTP* zahtjeva. Nakon kreiranja zahtjeva, odgovor od servera se pohranjuje pomoću objekta *bufferReader*. Rezultat zahtjeva pohranjuje se u objekt *stringBuilder* liniju po liniju odgovora. Na kraju, metoda vraća varijablu *data* koja je poprimila vrijednost objekta *stringBuilder*. Nakon pozadinskog izvođenja, poziva se metoda *onPostExecute* koja se nalazi na 35. slici.

```
@Override
protected void onPostExecute(String s) {
    if (s == null){
        Toast.makeText(mContext, text: "Provjerite internetsku vezu!", Toast.LENGTH_SHORT).show();
    } else {
        try {
            JSONObject object = new JSONObject(s);
            JSONArray jsonArray = object.getJSONArray( name: "results");

            for (int i = 0; i < jsonArray.length(); i++) {
                JSONObject jsonObject = jsonArray.getJSONObject(i);
                JSONObject location = jsonObject.getJSONObject("geometry").getJSONObject("location");
                String latitude = location.getString( name: "lat");
                String longitude = location.getString( name: "lng");

                latsNovo.add(latitude);
                lngsNovo.add(longitude);
                // ...
```

Slika 35. Metoda *onPostExecute* - programski kod

Rezultat zahtjeva pohranjen u varijablu se dodaje *JSON* objektu koji dohvaća *JSON* polje naziva *results.* Zatim, unutar *for* petlje se obrađuju rezultati zahtjeva, tako što se dohvaćaju *JSON* objekti rezultata i pohranjuju u liste. Primjerice, unutar polja *results* nalaze se objekti *location* koji predstavljaju geografsku lokaciju mjesta. Za svako mjesto, dohvaćaju se lokacije (*lat* i *lng*) koje se dodaju u liste. Slika 36 prikazuje dodavanje markera dohvaćenih mjesta.

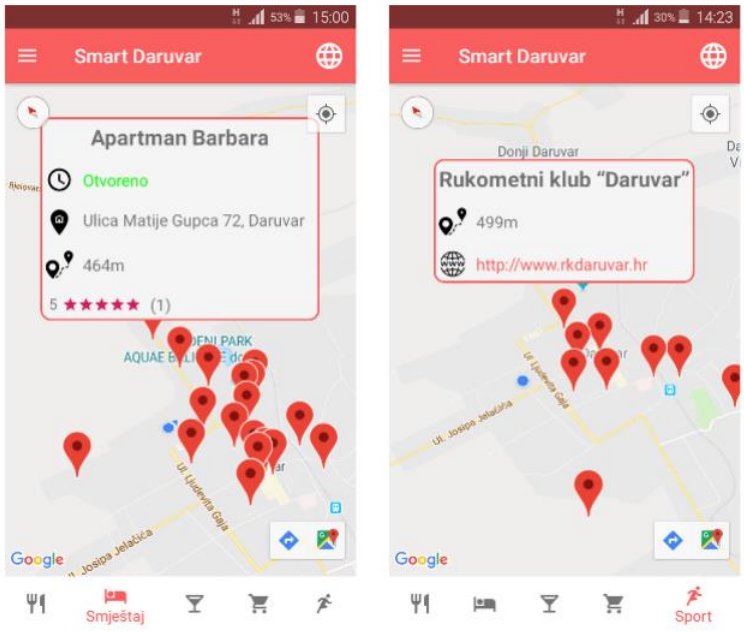
```
// ...
for (int i=0; i<latsNovo.size(); i++){
    LatLng latLng = new LatLng(Double.valueOf(latsNovo.get(i)), Double.valueOf(lngsNovo.get(i)));
    String id = String.valueOf(i);

    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.title(nameNovo.get(i));
    markerOptions.position(latLng);
    markerOptions.snippet(id);

    map.addMarker(markerOptions);
}

map.setInfoWindowAdapter(new InfoWindowAdapter(mContext, lat, lng, nameNovo, latsNovo, lngsNovo,
        vrijemeNovo, ocjenaNovo, brojOcjenaNovo, adresaNovo));
// ...
```

Slika 36. Postavljanje markera na kartu - programski kod

Nakon pohrane svih podataka mjesta u liste, unutar *for petlje* se dodaju markeri na kartu te se za svaki marker postavlja *InfoWindow* s njegovim podacima. To se ostvaruje

uz pomoć adaptera *InfoWindowAdapter* kojemu se prosljeđuju liste s podacima o mjestima.

Dodirom na marker, otvara se *InfoWindow* s detaljima odabranog mjesta, što je vidljivo na sljedećoj slici (Slika 37).



Slika 37. Mobilna aplikacija - detalji sadržaja ekrani

InfoWindow za hranu, smještaj, kafiće i trgovine izgleda kao *InfoWindow* na lijevom ekranu. Na vrhu se nalazi naziv objekta, a ostali podaci su strukturirani kao slika/tekst. Radno vrijeme je prikazano kao *Otvoreno* ili *Zatvoreno*. Sljedeći podatak je adresa, zatim udaljenost od korisnika do objekta te ocjene istog. Ocjene su prikazane na sljedeći način: prosječna ocjena, prikaz ocjene sa zvjezdicama, broj ocjena u zagradi. Za sportske sadržaje *InfoWindow* je drugačiji jer se podaci o objektu preuzimaju iz baze podataka koja sadrži koordinate, naziv i web stranicu sportskog sadržaja. Udaljenost se, kao i na ostalim sadržajima, izračunava u odnosu na lokaciju korisnika. Dodirom na *InfoWindow* za sportske sadržaje otvara se web preglednik, odnosno web stranica objekta ako ista postoji. Programski kod za postavljanje *InfoWindowa* se nalazi na 38. slici.

```
private void setData(Marker marker, View view, int position){
    mName = (TextView)view.findViewById(R.id.nameInfo);
    mIsOpen = (TextView)view.findViewById(R.id.timeInfo);
    mAdress = (TextView)view.findViewById(R.id.adressInfo);
    mDistance = (TextView)view.findViewById(R.id.distanceInfo);
    mRatingBar = (RatingBar)view.findViewById(R.id.ratingInfo);
    mNumberOfGrades = (TextView)view.findViewById(R.id.numberOfGradesInfo);
    mRating = (TextView)view.findViewById(R.id.rating);
    mName.setText(namesList.get(position));
    if (isOpenList.get(position).equals("true")){
        mIsOpen.setText("Otvoreno");
        mIsOpen.setTextColor(Color.GREEN);
    } else {
        mIsOpen.setText("Zatvoreno");
        mIsOpen.setTextColor(Color.RED);
    }
    mAdress.setText(adressList.get(position));
    float[] results = new float[1];
    Location.distanceBetween(userLatValue, userLngValue,
            Double.valueOf(latsList.get(position)), Double.valueOf(lngsList.get(position)), results);
    int distance = (int)results[0];
    if (distance < 1000){
        mDistance.setText(distance + "m");
    } else {
        float distanceLong = results[0]/1000;
        String distanceLongTwoDecimals = String.format("%.2f", distanceLong);
        mDistance.setText(distanceLongTwoDecimals + "km");
    }
    mRating.setText(ratingList.get(position));
    mRatingBar.setRating(Float.valueOf(ratingList.get(position)));
    mNumberOfGrades.setText(" (" + numberOfRatingsList.get(position) + ")");
}
```

Slika 38. Metoda za postavljanje *InfoWindow*-a - programski kod

Dodirom na marker, poziva se metoda *setData* koja postavlja vrijednosti iz lista proslijeđenih konstruktorom kase u elemente korisničkog sučelja. Podaci se formatiraju ili se postavljaju u obliku u kojemu su preuzeti s *Google* servera. Primjerice, radno vrijeme mjesta se prikazuje u obliku *Otvoreno* ili *Zatvoreno*, ovisno o vrijednosti u listi (*true* ili *false*).

5.2.4. *O DARUVARU*

Sljedeća opcija koju korisnik ima na raspolaganju je prikaz podataka o Daruvaru. Podaci koji se prikazuju su oblika slika/tekst koji se nalaze u bazi podataka. Izgled fragmenta *O Daruvaru* se nalazi na sljedećoj slici (Slika 39).

Slika 39. Mobilna aplikacija - o Daruvaru ekran

Tekst o Daruvaru se prikazuje u pet *TextViewa* te se ispod svakog nalazi fotografija Grada. Na dnu prikaza, nalazi se web poveznica na službene stranice grada Daruvara u slučaju potrage za detaljnijim informacijama. Na slici 40 se nalazi programski kod fragmenta *O Daruvaru*.

```
private void setData(){
    for (int i=0; i<textViews.size(); i++){
        if (i == textViews.size()-1){
            textViews.get(i).setText("https://daruvar.hr");
        } else {
            textViews.get(i).setText(texts.get(i));
        }
    }

    for (int j=0; j<images.size(); j++){
        Glide.with(mContext).load(images.get(j)).into(imageViews.get(j));
    }
}
```

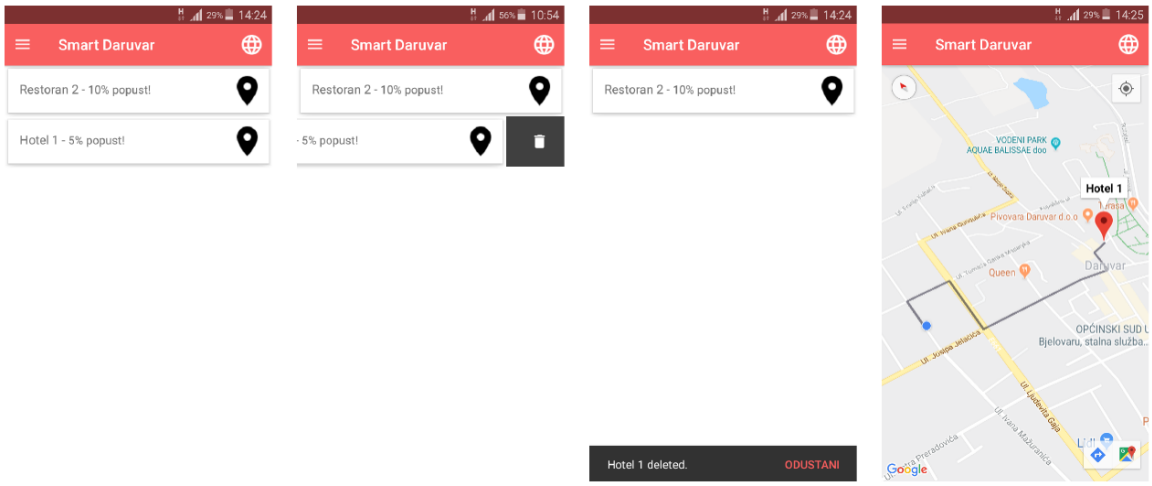
Slika 40. Postavljanje podataka o Daruvaru - programski kod

Metoda u prvoj *for* petlji postavlja tekst u elemente korisničkog sučelja za prikaz teksta. U posljednji *TextView* postavlja poveznicu na službene stranice grada Daruvara. U

sljedećoj *for* petlji, postavljaju se slike Grada u *ImageView* elemente uz pomoć *Glide* biblioteke.

5.2.5. *MOJE NAGRADE*

Posljednja opcija aplikacije je prikaz korisnikovih osvojenih nagrada, ako ih ima. Fragment *Moje Nagrade* prikazuje podatke kao listu *itema* gdje je *item* jedna nagrada. Korisnik ima opciju obrisati nagradu, ako ju je, primjerice, iskoristio te zatražiti upute do mjesta gdje nagradu može iskoristiti. Opisani scenariji su vidljivi na sljedećoj slici (Slika 41).



Slika 41. Mobilna aplikacija - *moje nagrade* ekrani

Na prvom prozoru se nalaze sve nagrade koje je korisnik ostvario, dok je na sljedećem prozoru demonstrirano brisanje nagrade. Brisanje je moguće povlačenjem *itema* lijevo ili desno. U slučaju da je nagrada slučajno obrisana, postoji opcija vraćanja *itema* dodirom na *odustani* na dnu prozora kao što je prikazano na trećem ekranu. Odabirom opcije za dohvaćanje uputa do mjesta za korištenje nagrade otvara se karta s iscrtanom rutom od korisnikove lokacije do odabranog mjesta (zadnji prozor). Programski kod za dohvaćanje podataka o osvojenim nagradama se nalazi na sljedećoj slici (Slika 42).

```
// ...
JSONArray jsonArray;
jsonArray = getSavedJSONArrayPrizes();
System.out.println("ARRAY: " + jsonArray);
if (jsonArray != null) {
    for (int i=0; i<jsonArray.length(); i++){
        try {
            prizes.add(i, jsonArray.getJSONObject(i).getString( name: "prize"));
            discounts.add(i,jsonArray.getJSONObject(i).getString( name: "discount"));
            lats.add(i, jsonArray.getJSONObject(i).getString( name: "lat"));
            lngs.add(i, jsonArray.getJSONObject(i).getString( name: "lng"));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    // ...
```

Slika 42. Dohvaćanje ostvarenih nagrada - programski kod

Podaci o osvojenim nagradama se pohranjuju u lokalnu datoteku nakon preuzimanja nagrade. Isti podaci se dohvaćaju u programskom kodu sa slike koji se formatiraju u *JSON* polje. Zatim se iz polja dohvaćaju *JSON* objekti s podacima o nazivu nagrade, popustu te lokaciji iste. Slika 43 prikazuje programski kod za iscrtavanje uputa na karti.

```
List<LatLng> directions = PolyUtil.decode(encoded);
PolylineOptions options = new PolylineOptions().width(5).color(R.color.appColor);

for (int i=0; i<directions.size(); i++){
    LatLng latLng = directions.get(i);
    options.add(latLng);
}

progressBar.setVisibility(View.INVISIBLE);

LatLng latLng = new LatLng(Double.valueOf(lat), Double.valueOf(lng));
map.addMarker(new MarkerOptions().position(latLng).title(prizeName));
polyline = map.addPolyline(options);
```

Slika 43. Iscrtavanje uputa na karti - programski kod

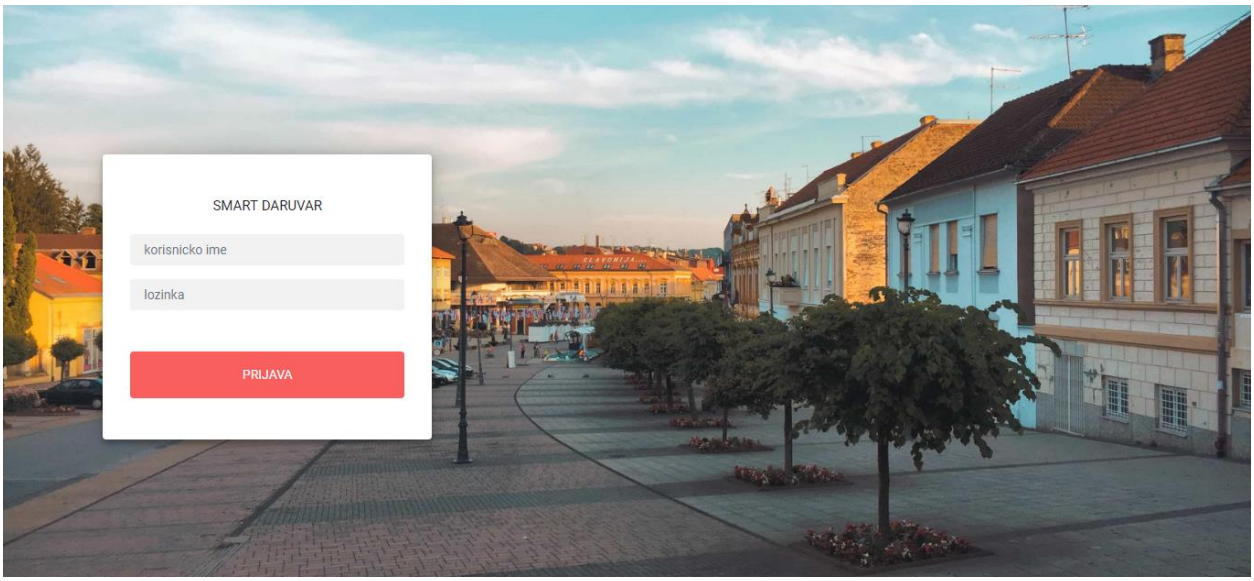
Upute od korisnikove lokacije do lokacije mjesta za korištenje osvojene nagrade su dohvaćeni na način kao i mjesta u blizini korisnika, odnosno procesom u pozadinskog dretvi. Nakon završenog procesa preuzimanja podataka, rezultati u obliku *stringa* se formatiraju u *JSON* format iz kojega se dohvaćaju upute. Iste se spremaju u listu dekodiranjem pomoću klase *PolyUtil* koja se nalazi u biblioteci *Maps Utils*. Na karti se zatim iscrtava *Polyline* grafički element između dvije lokacije.

5.3. WEB DIO APLIKACIJE

Web aplikacija *Smart Daruvar* namijenjena je osoblju objekata u kojima je moguće ostvariti popust igranjem kviza. Aplikacija je razvijena kao jednostavna web aplikacija za validaciju osvajanja nagrade. Sadrži dva prozora, jedan za prijavu u sustav i drugi za prikaz podataka. Implementirana je u nekoliko datoteka: *index.html* (za prozor prijave u sustav), *main.html* (za prozor prikaza podataka), *login.php* (za validaciju korisničkog imena i lozinke), *get_data.php* (za dohvaćanje podataka o poslovnom objektu), *get_table_data.php* (za dohvaćanje podataka o dobitnicima nagrade), *change_data.php* (za promjenu statusa iskorištenosti nagrade), *composer.json* i *composer.lock* (za upravljanje bibliotekama), *JSON* datoteke koja sadrži podatke o povezivanju s bazom podataka te *style.css* datoteke za definiranje izgleda aplikacije.

5.3.1. PRIJAVA

Sljedeća slika (Slika 44) prikazuje izgled prozora za prijavu u sustav web aplikacije *Smart Daruvar*.



Slika 44. Web aplikacija - prijava ekran

Prozor za prijavu sadrži *login* formu koja je postavljena na pozadinsku fotografiju grada Daruvara. Forma zahtjeva upis korisničkog imena i lozinke. U slučaju da jedno od polja nije uneseno, prijava nije moguća te će korisnik o tome biti obaviješten. Unosom

podataka u oba polja forme te klikom na gumb prijava, poziva se *PHP* skripta koja u bazi podataka provjerava ispravnost podataka. U slučaju netočnog korisničkog imena ili lozinke, prijava će biti odbijena, a korisnik obaviješten. Inače, ulazak u sustav će biti odobren. Ulaskom u sustav, otvara se tablica s podacima. Slika 45 prikazuje programski kod stranice za prijavu u sustav.

```
$('#loginForm').submit(function (e) {
  e.preventDefault();

  document.getElementById("loaderLogin").style.visibility = "visible";
  document.getElementById('login_message').innerHTML = "";

  var data = {
    'username': $('input[name=username]').val(),
    'password': $('input[name=password]').val()
  };
  $.ajax({
    url: "login.php",
    type: "POST",
    data: data,
    success: function (data, textStatus, jqXHR) {
      if (data == true) {
        document.getElementById("loaderLogin").style.visibility = "hidden";
        var subjectName = document.getElementById('username').value;
        setCookie("login", subjectName, 1);
        window.open ('main.html?name=' + subjectName,'_self',false)
      } else {
        document.getElementById("loaderLogin").style.visibility = "hidden";
        document.getElementById('login_message').innerHTML = data;
      }
    },
    error: function (jqXHR, textStatus, errorThrown) {
      document.getElementById('login_message').innerHTML = textStatus;
    }
  });
});
```

Slika 45. Metoda za prijavu u sustav - programski kod

Klikom na gumb *prijava* poziva se metoda, odnosno događaj sa slike. Element grafičkog sučelja postaje vidljiv, a tekst za unos pogrešnog korisničkog imena ili lozinke postane nevidljiv. Zatim se u varijablu *data* sprema uneseno korisničko ime i lozinka. Varijabla se prosljeđuje *AJAX* asinkronom zahtjevu za provjeru ispravnosti podataka, odnosno, proziva se skripta *login.php*. Unutar zahtjeva, implementirane su dvije metode, ona u slučaju uspješno vraćenog odgovora na zahtjev, odnosno ona u slučaju da zahtjev nije uspio. Uspješno obrađenim zahtjevom poziva se metoda koja provjerava je li rezultat

zahtjeva jednak *true*. Ako je *true*, *progress* element se uklanja s web stranice te se poziva metoda *setCookie* kojoj se prosljeđuje upisano korisničko ime i uspješno se otvara novi prozor za prikaz podataka. Ako rezultat nije jednak *true*, na stranici se ispisuje poruka, odnosno rezultat zahtjeva. Ako zahtjev nije uspješno završen, ispisuje se poruka greške. Metoda *setCookie* služi za spremanje podataka o korisniku, odnosno spremanje podataka za prijavu u sustav. Korisnik može napustiti ili osvježiti stranicu bez ponovnog prijavljivanja. Odjaviti se može klikom na gumb odjava ili će biti automatski odjavljen nakon 24 sata od zadnjeg posjeta stranici (istek sesije). Slika 46 prikazuje *PHP* skriptu za prijavu u sustav.

```php
if(isset($_POST['username'])) {
    $username = $_POST['username'];
} else {
    $username = "";
}

if(isset($_POST['password'])) {
    $password = $_POST['password'];
} else {
    $password = "";
}

$serviceAccount = ServiceAccount::fromJsonFile( filePath: __DIR__.'/secret/smart-daruvar-key...json');

$firebase = (new Factory)
    ->withServiceAccount($serviceAccount)
    ->create();

$database = $firebase->getDatabase();
$reference = $database->getReference();
$snapshot = $reference->getChild( path: "users");

$user = $snapshot->getChild($username)->getChild( path: 'username')->getValue();
$pass = $snapshot->getChild($username)->getChild( path: 'password')->getValue();

if ($username != $user || $password != $pass){
    echo "Pogrešno korisničko ime ili lozinka";
} else {
    echo true;
}
```

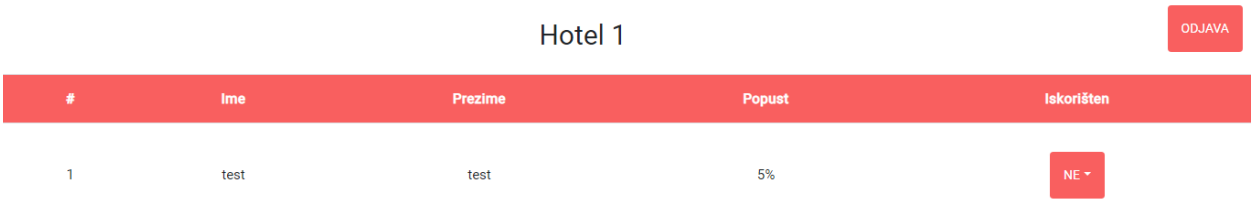
Slika 46. Prijava u sustav - *php* skripta - programski kod

Na početku skripte se provjerava da li je zahtjevu proslijeđeno korisničko ime i lozinka. Ako nije, oba podatka se postavljaju kao prazni, inače se proslijeđeni spremaju u varijable. Nakon toga se izvodi spajanje s bazom podataka te se u *JSON* objektu *users*, u bazi podataka, pronalazi objekt s proslijeđenim korisničkim imenom te objekt iste

vrijednosti kao i lozinka. Ako su podaci pronađeni, pristup je omogućen, odnosno skripta kao povratnu vrijednost vraća *true*. U suprotnom vraća poruku o pogrešnom korisničkom imenu ili lozinki.

5.3.2. SUSTAV

Slika 47 prikazuje korisničko sučelje nakon ulaska u sustav koji se sastoji od tablice s podacima o dobitnicima, naziva poslovnog objekta te gumba za odjavu iz sustava.



Slika 47. Web aplikacija - podaci ekran

Tablica sadrži podatke o korisniku koji je ostvario nagradu u njihovom objektu. Podaci koji se prikazuju su: redni broj koji se automatski generira, ime, prezime, popust te je li popust iskorišten. Osoblje ima opciju promijeniti vrijednost varijable iskorišten iz *NE* u *DA*, dok obrnuto nije moguće. Programski kod *PHP* skripte za promjenu podataka o iskorištenosti nagrade se nalazi na sljedećoj slici (Slika 48).

```php
$database = $firebase->getDatabase();
$reference = $database->getReference();
$snapshot = $reference->getChild($table)->getChildKeys();

$result = false;

for ($i=0; $i<sizeof($snapshot); $i++){
    if ($reference->getChild($table)->getChild($snapshot[$i])->getChild( path: "first_name")->getValue() == $first_name
    && $reference->getChild($table)->getChild($snapshot[$i])->getChild( path: "last_name")->getValue() == $last_name){
        $reference->getChild($table)->getChild($snapshot[$i])->update($arr);
        $result = true;
    }
}

echo $result;
```

Slika 48. Promjena statusa nagrade - php skripta - programski kod

Nakon klika na gumb za promjenu statusa nagrade, kao i kod prijave u sustav, poziva se *AJAX* metoda, odnosno *PHP* skripta kojoj se prosljeđuju sljedeći podaci: ime i prezime korisnika, status iskorištenosti nagrade kojemu je vrijednost *da* i naziv poslovnog objekta. Zatim, skripta u bazi podataka traži *JSON* objekt naziva jednakog kao naziv proslovnog objekta. Nakon toga, u *for* petlji koja se izvršava onoliko puta koliko *JSON* objekt ima djece, skripta traži dijete istog imena i prezimena. U slučaju pronalaska, tom djetetu, odnosno *JSON* objektu mijenja stanje iskorištenosti nagrade u *da*. Na kraju, skripta kao povratnu vrijednost vraća *true*.

6. ZAKLJUČAK

Iz svega navedenog može se zaključiti da turizam u današnje vrijeme ovisi o tehnologiji, odnosno da razvoj tehnologije doprinosi razvoju i promoviranju turizma. Postoje mnoge mobilne i web aplikacije namijenjene turizmu bez kojih bi danas bilo nezamislivo pretražiti, odnosno rezervirati smještaj u destinaciji koju korisnik namjerava posjetiti, kao i primjerice informirati se o načinima putovanja.

Cilj ovog diplomskog rada bio je razviti mobilnu *Android* aplikaciju za promociju turizma u Daruvaru te web aplikaciju namijenjenu djelatnicima poslovnih objekata u Gradu. Aplikacija *Smart Daruvar* je moguće proširenje postojeće turističke ponude Daruvara upravo zbog modernog, tehnološkog te, danas uobičajenog, načina pristupa informacijama. Aplikacija nudi edukativan, zabavan i rekreacijski način upoznavanja Grada uz mogućnost ostvarenja raznih pogodnosti.

Mobilna aplikacija je implementirana u *Android Studiou* korištenjem *Java* i *XML* programskih jezika. Prikaz karte je ostvaren korištenjem *Google Maps* biblioteke koja sadrži već implementirane metode, primjerice, postavljanje oznaka na karti. Jedna od funkcionalnosti koje aplikacija nudi je pregled mjesta (restorana, smještaja, kafića, trgovina i sportsko-rekreacijskih sadržaja) u blizini korisnika. Korištena biblioteka za tu svrhu je *Google Places*. Uz spomenute dvije *Google*-ove biblioteke, korištena je još i *Google Directions* s kojom se izračunava optimalna ruta između dva mjesta, odnosno između korisnikove lokacije i mjesta za ostvarivanje osvojene nagrade igranjem kviza.

Korištena baza podataka je *Firebase Realtime Database* u kojoj su podaci *JSON* formata. Ista služi za pohranu podataka o kvizu, sportsko-rekreacijskim sadržajima, Gradu itd.

Web aplikacija je implementirana u razvojnom okruženju *PHP Storm*. Uz *PHP* programski jezik koji je korišten za dohvaćanje i ažuriranje podataka iz baze, u implementaciji su upotrijebljeni i *HTML*, *CSS* i *JavaScript* programski jezici za definiranje sučelja web aplikacije te upravljanje korisničkim događajima.

Spoj mobilne i web *Smart Daruvar* aplikacije je sustav za promociju Grada i ostvarivanje

raznih pogodnosti igranjem kviza (mobilna) uz validaciju i mogućnost korištenja ostvarenih pogodnosti (web). Takav sustav bi osvježio turističku ponudu i mogao služiti domaćim i inozemnim korisnicima, odnosno turistima.

LITERATURA

Knjige:

1) Gržinić, J. (2019), *Uvod u turizam*, Sveučilište Jurja Dobrile u Puli, Pula
2) Gržinić, J. i Floričić, T. (2015), *Turoperatori i hotelijeri u suvremenom turizmu*, Sveučilište Jurja Dobrile u Puli, Pula
3) MarCon (2011), *Dijagnoza stanja te vizija i misija razvoja turizma za područje Daruvar - Papuk*, Grad Daruvar, Daruvar

Članci:

1) Bačić, I. i Medak, M., (2012). *Zdravstveni turizam u Hrvatskoj i Bjelovarsko-bilogorskoj županiji*, *Radovi Zavoda za znanstvenoistraživački i umjetnički rad u Bjelovaru*, (6), str. 211-214. Preuzeto s: https://hrcak.srce.hr/91584, Pristupljeno 19.8.2019.

Statističke publikacije:

1) *International Recommendations for Tourism Statistics* 2008, *Publikacija 978-92-1-161521-0*, UNWTO, New York (2010)

Internetski izvori:

1) Airbnb, Preuzeto s: https://press.airbnb.com/about-us/, Pristupljeno 19.8.2019.
2) Android, Preuzeto s: https://developer.android.com/about/dashboards, Pristupljeno 20.8.2019. te https://developer.android.com/guide/platform, Pristupljeno 9.8.2019.
3) Android Developers, Preuzeto s: https://developer.android.com/studio/intro, Pristupljeno 21.8.2019.
4) Booking.com, Preuzeto s: https://www.booking.com/content/about.html, Pristupljeno 19.8.2019.
5) Ipwatchdog, Preuzeto s: https://www.ipwatchdog.com/2014/11/26/a-brief-history-of-googles-android-operating-system/id=52285/, Pristupljeno 19.8.2019.
6) C#Corner, Preuzeto s: https://www.c-sharpcorner.com/article/what-is-the-most-popular-operating-system/, Pristupljeno 9.8.2019.
7) ComponentSource, Preuzeto s:

https://www.componentsource.com/product/phpstorm/about, Pristupljeno: 21.8.2019.

8) Firebase, Preuzeto s: https://firebase.google.com/docs/storage, Pristupljeno: 20.8.2019. te Preuzeto s: https://firebase.google.com/docs/database, Pristupljeno 20.8.2019.
9) Google, Preuzeto s: https://developers.google.com/maps/documentation/android-sdk, Pristupljeno 20.8.2019., Preuzeto s: https://developers.google.com/places/android-sdk/intro, Pristupljeno: 21.8.2019., Preuzeto s: https://developers.google.com/places/web-service/search#PlaceSearchRequests, Pristupljeno: 21.8.2019. te Preuzeto s: https://developers.google.com/maps/documentation/directions/intro, Pristupljeno: 21.8.2019.
10) SimilarWeb, Preuzeto s: https://www.similarweb.com/website/booking.com?competitors=airbnb.com, Pristupljeno 19.8.2019.
11) Statcounter, Preuzeto s: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201807-201907-bar, Pristupljeno 9.8.2019.
12) Tutorials Point, Preuzeto s: https://www.tutorialspoint.com/json/json_tutorial.pdf, Pristupljeno 20.8.2019.
13) TZ Daruvar, Preuzeto s: http://tz-daruvar.hr/zdravstveni-turizam-u-daruvaru/, Pristupljeno 19.8.2019.
14) Visitdaruvar, Preuzeto s: http://www.visitdaruvar.hr/o-daruvaru-1.aspx, Pristupljeno 19.8.2019.

PRILOZI

SAŽETAK

Spoj tehnologije i turizma čini neizostavan dio modernog doba te, kao takav, omogućuje lak, jednostavan i brz pristup informacijama u svrhu turističke promocije, odnosno zadovoljavanja svih elemenata turističke ponude i potražnje. Razvijena aplikacija *Smart Daruvar* je moderan, tehnološki pristup promocije grada Daruvara. Ista nudi korisnicima mogućnost ostvarivanja raznih pogodnosti igranjem kviza koji uključuje obilaženje najatraktivnijih lokacija u Gradu i odgovaranje na pitanja vezana uz lokacije. Jedna od mogućnosti je pretraga mjesta u blizini (restorani, smještaj, kafići, trgovine i sportsko-rekreacijski sadržaji) te prikaz informacija vezanih uz iste. Dokaz o uspješno odigranom kvizu se odvija preko web dijela aplikacije koja je namijenjena djelatnicima mjesta u kojima je moguće iskoristiti nagradu. Praktični dio rada čini razvoj i implementaciju aplikacije, a teorijski dio obuhvaća opis korištenih tehnologija i izrade aplikacije te korisničke scenarije.

Ključne riječi: mobilna aplikacija, web aplikacija, Android, turizam, Daruvar

ABSTRACT

The combination of technology and tourism is an indispensable part of modern times and, therefore, provides easy and fast access to information for the purpose of tourism promotion as well as meeting all elements of tourism supply and demand. The developed *Smart Daruvar* app is a modern and technological approach for the purpose of promotion of Daruvar. It offers users the opportunity to gain a variety of benefits by playing a quiz that involves visiting the most attractive locations in the City and answering location questions. One option is to search nearby places (restaurants, accommodation, cafes, shops and sports and recreational facilities) and view information related to them. The proof of successfully solved quiz is developed through the web part of the application, which is intended for employees of the place where it is possible to redeem the prize. The practical part of the master's thesis is the development and implementation of the application, and the theoretical part includes a description of the technologies used as well as the design of the application and user scenarios.

Keywords: mobile application, web application, Android, tourism, Daruvar