Mobilna Android aplikacija za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2

Bunoza, Domagoj

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

Sveučilišni studij

MOBILNA ANDROID APLIKACIJA ZA VIŠEKRITERIJSKU PROCJENU IZLOŽENOSTI ZARAZI VIRUSOM SARS-COV-2

Završni rad

Domagoj Bunoza

Osijek, 2021.

SADRŽAJ

1. UVOD

SARS-CoV-2 je prvi put identificiran potkraj 2019. godine u kineskom gradu Wuhanu kada je 40-ak osoba oboljelo od pneumonije bez tada poznatog uzroka. Nedugo zatim, došlo je do proširenja epidemije na ostatak Kine, Južnu Koreju, Iran i Italiju, dok je mnoštvo ostalih država brojalo dvoznamenkaste brojeve slučajeva. 11. ožujka 2020. proglašena je globalna pandemija zbog COVID-19 (eng. *Coronavirus Disease 2019*) od strane Svjetske zdravstvene organizacije. SARS-CoV-2 u mogućnosti je rasprostraniti se iznimno velikom brzinom jer se širi uglavnom kapljičnim putem. Glavni simptomi SARS-CoV-2 su, prema [1], visoka temperatura, suhi kašalj i umor. COVID-19 opasan je zbog toga što se sporo kreće plućima, te iza sebe ostavlja oštećeno plućno tkivo. Uz to, pridonosi niskoj količini šećera u krvi i šteti ostalih organa.

Za ostvarenje projekta nužno je prikupiti informacije o relevantnim čimbenicima koji u većoj mjeri utječu na vjerojatnost zaraze virusom SARS-CoV-2. Bit će osmišljen prikladan model procjene vjerojatnosti zaraze s obzirom na temeljito istražene relevantne čimbenike, čiji rezultat će se pratiti određen vremenski period. Sukladno procjeni vjerojatnosti infekcije virusom SARS-CoV-2, stvarat će se profil korisnika određene razine rizika infekcije, uz odgovarajuću preporuku. U radu će se analizirat programske okoline i alati za razvoj aplikacije, objasnit će se odabrana programska arhitektura aplikacije, obrazložit će se izbor operacijskog sustav za koji se razvija aplikacija, a nakon toga, bit će prikazano programsko rješenje mobilne aplikacije, te prikazani i analizirani rezultati ispitivanja mobilne aplikacije za više slučajeva korištenja.

U drugom poglavlju bit će prikazani i analizirani uzroci i moguće posljedice bolesti COVID-19, te potencijalni načini sprječavanja infekcije. Treće poglavlje daje opis osmišljenog modela programskog rješenja i postupaka za višekriterijsku procjenu izloženosti zarazi. Nakon toga se u četvrtom poglavlju daje osvrt na korištenje programske tehnologije, jezike i razvojnu okolinu i iscrpno objašnjava programsko rješenje na strani korisnika i na strani poslužitelja. U petom poglavlju opisan je način rada i korištenje mobilne aplikacije, te su analizirani rezultati dobiveni ispitivanjem za više korisničkih slučajeva.

1.1. Zadatak završnog rada

U završnom radu treba opisati i analizirati opasnosti od zaraze virusom SARS-CoV-2, te načine prijenosa virusa i mogućnosti koje može pružiti mobilna aplikacija u smislu smanjenja rizika zaraze. Na temelju globalne i lokalne epidemijske situacije, uvjeta širenja virusa i profila izloženosti korisnika zarazi, a koristeći prostorno-vremenske mogućnosti mobilnog uređaja i odgovarajućih usluga, odnosno geolokaciju i kalendarsko vrijeme, treba osmisliti i definirati model i postupak, te programsku arhitekturu mobilne aplikacije za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2. Također, treba opisati potrebne programske tehnologije, jezike i razvojne okvire, a u praktičnom dijelu rada, treba razviti mobilnu aplikaciju s bazom podataka koja implementira navedeni model višekriterijske procjene izloženosti zarazi. Nadalje, mobilna aplikacija treba omogućiti unos i pohranu potrebnih podataka, prikladan prikaz razine izloženosti zarazi ovim virusom, stvaranje preporuka prema korisniku i praćenje stanja korisnika. Mobilnu aplikaciju potrebno je ispitati i analizirati za odgovarajuće ulazne podatke, profile korisnika i slučajeve korištenja.

2. IZAZOVI PRI PROCJENI VJEROJATNOSTI ZARAZE SARS-COV-2

2.1. Opis bolesti COVID-19

COVID-19 je bolest uzrokovana virusom SARS-CoV-2, a pojavila se 2019. godine. Prema [2], to je jedan od virusa koji cirkulira među životinjama, no može prijeći na ljude. Neki od mogućih simptoma su:

- Povišena tjelesna temperatura
- Kašalj
- Otežano disanje
- Bol u mišićima
- Umor

Moguće je pojavljivanje pneumonije ili akutne upale pluća, sepsa, no najčešće kod pacijenata sa značajnim komorbiditetom. Ukoliko dođe do zaraze, osoba ne mora odmah pokazivati simptome, već je procijenjeno vrijeme prvih simptoma oko pet dana [3]. Učestalost pojavljivanja simptoma prema [1] može se vidjeti u tablici 2.1. Osim navedenih simptoma, također se često pojavljuje gubitak mirisa i okusa.

Tablica 2.1 Učestalost pojavljivanja simptoma

Simptom	Pojavnost
Vrućica	83–99%
Kašalj	59–82%
Gubitak apetita	40–84%
Umor	44–70%
Kratkoća daha	31–40%
Iskašljavanje sputuma	28–33%
Bol u mišićima	11–35%

2.2. Analiza čimbenika vjerojatnosti zaraze

Kao što je navedeno u [2], virus SARS-CoV-2 prenosi se uglavnom kapljičnim putem i mogućnost infekcije razlikuje se od osobe do osobe. Uzimajući u obzir nepotpune podatke o ovom virusu, u [4] se navodi kako postoji značajna razlika između mogućnosti infekcije žena i muškaraca. Žene su općenito otpornije na infekcije nego muškarci jer imaju snažniji imunološki sustav, a uz to

imaju odgovorniji stav prema pandemiji COVID-19. Muškarci češće u svoj životni stil ukomponiraju konzumiranje alkohola i pušenje u usporedbi sa ženama, što slabi imunitet i čini organizam podložniji zarazama. Samim time, i konzumacija alkohola i pušenje neizostavan su čimbenik pri procjeni vjerojatnosti zaraze. Prema [5], mogućnost hospitalizacije osobe u dobnom rangu od 65 do 74 godine čak je 40 puta veća i smrtnost je 1300 puta veća nego kod osoba između 5 i 17 godina. Stoga, dob je važan čimbenik za koji utječe na širenje virusa SARS-CoV-2. Bilo kakvo smanjenje učinkovitosti imunološkog sustava povećava rizik infekcije, što znači da ukoliko netko koristi lijekove kojima je jedna od nuspojava smanjenje učinkovitosti imunološkog sustava, imat će povećanu mogućnost zaraze virusom SARS-CoV-2. Nadalje, izlaganje društvu uz manjak fizičke distance čini rizik zaraze većim. Dakle, osoba koja posjeti 15 lokacija u danu imat će veću mogućnost zaraze od osobe koja je posjetila jednu lokaciju – odnos broja posjećenih lokacija i vjerojatnosti zaraze je proporcionalan. Broj posjećenih lokacija usko je povezan sa zanimanjem. Primjerice, radnik u zdravstvu imat će veću mogućnost zaraze nego programer upravo zbog prirode vlastitog zanimanja koji nalaže visoku ili nisku količinu fizičkih kontakata. Sličan odnos vrijedi i za trenutno stanje u državi i stanje na lokalnoj razini. Mogućnost zaraze je veća što više ima slučajeva zaraze u blizini, u ovom slučaju na razini županije.

2.3. Stanje u području i prikaz postojećih rješenja

Računalna rješenja u borbi protiv bolesti COVID-19 obuhvaćaju korištenje raznih tehnologija, postupaka i izvora. Jedna od često korištenih tehnologija je *BLE* ili niskoenergetski bluetooth (eng. *Bluetooth Low Energy*). To je bežična komunikacijska tehnologija koja se može koristiti na maloj udaljenosti između pametnih uređaja u svrhu komunikacije, što može biti korisno za određivanje fizičkih kontakata između ljudi, jer većina nosi svoj pametni telefon sa sobom. Neki od korištenih postupaka obuhvaćaju postupak interakcije s državnom bazom podataka testova na COVID-19. Prilikom dobivanja pozitivnog testa na COVID-19 može se dobiti i jedinstveni kod, koji služi za unos u aplikaciji. Unos jedinstvenog koda u aplikaciju jamči pozitivni test korisnika, kojim nakon čega se obavještavaju ljudi koji su bili prethodno u kontaktu sa zaraženim korisnikom, putem prethodne *BLE* tehnologije. Osim navedenih tehnologija i postupaka, koriste se sučelja za programiranje aplikacija (eng. *API*). *API* predstavlja alat za upravljanje određenim uslugama ili resursima kojima inače upravlja neki drugi sustav ili drugi program. Najčešće se koristi za ažurno dohvaćanje podataka s interneta, što bi u slučaju programskog rješenja za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2 značilo dohvaćanje dnevnog broja novozaraženih, broj ozdravljenih i broj preminulih osoba.

2.4. Postojeći sustavi i aplikacije

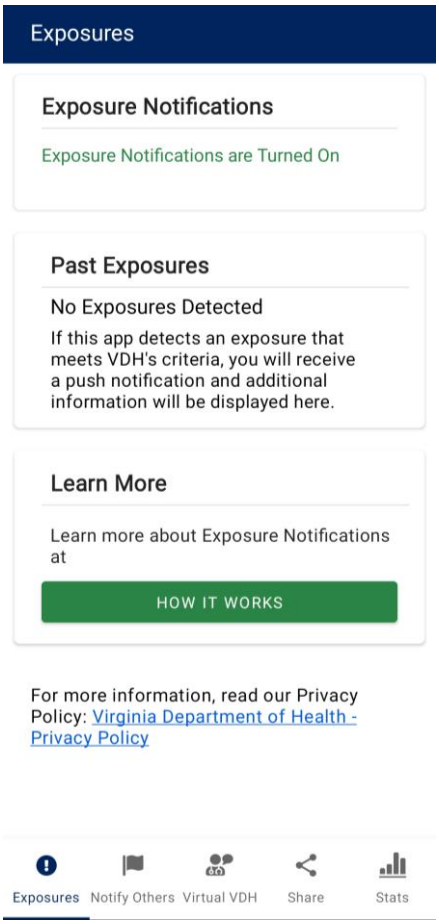
U ovom potpoglavlju analizirat će se tri postojeća programska rješenja u borbi protiv bolesti COVID-19. Programska rješenja *COVIDWISE*, *Stop COVID-19,* i *COVID-19 Event Risk Assessment Planning Tool* jedni su od najpopularnijih aktualnih rješenja.

2.4.1. COVIDWISE

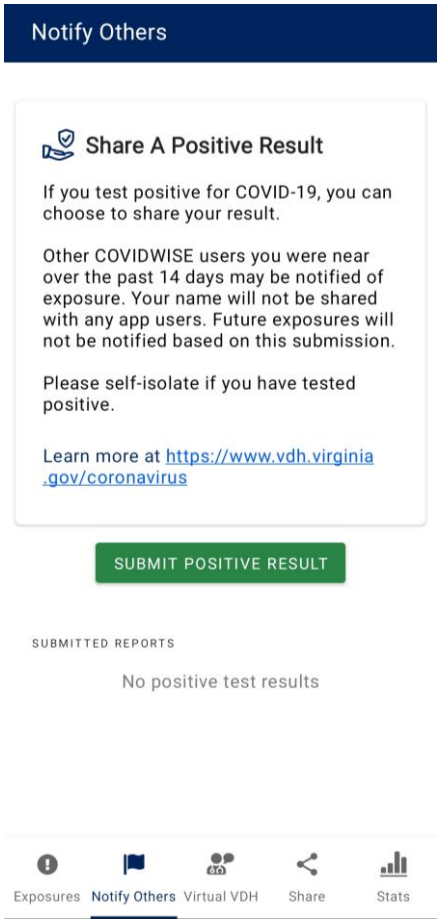
COVIDWISE je aplikacija tvrtke *Virginia's Department of Health (VDH)* iz savezne države Virginije, koja radi na principu odašiljanja signala između pametnih uređaja [6]. Ako netko unutar aplikacije prijavi pozitivnost na COVID-19, aplikacija signal aplikacije traži korisnike koji su dijelili taj signal. Signali imaju svojevrsnu vremensku oznaku i aplikacija na temelju snage signala procjenjuje koliko su dva uređaja bila blizu. Ako je vremenski period signala trajao barem 15 minuta i ako je procijenjena udaljenost između uređaja iznosila najviše 1.8 metara, korisnici dobivaju obavijest o mogućem izlaganju COVID-19. Aplikacija radi bez dohvaćanja lokacije i bez osobnih imena. Koristi se *BLE* radni okvir koji radi u pozadini čak i ako je aplikacija ugašena. Aplikacija ne troši bateriju više nego druge aplikacije koje za svoj rad zahtijevaju *Bluetooth* ili koje su konstanto upaljene ili otvorene. Aplikacija se brine za privatnost na način da bez imena i dohvaćanja lokacije emitira i prima signale okolnih korisnika aplikacije. Laboratorijski rezultati osoba koje su testirane pozitivno na COVID-19 se šalju u *VDH*, što nije povezano sa samom aplikacijom. Aplikacija omogućava anonimno objavljivanje pozitivnih testova na COVID-19. Ukoliko *VDH* zaprimi bilo koji pozitivni nalaz na COVID-19 koji je povezan s odgovarajućim brojem mobilnog uređaja, automatski se šalje poruka korisniku, koja pruža brze informacije i savjetuje korisnika da ostane kod kuće i da se udalji od drugih ljudi. Također, pozitivni korisnici mogu zatražiti osmeroznamenkasti verifikacijski kod na *COVIDWISE verification portal,* no pri tome je potrebno unijeti prezime, datum rođenja i broj mobitela koji je korisnik unio prilikom testa na COVID-19. Osmeroznamenkasti verifikacijski kod može se koristiti u svrhu anonimne prijave pozitivnog testa aplikaciji. Ta radnja sprječava lažne dojave pozitivnih rezultata, koje bi inače mogle stvoriti laže obavijesti o izloženosti virusu. COVIDWISE aplikacija je razvijena u suradnji s *SpringML* koristeći niskoenergetski *Bluetooth API* radni okvir stvoren jedinstvenom kolaboracijom Apple-a i Google-a.

Početni zaslon aplikacije prikazuje se prilikom prvog pokretanja aplikacije, te se na njemu mogu naći dva gumba. Prvi nudi objašnjenje na koji način aplikacija radi, drugi vodi prema zahtjevu za omogućavanje slanja obavijesti korisniku. Nakon pritiska na prvi gumb pojavljuju se informacije i upute za korištenje aplikacije.

Pritiskom na ikonu *X* u gornjem desnom uglu, korisnik se vraća na početni zaslon aplikacije. Nakon pritiska na gumb koji traži dopuštenje pojavljuje se dijalog koji sadrži obrazloženje zašto aplikacije treba pristup *Bluetooth*-u i ukratko se objašnjava način rada aplikacije. Nakon pritiska na gumb *Done* prikazuje se početni zaslon aplikacije koji se prikazuje pri svakom idućem ulazu u aplikaciju, ako je ispunjen uvjet dozvole prikazivanja obavijesti. U prvom dijelu aplikacije korisniku je prikazan popis prethodnih izlaganja kao na slici 2.1, ako je do istih došlo od omogućavanja korištenja *Bluetooth*-a i omogućavanja obavijesti. Pritiskom na tipku *How it works* korisniku se prikazuju zasloni s uputama i dodatnim informacijama. U drugom dijelu prikazanom na slici 2.2, korisnik može prijaviti vlastiti pozitivan rezultat testa na COVID-19, nakon čega će signali koji se uspostave s istim korisnikom obavijestiti druge korisnike o izlaganju COVID-19. Također, prikazana je povijest prijavljenih testova.



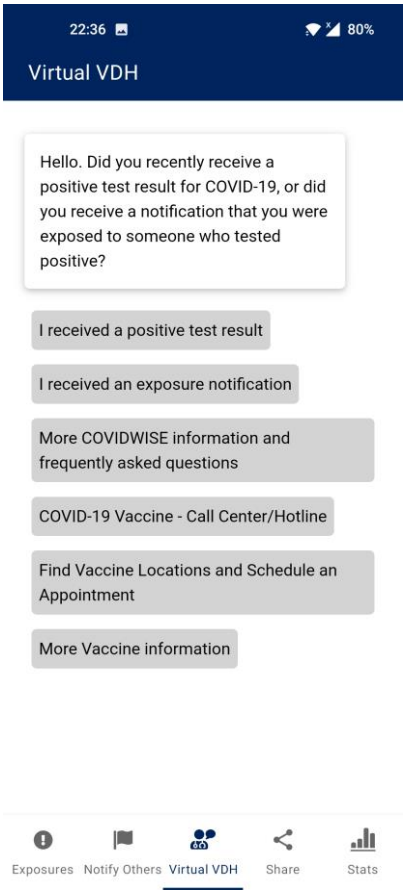
Slika 2.1 Izlaganja COVIDWISE aplikacije



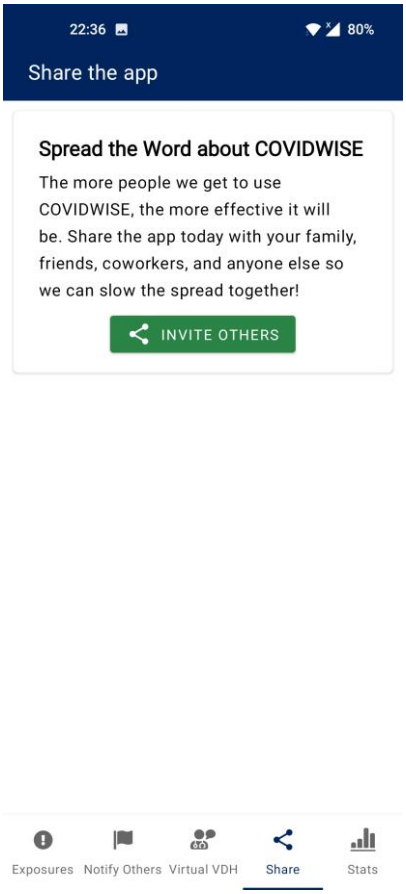
Slika 2.2 Obavijest drugima COVIDWISE aplikacijom

Na slikama 2.3 i 2.4 prikazani su zasloni trećeg i četvrtog dijela aplikacije. Treći dio predstavlja mogućnost kontaktiranja virtualnog pomagača tako da korisnik odabere jedan od predviđenih

zahtjeva. Na temelju odabranog zahtjeva nastavlja se virtualni razgovor. Četvrti dio služi za dijeljenje poveznice za preuzimanje aplikacije. Ovaj dio aplikacije je potreban jer funkcionalnost aplikacije doseže veći stupanj korisnosti većim brojem korisnika.

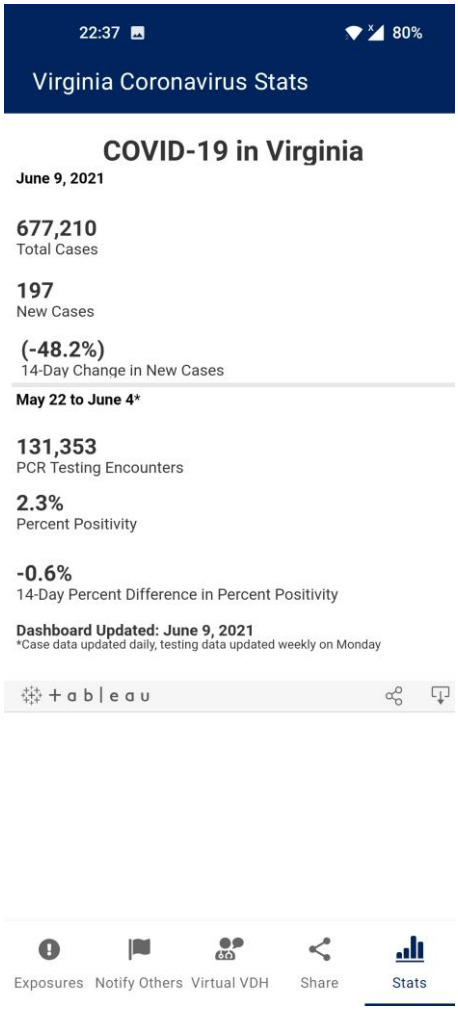


Slika 2.3 COVIDWISE Virtualni kontakt s vlastima



Slika 2.4 COVIDWISE Podijeli

Posljednji dio aplikacije, prikazan na slici 2.5, uvid je u statistiku o trenutnoj slici COVID-19 u saveznoj državi Virginiji. Prikazan je ukupni broj slučajeva, broj novih slučajeva, postotni prikaz promjene novih slučajeva u odnosu na prethodnih 14 dana. U drugom odjeljku vidljiv je broj testiranja u periodu od 14 dana, postotni prikaz pozitivnih slučajeva i postotni prikaz promjene postotka pozitivnih slučajeva u prethodnih 14 dana. Ovaj dio aplikacije bitan je za ovaj završni rad jer će aplikacija koja će se opisivati u ovom završnom radu sadržavati podatke koje će prikazivati na sličan način.



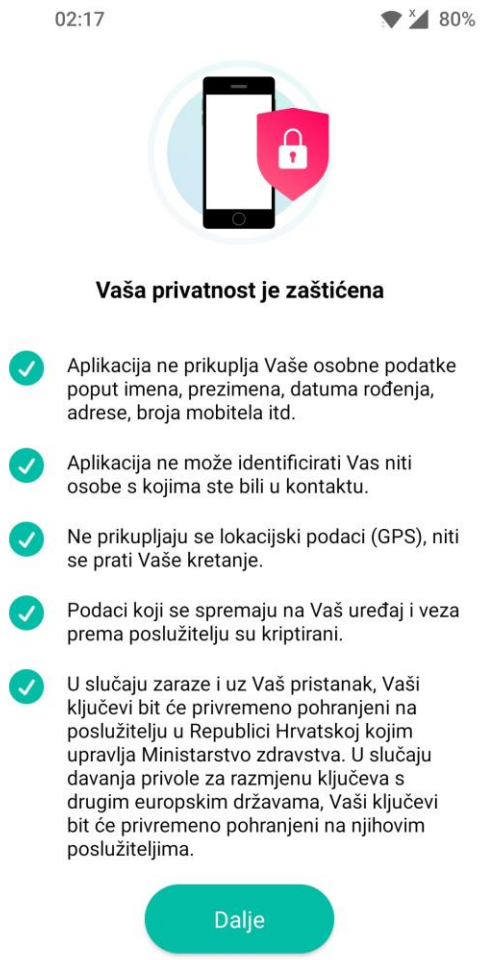
Slika 2.5 Prikaz statistike u COVIDWISE aplikaciji

2.4.2. Stop COVID-19

Mobilna Android aplikacija *Stop COVID-19* radi na identičnom principu kao aplikacija COVIDWISE [6]. Aplikaciju je objavilo Ministarstvo zdravstva Republike Hrvatske s ciljem obavještavanja korisnika od izloženosti COVID-19 [7].

Pri prvom pokretanju aplikacije prikazuje se zaslon sa slike 2.6. Korisnik odabire jezik na kojemu će sučelja biti prikazana tijekom korištenja. Pritiskom na gumb *Krenite* pojavljuje se zaslon sa slike 2.7 na kojemu se korisnik obavještava o zaštićenosti privatnosti tijekom korištenja.

Slika 2.6 Stop COVID-19 početni zaslon



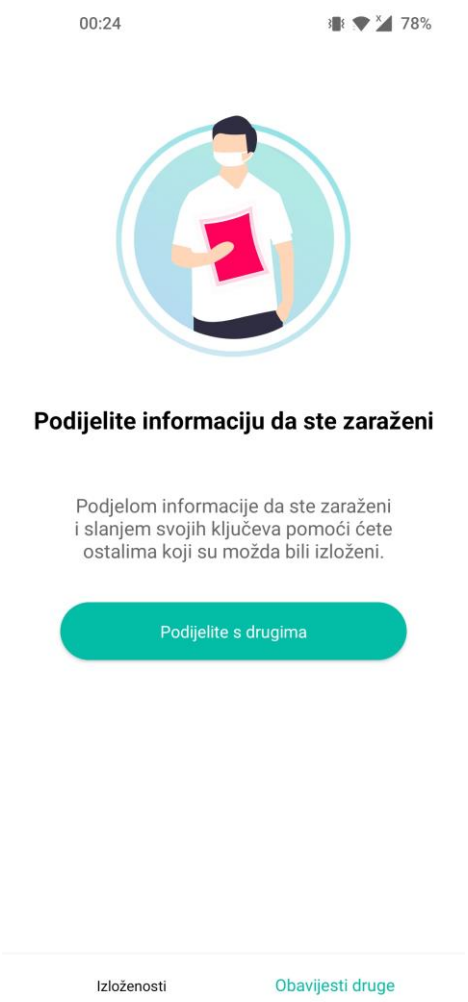
Slika 2.7 Stop COVID-19 obavijest korisniku

Dok korisnik ne uključi bilježenje kontakata, prikazuje se zaslon sa slike 2.8. Nakon uključivanja vidljiv je zaslon kao na slici 2.9, čime je omogućena funkcionalnost aplikacije. Uključivanje i isključivanje kontakata svojstvo je prvog dijela aplikacije, dok drugi dio služi za obavještavanje drugih korisnika o mogućoj izloženosti COVID-19.

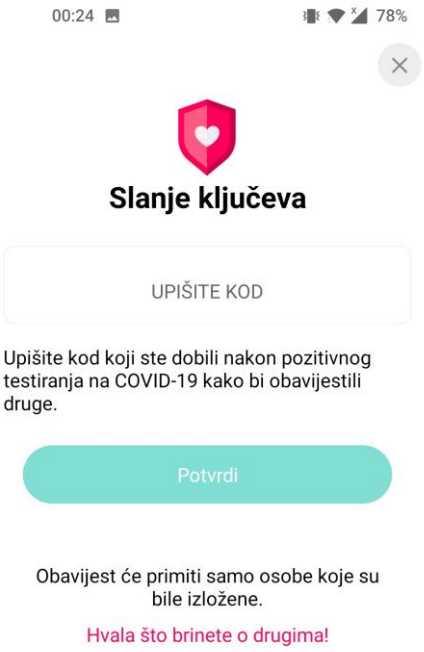
Slika 2.8 Stop COVID-19 isključen kontakt

Slika 2.9 Stop COVID-19 uključen kontakt

Pritiskom na *Obavijesti druge* korisniku se prikazuje zaslon sa slike 2.10 na kojemu se odvija preusmjeravanje korisnika na idući zaslon sa slike 2.11. Ovaj dio aplikacije omogućava korisniku upisivanje koda dobivenog nakon pozitivnog testa na COVID-19, nakon čega će korisnici koji budi u blizini dobiti obavijest o mogućoj izloženosti COVID-19. Upisivanjem koda korisnik aplikacijom dojavljuje rezultat svog negativnog testa na bolest uzrokovanu virusom SARS-CoV-2 na sličan način kao u [8].



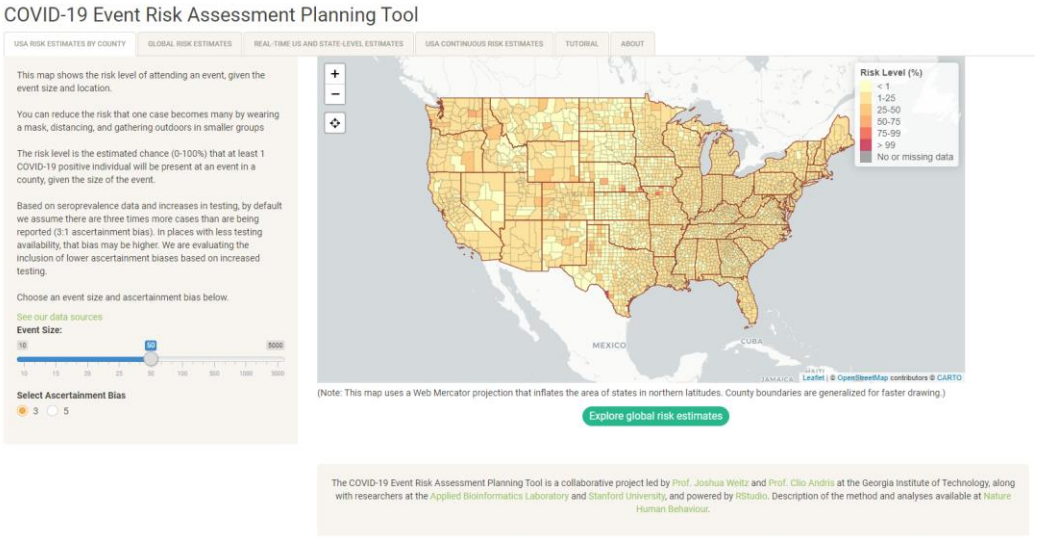
Slika 2.10 Stop COVID-19 obavijesti druge



Slika 2.11 Stop COVID-19 slanje ključeva

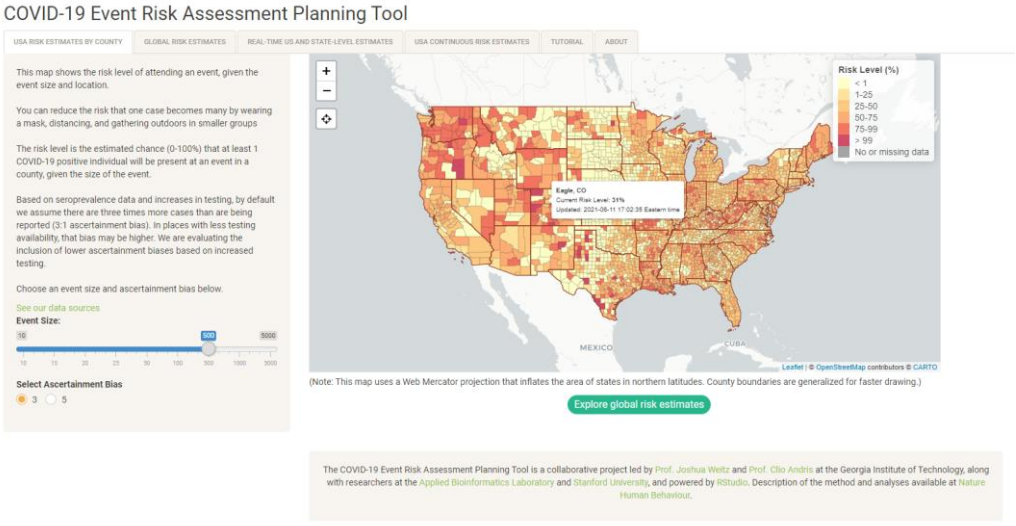
2.4.3. COVID-19 Event Risk Assessment Planning Tool

Osim mobilnih rješenja, postoji i rješenje u obliku mrežne stranice pod nazivom *COVID-19 Event Risk Assessment Planning Tool*. Web stranica pruža interaktivni kontekst u svrhu procjene rizika izlaganja jednog ili više pojedinaca bolesti COVID-19 [9]. Početni zaslon web stranice prikazan je na slici 2.12.



Slika 2.12 Event Risk Assessment Planning Tool zaslon

Ovaj alat radi na način povećavanja rizika zaraze virusom SARS-CoV-2 povećavanjem broja ljudi na potencijalnom događaju. Na krajnju procjenu utječe i trenutno stanje broja zaraženih u odabranom dijelu odabrane savezne države. Osim Sjedinjenih Američkih Država, ovo web rješenje radi i na području određenih europskih zemalja koje imaju dostupan *API*, a to su: Ujedinjeno Kraljevstvo, Italija, Švicarska, Austrija, Francuska, Češka, Irska, Švedska, Španjolska i Danska. Prema slici 2.13, prilikom povećanja veličine planiranog događaja, karta SAD-a se ažurira s odgovarajućim podacima. Budući da događaj s više uzvanika nosi veći rizik mogućnosti zaraze, mnogi dijelovi unutar saveznih država poprimaju tamniju boju koja po priloženoj legendi znači veću mogućnost zaraze.



Slika 2.13 Event Risk Assessment Planning Tool nakon povećane procjena

3. MODEL I ARHITEKTURA APLIKACIJE ZA VIŠEKRITERIJSKU PROCJENU IZLOŽENOSTI ZARAZI VIRUSOM SARS-COV-2

3.1. Potrebni postupci i programski pristupi za ostvarenje aplikacije

Ovim potpoglavljem opisani su potrebni postupci i programski pristupi za ostvarenje aplikacije, kao što je postupak kreiranja profila, dohvaćanja novih slučajeva COVID-19, dohvaćanja lokacije, prikaz posjećenih mjesta i prikaz procjene izloženosti zarazi virusom SARS-CoV-2.

3.1.1. Postupak kreiranje profila

Pri prvom pokretanju aplikacije za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2 od korisnika se zahtjeva svojevrsno kreiranje profila, kojom će se zabilježiti bitni podaci korisnika za rad aplikacije. Od korisnika se zahtjeva unos imena, prezimena, spola, datuma rođenja, te izjava korisnika o konzumaciji cigareta i lijekova koji narušavaju imunitet. Nakon unosa navedenih parametra, od korisnika se traži da iz padajućeg izbornika odabere svoj posao. Budući da su korisnici bili zabrinuti zbog sličnih aplikacija koje pohranjuju podatke [10], aplikacija podatke pohranjuje u lokalnu bazu podataka.

3.1.2. Postupak dohvaćanja lokacije

Kako bi aplikacija radila svojim potpunim potencijalom, od korisnika se traži dozvola dohvaćanja lokacije. Korisnik tada ima tri opcije: prihvaćanje stalnog pristupa lokaciji od strane aplikacije, prihvaćanje pristupa lokaciji od strane aplikacije samo dok je aplikacija upaljena, odbijanje pristupa lokaciji. Za kvalitetan rad aplikacije potrebno je odabrati prvu opciju, tj. prihvaćanje stalnog pristupa lokaciji. Dohvaćanje lokacije je bitno iz razloga što je jedan od čimbenika unutar algoritma za izračunavanje procjene izloženosti zarazi virusom SARS-CoV-2 broj posjećenih lokacija unutar jednog dana. Lokacija se dohvaća isključivo putem GPS-a, jer procjena lokacije koristeći mobilne mreže nije dovoljno precizna za predefinirani način rada aplikacije.

3.1.3. Postupak dohvaćanja novih slučajeva COVID-19

Dohvaćanje novih slučajeva COVID-19 odvija se koristeći strojno čitljive podatke (*API*) [11], tako da se posebno dohvaćaju podaci o novim slučajevima u cijeloj Republici Hrvatskoj, a posebno za sve županije. Dohvaćanje broja novih slučajeva važno je zbog implementacije algoritma procjene zaraze.

3.1.4. Prikaz posjećenih mjesta

Ova mogućnost aplikacije bazira se na 3.1.2, putem čega se posjećene lokacije spremaju u lokalnu bazu podataka. Prilikom zahtjeva za prikazom posjećenih mjesta, na prikladan način prikazuju se sve posjećene lokacije unutar 24 ili 12 sati, ovisno o odabiru korisnika.

3.1.5. Prikaz procjene izloženosti zarazi virusom SARS-CoV-2

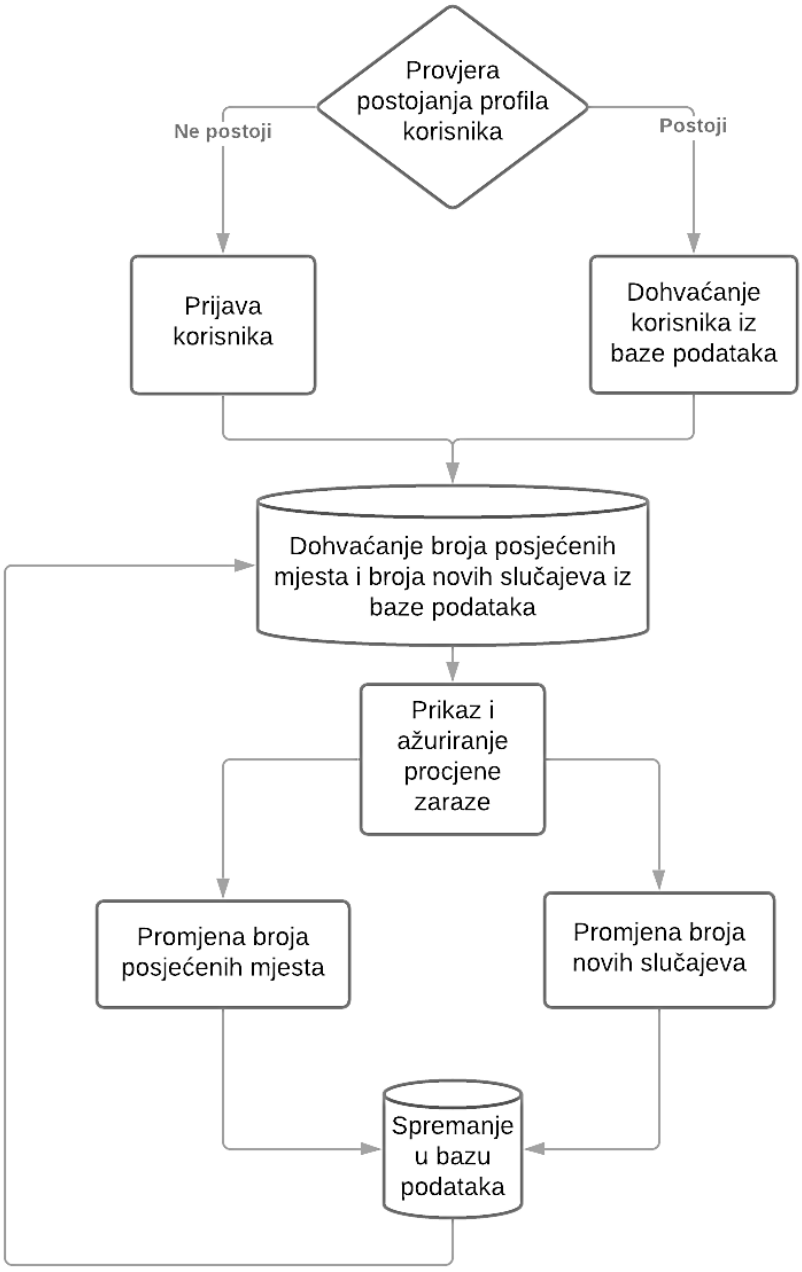
Budući da je prikaz procjene izloženosti zarazi virusom SARS-CoV-2 ključna značajka aplikacije za procjenu izloženosti zarazi virusom SARS-CoV-2, nalazi se na početnom zaslonu uz popratnu poruku korisniku. Postotak procjene izloženosti zarazi virusom SARS-CoV-2 iste je boje kao i popratna poruka korisniku, koja može biti zelena, narančasta i crvena. Boje se mijenjaju ovisno o dugoročnoj procjeni izloženosti zarazi virusom SARS-CoV-2 u periodu od sedam dana.

3.2. Idejno rješenje i arhitektura mobilne aplikacije

Ovo potpoglavlje sastoji se od obrazloženja idejnog rješenja i arhitekture mobilne aplikacije.

3.2.1. Idejno rješenje mobilne aplikacije

Slika 3.1 prikazuje dijagram toka [12] korištenja aplikacije za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2. Prilikom pokretanja aplikacije, unutar lokalne baze podataka potražuje se postojeći korisnički profil. Ako korisnički profil postoji, iz baze podataka dohvaćaju se podaci postojećeg korisnika. Ako traženi korisnički profil ne postoji, od korisnika se traži unos osobnih podataka, tj. kreiranje novog korisničkog profila koji se zatim pohranjuje u bazu podatka. Nakon dohvaćanja podataka iz baze podataka ili kreiranja novog profila, dohvaćaju se podaci korisni za rad aplikacije: broj novih slučajeva zaraženih virusom SARS-CoV-2 i broj posjećenih lokacija iz baze podataka. Neposredno zatim odvija se izračun procjene zaraze i ažurira se prikaz procjene zaraze. Aplikacije ostaje na ovom dijelu tijeka rada sve dok ne dođe do promjene broja posjećenih mjesta ili promjene broja novih slučajeva zaraze COVID-19. Kada dođe do takvih promjena, novi broj posjećenih mjesta ili novi broj novih slučajeva zaraze sprema se u lokalnu bazu podataka. Nakon spremanja u bazu podataka, podaci se iznova dohvaćaju iz baze podataka i računa se nova procjena izloženosti zarazi virusom SARS-CoV-2, nakon čega se ažurira prikaz procjene izloženosti zarazi virusom SARS-CoV-2. Važno je za napomenuti kako se do lokacije dolazi putem GPS-a, kojemu korisnik treba dozvoliti pristup. Do brojeva novih slučajeva zaraze virusom SARS-CoV-2 dolazi se putem otvorenih strojno čitljivih podataka Hrvatskog zavoda za javno zdravstvo.



Slika 3.1 Prikaz idejnog rješenja tijeka aplikacije

3.2.2. Arhitektura mobilne aplikacije

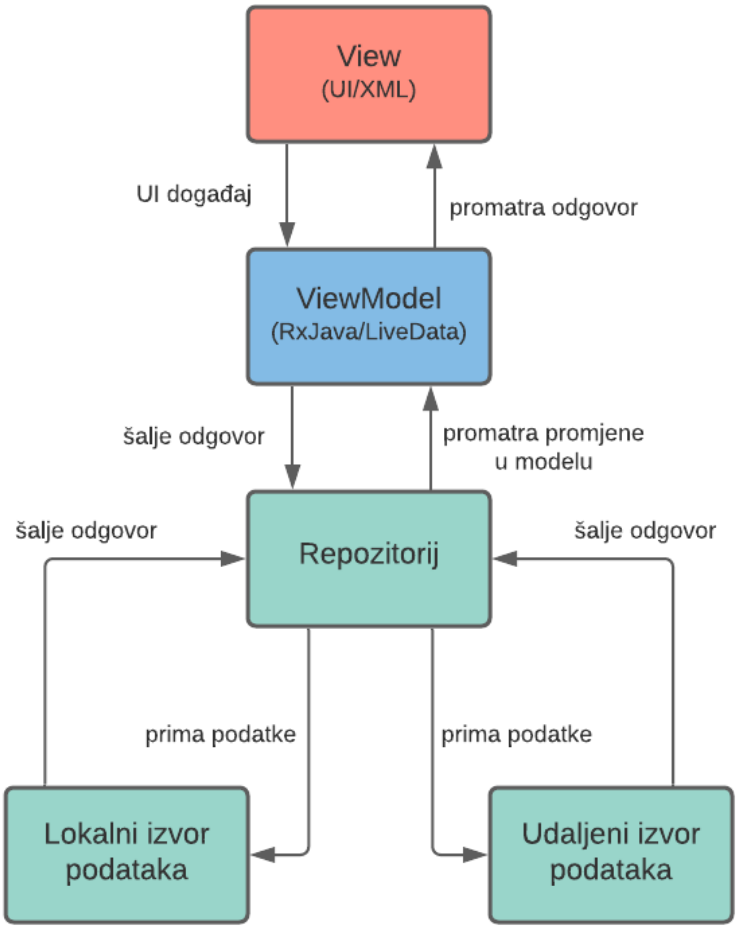
Iako se uvelike koristi arhitektonski obrazac *Model View Controller* (MVC) [13] u kojemu se *Activity* i *Fragement* koriste kao upravljači i modeli za podatke i njihovu dosljednost, za ovu aplikaciju je korištena arhitektura MVVM → *Model – View – ViewModel*. Radi se o pristupu koji služi za apstrakciju stanja i ponašanja od *View* komponente, što omogućuje odvajanje korisničkog sučelja od logike aplikacije [14]. Izvodi se uvođenjem *ViewModela*, čija je odgovornost

dohvaćanje objekata podataka iz modela i nošenje s logikom aplikacije povezane s komponentom *View* [15].

Ovaj pristup sastoji se od tri glavne sastavnice, od kojih svaka ima svoju zasebnu ulogu:

1. *Model* – sadrži logiku aplikacije i vjerodostojnosti podataka
2. *View* – definira strukturu, raspored i izgled zaslona
3. *ViewModel* – ponaša se kao poveznica između *Viewa* i *Modela* koja se bavi logikom povezanom sa *Viewom*.

Na slici 3.2 može se vidjeti način rada arhitekture MVVM [16]. Prilikom događaja na korisničkom sučelju poziva se određena metoda u *ViewModelu*, koja zatim šalje zahtjev modelu. U ovom slučaju model se sastoji od repozitorija, lokalnog izvora podataka i udaljenog izvora podataka.



Slika 3.2 MVVM arhitektura

Repozitorij vrši komunikaciju s lokalnim i udaljenim izvorom podataka, nakon čega *ViewModel* promatra promjenu vrijednosti koju vraća repozitorij. *View* promatra promjenu koju vraća repozitorij, prilikom čega se odvija određena radnja nad *View* komponentom.

U odnosu na ostale arhitekture mobilnih aplikacija, nema uskog povezivanja komponenti *View* i *ViewModel*, nema sučelja između istih komponenti, kod se lako testira i kod se bazira na događajima. Neki od nedostataka su veličina koda i potreba za stvaranjem promatranog objekta za svaku komponentu korisničkog sučelja.

3.3. Postupci za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2

Postupci za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2 koji će biti opisani ovim potpoglavljem su dohvaćanje dnevnog broja oboljelih od bolesti COVID-19, dohvaćanje lokacije i izračun procjene izloženosti zarazi virusom SARS-CoV-2.

3.3.1. Dohvaćanje dnevnog broja oboljelih od bolesti COVID-19

Dohvaćanje dnevnog broja oboljelih od bolesti COVID-19 odvija se putem *Retrofit* mrežne biblioteke koja se koristi za Android i Javu [17]. Odabrana biblioteka izvršava procese primanja, slanja i stvaranja *HTTP* zahtjeva i odgovora, mijenja IP adrese ako dođe do greške web servisa, hvata odgovore kako bi se izbjeglo slanje istih zahtjeva. Osim toga *Retrofit* pokušava sam riješiti problem prije slanja greške i rušenja aplikacije. Između ostalog, pruža podršku i za sinkrone i za asinkrone mrežne zahtjeve. Kako bi se biblioteka implementirala, potrebno je upisati linije koda u *build.gradle* datoteku unutar Android projekta i dozvoliti pristup internetu u *Android Manifestu* kao u programskim kodovima 3.3 i 3.4.

```
def retrofitVersion = "2.6.0"
implementation "com.squareup.retrofit2:retrofit:$retrofitVersion"
implementation "com.squareup.retrofit2:converter-gson:$retrofitVersion"
```

Programski kod 3.1 Implementacija biblioteke *Retrofit*

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

Programski kod 3.2 Dopuštenje pristupa aplikacije internetu

Nakon implementacije i davanja dozvole pristupa aplikacije internetu, aplikacija treba stvoriti instancu *Retrofita*, odrediti odredišne točke i kreirati model, što je prezentirano u potpoglavlju 4.3.2.

3.3.2. Dohvaćanje lokacije

Lokacija se dohvaća putem pružatelja lokacije (eng. *Location Provider).* Može se koristiti GPS pružatelj lokacije ili mrežni pružatelj lokacije [18]. Razlika između navedena dva pružatelja lokacije je u tome što GPS pružatelj brže troši bateriju, može biti spor prilikom rada u zatvorenim prostorima, no daje točnu lokaciju uređaja. Mrežni pružatelj lokacije brži je od GPS pružatelja, ali ovisi o najbližem ćelijskom tornju, čiju lokaciju dohvaća. Za dohvaćanje je implementirana Google-ova biblioteka za lokaciju i deklarirana su dopuštenja za korištenje grube i fine lokacije. Gruba lokacije koristi se za mrežne pružatelje lokacije, a fina lokacije koristi se GPS mrežnim poslužiteljima. Aplikaciji se daje dopuštenje pristupa lokaciji programskim kodovima 3.5 i 3.6.

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

Programski kod 3.3 Dopuštenja za dohvaćanje lokacije

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
```

Programski kod 3.4 Implementacija Google-ove biblioteke za dohvaćanje lokacije

Nakon navedenih koraka potrebno je instancirati *LocationManager* kao instancu usluge za lokacije. Tada je moguće dohvaćati lokaciju po potrebi ili prilikom promjene lokacije putem klase *LocationListener*.

3.3.3. Izračun procjene izloženosti zarazi virusom SARS-CoV-2

Nakon dohvaćanja lokacije i dnevnog broja oboljelih od bolesti COVID-19, poziva se metoda izračuna procjene izloženosti zarazi virusom SARS-CoV-2. Metoda radi na način da svim čimbenicima (broj posjećenih lokacija, broj oboljelih, dob, spol, pušenje, korištenje lijekova štetnih za imunitet i posao) daje određena težina. Nizom predefiniranih računskih operacija dolazi se do parametra *Ex* – procjene mogućnosti zaraze virusom SARS-CoV-2. Ukoliko je broj posjećenih lokacija jednak 1, smatra se da korisnik nije napustio stambeni prostor, što znači da je

mogućnost zaraze virusom SARS-CoV-2 minimalna. Postupak izračuna kojim se dobiva vrijednost *Ex* slijedi u izrazu 3.1.

$$Ex = \frac{(spol + dob + pušenje + lijekovi + posao + \frac{zaraženi}{10}) \cdot lokacije}{\frac{200}{3}} \tag{3.1}$$

gdje je:

- spol – određena vrijednost ovisno o muškom, ženskom ili neodređenom spolu
- dob – određena vrijednost ovisno o dobi, kategorizira se u intervalima od 15 godina
- pušenje – vrijednost koja se dodaje ukoliko korisnik konzumira duhanske proizvode
- lijekovi – broj koji označuje koristi li korisnik lijekove koji štete imunitetu
- posao – težina određena poslu kojom se korisnik bavi
- zaraženi – broj slučajeva određenog dana određene županije
- lokacije – broj posjećenih lokacija od strane korisnika određenog dana

Do procjene mogućnosti zaraze virusom SARS-CoV-2 se dolazi metodom logističke regresije [19]. Logistička regresija se uglavnom koristi za procjenu mogućnosti određenog događaja, gdje događaj ovisi o ulaznim parametrima, u ovom slučaju o *Ex*. U odnosu na ishodišnu formulu logističke regresije, eksponentu je dodana konstanta u vrijednosti -3.5, što empirijski daje kvalitetne i očekivane rezultate. Procjena mogućnosti zaraze virusom SARS-CoV-2 računa se na sljedeći način prema izrazu 3.2:

$$\partial(Ex) = \frac{e^{-(-3.5 + Ex)}}{e^{-(-3.5 + Ex)} + 1} = \frac{1}{1 + e^{-(-3.5 + Ex)}} \tag{3.2}$$

Izračun procjene mogućnosti zaraze virusom SARS-CoV-2 unutar koda prikazan je u potpoglavlju 4.3.3.

4. PROGRAMSKO RJEŠENJE MOBILNE APLIKACIJE ZA VIŠEKRITERIJSKU PROCJENU IZLOŽENOSTI ZARAZI VIRUSOM SARS-COV-2

4.1. Korištene programske tehnologije, jezici i razvojna okruženja

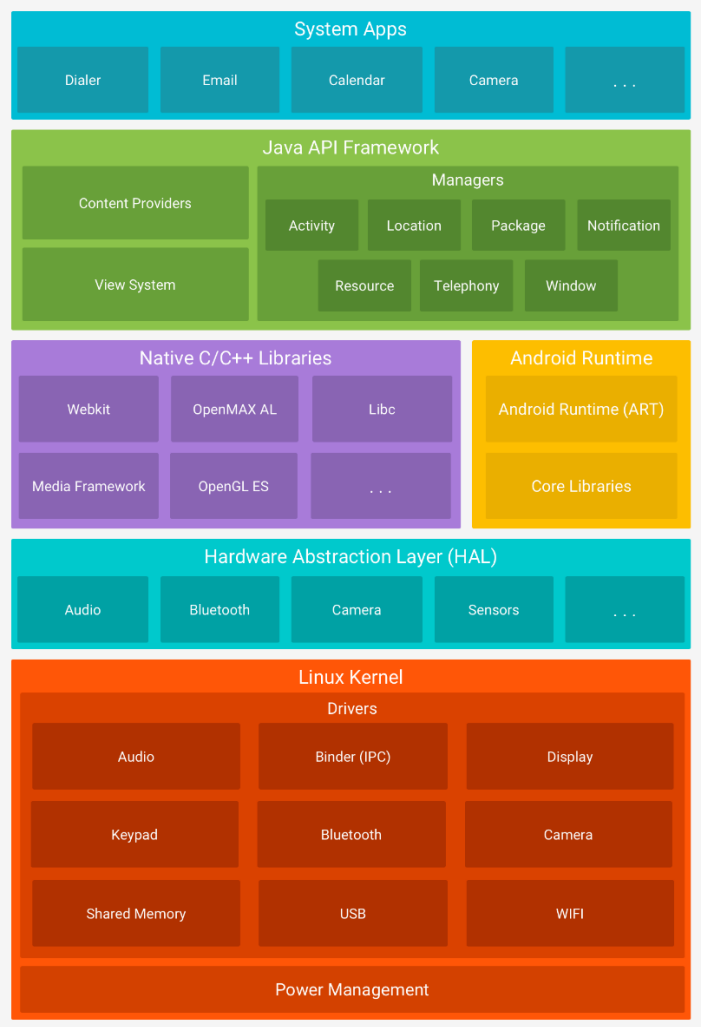
Ovim poglavljem opisane su korištene programske tehnologije, jezici i razvojna okruženja potrebna za razvoj mobilne aplikacije za višekriterijsku procjenu izloženosti zarazi virusom SARS-COV-2.

4.1.1. Operacijski sustav Android

Android je operacijski sustav prvenstveno namijenjen za korištenje na mobilnim uređajima sa zaslonom na dodir [20]. Dvije godine nakon što je Google kupio firmu Android 2005. godine, najavljen je prvi mobilni uređaj koji pokreće Android sustav na tržištu. Šest godina kasnije, 2011., Android postaje najprodavaniji mobilni operacijski sustav, a 2013. postaje i najprodavaniji operacijski sustav za tablete [21].

Budući da je jedna od značajki Android sustava otvorenost koda, aplikacije su u mogućnosti same pokretati druge aplikacije ili izvršavati radnje kao slanje poruka, uspostavljanja poziva ili fotografiranja. Prema slici 4.1 preuzetoj iz [22], arhitektura operacijskog sustava Android može se podijeliti na više razina. Najniža razina i temelj Android platforme je jezgra Linuxa, na koju se sustav oslanja prilikom pojavljivanja potrebe za nužnim funkcionalnostima kao što je *threading* i upravljanje memorijom niske razine. Razinu iznad Linux jezgre nalazi se *Hardware Abstraction Layer* (*HAL*), koji višim razinama sustava pruža sučelja, čime je moguće iskoristiti određenu komponentu uređaja, npr. kameru ili *bluetooth*. Kada radni okvir određene komponente želi pristupiti komponenti, Android sustav učitava biblioteku za određenu hardversku komponentu. Nakon *HAL*, slijede 2 paralelne razine Android arhitekture: *Android Runtime* i nativne C/C++ biblioteke. *ART* (*Android Runtime*) je napisan u svrhu pokretanja više virtualnih strojeva na uređajima niske memorije, s namjerom pokretanja *DEX* (eng. *Dalvik Executable format*) datoteka. Postoje razvojni alati kao što je *d8* [23], vrše prevođenje nad *Java* izvorima iz čega proizlazi *DEX* kod, koji može biti pokrenut na Android platformi. Glavne značajke *ART*-a su optimizirano prikupljanje smeća, pretvaranje *DEX* formata u praktičniji strojni kod, bolja podrška uklanjanja grešaka, detaljna dijagnostika grešaka i izvještavanje o rušenju. Nativne C/C++ biblioteke služe za omogućavanje komunikacije *Java API* radnog okvira sa funkcionalnostima ovih nativnih biblioteka, npr. dodavanje podrške za crtanje i manipuliranje 2D i 3D grafičkih objekata u

aplikaciji. Sljedeća razina Android arhitekture je *Java API* radni okvir, koji omogućava sve značajke sustava Android putem *API*-ja. *API* je sučelje koje čini gradivnu jedinicu Android aplikacije pojednostavljivanjem jezgre, komponenti modularnog sustava i servisa. Najvišu razinu arhitekture sustava Android čine aplikacije sustava. Aplikacije sustava su aplikacije koje funkcioniraju i kao aplikacija namijenjena za korisnika, ali i kao pružatelj ključnih mogućnosti kojima mogu pristupiti razvojni programeri. Ukoliko aplikacija treba poslati SMS poruku, nije potrebno izgrađivati tu funkcionalnost, jer je moguće iskoristiti postojeću SMS aplikaciju.



Slika 4.1 Građa operacijskog sustava Android, [19]

4.1.2. Programski jezik Java

Java je programski jezik visoke razine i kao programska platforma radi na milijardama uređaja, uključujući i prijenosna računala, mobilne uređaje, igraće konzole, medicinske uređaje i mnoge druge. Pravila i sintaksa *Java* jezika temelje se na onima *C* i *C++* jezika [24]. Velika prednost

razvoja programa u *Javi* je portabilnost. Kod koji je napisan na računalu jako je jednostavno prenijeti na mobilni uređaj. Kad je James Gosling 1991. godine osmislio Java programski jezik, primarni cilj bio je biti u mogućnosti „jednom napisati, pokrenuti bilo gdje“, tj. interoperabilnost između različitih uređaja. Ključni atribut *Jave* je skalabilnost platforme. Osim toga, bilo je potrebno da Java bude pouzdan programski jezik. Trebalo je smanjiti mogućnost kritičnih kvarova uzrokovanim programerskim greškama, zbog čega je predstavljeno objektno orijentirano programiranje. Uz pouzdanost, važan čimbenik programskog jezika je i sigurnost. Budući da je *Java* prvotno korištena za razmjenu podataka preko mreže između mobilnih uređaja, *Java* je izgrađena s visokim stupnjem sigurnosti.

4.1.3. Strukturni jezik XML

Prema [25], *XML* (eng. *Extensible Markup Language*) je strukturni jezik bez predefiniranih oznaka. Korisnik definira vlastite oznake (eng. *tagove*) dizajnirane isključivo korisnikovim potrebama. Ovo je iznimno koristan način za pohranu podataka u formatu koji se može spremati, pretraživati i dijeliti, i najvažnije, zato što je temeljni format *XML*-a standardiziran, *XML* se može dijeliti i koristiti na različitim sustavima i platformama, bilo lokalno ili preko interneta. Primatelj će uvijek moći uspješno analizirati podatke zbog standardizirane sintakse *XML*-a.

XML u razvoju Android aplikacija koristi se za definiranje izgleda sučelja aplikacije, od većih grupa komponenata do sitnih detalja. Korisničko sučelje aplikacije dizajnira se isto kao i kod web stranica – nizom ugniježđenih elemenata [26]. Svi elementi sučelja grade se koristeći hijerarhiju *View* i *ViewGroup* objekata, gdje su *View* objekti nešto što korisnik vidi, a *ViewGroup* objekti su objekti koji sadrže *View* objekte.

4.1.4. Integrirano razvojno okruženje Android Studio

Android Studio je službena integrirana razvojna okolina za razvoj Android aplikacija [27]. Osim programa za uređivanje koda i alata za razvoj, pruža niz drugih korisnih značajki kao što su brzi emulator, standardizirano okruženje u kojem se mogu razvijati aplikacije za Android uređaje, predlošci koda i *GitHub* integracija.

4.2. Programsko rješenje na strani korisnika

U ovom poglavlju opisuje se programsko rješenje na strani korisnika, pod što spada unos podataka o korisniku, prikaz zaslona s dnevnim brojevima slučajeva zaraze virusom SARS-CoV-2, prikaz posjećenih lokacija, prikaz zaslona s postavkama i prikaz početnog zaslona.

4.2.1. Unos podataka o korisniku

Pri prvom pokretanju aplikacije, provjerava se postoji li instanca baze podataka, nakon čega se stvara dohvaća postojeća instanca baze podataka ili se stvara nova instanca baze podataka, kao u programskom kodu 4.1.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    repo = Repository.getInstance(getApplicationContext());
    if(repo.isDBnull()){
        repo.initDB(getApplicationContext());
    }
    if(repo.getUserCount() != 0 ) {
            setContentView(R.layout.activity_main);
            initUI();
    }else{
        setContentView(R.layout.activity_welcome_screen);
        initWelcomeUI();
    }
}
```

```
private void initWelcomeUI() {
    btnContinue = findViewById(R.id.btnNastavi);
    btnContinue.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            deleteOldAndAddNewUser();
        }
    });
}
```

```
private void deleteOldAndAddNewUser(){
    Intent intent = new Intent( packageContext: MainActivity.this, ScrollingActivity.class);
    startActivityForResult(intent, REQUEST_CODE_SECOND_ACTIVITY);
}
```

Programski kod 4.1 Početna inicijalizacija u aktivnosti *MainActivity*

Koristeći dobivenu instancu, provjerava se postojanje profila korisnika unutar baze podataka. Ako profil postoji, aplikacija nastavlja s radom s postojećim podacima, prikazujući glavno korisničko sučelje i inicijalizirajući odgovarajuće elemente. Ako profil ne postoji, postavlja se određeno korisničko sučelje pod nazivom *activity_welcome_screen*, nakon čega se pokreće i metoda *initWelcomeUI()* koja inicijalizira elementa na istom sučelju. Svrha navedenog sučelja je navođenje korisnika na unos novog korisničkog profila, pritiskom na gumb, koji pokreće metodu *deleteOldAndAddNewUser()*. Dotična metoda pokreće drugu aktivnost, *ScrollingActivity,* pri čemu očekuje rezultat od navedene aktivnosti. Ukoliko se korisnik vrati iz nove aktivnosti

pritiskom na navigacijsku tipku za vraćanje nazad, dočekat će ga isti *activity_welcome_screen* jer *resultCode* neće biti jednak *RESULT_OK*. Od korisnika se očekuje da na *ScrollingActivity* aktivnosti unese svoje podatke koje su neophodne za rad aplikacije. Nakon unosa podataka i pritiska odgovarajućeg gumba, profil korisnika se sprema u bazu podataka, poziva se *onActivityResult()* metoda, koja omogućuje korisniku očekivan nastavak rada unutar aplikacije. Nakon uspješnog unosa podataka, poziva se *initUI()* metoda kao u programskom kodu 4.2, koja inicijalizira potrebne fragmente i navigacijsku traku koja se sastoji od 4 gumba za odgovarajuće fragmente.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == REQUEST_CODE_SECOND_ACTIVITY){
        if (resultCode == RESULT_OK) {
            setContentView(R.layout.activity_main);
            initUI();
        }
    }
}

private void initUI(){
    BottomNavigationView navView = findViewById(R.id.nav_view);
    AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
            R.id.fragment_home, R.id.fragment_covid, R.id.fragment_location, R.id.settingsFragment)
            .build();
    NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment);
    NavigationUI.setupActionBarWithNavController( activity: this, navController, appBarConfiguration);
    NavigationUI.setupWithNavController(navView, navController);
}
```

Programski kod 4.2 Inicijalizacija glavnog sučelja u aktivnosti *MainActivity*

U programskom kodu 4.3 može se vidjeti na koji način i što se sprema unutar profila korisnika u lokalnu bazu podataka. Prilikom pritiska na gumb *btnContinue*, dohvaća se ime, prezime, spol, datum rođenja, izjava o pušenju, izjava o korištenju lijekova koji narušavaju imunitet i posao. Dohvaćene vrijednosti privremeno se pohranjuju unutar lokalnih varijabli odgovarajućih tipova. Nakon dohvaćanja podataka, podaci se pohranjuju u lokalnu bazu podataka tako da se parametarskom konstruktoru predaju odgovarajuće lokalne varijable kao na slici. Ukoliko ne dođe do greške, aktivnost završava. Također, može se vidjeti dodana funkcionalnost koja omogućuje korisniku povratak na prethodnu aktivnost bez neželjene interferencije s lokalnom bazom podataka.

```
btnContinue.setOnClickListener(view -> {
    String ime = etIme.getText().toString();
    String prezime = etPrezime.getText().toString();
    char spol;
    if(rbMusko.isChecked()){ spol = 'M'; }
    else if(rbZensko.isChecked()){ spol = 'Z'; }
    else{ spol = 'X'; }
    String datum = dpDate.getDayOfMonth() +"/"+ (dpDate.getMonth()+1) +"/"+dpDate.getYear();
    boolean pusenje;
    pusenje = rbPusenjeDa.isChecked();
    boolean lijekovi;
    lijekovi = rbLijekoviDa.isChecked();
    int tempIndex = spinner.getSelectedItemPosition();
    String posao = getResources().getStringArray(R.array.poslovi_array)[tempIndex];
    repo = Repository.getInstance();
    insertedFlag = false;
    if (repo.isUserPopulated()) { repo.deleteUser(); }
    repo.insertUser(new User(ime, prezime, spol, datum, pusenje, lijekovi, posao));
    setResult(Activity.RESULT_OK);
    finish();
});
```

Programski kod 4.3 Stvaranje profila korisnika unutar aktivnosti *ScrollingActivity*

4.2.2. Prikaz zaslona s dnevnim brojevima slučajeva zaraze virusom SARS-CoV-2

Pritiskom na drugi gumb na navigacijskoj traci na dnu zaslona, aplikacija pokreće fragment koji se sastoji od detalja vezano za trenutno stanje slučajeva zaraženih virusom SARS-CoV-2, opisan programskim kodom 4.4. Nakon stvaranja fragmenta, inicijaliziraju se potrebni elementi stvorenog fragmenta.

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

    initUI(view);

    covidViewModel.getRepository().getShouldSetRefreshFalse().observe(getViewLifecycleOwner(),
            aBoolean -> {
        if(aBoolean){
            if(swipeRefreshLayout.isRefreshing()){
                swipeRefreshLayout.setRefreshing(false);
            }
        }
    });
    swipeRefreshLayout.setOnRefreshListener(() -> {
        if(getActivity() != null){
            ConnectivityManager cm = (ConnectivityManager) getActivity().getSystemService(
                    Context.CONNECTIVITY_SERVICE);
            NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
            isConnected = activeNetwork != null && activeNetwork.isConnectedOrConnecting();
        }
        if(isConnected) {
            covidViewModel.getRepository().getData();
        }else{
            swipeRefreshLayout.setRefreshing(false);
            Toast.makeText(getContext(), text: "Nemoguć pristup internetu", Toast.LENGTH_SHORT)
                    .show();
        }
    });

    covidViewModel.getRepository().getLatestCases().observe(getViewLifecycleOwner(), covidDB ->
    {
        if(covidDB != null){
            tvHrSlucajevi.setText(String.format(getString(R.string.slucajevi_u_hr), covidDB.hr_zarazeni));
            tvHrIzlijeceni.setText(String.format(getString(R.string.ozdravljeni_hr), covidDB.hr_izlijeceni));
            tvHrUmrli.setText(String.format(getString(R.string.umrli_hr), covidDB.hr_umrli));
            tvZupSlucajevi.setText(String.format(getString(R.string.zarazeni_obz), covidDB.zup_zarazeni));
            tvZupIzlijeceni.setText(String.format(getString(R.string.ozdravljeni_obz), covidDB.zup_izlijeceni));
            tvZupUmrli.setText(String.format(getString(R.string.umrli_obz), covidDB.zup_umrli));
        }
        swipeRefreshLayout.setRefreshing(false);
    });
```

Programski kod 4.4 Postavljanje promatranih vrijednosti unutar *CovidFragmenta*

Budući da je implementirana biblioteka *swipeRefreshLayout*, korisnik može osvježiti podatke o trenutnom broju slučajeva novozaraženih povlačenje prstom prema dolje, što je prepoznatljiva karakteristika aplikacija za društvene mreže poput Facebooka ili Instagrama. Nakon povlačenja prstom prema dolje, provjerava se povezanost uređaja na internet. Ukoliko je uređaj može pristupiti internetu, dohvaćaju se podaci o trenutnom stanju oboljelih od bolesti izazvanom virusom SARS-CoV-2. U protivnom, ispisuje se *Toast* poruka upućena korisniku, s odgovarajućom porukom. Pri promjeni dohvaćenih podataka, njihove vrijednosti postavljaju se u

odgovarajuće elemente. Podaci koji se dohvaćaju su dnevni slučajevi zaraze u Republici Hrvatskoj, dnevni broj izliječenih od bolesti izazvane virusom SARS-CoV-2, dnevni broj umrlih od bolesti COVID-19, dnevni slučajevi zaraze u odabranoj županiji, dnevni broj izliječenih u odabranoj županiji i dnevnih broj umrlih u određenoj županiji. U programskom kodu 4.5 prikazana je definicija putanje korisnika nakon pritiska gumba *Detalji*.

```
details.setOnClickListener(view1 -> {
    Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri
            .parse("https://www.koronavirus.hr/podaci/489"));
    startActivity(browserIntent);
});

zupanijeSpinner = view.findViewById(R.id.spinnerZupanija);
zupanijeSpinner.setSelection(Integer.parseInt(sharedPreferences
        .getString( s: "zupanija",  s1: "10")));
zupanijeSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        covidViewModel.notifySpinnerChanged(i);
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
    }
});
```

Programski kod 4.5 Definicija gumba i padajućeg izbornika unutar *CovidFragmenta*

Gumb vodi korisnika na službenu mrežnu stranicu [28] Republike Hrvatske o stanju virusa SARS-CoV-2 u državi, koju su omogućili Hrvatski zavod za javno zdravstvo, Ministarstvo unutarnjih poslova i Ministarstvo zdravstva. Web stranica otvara se u pregledniku zadanom od strane korisnika. Također, prikazan je način rada elementa *Spinner*, ili padajućeg izbornika. *Spinner* omogućuje korisniku pregled stanja broja umrlih, novooboljelih i ozdravljenih tog dana odabrane županije. Budući da je zadana vrijednost definirana kao Osječko-baranjska županija, unutar implementirane metode *onNothingSelected()* nije potrebno definirati što se događa u tom slučaju, jer do njega ne može doći.

4.2.3. Prikaz posjećenih lokacija

Treći gumb omogućuje prikaz fragmenta s popisom posjećenih lokacija unutar dana. Prilikom stvaranja fragmenta za prikaz lokacija, prvo se stvara *Options menu* koji ima funkcija gumb na

akcijskoj traci. Za prikaz posjećenih lokacija tijekom dana koristi se *RecyclerView*, što je prikazano programskim kodom 4.6.

```
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    setHasOptionsMenu(true);
    textView = view.findViewById(R.id.tvAdrese);
    recycler = view.findViewById(R.id.rvRecycler);
    LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getContext());
    recycler.setLayoutManager(linearLayoutManager);
    adapter = new RecyclerAdapter( clickListener: this);
    recycler.setAdapter(adapter);
    DividerItemDecoration dividerItemDecoration = new DividerItemDecoration(requireContext(),
            linearLayoutManager.getOrientation());
    recycler.addItemDecoration(dividerItemDecoration);

    locationViewModel.getAreLocationsPopulated().observe(getViewLifecycleOwner(), aBoolean -> {
        if (aBoolean){
            textView.setVisibility(View.GONE);
        }else{
            textView.setVisibility(View.VISIBLE);
            textView.setText(getResources().getString(R.string.gps_not_available));
        }
    });
}
```

Programski kod 4.6 Inicijalizacija *RecyclerView*-a unutar *LocationFragmenta*

Nakon definiranja upravljača izgleda, *RecyclerView-a* i njegovog adaptera, postavlja se i dekoracija za *RecyclerView* u obliku horizontalne crte, kako bi korisnik imao ugodnije iskustvo pregledavanja lokacija. Za implementaciju *RecyclerView*-a potrebne su tri komponente: sučelje koje pruža klikabilnost u programskom kodu 4.7, adapter koji upravlja podacima koji će se naći unutar *RecyclerView*-a i klasa koja opisuje jedan element *RecyclerView*-a.

```
public interface ClickListener {
    public void onClick(String s);
}
```

Programski kod 4.7 Sučelje za klikabilnost

Adapter koji upravlja podacima treba nasljeđivati klasu *RecyclerView.Adapter<AddressHolder>*, gdje se unutar znakova manje od (<) i više od (>) nalazi ime klase koja pruža definiciju jednog elementa *RecyclerView*-a. Implementacija *Recycler Adaptera* nalazi se u programskom kodu 4.8.

```
public RecyclerAdapter(ClickListener clickListener) {
    repo = Repository.getInstance();
    this.clickListener = clickListener;
    observer = locationsModels -> {
        addData(repo.getLocationsDead());
        notifyDataSetChanged();
    };
    repo.getLocations().observeForever(observer);
}
```

```
public void addData(ArrayList<LocationsModel> locationsModel){
    if(locationsModel != null){
        raw.clear();
        int i;
        for(i = 0; i < repo.getLocationsDead().size(); i++){
            raw.add(repo.getLocationsDead().get(repo.getLocationsDead().size() -1 -i).address);
        }
    }
}
```

```
@NonNull
@Override
public AddressHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.address_row, parent,
            attachToRoot: false);
    return new AddressHolder(view, clickListener);
}
```

```
@Override
public void onBindViewHolder(@NonNull AddressHolder holder, int position) {
    if(repo.getLocationsDead() != null)
    holder.setName(raw.get(position));
}
```

```
@Override
public int getItemCount() {
        return raw.size();
}
```

Programski kod 4.8 Implementacija adaptera *RecyclerAdapter*

RecyclerView implementira se na način se, nakon nasljeđivanja, prepišu metode koje pruža roditeljska klasa, to su *onCreateViewHolder, onBindViewHolder i getItemCount*. Prve dvije navedene metode bave se ispunjavanjem određenog elementa željenim podacima i njegovim prikazom, dok *getItemCount* vraća broj elemenata unutar *RecyclerView*-a. Osim prepisanih

metoda, važna je i metoda *addData*, kojom se u adapter postavljaju podaci o posjećenim lokacijama ravno iz baze podataka. Prikaz posjećenih lokacija ažurira se pomoću metode *notifyDataSetChanged*. Ukoliko se želi postići klikabilnost elemenata *RecyclerView*-a, potrebno je pomoću konstruktora predati instancu već navedenog sučelja klasi *AddressHolder*, koja je opisana u programskom kodu 4.9.

```
public class AddressHolder extends RecyclerView.ViewHolder  implements View.OnClickListener {
    TextView textView;
    ClickListener clickListener;

    public AddressHolder(@NonNull View itemView, ClickListener clickListener) {
        super(itemView);
        textView = itemView.findViewById(R.id.textView);
        this.clickListener = clickListener;
        textView.setOnClickListener(this);
    }

    public void setName(String name){
        textView.setText(name);
    }

    @Override
    public void onClick(View view) {
        clickListener.onClick(textView.getText().toString());
    }
}
```

Programski kod 4.9 Klasa *AddressHolder*

Klasa koja predstavlja jedan element unutar *RecyclerView*-a treba nasljeđivati klasu *RecyclerView.ViewHolder* i ako se želi postići klikabilnost potrebno je implementirati sučelje *View.OnClickListener*. Kako bi se implementiralo sučelje *View.OnClickListener,* potrebno je prepisati metodu *onClick*, koja omogućuje definiranje željene radnje prilikom pritiska na jedan element *RecyclerVi*ew-a. Budući da je navedeno sučelje implementirano, kao argument za metodu *setOnClickListener* može se predati trenutna instanca klase. Prilikom pritiska na jedan element unutar *RecyclerView*-a, poziva se metoda *onClick* nad varijablom *clickListener*, što je instanca *LocationFragmenta*. Prilikom tog poziva, poziva se metoda iz programskog koda 4.10. Unutar *try* bloka definira se poziv upućen aplikaciji *Google Maps*, pri čemu se pokušava upaliti točna lokacija

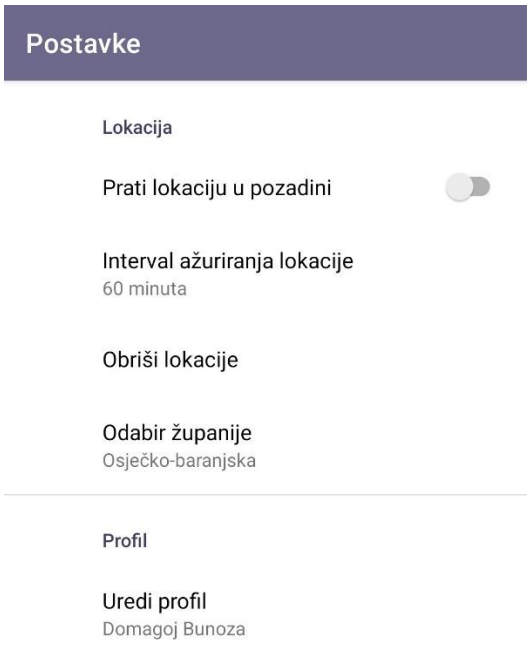
sadržana u pritisnutom elementu *RecyclerView*-a. Ukoliko *Google Maps* nije instaliran na mobilnom uređaju korisnika, baca se greška i korisniku se na zaslon ispisuje odgovarajuća poruka.

```
@Override
public void onClick(String s) {
    try {
        Uri gmmIntentUri = Uri.parse("geo:0,0?q=" + s);
        Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
        mapIntent.setPackage("com.google.android.apps.maps");
        startActivity(mapIntent);
    }catch(Exception e){
        Toast.makeText(getContext(), getResources().getString(R.string.nemoguce_pokrenuti),
                Toast.LENGTH_SHORT).show();
    }
}
```

Programski kod 4.10 Metoda *onClick* unutar fragmenta *LocationFragment*

4.2.4. Prikaz zaslona s postavkama

Na slici 4.2 može se vidjeti izgled korisničkog sučelja fragmenta koji sadrži mogućnost promjene postavki.



Slika 4.2 Fragment s postavkama

Prva postavka omogućuje praćenje lokacije u pozadini, što jedna od glavnih funkcionalnosti aplikacije. Prilikom prvog pokušaja uključivanja praćenja lokacije u pozadini korisniku će se pokazati dijalog u namjeri traženja dopuštenja praćenja lokacije. Kako bi se omogućilo praćenje lokacije u pozadini, čak i kad je sama aplikacija obrisana iz radne memorije, bilo je potrebno definirati odgovarajuću uslugu (*eng. service),* čije stvaranje je vidljivo u programskom kodu 4.11.

```
private void startMyOwnForeground(){
    String NOTIFICATION_CHANNEL_ID = "com.example.simpleapp";
    String channelName = "My Background Service";
    NotificationChannel chan = new NotificationChannel(NOTIFICATION_CHANNEL_ID, channelName,
            NotificationManager.IMPORTANCE_NONE);
    chan.setLightColor(Color.BLUE);
    chan.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    NotificationManager manager = (NotificationManager) getSystemService(
            Context.NOTIFICATION_SERVICE);
    assert manager != null;
    manager.createNotificationChannel(chan);

    NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(
            context: this, NOTIFICATION_CHANNEL_ID);
    Notification notification = notificationBuilder.setOngoing(true)
            .setContentTitle("Aplikacija radi u pozadini")
            .setPriority(NotificationManager.IMPORTANCE_MIN)
            .setSmallIcon(R.drawable.ic_baseline_coronavirus_24)
            .setCategory(Notification.CATEGORY_SERVICE)
            .build();
    startForeground( id: 2, notification);
}
```

Programski kod 4.11 Stvaranje usluge

Usluga može nastaviti raditi sve dok se na traci za obavijesti prikazuje obavijest koja je već definirana. Ako sam sustav ugasi uslugu zbog optimizacije potrošnje baterije, korisnik će znati tako što više neće vidjeti obavijest u traci s obavijestima. Pri svakom dohvaćanju lokacije, poziva se i metoda *checkTimestamps*, koja provjerava je li počeo novi dan. Ako je uvjet ispunjen, briše se povijest lokacija unutar fragmenta s lokacijama, odnosno unutar baze podataka. Lokacija se dohvaća putem *fn_update* metode, koja se koristi *Geocoder*-om kako bi za određene vrijednosti geografske duljine i širine dohvatila potencijalnu adresu trenutne lokacije, što je ostvareno kao u programskom kodu 4.12. Ukoliko dohvaćena lokacija nije jednaka posljednoj u bazi podataka, dohvaćena lokacija pohranit će se u bazu podataka. Konstruktoru *LocationsModel*-a predaje se i trenutno vrijeme.

```
private void fn_update(Location location) {
    if(!sharedPreferences.getBoolean( s: "gpsSwitch",  b: false)){
        stopForeground( removeNotification: true);
        stopSelf();
    }
    Geocoder geocoder = new Geocoder(getApplicationContext(), Locale.getDefault());
    int i;
    try{
        String currentAddress = geocoder.getFromLocation(location.getLatitude(),
                location.getLongitude(),  maxResults: 1).get(0).getAddressLine( index: 0);
    if (repo.getLocationsDead() != null && repo.getLocationsDead().size() > 0){
        if(!currentAddress.equals(repo.getLocationsDead().get(repo.getLocationsDead().size()-1)
                .address)) {
            locations = new ArrayList<>();
            for(i = 0; i < repo.getLocationsDead().size(); i++){
                locations.add(repo.getLocationsDead().get(i));
            }
            locations.add(new LocationsModel(String.valueOf(new Date().getTime()),
                    currentAddress));
            LocationsModel temp = new LocationsModel(String.valueOf(new Date().getTime()),
                    currentAddress);
            repo.insertData(temp);
        }
    }else{
        LocationsModel temp = new LocationsModel(String.valueOf(new Date().getTime()),
                currentAddress);
        repo.insertData(temp);
    }
        checkTimestamps();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Programski kod 4.12 Ažuriranje lokacije unutar usluge

Za vrijeme kreiranja usluge, pokreće se sustav za mjerenje vremena *(eng. timer)* koji omogućuje dohvaćanje lokacije u odabranog intervalu. Programski kod sustava za mjerenje vremena opisan je u programskom kodu 4.13. Korisniku se pruža mogućnost odabira raznih intervala dohvaćanja lokacije: 3 sekunde, 5, 10, 15, 30 i 60 minuta. Intervali od 15 minuta i kraći služili su isključivo kao pomoć pri razvoj aplikacije i ne bi se trebali koristiti zbog povećane potrošnje baterije i mogućih neispravnosti u rezultatu procjene mogućnosti zaraze virusom SARS-CoV-2.

```
sharedPreferences = PreferenceManager.getDefaultSharedPreferences( context: this);
if(sharedPreferences.getBoolean( s: "gpsSwitch",  b: false)) {
    notify_interval = 1000 * Integer.parseInt(sharedPreferences.getString(
            s: "interval",  s1: "1000"));

    mHandler = new Handler();
    mTimer = new Timer();
    mTimerToGetLoc = new TimerTaskToGetLocation();
    mTimer.schedule(mTimerToGetLoc,  delay: 5, notify_interval);
    repo = Repository.getInstance();
    fn_getlocation();

    Log.d(TAG,  msg: "onCreate: " + notify_interval);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
        startMyOwnForeground();
    else
        startForeground( id: 1, new Notification());
}
```

Programski kod 4.13 Sustav za mjerenje vremena

Idući na popisu postavki nalazi se gumb *Obriši lokacije*, koji je definiran programskim kodom 4.14.

```
delete.setOnPreferenceClickListener(preference -> {
    new AlertDialog.Builder(getContext())
            .setTitle("Želite obrisati lokacije?")
            .setMessage("Obrisani podaci ne mogu se vratiti.")
            .setPositiveButton(android.R.string.yes, (dialog, which) -> repo.
                    deleteLocationData())
            .setNegativeButton( text: "Odustani",  listener: null)
            .setIcon(R.drawable.ic_baseline_warning_24)
            .show();
    return false;
});
```

Programski kod 4.14 Postupak brisanja lokacija

Ukoliko korisnik odluči potvrdno odgovoriti na prikazani dijalog, iz baze podataka se briše povijest lokacija, a samim time i iz fragmenta koji sadrži popis lokacija. Četvrta stavka u postavkama je odabir županije, pritiskom na koju se prikazuje popis hrvatskih županija. Izbor županije se pohranjuje unutar postavki. Prikaz županije na početnom zaslonu i na fragmentu s

informacijama o dnevnom broju oboljelih od bolesti izazvanom SARS-CoV-2 mijenja se ovisno o navedenom odabiru korisnika unutar postavki.

Prilikom pritiska na opciju *Uredi profil*, pokreće se aktivnost unosa podataka kao što prikazuje programski kod 4.3, a kao što je opisano u programskom kodu 4.15, pri čemu je moguće izmijeniti profil korisnika.

```
profil.setOnPreferenceClickListener(new Preference.OnPreferenceClickListener() {
    @Override
    public boolean onPreferenceClick(Preference preference) {
        Intent intent = new Intent(getContext(), ScrollingActivity.class);
        startActivityForResult(intent, REQUEST_CODE_SECOND_ACTIVITY);
        return false;
    }
});
```

Programski kod 4.15 Uređivanje profila

4.2.5. Prikaz početnog zaslona

Na početnom zaslonu, opisanom u programskom kodu 4.16, prikazuje se današnji broj oboljelih od bolesti uzrokovanom virusom SARS-CoV-2 u odabranoj županiji, u cijeloj Republici Hrvatskoj, zadnja posjećena lokacija, procjena vjerojatnosti zaraze virusom SARS-CoV-2 i odgovarajuća preporuka, ovisna o razinama vjerojatnosti zaraze virusom SARS-CoV-2 kroz sedam dana. Dohvaćena trenutna mogućnost zaraze virusom SARS-CoV-2 prikazuje se korisniku u obliku postotka sa dva decimalna mjesta, obojen u odgovarajuću boju. Vjerojatnost zaraze manja od 15 posto bit će obojena zelenom bojom, što znači nisku mogućnost zaraze. Vjerojatnost od 15 do 40 posto znači umjeren rizik zaraze virusom SARS-CoV-2, koji će biti obojen narančastom bojom. Postotak u vrijednosti od 40 ili više znači visoku vjerojatnost zaraze virusom SARS-CoV-2. Takav postotak bit će obojen crveno. U navedenom programskom kodu može se vidjeti kako se u aplikaciji koristi obrazac „promatrač“ (eng. *observer*), što je jedan on čimbenika MVVM arhitekture mobilnih aplikacija.

```
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    sharedPreferences = PreferenceManager.getDefaultSharedPreferences(getContext());
    Repository repo = Repository.getInstance();

    location = view.findViewById(R.id.text_home);
    covidHR = view.findViewById(R.id.tvCovidDanasHR);
    covidZUP = view.findViewById(R.id.tvCovidDanasZUP);
    procjena = view.findViewById(R.id.procjena);
    prosjek = view.findViewById(R.id.tvProsjek);
    preporuka = view.findViewById(R.id.tvPreporuka);

    homeViewModel.getApproximation().observe(getViewLifecycleOwner(), aDouble -> {
        procjena.setText(String.format("%.2f%%", aDouble));
        setValidColor(aDouble);
    });
    homeViewModel.lastApproximations.observe(getViewLifecycleOwner(), approximations -> {
        Double sum = 0.0;
        for(int i = 0; i < approximations.size(); i++){
            sum += approximations.get(i).value;
        }
        if(approximations.size() == 1){
            prosjek.setText(String.format("Prosjek procjene zaraze unazad %d dan: %.2f%%",
                    approximations.size(), sum / approximations.size()));
        }else{
            prosjek.setText(String.format("Prosjek procjene zaraze unazad %d dana: %.2f%%",
                    approximations.size(), sum/approximations.size()));
        }
        if(sum / approximations.size() < 15){
            preporuka.setText(getResources().getString(R.string.nizak_prosjek));
            preporuka.setTextColor(Color.rgb( red: 0, green: 200, blue: 0));
        }else if( sum/approximations.size() < 40){
            preporuka.setText(getResources().getString(R.string.medium_prosjek));
            procjena.setTextColor(Color.rgb( red: 255, green: 165, blue: 0));
        }else{
            preporuka.setText(getResources().getString(R.string.high_prosjek));
            procjena.setTextColor(Color.RED);
        }
    });
```

Programski kod 4.16 Dohvaćanje mogućnosti zaraza iz baze podatka na početnom zaslonu

U programskom kodu 4.17 definiran je način dohvaćanja posljednje lokacije, broja zaraženih u odabranoj županiji i u cijeloj Republici Hrvatskoj, i imena korisnika. Ime korisnika dohvaća se kako bi se isto prikazalo u naslovu zaslona, tako da svaki korisnik ima personaliziranu poruku.

```
homeViewModel.getLastLocation().observe(getViewLifecycleOwner(), s -> {
    location.setText(s);
    Log.d(TAG,  msg: "onChanged: " + s);
});
homeViewModel.getCovidHr().observe(getViewLifecycleOwner(), integer -> covidHR
        .setText("Broj slučajeva danas: " + integer.toString()));
homeViewModel.getCovidZup().observe(getViewLifecycleOwner(), integer -> covidZUP
        .setText("Broj slučajeva danas u županiji " +
        (getResources().getStringArray(R.array.zupanije))
                [Integer.parseInt(sharedPreferences
                .getString( s: "zupanija",  s1: "10"))] + ": " + integer.toString()));
homeViewModel.getUsers().observe(getViewLifecycleOwner(), user ->
        ((MainActivity) getActivity()).getSupportActionBar()
                .setTitle("Dobar dan, " + user.ime));
repo.storeData(Integer.parseInt(sharedPreferences.getString( s: "zupanija",  s1: "10")));
```

Programski kod 4.17 Dohvaćanje lokacije, brojeva zaraženih i imena korisnika

4.3. Programsko rješenje na strani poslužitelja

Ovo potpoglavlje pruža detaljno objašnjenje programskog rješenja na strani poslužitelja, kao što su interakcije korisničkog sučelja i baze podataka, dohvaćanje podataka putem REST klijenta i izračun procjene mogućnosti zaraze virusom SARS-CoV-2.

4.3.1. Interakcije korisničkog sučelja i baze podataka

Za interakciju korisnika s podacima koji se trajno pohranjuju u bazu podataka i za implementaciju same baze podataka koristi se baza podataka Room. Radi se o lokalnoj bazi podataka koja se koristi za pohranu strukturiranih podataka. Budući da se radi o lokalnoj bazi podataka, nema potrebe za povezivanjem na internet. Razvojnim programerima se preporuča korištenje Room baze podataka umjesto direktnog pozivanja *SQLite API*-ja iz nekoliko razloga, neki od kojih su skraćeno vrijeme verifikacije *SQL* zahtjeva, umanjivanje repetitivnog koda sklonog greškama i pojednostavljene migracijske putanje baze podatka. Kako bi se omogućilo korištenje Room baze podataka, u Grade datoteci potrebno je definirati na isti načina kao u programskom kodu 4.18.

```
def room_version = "2.3.0"
implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
```

Programski kod 4.18 *Gradle Room database*

Baza podataka *Room* sastoji se od 3 glavne komponente: klasa baze podataka, podatkovni entiteti i objekti za pristup podacima. Klasa baze podataka pruža aplikaciji objekte za pristup podacima, samim time i mogućnost manipulacije i upravljanja podataka. Iz priloženog programskog koda 4.19 može se vidjeti kako su entiteti, tj. tablice definirane pomoću klasa *LocationsModel, CovidDB, User* i *Approximation*.

```
@Database(entities = {LocationsModel.class, CovidDB.class, User.class, Approximation.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract CovidDBDao covidDBDao();
    public abstract LocationsDao locationsDao();
    public abstract UserDao userDao();
    public abstract ApproxDao approxDao();
}
```

Programski kod 4.19 Klasa baze podataka

Svakom entitetu priložen je jedan objekt za pristupanje podacima (*eng. Data Access Object, DAO*). U programskom kodu 4.20 može se vidjeti programski kod jednog od objekata za pristup podacima. Prva metoda služi za dohvaćanje svih korisnika pohranjenih unutar baze podataka. Druga metoda koristi se kao objekt promatrača, budući da se ista metoda promatra od strane više promatrača unutar aplikacije. Metode *insert* i *delete* služe za pohranu korisnika u bazu podataka i brisanje korisnika iz baze podataka. Metoda *insert* sadrži i dodatni argument koji znači da će se pri umetanju novog korisnika, a već postoji korisnik pohranjen u bazi podataka, stari obrisati prije unosa novog, iz razloga što samo jedan korisnik može koristiti aplikaciju u isto vrijeme.

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAllDead();

    @Query("SELECT * FROM user")
    LiveData<List<User>> getAll();

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insert(User user);

    @Delete
    void delete(User user);

    @Query("SELECT COUNT(*) FROM user")
    int getElementCount();
}
```

Programski kod 4.20 Objekt za dohvaćanje podataka o korisniku

U programskom kodu 4.21 može se vidjeti kako je definiran jedan od modela baze podataka. Radi se o pohranjivanju podataka o korisniku, gdje se mogu vidjeti tipovi pojedine varijable i imena stupaca tablice unutar baze podataka. Varijabla *id* definirana je kao primarni ključ tablice, koji ne može biti *null* i njegova vrijednost ne može biti jednaka nekom od drugih primarnih ključeva. Postoje atributi tablice pod nazivima: *Ime*, *Prezime*, *Spol*, *Datum*, *Pusenje*, *Lijekovi*, *Posao*. Budući da u aplikaciji smije postojati samo jedan korisnik, unutar konstruktora *User*-a primarni ključ se postavlja na konstantnu vrijednost. Time se postiže to da se svaki novi korisnik piše u bazu podataka preko starog zahvaljujući dodatnom argumentu metode *insert* unutar objekta za pristup objektu.

```java
@Entity
public class User {

    @PrimaryKey
    private int id;

    @ColumnInfo(name = "Ime")
    public String ime;

    @ColumnInfo(name = "Prezime")
    public String prezime;

    @ColumnInfo(name="Spol")
    public char spol;

    @ColumnInfo(name="Datum")
    public String datum;

    @ColumnInfo(name="Pusenje")
    public boolean pusenje;

    @ColumnInfo(name="Lijekovi")
    public boolean lijekovi;

    @ColumnInfo(name="Posao")
    public String posao;
```

Programski kod 4.21 Model podataka o korisniku

4.3.2. Dohvaćanje podataka putem REST klijenta

Za potrebe aplikacije za procjenu mogućnosti zaraze virusom SARS-CoV-2 korišten je *REST* klijent *Retrofit 2*. *Retrofit 2* radi na način da se definiraju krajnje točke *API* poziva, što je vidljivo u programskom kodu 4.22, stvara se sučelje koje definira moguće *HTTP* operacije te se definira model podataka koji se dohvaćaju.

```
public interface RetrofitInterface {
    @GET("json/?action=podaci")
    Call<List<CovidZadnjiPodaci>> getLatest();

    @GET("json/?action=po_danima_zupanijama")
    Call<List<CovidZadnjiZupanije>> getByCounty();
}
```

Programski kod 4.22 Sučelje *Retrofit* za dohvaćanje podataka s interneta

Budući da je za rad aplikacije potrebno dohvatiti podatke s dvije različite lokacije, potrebno je definirati dvije metode s oznakom *GET*. Programski kod 4.23 prikazuje klasu *Retrofit* klijenta.

```
public class RetrofitClient {
    private static final String BASE_API =
            "https://www.koronavirus.hr/";
    private static RetrofitInterface retrofitInterface;

    public static RetrofitInterface getRetrofitInterface() {
        if (retrofitInterface == null) {

            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_API)
                    .addConverterFactory(GsonConverterFactory.create())
                    .client(new OkHttpClient.Builder()
                            .connectTimeout( timeout: 5, TimeUnit.SECONDS)
                            .readTimeout( timeout: 5, TimeUnit.SECONDS)
                            .writeTimeout( timeout: 5, TimeUnit.SECONDS).build()
                    )
                    .build();
            retrofitInterface = retrofit.create(RetrofitInterface.class);
        }
        return retrofitInterface;
    }
}
```

Programski kod 4.23 Definicija klase *Retrofit* klijenta

Unutar klase definira se osnova API-ja, te *Retrofit* sučelje. Za pružanje instance *Retrofit* sučelja koristi se obrazac stvaranja *singleton*. *Singleton* provjerava postoji li već instance željenog sučelja i ako ne postoji, stvara ga. Također, prilikom stvaranja instance *Retrofita*, dodaje se *OkHttpClient*, koji omogućuje postavljanje *Timeout* vremena. Nakon *Timeout* vremena zahtjev za podacima s interneta se poništava.

4.3.3. Izračun procjene mogućnosti zaraze virusom SARS-CoV-2

Programski kod 4.24 definira prvi dio izračuna mogućnosti zaraze virusom SARS-CoV-2. Nakon inicijalizacije varijable koje predstavlja trenutnu sumu, koja će kasnije služiti kao *Ex* iz (3.1), dohvaća se spol korisnika iz baze podataka. Ovisno o spolu, sumi se dodaje odgovarajuća vrijednost. Nakon spola, dohvaća se datum rođenja korisnika. Obrađivanjem datuma rođenja dolazi se do dobi korisnika.

```
public void setNewApproximation(){
    double sum = 0;
    int age = 0;
    if(user.getValue() != null){
        if (user.getValue().spol == 'M') {
            sum += 5;
        } else if (user.getValue().spol == 'Z') {
            sum += 3;
        } else {
            sum += 4;
        }
    }
    if(user.getValue() != null) {
        try {
            List<String> datumi = Arrays.asList(user.getValue().datum.split( regex: "/"));
            age = getAge(Integer.parseInt(datumi.get(2)), Integer.parseInt(datumi.get(1)),
                    Integer.parseInt(datumi.get(0)));
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (age < 15) {
            sum += 5;  } else if (age < 30) {
            sum += 10; } else if (age < 45) {
            sum += 15; } else if (age < 60) {
            sum += 20; } else {
            sum += 25;
        }
        if (user.getValue().pusenje) {
            sum += 10;
        }
        if (user.getValue().lijekovi) {
            sum += 15;
        }
        List<String> list = new ArrayList<String>(Arrays.asList(context.getResources()
                .getStringArray(R.array.poslovi_array)));
        int index = list.indexOf(user.getValue().posao)  ;
        sum += Integer.parseInt(Arrays.asList(context.getResources()
                .getStringArray(R.array.posloviValues_array)).get(index));
    }
```

Programski kod 4.24 Prvi dio izračuna nove mogućnosti zaraze

Dob utječe na sumu na sljedeći način, dodajući težinu iz tablice 4.1 postojećoj sumi:

Tablica 4.1 Težine ovisno o dobi korisnika

Dob	Težina
0 - 14	5
15 - 29	10
30 - 44	15
45 - 59	20
60 ili više	25

Nakon dohvaćanja dobi, slijedi dohvaćanje pušenja i konzumacije lijekova iz baze podataka. Ukoliko je dohvaćena istinita vrijednost, sumi se dodaje konstanta 10, odnosno 15. Zatim, potrebno je dohvatiti odabrani posao korisnika, samim time i težinu odabranog posla. Težine se pridaju ponuđenim poslovima kao u tablici 4.2.

Nakon toga, dohvaća se broj zaraženih u odabranoj županiji, čija se desetina dodaje postojećoj sumi, što se vidi u programskom kodu 4.25. Broj posjećenih lokacija utječe na sumu tako što ju množi. Ako je broj posjećenih lokacija 0, znači da korisnik nije aktivirao GPS praćenje, a ako je broj posjećenih lokacija jednak 1, smatra se da je korisnik kod kuće pri čemu je mogućnost zaraze minimalna. Nadalje se varijabla koja je do sad predstavljala sumu koristi kao *Ex*, definiran kao izraz (3.1).

Nakon implementacije logističke regresije, rezultat se postavlja kao vrijednost nove procjene mogućnosti zaraze. Budući da nije moguća mogućnost zaraze od 100 posto, ako je postotak vrijednosti 99 ili više, postotak ostaje na vrijednosti od 99. Nova vrijednost procjene mogućnosti zaraze pohranjuje se u bazu podataka ako je prva mogućnost zaraze trenutnog dana, ili ako je veća od prethodne mogućnosti zaraze istog dana. Uz vrijednost izraženu u postotku, pohranjuje se i vrijeme izračuna mogućnosti zaraze kako bi se istom mogućnošću moglo upravljati u budućnosti ovisno o njenom vremenu izračuna.

Tablica 4.2 Težina ponuđenih poslova

Posao	Težina
Administracija	5
Arhitektura	4
Botanika	2
Dizajn	5
Financije	3
Fitness	15
Komercijalist	10
Kuhar	5
Ljudski resursi	10
Maloprodaja	20
Marketing	6
Mediji	10
Menadžment	10
Nekretnine	10
Obrazovanje	20
Odvjetnik	10
Osiguranje	7
Pisanje	2
Prevođenje	2
Proizvodnja	5
Salon za ljepotu	10
Služba za korisnike	5
Software	3
Spa	7
Sustavi i mreže	7
TV/Film/Video	5
Tehnička podrška	7
Transport	20
Umjetnost	5
Veleprodaja	20
Vlada/Sabor	15
Web dizajn	3
Zdravstvo	30

```
if(covidZup.getValue() != null){
    sum += (covidZup.getValue()*1.0)/10;
}

if(repo.getLocationsDead() != null){
    ArrayList<LocationsModel> list = new ArrayList<>(repo.getLocationsDead());
    sum *= list.size();
    if(list.size() == 1 || list.size() == 0){
        sum = 0;
    }
}else {
    sum = 0;
}
sum /= 200.0 /3;
sum = (1)/(1+Math.exp(-(-3.5+(sum))));
if(sum >= 0.999){
    sum = 0.999;
}
approximation.setValue(sum*100);

@SuppressLint("SimpleDateFormat") SimpleDateFormat formatter =
        new SimpleDateFormat( pattern: "dd/MM/yyyy");
String dateString = formatter.format(new Date(Calendar.getInstance().getTimeInMillis()));
if(repo.getApproxDead().size() > 0){
    if(!dateString.equals(repo.getApproxDead().get(repo.getApproxDead().size()-1).date)){
        repo.deleteLocationData();
        sum = 0;
        repo.insertApprox(new Approximation(sum, dateString));
    }else if(dateString.equals(repo.getApproxDead().get(repo.getApproxDead().size()-1).date)
            && sum > repo.getApproxDead().get(repo.getApproxDead().size()-1).value){
        repo.deleteLastApproximation();
        repo.insertApprox(new Approximation(sum, dateString));
    }
}else {
    repo.insertApprox(new Approximation(sum, dateString));
}
repo.checkApproxCount();
```

Programski kod 4.25 Drugi dio izračuna nove mogućnosti zaraze

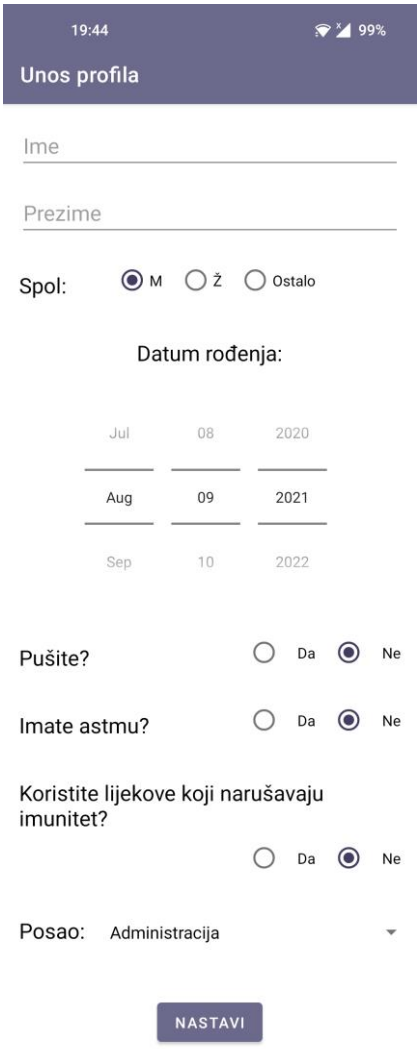
5. PRIKAZ NAČINA RADA APLIKACIJE, ISPITIVANJE I ANALIZA REZULTATA

U ovom poglavlju nalazi se detaljan prikaz korištenja aplikacije i objašnjenja svih funkcionalnosti, od unosa osobnih podataka korisnika, do prikaza mogućnosti zaraze virusom SARS-CoV-2, prikaza dnevnog broja zaraženih i prikaza posjećenih lokacija.

5.1. Prikaz načina rada aplikacije

Prilikom pokretanja aplikacije, provjerava se je li već neki korisnički profil pohranjen u bazi podataka, ako profila nema, prikazuje se zaslon sa slike 5.1. Nakon pritiska na gumb *Novi profil*, korisniku se prikazuje zaslon sa slike 5.2, pri čemu se od korisnika očekuje unos svih potrebnih parametara.

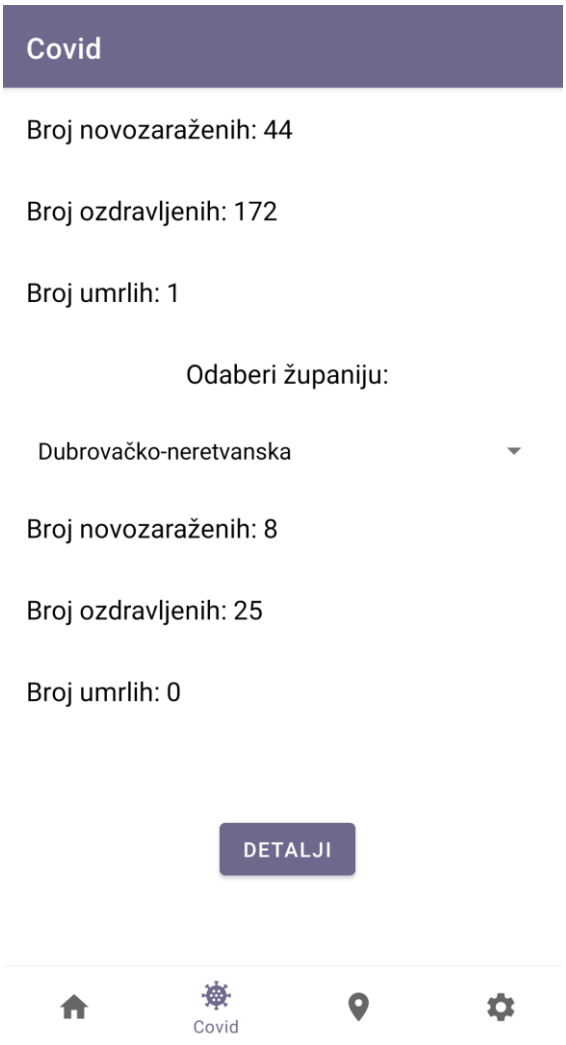
Slika 5.1 Početni zaslon bez korisnika u bazi podataka



Slika 5.2 Unos podataka o novom korisniku

Pritiskom na gumb *Nastavi*, prikazuje se početni zaslon aplikacije (slika 5.3). Na početnom zaslonu postoje tri retka teksta, koji pružaju najvažnije podatke o trenutnoj situaciji za korisnika, osim procjene mogućnosti zaraze virusom SARS-CoV-2. *Broj slučajeva danas* odnosi se na broj zaraženih određenog dana na razini Republike Hrvatske, a *broj slučajeva danas u županiji* prikazuje broj zaraženih određenog dana na razini županije izabrane od strane korisnika na zaslonu s postavkama. *Zadnja lokacija* ispisuje adresu zadnje dohvaćene lokacije tijekom jednog određenog dana. *Mogućnost zaraze* prikazuje procjenu vjerojatnosti zaraze virusom SARS-CoV-2, tj. prikazuje rezultat algoritma opisanog u (3.2). Osim toga, može se vidjeti i prosjek procjene zaraze unazad najviše sedam dana, na temelju čega se stvara preporuka s odgovarajućom porukom i tekstom. Slika 5.4 prikazuje drugi zaslon aplikacije, gdje se mogu vidjeti dodatne brojke u vezi određenog dana. Može se izabrati županija iz padajućeg izbornika (eng. *Spinner*), nakon čega se područja s tekstom ažuriraju odgovarajućim brojevima. Pritisak gumba *Detalji* vodi na [28].

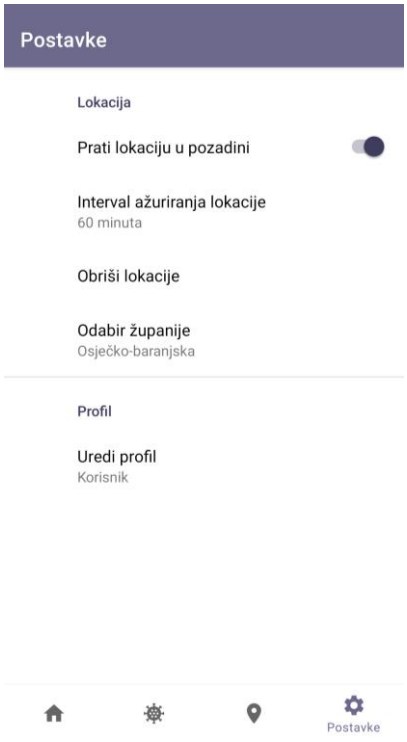
Slika 5.3 Prikaz početnog zaslona



Slika 5.4 Prikaz zaslona s brojevima zaraženih

Pritiskom na treći gumb navigacijske trake na dnu zaslona, prikazuje se zaslon s popisom posjećenih lokacija određenog dana (slika 5.5). Korisnik može na brz i učinkovit dobiti prikaz adrese na *Google Maps* pritiskom na adresu. Pritiskom na gumb s informacijama u gornjem desnom uglu, na alatnoj traci, korisniku se prikazuje prozor s uputama o navedenoj funkciji prikaza adrese na *Google Maps*. Četvrti gumb navigacijske trake vodi na zaslon s postavkama sa slike 5.6. Prva kategorija postavki odnosi se na dohvaćanje i upravljanje lokacijom, a druga kategorija koja se sastoji od jedne postavke služi za uređivanje korisničkog profila. Prva postavka je paljenje ili gašenje praćenja lokacije u pozadini putem GPS-a. Uključeno praćenje lokacije u pozadini nužno je za ispravan rad aplikacije. Interval ažuriranja lokacije može se prilagoditi korisnikovim potrebama, no najbolje je izabrati opciju *60 minuta* ili *30 minuta*, pri čemu se dobivaju najkvalitetniji rezultati. Odabirom županije korisnik može prilagoditi dnevne brojeve zaraženih virusom SARS-CoV-2 koje županije želi vidjeti na svom početnom zaslonu, također koja će županija biti odabrana na drugom, *Covid*, zaslonu.

Slika 5.5 Prikaz posjećenih lokacija u određenom danu



Slika 5.6 Prikaz zaslona s postavkama

5.2. Korisnički slučajevi ispitivanja mobilne aplikacije

Za ispitivanje aplikacije odabrana su tri modela korisničkih profila, koja predstavljaju tri različita načina života. Prvi model korisnika je pisac u kasnim godinama života, provodi najviše vremena

kod kuće. Drugi model je komercijalist u srednjim godinama, ne posjećuje puno lokacija tijekom dana, ali se ne zadržava dugo kod kuće. Treći model je mlada radnica u salonu za ljepotu, posjećuje puno lokacija dnevno. Parametri su objašnjeni u potpoglavlju 3.3.3.

5.2.1. Korisnički slučaj 1

Za korisnički slučaj 1 uneseni su parametri iz tablice 5.1.

Tablica 5.1 Unos parametara za prvi korisnički slučaj i očekivana vrijednost

Parametar	Vrijednost
Spol	muško
Dob	70
Pušenje	ne
Lijekovi	da
Posao	pisanje
Županija	Splitsko-dalmatinska
Broj posjećenih lokacija	2
Očekivana procjena [%]	1 - 15

Prema slici 5.7, rezultat prvog korisničkog slučaja iznosi 13.24%, što se smatra niskim rizikom.

Slika 5.7 Prikaz rezultata prvog korisničkog slučaja

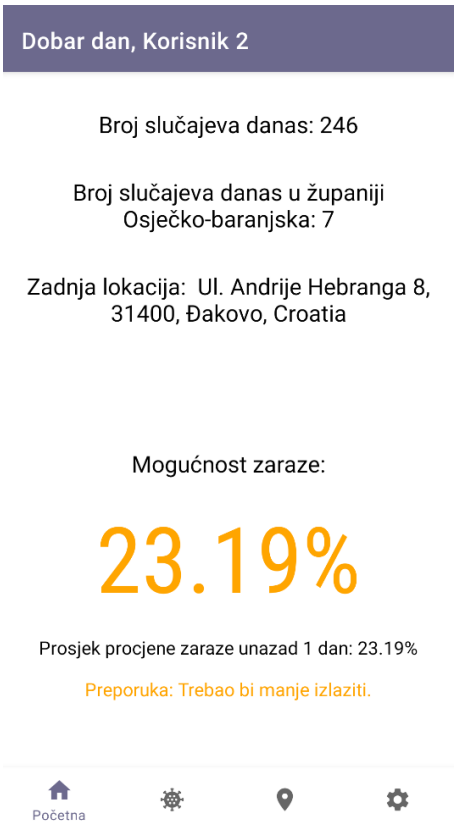
5.2.2. Korisnički slučaj 2

Za korisnički slučaj 2 uneseni su parametri iz tablice 5.2.

Tablica 5.2 Unos parametara za drugi korisnički slučaj i očekivana vrijednost

Parametar	Vrijednost
Spol	muško
Dob	42
Pušenje	ne
Lijekovi	ne
Posao	komercijalist
Županija	Osječko-baranjska
Broj posjećenih lokacija	5
Očekivana procjena [%]	16 - 40

Slika 5.8, prikazuje rezultat drugog korisničkog slučaja, koji iznosi 23.19%. Budući da se postotak nalazi između 15 i 40 posto, smatra se osrednjim rizikom zaraze virusom SARS-CoV-2. Dodatan znak osrednje razine rizika je narančasta boja, koju su poprimili prikaz postotka i preporuke.



Slika 5.8 Prikaz rezultata drugog korisničkog slučaja

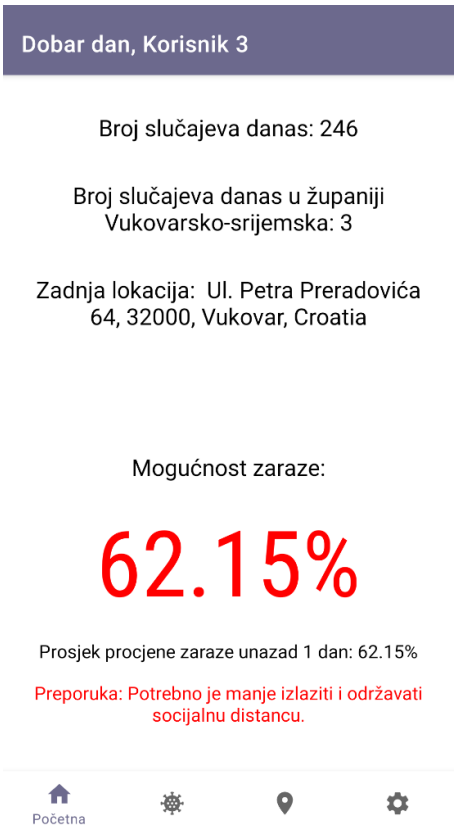
5.2.3. Korisnički slučaj 3

Za korisnički slučaj 3 uneseni su parametri iz tablice 5.3.

Tablica 5.3 Unos parametara za treći korisnički slučaj i očekivana vrijednost

Parametar	Vrijednost
Spol	žensko
Dob	21
Pušenje	da
Lijekovi	ne
Posao	salon za ljepotu
Županija	Vukovarsko-srijemska
Broj posjećenih lokacija	8
Očekivana procjena [%]	Više od 40

Rezultat trećeg korisničkog slučaja iznosi 62.15%, što je prikazano na slici 5.9. Visoka razina rizika zaraze virusom SARS-CoV-2 karakterizirana je crvenom bojom prikaza mogućnosti zaraze i preporuke.



Slika 5.9 Prikaz rezultata trećeg korisničkog slučaja

Ispitivanjem ostvarene mobilne Android aplikacije za višekriterijsku procjenu mogućnosti zaraze virusom SARS-CoV-2 dokazana je njena ispravnost. U prvom korisničkom slučaju očekivana procjena mogućnosti zaraze je niska, tj. manje od 15%, što je i dokazano rezultatom procjene od 13.24%. Za drugi korisnički slučaj previđen postotak procjene mogućnosti zaraze iznosi 15 – 40. Budući da rezultat ispitivanja drugog korisničkog slučaja iznosi 23.19%, dokazana je ispravnost rada programskog rješenja za drugi korisnički slučaj. Očekivani rezultat trećeg korisničkog slučaja je iznad 40%, što je potvrđeno i rezultatom ispitivanja koji iznosi 62.15%. Ispitivanjem rada aplikacije empirijski je dokazana valjanost primijenjenog algoritma za procjenu mogućnosti zaraze virusom SARS-CoV-2.

6. ZAKLJUČAK

Zbog širenja i sve veće rasprostranjenosti bolesti koju uzrokuje virus SARS-CoV-2, ljude zanima kako se mogu zaštititi i kako mogu spriječiti širenje virusa. Neki od čimbenika sprječavanja širenja virusa su samoizolacija i društvena udaljenost. Ova mobilna Android aplikacija nudi korisniku uvid u posjećene lokacije, brz pregled najnovijih brojeva slučajeva zaraze virusom SARS-CoV-2, prikaz procjene mogućnosti zaraze i prikaz prosjeka mogućnosti zaraze unazad sedam dana. Aplikacija može pomoći korisnicima tako što im daje uvid u kretanja tijekom dana, nakon čega se dobiva određena povratna informacija s procjenom mogućnosti zaraze virusom SARS-CoV-2, a ona ovisi isključivo o prosjeku tih procjena.

Aplikacija je razvijena u razvojnoj okolini Android Studio, korištenjem jezika XML i Java. XML korišten je za definiranje strukture i semantike stvorenih zaslona aplikacije, a Java je korištena za prilagodbu elemenata za prikazivanje korisniku, upravljanje podacima s interneta i iz baze podataka, dohvaćanje lokacije i sve ostale funkcionalnosti aplikacije. Dohvaćanje podataka omogućeno je službenim otvorenim strojno čitljivim podacima Vlade Republike Hrvatske, a trenutna lokacija korisnika dohvaća se koristeći GPS. Nakon ostvarenja mobilne aplikacije, ona je bila ispitana za određene slučajeve korištenja, a ispitivanje je potvrdilo ispravnost rada aplikacije. Neka od mogućih poboljšanja su uzimanje u obzir gustoće stanovništva naselja na dohvaćenim lokacijama, pohranjivanje korisničkog profila na internetskoj bazi podataka i stvaranje privlačnijeg korisničkog sučelja.

LITERATURA

[1] »Clinical Care Guidance,« Centers for Disease Control and Prevention, 16 veljača 2021. [Mrežno]. Dostupno na: https://www.cdc.gov/coronavirus/2019-ncov/hcp/clinical-guidance-management-patients.html. [posjećeno 8. lipanj 2021.].

[2] Hrvatski zavod za javno zdravstvo, »Pitanja i odgovori o bolesti uzrokovanoj novim koronavirusom,« 13. ožujak 2020.. [Mrežno]. Dostupno na: https://www.hzjz.hr/priopcenja-mediji/pitanja-i-odgovori-o-bolesti-uzrokovanoj-novim-koronavirusom/. [posjećeno 7. lipanj 2021.].

[3] T.P. Velavan i C.G. Meyer, »The COVID-19 Epidemic,« *Trop Med Int Health,* 2020, pp. 278-280.

[4] G.M. Bwire, »Coronavirus: Why Men are More Vulnerable to Covid-19 Than Women?,« SN comprehensive clinical medicine, 4. lipanj 2020. [Mrežno]. Dostupno na: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7271824/. [posjećeno 7. lipanj 2021.].

[5] »People at Increased Risk,« Centers for Disease Control and Prevention, 14 svibanj 2021. [Mrežno]. Dostupno na: https://www.cdc.gov/coronavirus/2019-ncov/need-extra-precautions/older-adults.html. [posjećeno 8. lipanj 2021.].

[6] »COVIDWISE,« Virginia's Department of Health, 2021. [Mrežno]. Dostupno na: https://play.google.com/store/apps/details?id=gov.vdh.exposurenotification&hl=en&gl=US. [posjećeno 9. lipanj 2021.].

[7] »Stop COVID-19,« Ministarstvo zdravstva Republike Hrvatske, 2020. [Mrežno]. Dostupno na: https://play.google.com/store/apps/details?id=hr.miz.evidencijakontakata. [posjećeno 10. lipanj 2021.].

[8] T. Varsavsky et al., »Detecting COVID-19 Infection Hotspots in England Using Large-Scale Self-Reported Data from a Mobile Application: a Prospective, Observational Study,« *Lancet Public Health,* br. 6, 2020, pp. 21-26.

[9] A. Chande et al., »Interactive COVID-19 Event Risk Assessment Planning Tool,« Georgia Institute of Technology, 7. srpanj 2020. [Mrežno]. Dostupno na: http://covid19risk.biosci.gatech.edu/. [posjećeno 10. lipanj 2021.].

[10] S.L. Zhou, »Lessons on Mobile Apps for COVID-19 from China,« *Journal of Safety Science and Resilience,* svez. 3, br. 2, 2021, pp. 40-49.

[11] »Koronavirus API,« Hrvatski zavod za javno zdravstvo, [Mrežno]. Dostupno na: https://www.koronavirus.hr/podaci/otvoreni-strojno-citljivi-podaci/526. [posjećeno 29. lipanj 2021.].

[12] N. Chapin, »Flowchart,« u *Encyclopedia of Computer Science*, John Wiley and Sons Ltd., 2003, pp. 714–716.

[13] »Architecture of Android Apps,« Codepath, [Mrežno]. Dostupno na: https://guides.codepath.com/android/Architecture-of-Android-Apps. [posjećeno 5. srpanj 2021.].

[14] J. Birch, »Approaching Android with MVVM,« Ribot, 21 rujan 2015. [Mrežno]. Dostupno na: https://labs.ribot.co.uk/approaching-android-with-mvvm-8ceec02d5442. [posjećeno 5. srpanj 2021.].

[15] A. Syromiatnikov i D. Weyns, »A Journey Through the Land of Model-View-* Design Patterns,« u *IEEE/IFIP Conference on Software Architecture*, Sydney, 2014, pp. 21-30.

[16] V. Srivastava, »MVC vs MVP vs MVVM Architecture in Android,« MindOrks, 7. listopad 2019. [Mrežno]. Dostupno na: https://blog.mindorks.com/mvc-mvp-mvvm-architecture-in-android. [posjećeno 5. srpanj 2021.].

[17] B. Nzivu, »Simple GET Request Using Retrofit in Android,« Section, 5. siječanj 2021. [Mrežno]. Dostupno na: https://www.section.io/engineering-education/making-api-requests-using-retrofit-android/. [posjećeno 5. srpanj 2021.].

[18] »Get Current Location in Android,« Javapapers, 24. listopad 2014. [Mrežno]. Dostupno na: https://javapapers.com/android/get-current-location-in-android/. [posjećeno 5. srpanj 2021.].

[19] »Logistic Regression — Detailed Overview,« Towards Data Science, 15. ožujak 2018. [Mrežno]. Dostupno na: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc. [posjećeno 7. srpanj 2021.].

[20] »Android Operating System,« Investopedia, 3. veljača 2021. [Mrežno]. Dostupno na: https://www.investopedia.com/terms/a/android-operating-system.asp. [posjećeno 7. srpanj 2021.].

[21] D. Poeter, »Android Tablet Sales Overtake Apple's iPad in 2013,« PCMag, 3. ožujak 2014. [Mrežno]. Dostupno na: https://www.pcmag.com/news/android-tablet-sales-overtake-apples-ipad-in-2013. [posjećeno 7. srpanj 2021.].

[22] »Platform Architecture,« Android Developers, 11. ožujak 2021. [Mrežno]. Dostupno na: https://developer.android.com/guide/platform. [posjećeno 7. srpanj 2021.].

[23] »User Guide,« Android Developers, [Mrežno]. Dostupno na: https://developer.android.com/studio/command-line/d8. [posjećeno 8. srpanj 2021.].

[24] IBM Cloud Education, »What is Java?,« IBM, 8. svibanj 2019. [Mrežno]. Dostupno na: https://www.ibm.com/cloud/learn/java-explained. [posjećeno 8. srpanj 2021.].

[25] »XML Introduction,« Mozilla, 17. ožujak 2021. [Mrežno]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction. [posjećeno 16. srpanj 2021.].

[26] »Layouts,« Android Developers, 7. siječanj 2020. [Mrežno]. Dostupno na: https://developer.android.com/guide/topics/ui/declaring-layout. [posjećeno 16. srpanj 2021.].

[27] »Meet Android Studio,« Android Developers, 15. srpanj 2021. [Mrežno]. Dostupno na: https://developer.android.com/studio/intro. [posjećeno 16. srpanj 2021.].

[28] »Podaci,« Hrvatski zavod za javno zdravstvo, [Mrežno]. Dostupno na: https://www.koronavirus.hr/podaci/489. [posjećeno 18. srpanj 2021.].

POPIS SLIKA

POPIS PROGRAMSKIH KODOVA

SAŽETAK

U završnom radu programski je ostvarena mobilna Android aplikacija za višekriterijsku procjenu izloženosti zarazi virusom SARS-CoV-2. Pri tom je korištena integrirana razvojna okolina Android Studio i jezici Java i XML. Aplikacija korisniku omogućuje pohranu osobnih podataka u lokalnu bazu podataka Room, dohvaćanje dnevnog broja slučajeva zaraze virusom SARS-CoV-2 u određenoj županiji i Republici Hrvatskoj, dohvaćanje lokacije, prikaz procjene izloženosti zarazi virusom SARS-CoV-2 i prikaz preporuke temeljene na prosjeku procjena izloženosti zarazi virusom SARS-CoV-2. Podaci pohranjeni u lokalnoj bazi podataka koriste se za višekriterijsku procjenu izloženosti zarazi temeljenoj na logističkoj regresiji kako bi se došlo do višekriterijske procjene izloženosti zarazi virusom SARS-CoV-2. Ispitivanjem mobilne aplikacije za više korisničkih slučajeva potvrđena je njena ispravnost i pogodnost za korištenje.

Ključne riječi: logistička regresija, mobilna Android aplikacija, MVVM, virus SARS-CoV-2, višekriterijska procjena.

ABSTRACT

A mobile Android application for multi-criteria assessment of exposure to SARS-CoV-2 virus infection

The goal of bachelor's thesis is to develop an Android mobile application for multi-criteria assessment of exposure to SARS-CoV-2 virus infection. The application is implemented in the integrated development environment Android Studio, using languages Java and XML. The application allows user to store personal dana in Room local database, retrieve the daily number of SARS-CoV-2 infection cases in a particular county and Republic of Croatia, retrieve the location, display of the SARS-CoV-2 virus exposure assessment and display of the recommendation based on average SARS-CoV-2 infection estimations. The data stored in the local database is used for multi-criteria assessment of infection exposure based on logistic regression in order to calculate multi-criteria assessment of SARS-CoV-2 virus infection exposure. Testing the mobile application for multiple user cases confirmed its correctness and suitability.

Key words: logistic regression, mobile Android application, MVVM, SARS-CoV-2 virus, multi-criteria estimation.

ŽIVOTOPIS

Domagoj Bunoza rođen je 7. veljače 2000. godine u Našicama, Hrvatska, a živi u Đakovu. Nakon završetka Osnovne škole Josipa Antuna Ćolnića u Đakovu, upisuje opći smjer Gimnazije Antuna Gustava Matoša. Gimnaziju završava 2018. godine i upisuje sveučilišni preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Od kvaliteta ističe znanje engleskog jezika, sposobnost rada pod pritiskom, upornost i odgovornost. Poznaje programske jezike C, Javu, C++ i C#.

_____

Potpis autora

PRILOZI

Prilog 1: Završni rad u formatu docx

Prilog 2: Završni rad u formatu pdf

Prilog 3: Projekt mobilne aplikacije u Android Studiu