Android aplikacija s uputama za pružanje prve pomoći

Lukac, Stjepan

Master's thesis / Diplomski rad

2016

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:651975

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

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

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

Android aplikacija s uputama za pružanje prve pomoći

Diplomski rad

Stjepan Lukac

Osijek, 2016.

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek,

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Stjepan Lukac
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-661R, 23.10.2013.
Mentor:	Doc.dr.sc. Krešimir Nenadić
Sumentor:	
Predsjednik Povjerenstva:	
Član Povjerenstva:	
Naslov diplomskog rada:	Android aplikacija s uputama za pružanje prve pomoći
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi Android aplikaciju koja će korisniku osigurati upute za pružanje prve pomoći. Ovisno o vrsti ozljede aplikacija treba korisniku pružiti osnovne informacije o pružanju prve pomoći. Osim opisa postupka pružanja prve pomoći osigurati i slikovni prikaz.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: Postignuti rezultati u odnosu na složenost zadatka: Jasnoća pismenog izražavanja: Razina samostalnosti:
Datum prijedloga ocjene mentora:	

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

IZJAVA O ORIGINALNOSTI RADA

Osijek, 29.06.2016.

Ime i prezime studenta:	Stjepan Lukac
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-661R, 23.10.2013.
Ephorus podudaranje [%]:	

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija s uputama za pružanje prve pomoći**

izrađen pod vodstvom mentora Doc.dr.sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

1. UVOD 1
1.1. Tekst zadatka 1
2. ANDROID OPĆENITO 2
2.1. Android operacijski sustav 2
2.2. Sučelje Android uređaja 4
3. TEHNIČKE ZNAČAJKE APLIKACIJE 6
3.1. Programski jezik Java 6
3.2. Android Studio 6
3.3. Baza podataka 7
3.4. Fluid User Interface 8
3.5. Opisni jezik - XML 9
3.6. Fragmenti 9
4. APLIKACIJA „PrvaPomoc“ 11
4.1. Korisnički zahtjevi 11
4.2. Model aplikacije 12
4.3. Aktivnost i Android Manifest 13
4.4. Glavno sučelje 15
4.5. Upute prve pomoći 18
4.6. Lokacija 22
4.7. Baza podataka - registriranje i logiranje 28
4.8. Korisnički podaci 32
5. ZAKLJUČAK 35
LITERATURA 36
SAŽETAK 37
ANDROID APPLICATION WITH INSTRUCTION FOR FIRST AID 38
ŽIVOTOPIS 39

1. UVOD

Razvojem tehnologije ljudi su razvili razna pomagala kako bi što lakše savladali prepreke pa su tako mobilni uređaji postali svakodnevno pomagalo za dnevne poslove. Samim razvojem mobitela, paralelno se razvijaju te usavršavaju razne aplikacije s kojima se mogu raditi nezamislive stvari. Ljudi su današnjim načinom života primorani živjeti vrlo ubrzanim tempom. Razni opasni poslovni, žurba, brza vožnja, nesmotrenost i mnogi drugi faktori su glavni krivci za nesreće koje se dogode te brojne ozlijede koje nastanu. Često u raznim nezgodama, ljudi su primorani pomoći unesrećenima dok na mjesto nezgode ne dođe hitna medicinska pomoć. U takvim situacijama, često nedostaje znanja za pomoć unesrećenom stoga će ovaj diplomski rad u nastavku obrazložiti kako Android aplikacija može pomoći u takvim presudnim trenucima.

Mobilna aplikacija se sastoji od nekoliko funkcionalnih dijelova. Glavna funkcija aplikacije je brzo i jednostavno doći do uputa prve pomoći kako bi se ozlijeđenoj osobi moglo pomoći na licu mjesta dok ne dođe hitna pomoć. Uz upute, aplikacija sadrži funkciju lociranja ozlijeđene osobe putem GPS signala ili mrežnog pružatelja usluge. Također, korisnik se može registrirati sa svojim podacima u aplikaciju te svakodnevno imati pristup svojim podacima.

Nastavak diplomskog rada prikazuje opis Android operacijskog sučelja, tehničke značajke pomoću kojih je realizirana te korake nastanka same aplikacije.

1.1. Tekst zadatka

Izraditi Android aplikaciju koja će korisniku osigurati upute za pružanje prve pomoći. Ovisno o vrsti ozljede aplikacija treba korisniku pružiti osnovne informacije o pružanju prve pomoći. Osim opisa postupka pružanja prve pomoći osigurati i slikovni prikaz.

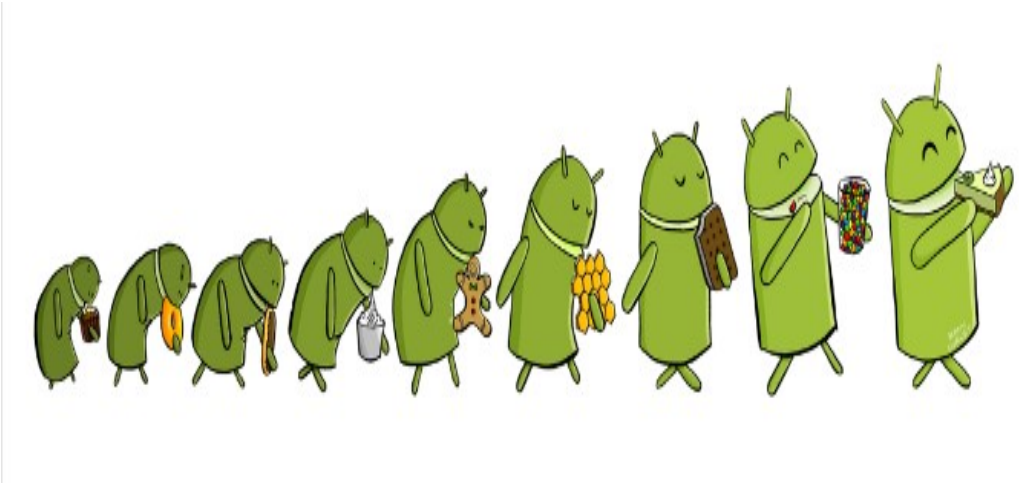
2. ANDROID OPĆENITO

2.1. Android operacijski sustav

Android je operacijski sustav temeljen na Linux platformi te je dizajniran za uređaje kao što su pametni telefoni i tableti. Razvila ga je 2003. godine Android Inc. kompanija koju je u početku Google financirao, a kasnije i otkupio 2005. godine. Prvo javno predstavljanje Androida kao operativnog sustava bilo je u studenome 2007. godine, a prvi komercijalni uređaj u koji je bio ugrađen Android operativni sustav bio je „T-Mobile G1“. Proizveo ga je tajvanski proizvođač pametnih telefona HTC (poznat i pod nazivom HTC Dream). Plasiranjem prvog uređaja na tržište, Android je ušao u izravnu utrku s ostalim mobilnim platformama tipa iOS (Apple), Windows Phone (Microsoft), Samsung itd.

Android je od samih početaka zamišljen kao projekt otvorenog koda (eng. open source project) što znači da svatko može uzeti objavljeni kod Android sustava te ga uređivati, unaprijeđivati, uljepšavati i to je ujedno dobra strana Androida jer se na taj način mogu prilagođavati kodovi za razne inačice Androida. Evolucijski napredak Androida je zaista izvanredan i jedinstven te, za razliku od iOS-a koji je u principu ostao isti od 2007. godine, doživljava funkcionalni i dizajnerski preporod.

Nazivi inačica Android sustava su posebno zanimljivi jer je Google poznat po tome što Android inačicama daje imena po slatkišima što se može vidjeti na slici 2.1.[1] Neki od popularnih „slatkiša“ su Gingerbread (2.3), Ice Cream Sandwich (4.0), Jelly Bean (4.1, 4.2) te onaj od koga je sve počelo – Cupcake (1.5).



Slika 2.1. Android inačice nazvane po „slatkišima“[1]

Popis ostalih Android inačica se može vidjeti u tablici 2.1. [2] Tablica prikazuje samu inačicu Androida, prikazuje njegovo kodno ime, navodi njegov datum plasiranja na tržite te prikazuje API razinu.[3]

Tablica 2.1. Android inačice

Inačica	**Kodno ime**	**Datum izlaska**	**API razina**
Beta		5. studenoga 2007.	
1.0	Apple Pie	23. rujna 2008.	1
1.1	Banana Bread	9. veljače 2009.	2
1.5	Cupcake	30. travnja 2009.	3
1.6	Donut	15. rujana 2009.	4
2.0/2.1	Eclair	26. listopada 2009.	5-7
2.2	Froyo	20. svibnja 2010.	8
2.3.x	Gingerbread	6. prosinca 2010.	9-10
3.x	Honeycomb	22. veljače 2011.	11-13
4.0.x	Ice Cream Sandwich	19. listopada 2011.	14-15
4.1/4.2/4.3	Jelly Bean	9. srpnja 2012.	16-18
4.4	KitKat	3. rujna 2013.	19-20
5.0/5.1	Lollipop	25. lipnja 2014.	21-22
6.0	Marshmallow	5. listopada 2015.	23

2.2. Sučelje Android uređaja

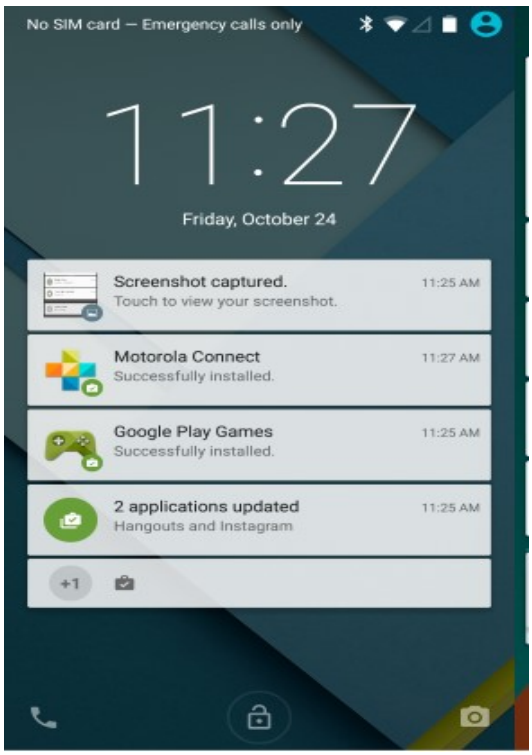
Android korisničko sučelje je bazirano na direktnoj manipulaciji, koristeći dodirne impulse koji odgovaraju akcijama u stvarnome svijetu kao što su klizanje, dodirivanje i kuckanje pomoću kojih se manipulira objektima na zaslonu. Odgovor na korisnički ulaz dizajniran je za trenutnu reakciju, često koristeći vibracije kojima daje korisniku do znanja da je njegov pokret zapažen. Unutrašnje sklopovlje se između ostaloga sastoji i od akcelerometra, žiroskopa i mnogih drugih senzora koji se koriste kroz neke aplikacije kako bi odgovarali dodatnim korisnikovim pokretima, kao npr. prilagođavanje zaslona iz okomitog u vodoravan položaj ovisno o tome kako je uređaj orijentiran, simulirajući volan automobila u raznim igricama itd.

Početni zaslon Android uređaja obično se sastoji od aplikacijskih ikona i dodatnih programa (eng. Widget). Ikone pokreću određenu aplikaciju, dok programi prikazuju trenutni, automatski ažurirani sadržaj, kao što je vremenska prognoza, kalendar ili bilo što što korisniku treba u svakodnevnom životu. Početni zaslon može biti napravljen od nekoliko stranica koje korisnik mijenja klizanjem lijevo-desno po zaslonu i time je Android omogućio korisnicima prilagodljivost te da korisnici sami mogu prilagoditi izgled i funkcije uređaja po svojim potrebama i željama.

Na vrhu zaslona nalazi se statusna straka koja prikazuje informacije o uređaju i njegovoj povezanosti. Ona se može povući prema dolje kako bi se otkrio zaslon obavijesti gdje aplikacije prikazuju važne informacije ili ažuriranje, kao npr. novi SMS ili e-mail. Na slikama 2.2.[4] i 2.3.[5] možemo vidjeti kako općenito izgleda zaslon nekog Android uređaja.



Slika 2.2. Zaslon Android uređaja



Slika 2.3. Notifikacije

3. TEHNIČKE ZNAČAJKE APLIKACIJE

Ovo poglavlje će opisati tehničke stvari s kojima je napravljena aplikacija koju opisuje ovaj diplomski rad.

3.1. Programski jezik Java

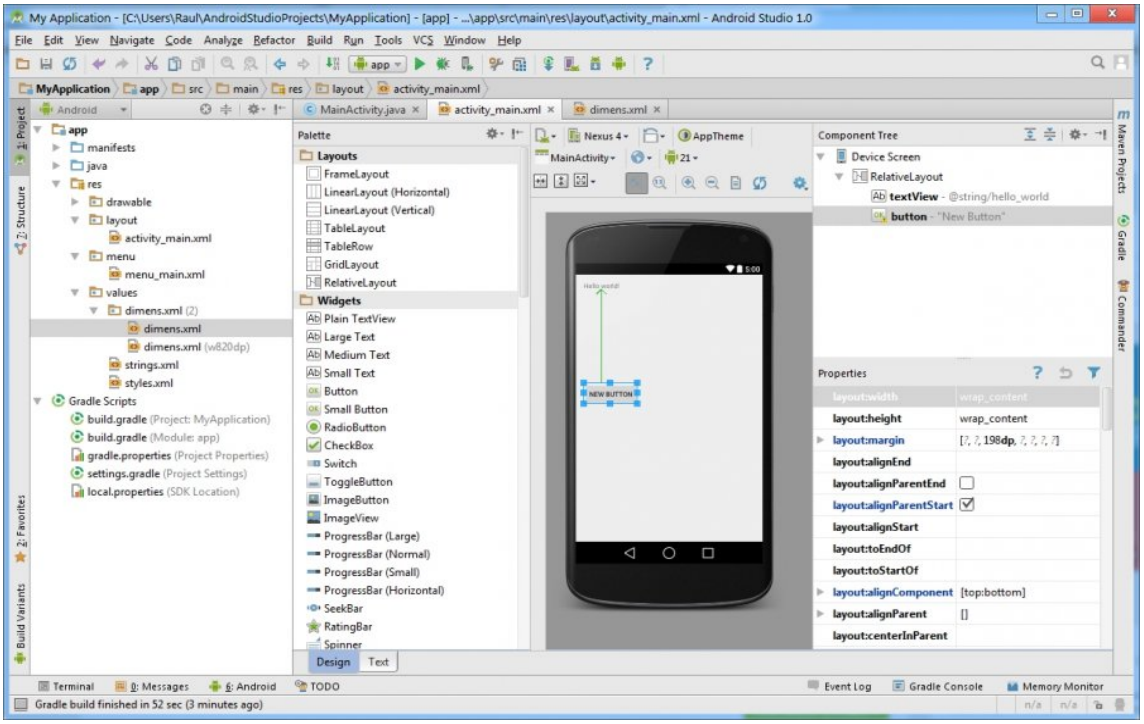
Java je programski jezik koji je objektno orijentiran te se koristi gotovo na svim operacijskim sustavima za koje postoji *JVM* (*Java Virtual Machine*). Razvijen je u timu predvođenim Jamesom Goslingom u kompaniji Sun Microsystems početkom 1990-tih. Glavna ideja nastanka je bila stvaranje programskog jezika koji bi bio nezavisan od operativnog sistema, baziran na C++-u, ali sa pojednostavljenom sintaksom, stabilnijim „*runtime*“ sistemom i pojednostavljenom kontrolom memorije. Pet osnovnih razloga za razvoj *Jave* jesu:

- koristi se objektno orijentirana metodologija
- isti program se izvršava na više operacijskih sustava
- moguća nadogradnja podrške za računalne mreže
- izvršava programe na udaljenim poslužiteljima s velikom sigurnošću
- jednostavan za upotrebu

Sav izvorni kod napisan je unutar klasa jer je to ujedno i osnovni koncept *Jave*. Svaka klasa je deklarirana unutar datoteke s istim imenom i sufiksom **.java**. Prednost *Jave* nad ostalim jezicima je ta što za razliku od npr. C++-a, memorija namijenjena objektu je automatski počišćena, tj. vraćena sistemu čim se taj objekat više ne koristi pa je taj objekat *„van dosega“* (eng. *out of scope*).

3.2. Android Studio

Integrirano razvojno okruženje (eng. *Integrated Development Environment (IDE)*) za *Android* platformu čini *Android Studio* s kojim se razvijaju razne aplikacije. Dizajniran je na temelju *JetBrains' IntelliJ IDEA* softvera te je dostupan za preuzimanje na Windows, Mac OS X i Linux platformama. U prosincu 2014. godine objavljena je prva stabilna inačica *Android Studija* (Sl.3.1)[6] te je glavni razlog nastanka bio zamjena dotadašnjeg *Eclipse*-a kao razvojnog okruženja.



Slika 3.1. Android Studio

Jedan od bitnijih razloga zašto se koristio Android Studio kao razvojno okruženje u ovom diplomskom radu je taj što on omogućuje da programer vidi bilo kakve vizualne promjene u realnom vremenu koje se naprave u aplikaciji, a također se može vidjeti kako će istovremeno izgledati aplikacija na više različitih Android uređaja s različitim konfiguracijama i rezolucijama. Za testiranje aplikacija kreira se virtualni uređaj koji simulira napravljenu aplikaciju. U diplomskom radu se za potrebe testiranja koristio stvarni Android uređaj naziva *LG G3*. Android Studio ujedno pruža:

- Široko razvojno okruženje
- Jednostavan način za testiranje performanski na drugim uređajima
- Razne predloške i „čarobnjake“ za lakše programiranje
- Mnogo dodatnih alata za razvoj aplikacija

3.3. Baza podataka

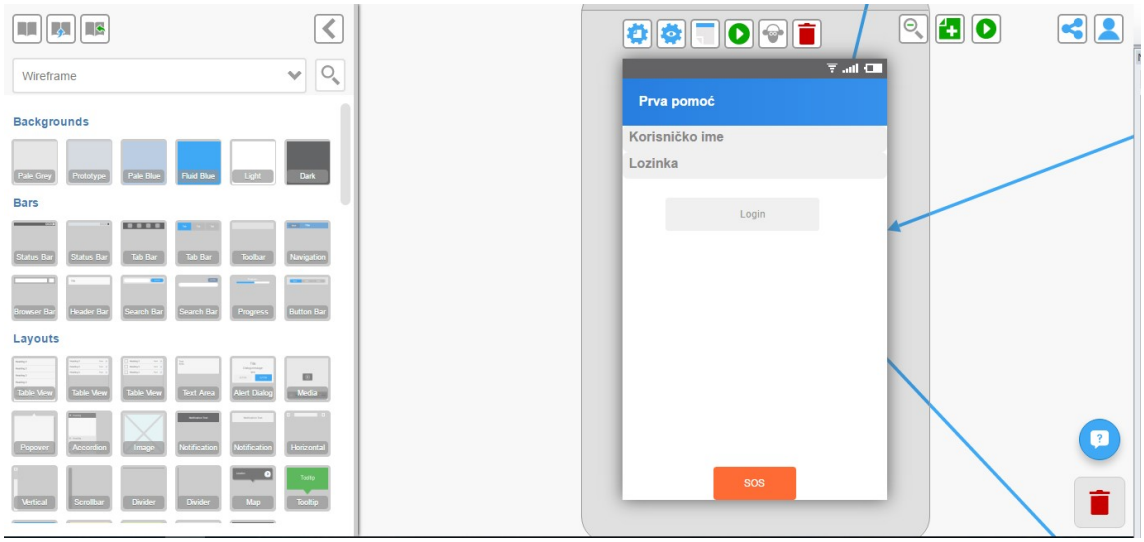
Android Studio najčešće koristi SQLite relacijsku bazu podataka koja se nalazi na svakom Android uređaju gdje nije potrebna posebna instalacija. Sav programski kod SQLite zasnovan je na načelu javnog dobra što znači da se kod i njegovi dijelovi u cijelosti mogu

kopirati, modificirati, koristiti, prodavati te objavljivati kao otvoreni kod u kompajliranom obliku za komercijalnu ili nekomercijalnu svrhu.

Kako je ideja diplomskog rada bila napraviti aplikaciju koja bi se kasnije mogla pretvoriti u ozbiljan projekt (kasnije u radu će biti detaljnije objašnjeno), autor ovog diplomskog rada je koristio online bazu upravo zato što je takva baza uvijek dostupna za razliku od *SQLite* baze koja je dostupna samo aplikaciji koja ju je kreirala. Radi se o stranici naziva *„000webhost“* koja pruža usluge besplatnog poslužitelja i besplatne baze. Funkcionira na način da korisnik napravi račun sa svojim podacima te mu tada pružatelj šalje poruku elektroničkim putem s podacima s kojima će pristupati web poslužitelju i web bazi. Funkcioniranje baze će biti detaljnije objašnjeno u poglavlju 4.7.

3.4. Fluid User Interface

Za svaki projekt je najbitnija ideja, a tu ideju je u prvim koracima najbitnije prenijeti na papir. Upravo za tu svrhu je u ovom diplomskom radu poslužio online alat naziva *Fluid User Interface*[7] pomoću kojega je dizajniran okvirni model aplikacije u digitalnom obliku. Alat funkcionira tako da se korisnik registrira sa svojim podacima gdje mu se u sljedećem koraku otvara sučelje za dizajniranje modela aplikacije.



Slika 3.2. Fluid UI [7]

Slika 3.2. [8] prikazuje s lijeve strane sučelje koje se sastoji od raznih pozadina, barova, dugmadi i drugih programa koje korisnik može dodavati na svoj radni prostor za dizajniranje aplikacija koji se nalazi desno od panela. Također, korisnik pomoću linkova može povezati određene programe sa drugim prozorima kako bi stvorio prototip željene aplikacije. Nakon što je

dizajn gotov, korisnik može pokrenuti simulaciju te vidjeti kako bi aplikacija okvirno trebala izgledati i funkcionirati.

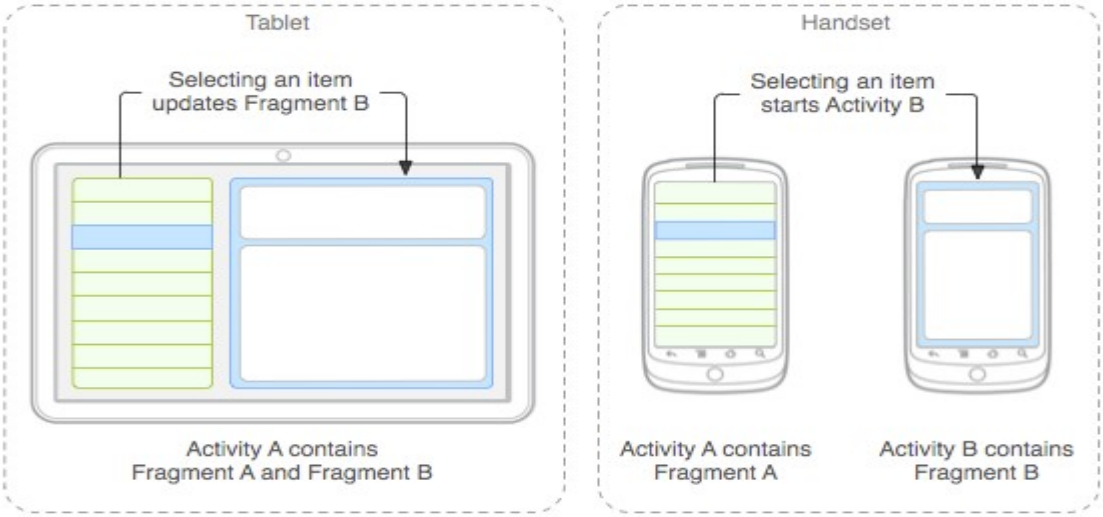
3.5. Opisni jezik - XML

XML (Eng. *Extensible Markup Language*) je opisni jezik koji je napravljen kako bi bio jednostavno čitljiv i ljudima i računalnim programima. Vrlo je sličan *HTML* (eng. *HyperText Markup Language)* jeziku i koristi se za širok spektar primjena. *XML* je standardizirani jezik, a za to se brine *World Wide Web Consortium*. Iako su *HTML* i *XML* dosta slični u sintaksi, sintaksna pravila *XML*-a su vrlo stroga i ako dokument nije formatiran u skladu s njima, računalni program neće moći pročitati *XML* dokument. S druge strane, *HTML* dokument koji nije sintaksno ispravan će svejedno biti pročitan i prikazan na najbolji mogući način.

Kako je u prijašnjem odlomku 3.4. prikazan alat s kojim je napravljen prototip dizajna, upravo XML jezikom je opisan dizajn cijele aplikacije koju nudi ovaj diplomski rad.

3.6. Fragmenti

Fragmenti predstavljaju modularne dijelove aktivnosti (eng. *Activity*) koji mogu, ali i ne moraju imati sučelje. Moguće ih je koristiti više unutar jedne aktivnosti, ali i kroz više njih. Fragmenti imaju vlastiti životni ciklus. Kada se dodaju kao elementi korisničkog sučelja, tada se njihov izgled definira u *XML*-u. Mogu se koristiti statički i dinamički. Korištenje fragmenta prikazuje slika 3.3.[9]



Slika 3.3. Fragmenti[9]

Kod statičkog dodavanja, svaki fragment je klasa koja nasljeđuje osnovnu fragment klasu i implementira njene metode. Komunikacija između fragmenata obavlja se preko aktivnosti, a pojedinim fragmentima je moguće pruistupiti preko *FragmentMenagera* čija se instanca dobiva pozivanjem *getFragmentMenager()* metode.

Osim statičkog, postoji i dinamičko dodavanje fragmenata. U tom slučaju moguće je mijenjati fragmente pomoću fragment izbornika tijekom izvođenja aplikacije bez da se ponovo pokreće aktivnost u kojoj se fragmenti nalaze. U ovom diplomskom radu, glavno sučelje je sastavljeno od tri kartice koji su napravljeni pomoću fragmenata.

4. APLIKACIJA „PrvaPomoc“

4.1. Korisnički zahtjevi

Glavna ideja i funkcija aplikacije je brzo i jednostvno doći do uputa prve pomoći kako bi se ozlijeđenoj osobi moglo pomoći na licu mjesta dok ne dođe hitna pomoć. Aplikacija treba sadržavati funkciju gdje korisnik pritiskom na dugme može pozvati hitnu pomoć. U slučaju da korisnik doživi nesreću, a u tom trenutku ne zna gdje se nalazi, preko aplikacije bi se trebalo doći do trenutne lokacije gdje se uređaj/korisnik nalazi te isto tako omogućiti hitan poziv. Popis svih mogućih zahtjeva te njihov prioritet koje korisnik može koristiti u aplikaciji vjerno pokazuje tablica 4.1.[10]

Tablica 4.1. Popis zahtjeva

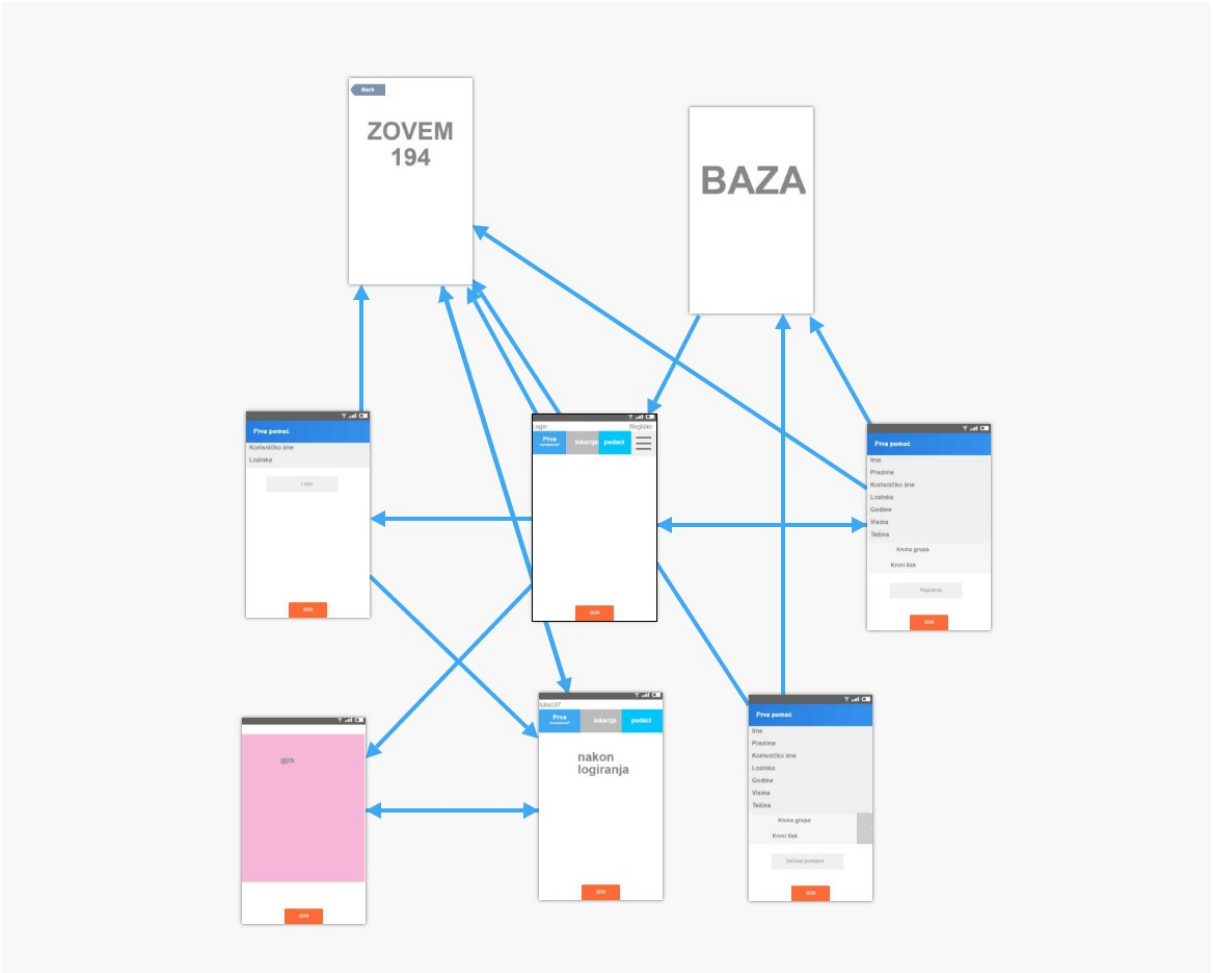
ID	Status	Prioritet	Opis
			Generalni zahtjevi korisnika
1	A	1	U sustavu postoje 3 različite kartice i jedan menu: Prva pomoc, Lokacija i Korisnički podaci
2	A	1	Na kartici PrvaPomoc nalaze se ozljede koje korisnik može odabrati te vidjeti željene upute
3	A	1	Korisnik klikom dolazi do uputa prve pomoći za željenu ozljedu
4	A	1	Korisnik može klikom na dugme nazvati Hitnu pomoć
5	A	1	Korisnik povlačenjem ili klikom dolazi do kartice Lokacija
6	A	1	Korisnik klikom na dugme Lokacija dobiva koordinate svoje lokacije
7	A	1	Korisnik povlačenjem ili klikom dolazi na treću karticu
8	A	2	Na 3. kartici korisnik može vidjeti svoje podatke
9	A	2	Korisnik se može registrirati u aplikaciju
10	A	2	Korisnik se može logirati u aplikaciju
11	A	3	Korisnik se može odlogirati iz aplikacije

Pri pokretanju aplikacije treba se otvoriti prozor s tri kartice i *menu* dugmetom. Na prvoj kartici se nalaze upute prve pomoći koje su sortirano prikazane po vrstama ozljeda, ovisno o kakvoj je ozljedi riječ. Na drugoj kartici se nalazi dugme koji vodi do lokacije na kojoj se uređaj trenutno nalazi, a na trećoj kartici se korisniku ispisuju njegovi osobni podaci ukoliko se korisnik registrira i prijavi u aplikaciju.

Korisnik se može registrirati sa svojim podacima te prijaviti s istim ako želi u aplikaciju. Klikom na menu dugme otvara se padajući izbornik te se preko njega dolazi na prozor za registriranje i prijavu.

4.2. Model aplikacije

Kako je već ranije rečeno, prije početka bilo kakve izrade aplikacije, najbitnije je imati ideju i tu ideju prebaciti na papir. Model aplikacije koju opisuje ovaj diplomski rad je rađen je u online alatu naziva *Fluid UI* koji je već ranije opisan u poglavlju 3.4. *Fluid User Interface.* Prikaz aktivnosti, programa (eng. Widget) te poveznica između njih prikazan je na slici 4.1.[11]



Slika 4.1. Model aplikacije

4.3. Aktivnost i Android Manifest

Aktivnost (eng. *Activity*) je jedan od osnovnih gradivnih elemenata *Android* aplikacije. Svaki ekran koji sadrži aplikacija je u osnovi nova aktivnost. Pri programiranju je definirano da se funkcionalnost aktivnosti i njen izgled rade odvojeno. Izgled se uobičajeno definira u *XML*-u (o čemu se već pričalo u poglavlju 3.5), dok se funkcionalnost aktivnosti radi unutar paketa u određenoj *.java* datoteci (unutar istoimene klase, budući da je *Java* u potpunosti objektno orijentirana). [12]

S druge strane, svaki Android projekt mora sadržavati tzv. *Manifest* datoteku imena *AndroidManifest.xml* koja čini korijen svakog Android projekta. Bilo koji gradivni element koji se napravi u aplikaciji mora biti prijavljen i definiran u *AndroidManifest.xml* datoteci. Ako se u projekt nove aktivnosti dodaju kroz izbornik *Android Studija*, manifest datoteka se automatski ažurira tako da uključi nove aktivnosti, ali ju je također moguće i ručno uređivati. [13]

Na početku svakog manifesta definiran je cijeli broj koji označava trenutnu inačicu i služi za ažuriranje aplikacije. Kako *Android* aplikacija nema automatski uključene dozvole za razne stvari, u manifestu se moraju uključiti određene dozvole kako bi aplikacija mogla pravilno funkcionirati. Naredbom <*uses-permission...*> u ovom diplomskom radu su uključene dozvole za spajanje na Internet, dohvat lokacije te za hitno pozivanje prve pomoći (Sl. 4.3.1).

Najbitniji dio manifest datoteke je <*aplication..*> jer on upravlja svim dijelovima manifesta te koristi određene komponente aplikacije. Preko atributa *android:label* je označeno ime aplikacije, a atribut *android:icon* označava ikonu aplikacije. <*aplication..*> je također jako bitan dio manifest datoteke jer su u njemu prijavljene sve aktivnosti koje se nalaze u aplikaciji preko atributa *activity android:name.* Ona aktivnost koji ispod sebe ima atribut <*intent-filter...*> te u njemu funkcije *action.MAIN* i *category.LAUNCHER* je glavna aktivnost i on će pri prvom pokretanju aplikacije prvi prikazati.

AndroidManifest.xml datoteka također sadrži i atribut *android:theme* kojim se definira tema korištena u aplikaciji. Izgled cijele *AndroidManifest.xml* datoteke prikazuje slika 4.2.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.stjepanlukac.prvapomoc">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".LoginActivity" />
        <activity android:name=".RegisterActivity" />
        <activity android:name=".UserAreaActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

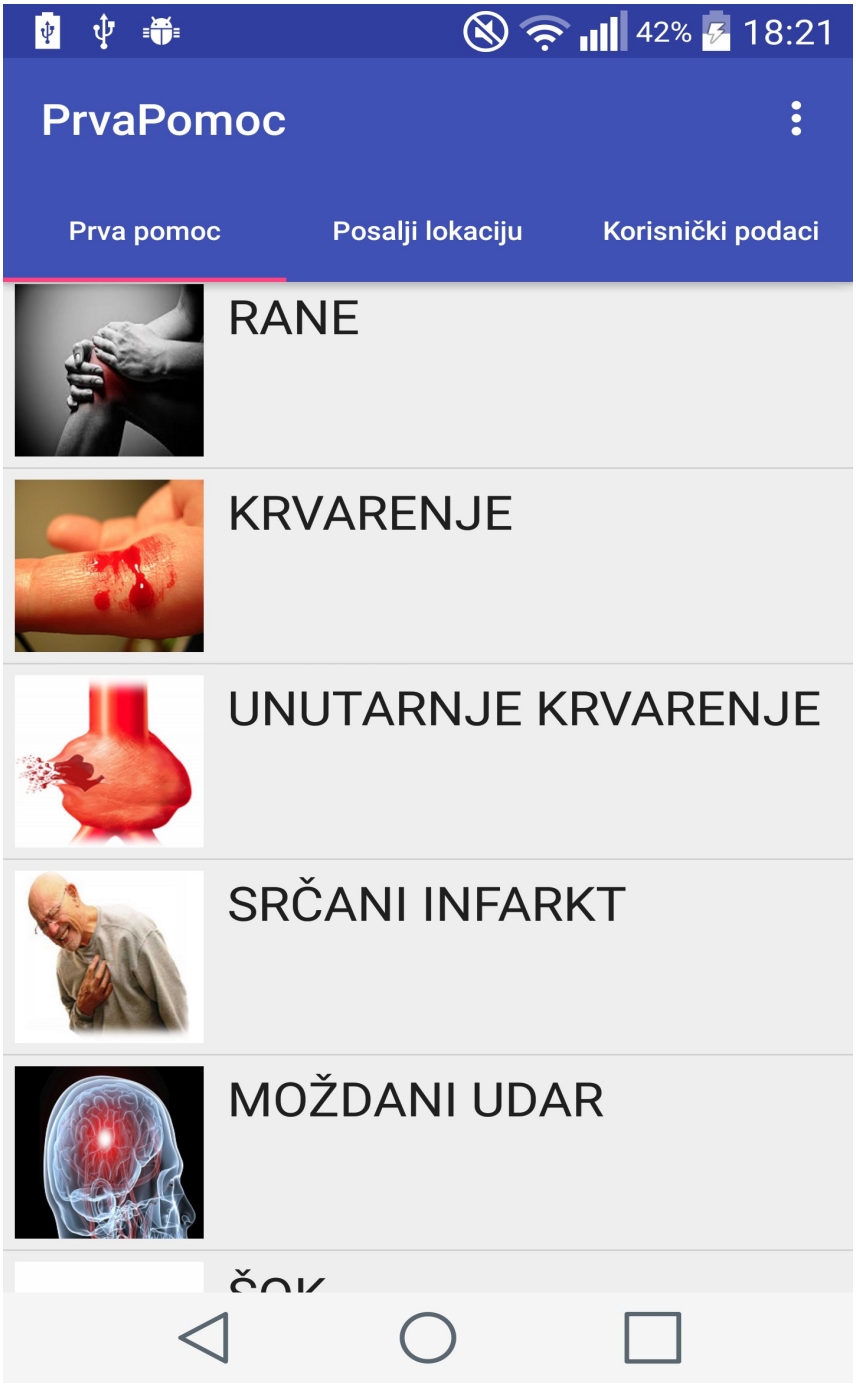
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SrcaniInfarkt" />
        <activity android:name=".MozdaniUdar" />
        <activity android:name=".Sok" />
        <activity android:name=".Rane" />
        <activity android:name=".Krvarenje" />
        <activity android:name=".UnutarnjeKrvarenje" />
        <activity android:name=".EpilepticniNapad" />
        <activity android:name=".Gusenje" />
        <activity android:name=".Opekline" />
        <activity android:name=".StrujniUdar" />
        <activity android:name=".UdarGroma" />
        <activity android:name=".ToplinskiUdar" />
        <activity android:name=".Vrucica" />
        <activity android:name=".Smrzotine" />
        <activity android:name=".Pothladenost" />
        <activity android:name=".Prijelomi" />
        <activity android:name=".OzljedeKraljeznice" />
        <activity android:name=".AmputacijskeOzljede" />
        <activity android:name=".BocniPolozaj" />
        <activity android:name=".Otrovanja" />
        <activity android:name=".Lokacija" />
        <activity android:name=".Dohvati_User_Data" />
        <activity android:name=".Posalji_User_Data"></activity>
    </application>

</manifest>
```

Slika 4.2. AndroidManifest.xml datoteka.

4.4. Glavno sučelje

Pri prvom pokretanju aplikacije otvara se glavno sučelje koje se sastoji od tri kartice i jednog *menu* dugmeta. Naziv prve kartice je *Prva Pomoc* i sadrži upute sa svim kronološki poredanim ozljedama. Druga kartica ima naziv *Lokacija* te se preko nje dolazi do željene lokacije i treća kartica nosi naziv *Korisnički Podaci* i na njoj korisnik u svakom trenutku može doći do svojih podataka. Izgled glavnog sučelja prikazan je na slici 4.3.



Slika 4.3. Izgled glavnog sučelja

Kartice su rađene radi lakše navigacije u aplikaciji i napravljene su pomoću fragmenta koji su opisani u poglavlju 3.6. Izgled kartica je napravljen u *XML* programskom jeziku. Prema slici 4.4. je napravljena nova *XML* datoteka imena *toolbar_layout.xml* u kojoj je dizajnirana alatna traka glavnog sučelja.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="wrap_content"
    android:background="?attr/colorPrimary"
    android:minHeight="?attr/actionBarSize"
    android:fitsSystemWindows="true"
    android:id="@+id/toolBar"
    app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
    >

</android.support.v7.widget.Toolbar>
```

Slika 4.4. Dizajn toolbar-a

Nakon toga, *toolbar_layout.xml* je atributom *<include...>* implementiran u *XML* datoteku naziva *activity_user_area* koja čini cijeloukupni dizajn glavne alatne trake u kojem se nalaze kartice.

Kako bi kartice bile kreirane i vidljive, kreirana je klasa *ViewPagerAdapter.java* koja popunjava listu fragmentima koji imaju svoje ime. Slika 4.5. prikazuje *ViewPagerAdapter* klasu.

```
public class ViewPagerAdapter extends FragmentPagerAdapter {

    ArrayList<Fragment> fragments = new ArrayList<>();
    ArrayList<String> tabTitles = new ArrayList<>();

    public void addFragments(Fragment fragments, String titles){

        this.fragments.add(fragments);
        this.tabTitles.add(titles);
    }
```

Slika 4.5. ViewPagerAdapter klasa

Nakon kreiranja liste fragmenata u *UserAreaActivity* klasi se poziva *activity_user_area layout* koji je ranije dizajniran XML jezikom. Pozivanjem metode *viewPagerAdapter* i pozivom metode *AddFragments* kreiraju se kartice na mjestu alatne trake glavnog sučelja (Sl.4.6.).

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_area);
```

```
    //referenca na toolbar kad sam radio tabove
    toolbar = (Toolbar) findViewById(R.id.toolBar);
    setSupportActionBar(toolbar);

    tabLayout = (TabLayout) findViewById(R.id.tabLayout);
    viewPager = (ViewPager) findViewById(R.id.viewPager);


    viewPagerAdapter = new ViewPagerAdapter(getSupportFragmentManager());

    viewPagerAdapter.addFragments(new PrvaPomoc(), "Prva pomoc");
    viewPagerAdapter.addFragments(new PosaljiLokaciju(), "Dohvati lokaciju");
    viewPagerAdapter.addFragments(new TreciFragment(), "Korisnički podaci");

    viewPager.setAdapter(viewPagerAdapter);
    tabLayout.setupWithViewPager(viewPager);
}
```

Slika 4.6. Inicijalizacija i kreiranje kartica pomoću fragmenata

Menu dugme koji je također sastavni dio glavnog sučelja dizajniran je *XML* jezikom u datoteci *menu_main.xml*. Zamišljen je kao padajući izbornik preko kojeg korisnik dolazi do prostora za registriranje/prijavu ili preko kojeg se može odjaviti iz aplikacije. *menu_main.xml* datoteka je napravljena od atributa *android:title* s kojim je definirano ime *te android:orderInCategory* kojim je definiran položaj u padajućem izborniku. (Sl. 4.7.).

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

        <item
            android:id="@+id/menu_Log_Reg"
            android:orderInCategory="1"
            app:showAsAction="never"
            android:title="@string/Log_Reg" />

    <item
        android:id="@+id/menu_Logout"
        android:orderInCategory="2"
        app:showAsAction="never"
        android:title="@string/menu_Logout" />
</menu>
```

Slika 4.7. Dizajn menu dugmeta

Kako bi *menu* dugme bilo kreirano i funkcionalano, u *UserAreaActivity* klasi je pozvana *menu_main.xml* datoteka. *Switch* petljom je zadano da se pokrene nova aktivnost ako korisnik klikne na određeni item koji je napravljen u *XML* datoteci (Sl. 4.8.).

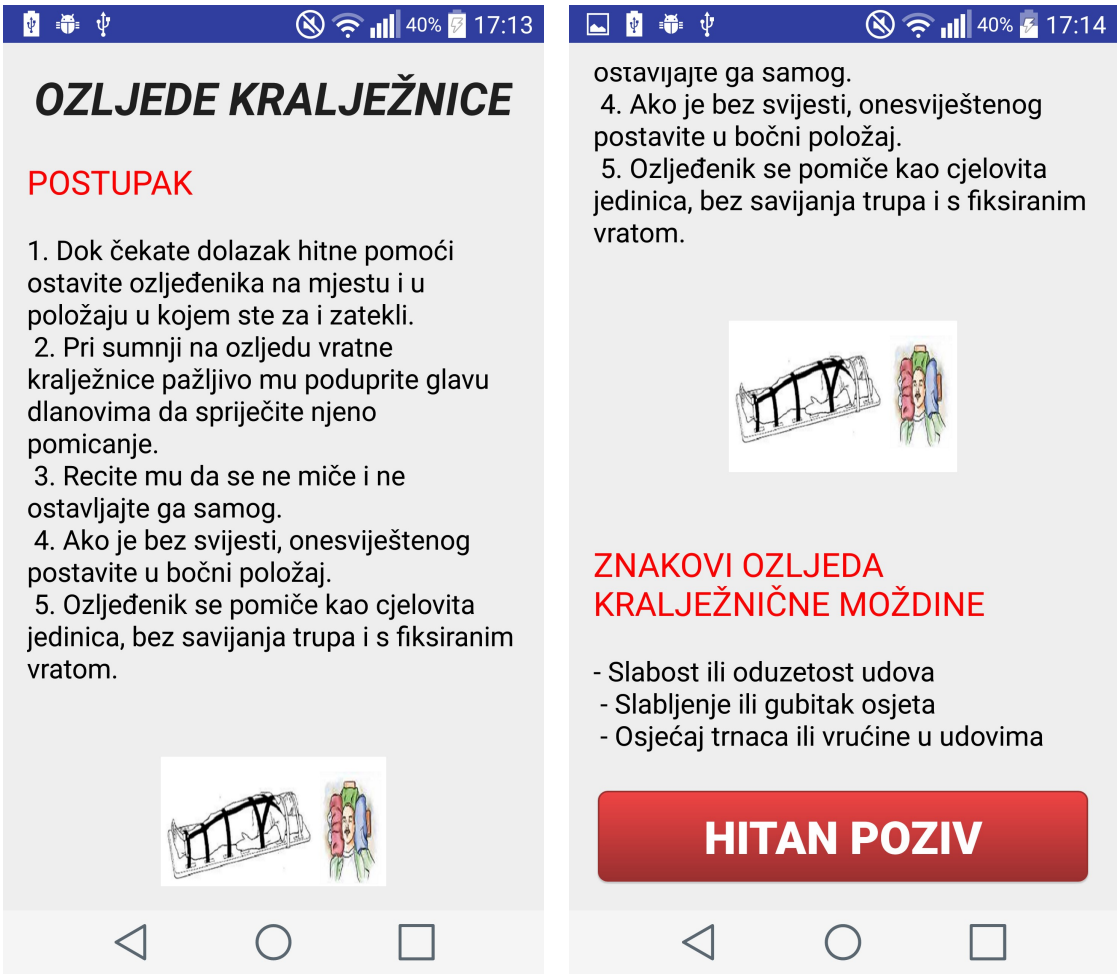
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {

    //meni koji dovodi do registriranja/logiranja
    switch (item.getItemId()){
        case R.id.menu_Log_Reg:
            if(item.isChecked())
                item.setChecked(false);
            else
                item.setChecked(true);
            Intent intent = new Intent(UserAreaActivity.this,
LoginActivity.class);
            UserAreaActivity.this.startActivity(intent);
            return true;
```

Slika 4.8. Kreiranje menu dugmeta

4.5. Upute prve pomoći

Kao što je već spomenuto, na prvoj kartici glavnog sučelja nalaze se upute prve pomoći. Poredane su po vrstama ozljeda gdje svaka ozljeda sadrži popis uputa kako pomoći unesrećenoj osobi u tom trenutku dok ne dođe hitna pomoć na mjesto nezgode. Na kraju uputa se nalazi dugme *„HITAN POZIV“* gdje se pritiskom na njega automatski poziva hitna pomoć. Popis ozljeda prikazan je na slici 4.3 u poglavlju 4.4, a same upute i dugme prikazani su na slici 4.9.



a) Prvi dio *b) Drugi dio*

Slika 4.9. Upute prve pomoći

Svaka ozljeda predstavlja posebnu aktivnost , a sve aktivnosti su implementirane u aktivnost *PrvaPomoc* koji je napravljen preko fragmenata. Dizajn aktivnosti *Prva Pomoc* definiran je *fragment_prva_pomoc.xml*-om koji se sastoji od jednog <ListView...> u kojemu su stavljeni popisi ozljeda. *<ListView...>* se sastoji od atributa *android:layout_width, android:layout_height* kojima je definirana visina i širina te atributa *android:paddingTop* kojim je definiran razmak između alatne trake i prvog ikone. (Sl.4.10.).

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.stjepanlukac.prvapomoc.PrvaPomoc">

    <!-- TODO: Update blank fragment layout -->

    <!-- ListView u kojeg ćemo staviti popis ozljeda -->

    <ListView
```

```
        android:paddingTop="100dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/ozljedeListView"
        >
    </ListView>
</FrameLayout>
```

Slika 4.10. fragment_prva_pomoc.xml

Kako bi se mogla staviti slika i tekst u *listView*, kreiran je *custom_row_ozljede.xml* koji se sastoji od jednog *<ImageView...>* i jednog *<TextView...>*. *<ImageView...>* sadrži atribute *android:layout_width, android:layout_height* kojima je definirana visina i širina, zatim *android:src* preko kojega se učita slika te atributa *android:id* preko kojeg se svaka ta slika poziva u *PrvaPomoc.java* datoteci. Izgled *custom_row_ozljede.xml* datoteke prikazuje slika 4.11.

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

    <ImageView
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:id="@+id/ozljedeSlika"
        android:src="@drawable/slika1"
        android:layout_margin="5dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Large Text"
        android:id="@+id/ozljedeText"
        android:layout_margin="5dp" />
</LinearLayout>
```

Slika 4.11. custom_row_ozljede.xml

U datoteci *CustomAdapterOzljede.java* pozvan je *custom_row_ozljede.xml* metodom *getView()*. U toj metodi su također i reference na *<TextView...>* i *<ImageView ...>*koji su definirani u *XML*-u. Klasa *CustomAdapterOzljede* sadrži konstuktor koji prima popis ozljeda i slike koje su spremljene u listu. Slika 4.12. prikazuje izgled *CustomAdapterOzljede.java* klase.

```
public class CustomAdapterOzljede extends ArrayAdapter<String> {

    private final int[] imgs;
```

```
    public CustomAdapterOzljede(Context context, String[] popis_ozljeda,
int[] imgs) {
        super(context, R.layout.custom_row_ozljede, popis_ozljeda);
        this.imgs = imgs;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        LayoutInflater ozljedeInflater = LayoutInflater.from(getContext());
        View customOzljedeView =
ozljedeInflater.inflate(R.layout.custom_row_ozljede, parent, false);

        //reference za niz, sliku i tekst
        String singleOzljedeItem = getItem(position);
        TextView ozljedeText = (TextView)
customOzljedeView.findViewById(R.id.ozljedeText);
        ImageView ozljedeSlika = (ImageView)
customOzljedeView.findViewById(R.id.ozljedeSlika);

        ozljedeText.setText(singleOzljedeItem);
        ozljedeSlika.setImageResource(imgs[position]);

        return customOzljedeView;
    }
}
```

Slika 4.12. CustomAdapterOzljede.java

Na kraju, u *PrvaPomoc.java* datoteci poziva se *ListView* koji je definiran u *fragment_prva pomoc* datoteci. Pozivanje se obavlja u *OnCreateView* metodi budući da se radi o fragmentima te se ujedno kreira lista s popisom ozljeda. Kako se radi s listama, ovisno o položaju, if/else naredbom se otvara ona aktivnost (ozljeda) na koju je korisnik kliknuo. Dio *PrvaPomoc.java* datoteke pokazuje slika 4.13.

```
 @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_prva_pomoc, container,
false);

        final int[] imgs = {R.drawable.rane, R.drawable.krvarenje,
R.drawable.unutarnje_krvarenje,
                R.drawable.srcani_infarkt, R.drawable.mozdani_udar,
R.drawable.sok, R.drawable.epilepticni_napad, R.drawable.gusenje,
                R.drawable.opekline, R.drawable.strujni_udar,
R.drawable.udar_groma, R.drawable.toplinski_udar, R.drawable.vrucica,
                R.drawable.smrzotine, R.drawable.pothladenost,
R.drawable.prijelomi, R.drawable.ozljede_kraljeznice,
R.drawable.amputacijske_ozljede,
                R.drawable.bocni_polozaj, R.drawable.otrovanja
```

```
        };

        //popis ozljeda
        String[] popis_ozljeda = { "RANE", "KRVARENJE", "UNUTARNJE
KRVARENJE", "SRČANI INFARKT",
        "MOŽDANI UDAR", "ŠOK", "EPILEPTIČNI NAPAD", "GUŠENJE", "OPEKLINE",
"STRUJNI UDAR", "UDAR GROMA", "TOPLINSKI UDAR",
        "VRUĆICA", "SMRZOTINE", "POTHLAĐENOST", "PRIJELOMI", "OZLJEDE
KRALJEŽNICE", "AMPUTACIJSKE OZLJEDE", "BOČNI POLOŽAJ",
        "OTROVANJA"};

        ListAdapter ozljedeAdapter = new CustomAdapterOzljede(getContext(),
popis_ozljeda, imgs);

        ListView ozljedeListView = (ListView)
view.findViewById(R.id.ozljedeListView);
        ozljedeListView.setAdapter(ozljedeAdapter);

        //Listener kad se klikne na pojedini item
        ozljedeListView.setOnItemClickListener(
                new AdapterView.OnItemClickListener(){
                    @Override
                    public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
                        String ozljede =
String.valueOf(parent.getItemAtPosition(position));
                        Toast.makeText(getActivity(), ozljede,
Toast.LENGTH_SHORT).show();


                        //otvaranje novog aktivitija na klikom na pojedini
item u listi
if (position == 0) {
Intent lala = new Intent(view.getContext(), Rane.class);
startActivityForResult(lala, 0)
}else if (position == 1){
 Intent lala = new Intent(view.getContext(), Krvarenje.class);
 startActivityForResult(lala, 0);

 }else if (position == 2){
 Intent lala = new Intent(view.getContext(),

.

.

.

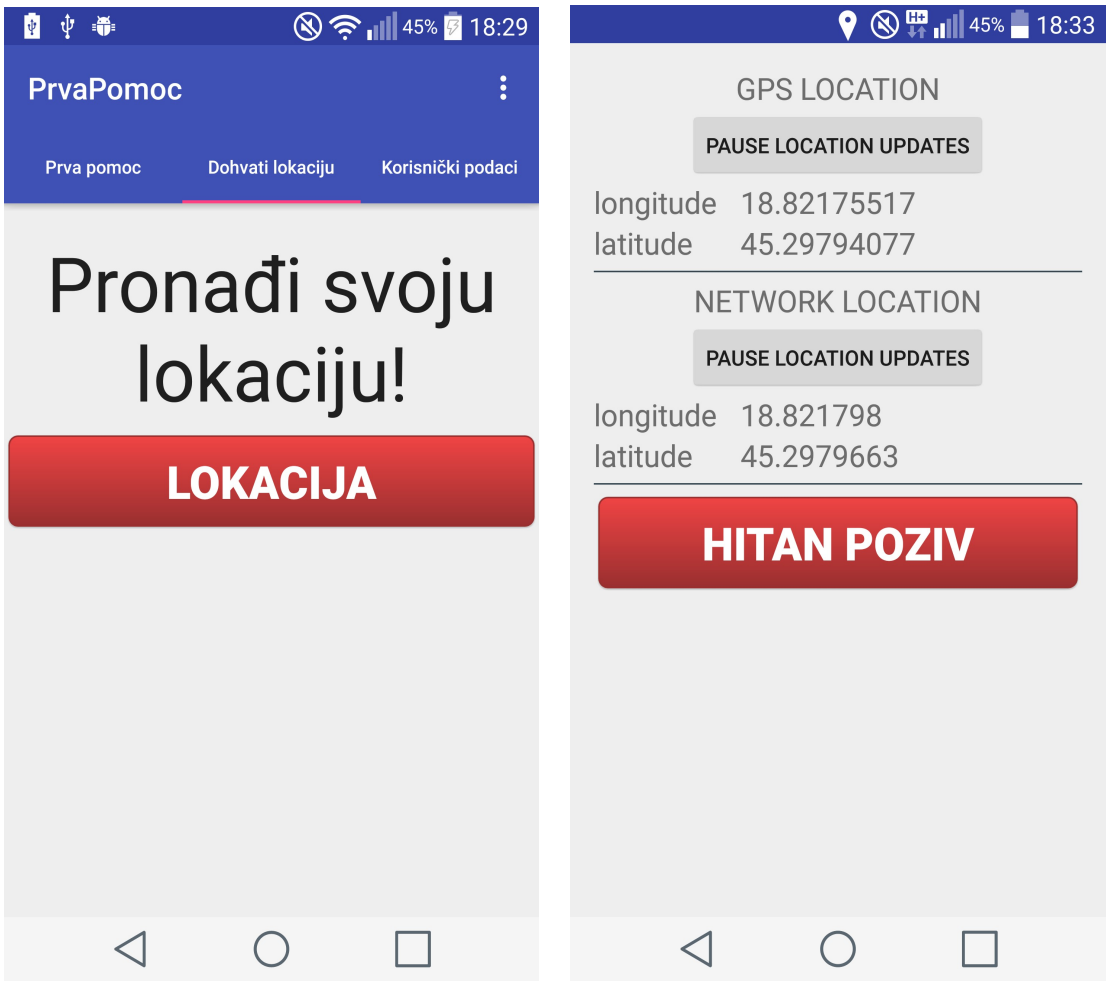
.
```

Slika 4.13. PrvaPomoc.java

4.6. Lokacija

Aktivnost *Dohvati Lokaciju* nalazi se na drugoj kartici te je jedan od funkcionalnih dijelova aplikacije preko kojeg se dolazi do koordinata trenutne lokacije gdje se nalazi uređaj. Odabirom se može ili preko GPS signala ili preko pružatelja mrežnih usluga doći do koordinata

trenutne lokacije uređaja odnosno korisnika te se ujedno može uputiti hitan poziv kako bi se pozvala hitna pomoć. Glavna ideja i svrha ove funkcije je da unesrećena osoba u trenutku nesreće može na što brži način pronaći svoju lokaciju te pozvati hitnu pomoć. Klikom na dugme „LOKACIJA“ se odlazi na aktivnost gdje se ispisuju koordinate trenutne lokacije (Sl.4.14.).



a) Dugme lokacija b) Koordinate

Slika 4.14. Dohvati Lokaciju

Izgled aktivnosti *Dohvati Lokaciju* definiran je *fragment_posalji_lokaciju.xml* datotekom, koja se sastoji od jednog *<TextView..>* i jednog dugmeta. *<TextView...>* sadrži atribute *android:textSize* i *android:textAlignment* kojim su definirani veličina i položaj teksta, a *<Button...>* uz atribute za definiranje širine i visine, sadrži atribut *android:background* kojim se definira stil oblikovanja samog dugmeta. Kako bi se pritiskom na dugme „*LOKACIJA*“ otišlo na aktivnost koja ispisuje koordinate, *PosaljiLokaciju.java* datoteka je u *onCreateView* (opet se radi

s fragmentima) metodi pozvala *XML* datoteku te je sa *setOnClickListener* metodom omogućen klik na dugme (Sl. 4.15.).

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {


    // Inflate the layout for this fragment
    final View view = inflater.inflate(R.layout.fragment_posalji_lokaciju,
container, false);

    posalji_Lokaciju = (Button) view.findViewById(R.id.posalji_lokaciju);

    posalji_Lokaciju.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent posalji_Lokaciju = new Intent(view.getContext(),
Lokacija.class);
            view.getContext().startActivity(posalji_Lokaciju);
        }
    });
    return view;
}
```

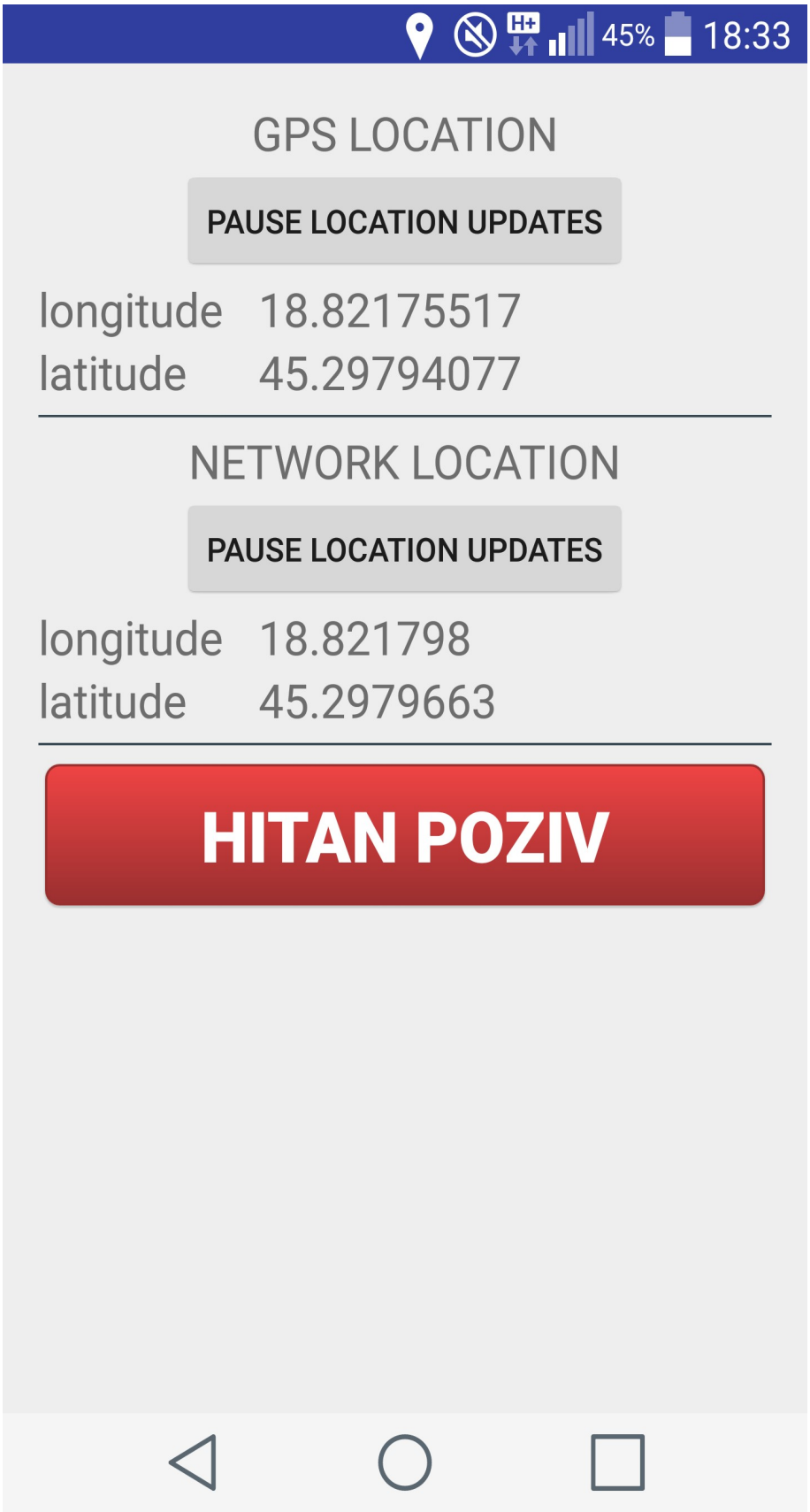
Slika 4.15. setOnClicListener medota

Kako je za dohvaćanje lokacije potrebno spajanje na Internet, u *AndroidManifest.xml* datoteci su dodane dvije dozvole (eng. *Permission*). Prva dozvola se odnosi na pristup internetu, a druga za dohvaćanje same lokacije uređaja. Korištenje dozvola prikazuje slika 4.16.

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

Slika 4.16. Dodavanje dozvola u AndroidManifest.xml

Izgled aktivnosti koji pritiskom na dugme ispisuje koordirane trenutne lokacije dizajniran je *activity_lokacija.xml* datotekom. Sastoji se od deset *<TextView...>*, dva separatora koji odvajaju prostor za dohvaćanje lokacije preko *GPS*-a i prostor za dohvaćanje lokacije preko pružatelja mrežnih usluga te od tri *<Button...>*. Separatori su definirati tagom *<View...>* pod kojim se nalaze atributi za definiranje širine i visine te atribut *android:background* koji je definirana pozadina. Izgled *activity_lokacija.xml* datoteke prikazuje slika 4.17.



Slika 4.17. activity_lokacija.xml

Reference na programe (eng. *Widget*) *iz activity_lokacija.xml* datoteke se nalaze u *Lokacija.java* datoteci. Moguće je da je u postavkama uređaja defaultno isključena opcija lokacije pa se zato prije dohvaćanja informacija o lokaciji treba provjeriti je li opcija uključena ili isključena. Kako bi se to provjerilo, implementirana je metoda *isLocationEnabled()* (Sl. 4.18.).

```
private boolean isLocationEnabled() {
    return locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||

locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
}
```

Slika 4.18. Provjera postavke lokacije

Kako bi to funkcioniralo, šalje se upit *LocationMenager*-u je li dostupan *GPS* ili pružatelj mrežnih usluga. U slučaju da je lokacija u postavkama isključena, pomoću *showAlert()* metode se otvara dijalog s postavkama i lokacija se može uključiti te se vratiti u aplikaciju (Sl. 4.19.).

```
//ako je lokacija isključena, otvori dijalog kako bi korisnik mogao uključiti
lokaciju u postavkama
private void showAlert() {
    final AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setTitle("Enable Location")
            .setMessage("Your Locations Settings is set to 'Off'.\nPlease
Enable Location to " +
                    "use this app")
            .setPositiveButton("Location Settings", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface paramDialogInterface, int
paramInt) {
                    Intent myIntent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                    startActivity(myIntent);
                }
            })
            .setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface paramDialogInterface, int
paramInt) {
                }
            });
    dialog.show();
}
```

Slika 4.19. Uključivanje lokacije u postavkama

Kako bi se ostvarilo ažuriranje dohvaćanja koordinata preko *GPS* ili pružatelja mrežnih usluga, koriste se *requestLocationUpdates()* metode. Za potrebe ovog diplomskog rada, koristi

se *requestLocationUpdates(String provider, int updateTime, int updateDistance, LocationListener listener)* metoda. S *updateTime* se mjeri frekvencija koliko se često traži zahtjev za ažuriranje, dok se *updateDistance* odnosi na duljinu koja je pređena prije nego što se tražilo ažuriranje.

Važna stvar na koju se treba misliti je ta što funkcija dohvaćanja lokacije crpi ogromni dio baterije uređaja. Kako bi se to spriječilo u slučaju kada korisnik ne treba dohvatiti lokaciju, a ostala mu je aplikacija raditi u pozadini, implementirana je *removeUpdates()* metoda gdje se klikom na dugme može zaustaviti i pokreniti funkcija dohvaćanja (Sl. 4.20.).

```
public void toggleGPSUpdates(View view) {
    if (!checkLocation())
        return;
    Button button = (Button) view;
    if (button.getText().equals(getResources().getString(R.string.pause))) {
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            // TODO: Consider calling
            //    ActivityCompat#requestPermissions
            // here to request the missing permissions, and then overriding
            //   public void onRequestPermissionsResult(int requestCode,
String[] permissions,
            //                                          int[] grantResults)
            // to handle the case where the user grants the permission. See
the documentation
            // for ActivityCompat#requestPermissions for more details.
            return;
        }
        locationManager.removeUpdates(locationListenerGPS);
        button.setText(R.string.resume);
    } else {
        locationManager.requestLocationUpdates(
                LocationManager.GPS_PROVIDER, 2 * 60 * 1000, 10,
locationListenerGPS);  //2*60*1000 - update svakih 5 minuta
        button.setText(R.string.pause);
    }
}
```

Slika 4.20. Uključivanje/isključivanje dohvaćanja lokacije

LocationListener je sučelje za primanje ažuriranja lokacije iz *LocationManager*-a. Sadrži četiri metode, *onLocationChanged()*, *onStatusChanged()*, *onProviderDisabled()* i *onProviderEnabled()*. U ovom diplomskom radu koristila se samo metoda *onLocationChanged()* koja se pozivala svaki puta kada je došlo ažuriranje od *LocationManager*-a (Sl. 4.21.).

```
private final LocationListener locationListenerGPS = new LocationListener() {
    public void onLocationChanged(Location location) {
```

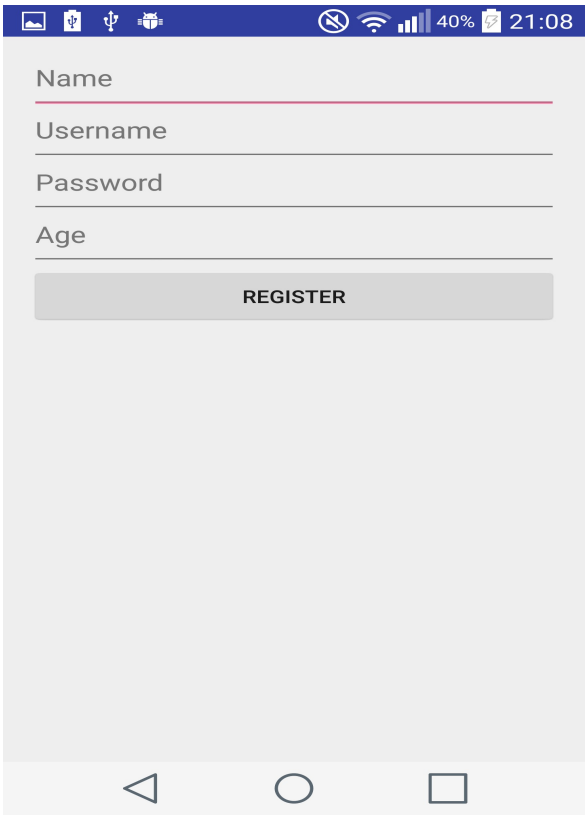
```
        longitudeGPS = location.getLongitude();
        latitudeGPS = location.getLatitude();

        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                longitudeValueGPS.setText(longitudeGPS + "");
                latitudeValueGPS.setText(latitudeGPS + "");
                Toast.makeText(Lokacija.this, "GPS Provider update",
Toast.LENGTH_SHORT).show();
            }
        });
    }
```

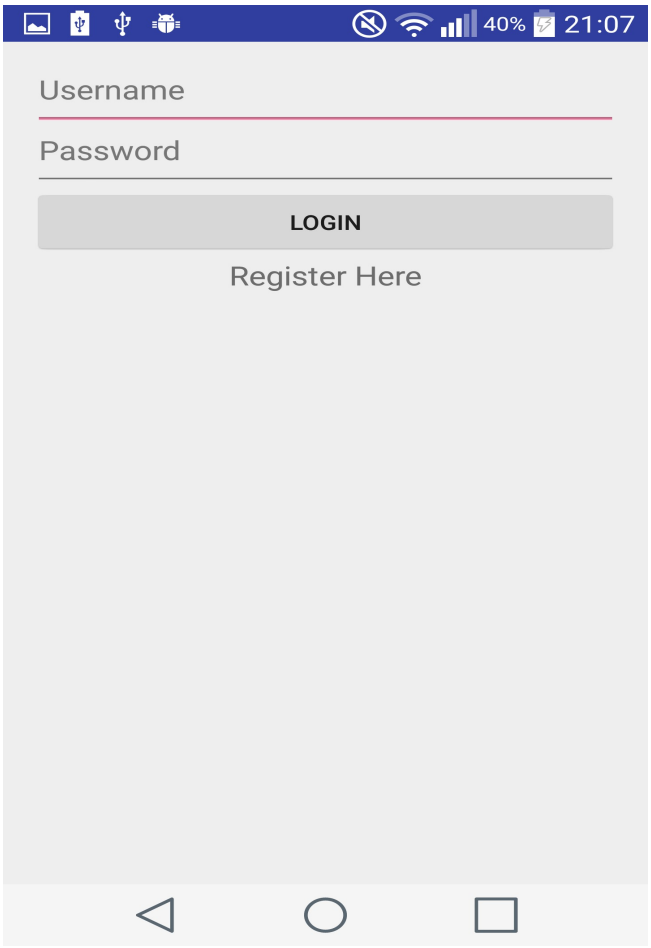
Slika 4.21. LocationListener

4.7. Baza podataka - registriranje i logiranje

Aplikacija nudi korisniku mogućnost registriranja i logiranja u aplikaciju. Nakon što se korisnik logira sa svojim podacima, aplikaciju mu omogućuje da upiše svoje podatke poput visine, mase, tlaka itd. Slika 4.22. prikazuje aktivnost za registriranje, a slika 4.23. aktivnost za logiranje u aplikaciju.



Slika 4.22. activity_register.xml



Slika 4.23. activity_login.xml

Obje aktivnosti su dizajnirane *XML* jezikom, a sastoje se od <*TextView...*>, <*Button...*> i <*EditText...*> elemenata. <*EditText...*> omogućuje korisniku da upisuje podatke u aplikaciju.

Kako bi registriranje i logiranje radilo, potrebna je baza u koju će se upisivati i ispisivati unešeni podaci. Za potrebe ovog diplomskog rada, korištena je online baza na besplatnom serveru naziva 000webhost. [14] Pomoću *PhpMyAdmin*-a kreirane su dvije talice naziva *user* i *podaci.* Tablica *user* služi za upisivanje korisničkih podataka, a sastoji se od polja *user_id*, *name*, *username*, *age* i *password*. Tablica *podaci* se sastoji od polja *id*, *username*, *password*, *krv*, *tlak1*, *tlak2*, *tezina*, *visina* i *user_id*. Slika 4.24. prikazuje izgled tablica na serveru.

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	user_id	int(4)			No		auto_increment	
☐	name	varchar(16)	latin1_general_ci		No			
☐	username	varchar(16)	latin1_general_ci		No			
☐	age	tinyint(4)			No			
☐	password	varchar(16)	latin1_general_ci		No			

	Field	Type	Collation	Attributes	Null	Default	Extra	Action
☐	id	int(100)			No		auto_increment	
☐	username	varchar(20)	latin1_general_ci		Yes	*NULL*		
☐	password	varchar(20)	latin1_general_ci		Yes	*NULL*		
☐	krv	varchar(20)	latin1_general_ci		Yes	*NULL*		
☐	tlak1	int(11)			Yes	*NULL*		
☐	tlak2	int(11)			Yes	*NULL*		
☐	tezina	double			Yes	*NULL*		
☐	visina	double			Yes	*NULL*		
☐	user_id	int(100)			No			

Slika 4.24. Tablice user i podaci

Povezivanje servera, baze podataka i same aplikacije napravljeno je pomoću php datoteka koje su digute na server, a ujedno pozvane u aplikaciji. Naredba *mysqli_connect("mysql6.000webhost.com", "a1665450_lukac07", "desert07", "a1665450_lukac07")* sadrži četiri parametra koja su potrebna za povezivanje. Prvi parametar definira ime servera koji se koristi, drugi parametar je korisničko ime, treći je lozinka, a četvrti parametar je ime baze na koju se aplikacija spaja. Upisivanje podataka u bazu je napravljeno pomoću naredbe *„INSERT INTO“*, a ispisivanje iz baze pomoću naredbe *„SELECT FROM“*. Sve varijable su spremane u *JSON* objekt koji se poziva u samoj aplikaciji.

RegisterRequest.java datoteka sadrži *url* preko kojega se aplikacija spaja na bazu. Isto tako u datoteci je konstuiran kostruktor *RegisterRequest* gdje se preko *POST* metode šalju podaci u *Register.php* te se tamo daje odgovor na poslan upit. Prikaz slanja upita u bazu pokazuje slika 4.25.

```
//ova klasa pravi request na register.php file na serveru i dobiva odgovor
kao string
public class RegisterRequest extends StringRequest {

    private static final String REGISTER_REQUEST_URL =
"http://stjepanlukac.comli.com/Register.php";
    private Map<String, String> params;

    //pravimo konstruktor koji će se prvi izvršiti kada se stvori instanca na
RegisterRequest klasu
    public RegisterRequest(String name, String username, int age, String
password, Odgovor.Listener<String> listener){

        /*preko POST metode šaljemo podatke u register.php i on nam daje
reposnse
        i kada volley završi request tada preko listenera obavještavamo da je
sve prošlo okej*/

        super(Method.POST, REGISTER_REQUEST_URL, listener, null);
        params = new HashMap<>();   //preko params pristupamo podacima
        params.put("name", name);
        params.put("username", username);
        params.put("password", password);
        params.put("age", age + "");
    }
    @Override
    public Map<String, String> getParams() {
        return params;
    }
}
```

Slika 4.25. RegisterRequest.java

Kada je upit poslan, u *RegisterActivity.java* datoteci se preko *JSON* objekta prima odgovor. Ako je odgovor uspješno prošao, registracija je uspješno obavljena, a ako nije, aplikacija će javiti iznimku (eng.*exception*) te će korisnik ponovo moći ponoviti registraciju (Sl.4.26.). Na isti način je obavljeno logiranje u aplikaciju.

```
@Override
public void onResponse(String response) {
        try {
            JSONObject jsonResponse = new JSONObject(response);
            boolean success = jsonResponse.getBoolean("success");

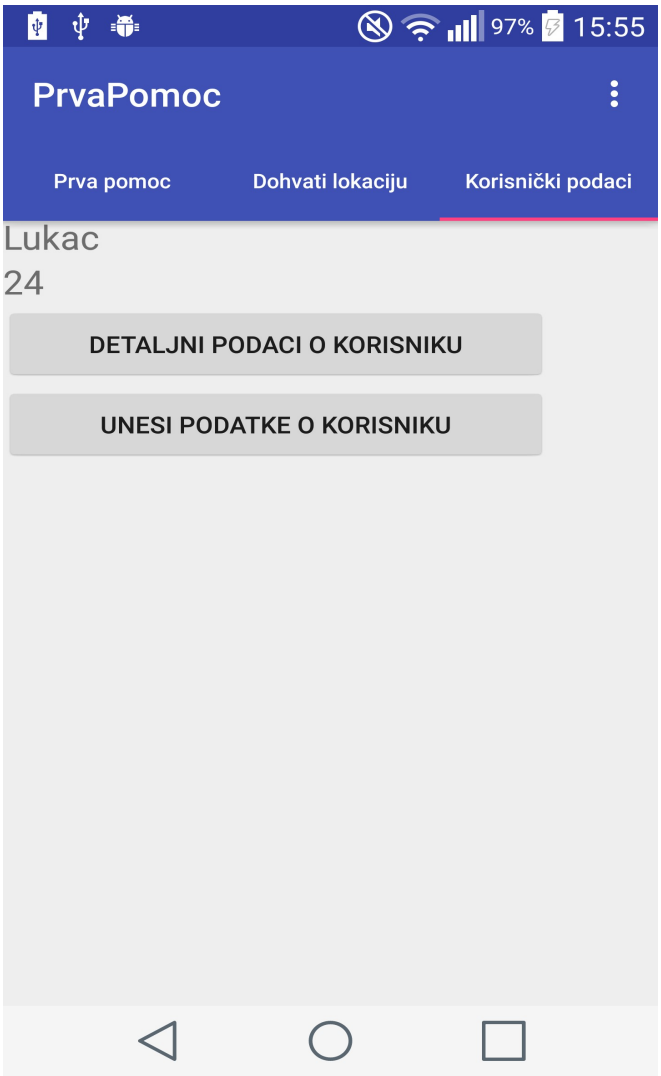
            if (success){
                Intent intent = new Intent(RegisterActivity.this,
LoginActivity.class);
                RegisterActivity.this.startActivity(intent);
            } else {
                AlertDialog.Builder builder = new
AlertDialog.Builder(RegisterActivity.this);
                builder.setMessage("Register Failed")
                        .setNegativeButton("Retry", null)
                        .create()
                        .show();
            }
```

```
        } catch (JSONException e) {
            e.printStackTrace();
        }
}
```

Slika 4.26. Odgovor – odgovor na upit

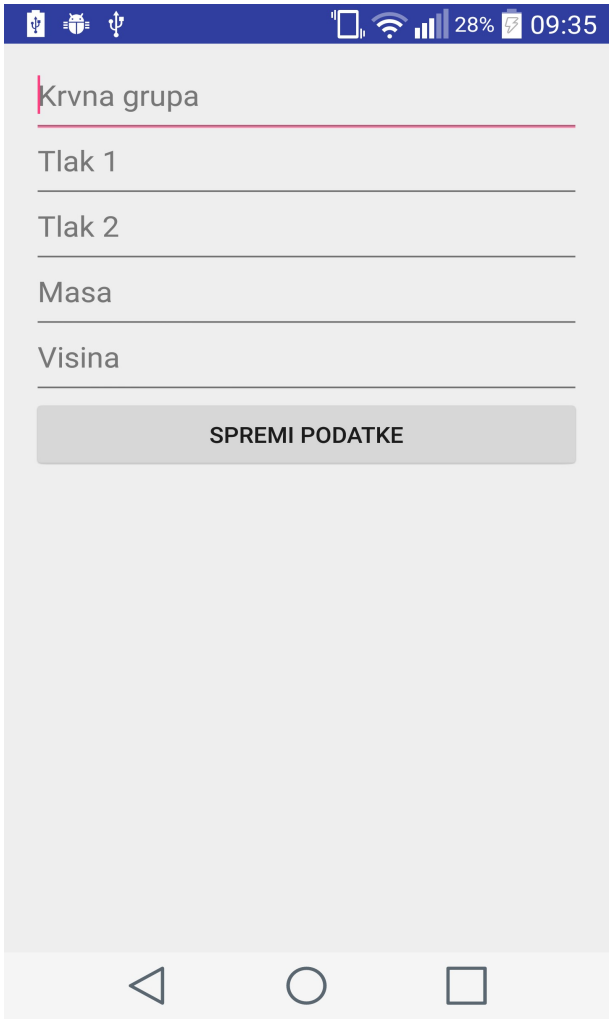
4.8. Korisnički podaci

Nakon registriranja i logiranja u aplikaciju, korisniku se na trećoj kartici *Korisnički podaci* otvaraju dodatne mogućnosti. Na početku se automatski prikazuje korisničko ime korisnika te njegove godine, a ispod toga otvaraju mu se opcije za izlistanje te upis dodatnih podataka. Prikaz kartice o korisničkim podacima pokazuje slika 4.27.



Slika 4.27. Kartica – Korisnički podaci

Klikom na dugme *„UNESI PODATKE O KORISNIKU"* korisniku se otvara *Posalji_User_data* aktivnost u kojoj može nadopuniti svoje podatke o krvnoj grupi, visokom i niskom tlaku, težini i visini. Izgled aktivnosti dizajniran je isto *XML* jezikom (Sl. 4.28.).



Slika 4.28. Kartica – Unos dodatnih korisničkih podataka

Upis i ispis podataka realiziran je na isti način kao i kod registriranja/logiranja u aplikaciju. Uz bazu na serveru, korišten je *Shared Preferances* pomoću kojeg su se zapisivali i dohvaćali trenutno zapisani podaci. Za upis podataka poziva se objekt *SharedPreference* gdje se metodom *edit()* dodaju vrijednosti koje korisnik unese i naredbom *commit()* sačuvaju promjene.

Pritiskom na*" SPREMI PODATKE"* u tablicu *podaci* na serveru se upisuju unešeni podaci te se automatski u *SharedPreferences* spremaju dvije vrijednosti: *username* i *password* (Sl. 4.29.).

```
takeData = getSharedPreferences(PREFS, Context.MODE_PRIVATE);
```

```
username = takeData.getString("username", "");
password = takeData.getString("password", "");
```

Slika 4.29. SharedPreferences

Kako je u *SharedPreferences*-u sačuvano o koji se korisnik logirao pomoću *username*-a i *password*-a, u *Dohvati_User_Data.java* datoteci se isto preko *jsonOdgovor*-a dohvaćaju podaci iz tablice podaci te se s naredbom *setText* šalju na ekran korisnika (Sl.4.30.).

```
bLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Response.Listener<String> responseListener = new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {

                //moramo pretvoriti request u JSON objekt kako bismo mogli
raditi s njim
                try {
                    JSONObject jsonResponse = new JSONObject(response);
                    boolean success = jsonResponse.getBoolean("success");
                    if (success) {
                        String krv = jsonResponse.getString("krv");
                        int tlak1 = jsonResponse.getInt("tlak1");
                        String tlakk1 = Integer.toString(tlak1);
                        int tlak2 = jsonResponse.getInt("tlak2");
                        String tlakk2 = Integer.toString(tlak2);
                        double tezina = jsonResponse.getDouble("tezina");
                        String tez = Double.toString(tezina);
                        double visina = jsonResponse.getDouble("visina");
                        String vis = Double.toString(visina);

                        tvKrv.setText(krv);
                        tvTlak1.setText(tlakk1);
                        tvTlak2.setText(tlakk2);
                        tvTezina.setText(tez);
                        tvVisina.setText(vis);
                    } else {
                        AlertDialog.Builder builder = new
AlertDialog.Builder(Dohvati_User_Data.this);
                        builder.setMessage("Login Failed")
                                .setNegativeButton("Retry", null)
                                .create()
                                .show();
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
```

Slika 4.30. Prikaz korisničkih podataka

5. ZAKLJUČAK

Ubrzani tempo, trčanje s jednog mjesta na drugo, nesmotrenost i brojni drugi faktori uzrokovani današnjim životom su dovoljan okidač za trenutak nepažnje koji dovodi do neželjenih ozljeda i nesreća. Upravo u tim trenucima, ljudi često budu u stanju šoka te ne znaju koje mjere trebaju poduzeti kako bi pomogli unesrećenoj osobi. Ovaj diplomski rad prikazuje jedno od rješenja tog problema u obliku android aplikacije s uputama prve pomoći.

Na početku diplomskog rada, dan je općeniti uvod o andrid operacijskom sučelju gdje je ukratko opisana povijest razvijanja androida, razne zanimljivosti te pojedine inačice androida koje su se razvijale postepeno kroz povijest. Nakon toga, opisane su tehničke značajke s kojima je realizirana android aplikacija. Opisan je programski jezik *Java* kojim je pisano i razvojno okruženje *Android Studio* pomoću kojeg je realizirana aplikacija. Navedena su i dva poslužitelja koja su se koristila. *Fluid User Interface* pomoću kojega je prvotna ideja pretvorena u model aplikacije te online baza podataka *„000webhost“* pomoću koje se realiziralo spremanje i prikazivanje korisničkih podataka.

Nakon uvoda i definiranja tehničkih značajki, opisan je detaljni razvoj aplikacije. Svaki dio aplikacije (upute prve pomoći, dohvaćanje lokacije, korisnički podaci) je opisan kroz opisni jezik *XML* nakon čega slijedi osvrt na realizaciju funkcionalnog dijela aplikacije. Uz sve to, svaki opisni i funkcionalni dio aplikacije je popraćen slikom te interpretacijom bitnih dijelova programskog koda.

Funkcije koje su prikazane ovom aplikacijom samo su temelj onoga što bi se moglo napraviti u budućnosti. Ideja je daljnjim nadogradnjama, stvoriti softver koji bi povezao bolnice i korisnike aplikacije kako bi se ozlijeđenoj osobi mogla pružiti bolja i kvalitetnija hitna pomoć. Uz još naprednije znanje programiranja, ovaj diplomski rad bi mogao prerasti u ozbiljan projekt.

LITERATURA

[1] http://giga.geek.hr/1/post/2013/05/upoznajmo-android-1-opcenito-o-androidu.html [22. travnja 2016.]

[2] https://hr.wikipedia.org/wiki/Dosada%C5%A1nje_ina%C4%8Dice_sustava_Androida [22. travnja 2016.]

[3] https://hr.wikipedia.org/wiki/API [22. travnja 2016.]

[4] http://giga.geek.hr/1/post/2013/07/samsung-galaxy-s4-zivotni-suputnik.html [22. travnja 2016.]

[5] http://mobile-place.info/sto-sve-donosi-novi-android-5-0-lollipop/[10. svibanja 2016.]

[6] http://android-studio.en.uptodown.com/windows[10. svibanja 2016.]

[7] https://www.fluidui.com/[10. svibanja 2016.]

[8] https://www.fluidui.com/editor/live/[10. svibanja 2016.]

[9] https://developer.android.com/guide/components/fragments.html[22. svibanja 2016.]

[10] https://loomen.carnet.hr/course/view.php?id=3719[22. svibanja 2016.]

[11] https://en.wikipedia.org/wiki/SQLite[9. lipnja 2016.]

[12] https://developer.android.com/reference/android/app/Activity.html[9. lipnja 2016.]

[13] https://developer.android.com/guide/topics/manifest/manifest-intro.html[9. lipnja 2016.]

[14] https://members.000webhost.com/login[9. lipnja 2016.]

SAŽETAK

Razvojem tehnologije, razvile su se razne aplikacije namijenjene za Android uređaje kako bi pomogle ljudima odraditi brojne svakodnevne zadatke. Žureći, ljudi su u današnjem vremenu sve više skloni brojnim ozljedama koji se događaju. Prijelomi, uganuća, prometne nesreće i gušenja samo su neke od brojnih ozljeda koje se mogu dogoditi. Ovaj diplomski rad je jedno od rješenja kako Android aplikacija u slučaju nesreće može biti jako koristan alat za pomoć.

Aplikacija omogućava korisniku da brzo i jednostavno dođe do uputa prve pomoći kako bi mogao pomoći unesrećenoj osobi dok hitna pomoć ne dođe na mjesto nesreće. Odabirom ozljede, korisniku se detaljno ispisuje postupak što i kako treba uraditi. Isto tako, korisnik može preko aplikacije u svakom trenutku pozvati hitnu pomoć. Prilikom nesreće, aplikacija nudi pronalazak lokacije uređaja gdje se nalazi te ispisivanje koordinata geografske širine i dužine kako bi korisnik znao gdje je stradao. Na kraju, korisnik se može registrirati u aplikaciju kako bi u svakome trenutku imao prikaz svojih podataka (tlak, krvna grupa itd.).

Aplikacija je pisana u *Java* programskom jeziku, a rađena je pomoću *Android Studio* razvojnog okruženja.

Ključne riječi: Android Studio, XML, Java, Aktivnost, Baza

ABSTRACT

ANDROID APPLICATION WITH INSTRUCTION FOR FIRST AID

With the development of technology, a variety of applications have been developed for Android devices to help people do many everyday tasks. People nowadays are more and more prone to many injuries. Fractures, sprains, traffic accidents and suffocations are just a few of the many injuries that can occur. This application is one of the solutions to the Android application in the event of an accident that can be a very useful tool to help.

The application allows the user to quickly and easily receive the instructions of first aid in order to assist injured person until the ambulance arrives. Choosing the injury, user is given a detailed instructions about what to do. Also, the user can call the ambulance through the application at any time. During the accident, the application is giving exact location of the device and shows coordinates of latitude and longitude where the injury occurred. User can register and log in into the application in order to have an overview of their personal information (blood pressure, blood type, etc.) The application is written in Java, and was built using Android Studio.

Keywords: Android Studio, XML, Java, Activity, Base

ŽIVOTOPIS

Stjepan Lukac rođen je 30.07.1991. godine u Vinkovcima. Nakon osnovne škole koju je završio u Levanjskoj Varoši, 2006. godine upisuje opću gimnaziju Matije Antuna Reljkovića u Vinkovcima. 2010. godine završava gimnaziju i upisuje Preddiplomski studij elektrotehnike i računarstva u Osijeku, smjer Računarstvo. Godine 2013. završava preddiplomski studij te upisuje diplomski studij, smjer Procesno računarstvo.