Android aplikacija za praćenje nekih parametara kod dijabetičara

Kordić, Luka

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

Sveučilišni studij računarstva

ANDROID APLIKACIJA ZA PRAĆENJE NEKIH PARAMETARA KOD DIJABETIČARA

Završni rad

Luka Kordić

Osijek, 2016.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 22.09.2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Luka Kordić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3573, 01.08.2013.
OIB studenta:	18273456504
Mentor:	Doc.dr.sc. Krešimir Nenadić
Sumentor:	
Naslov završnog rada:	Android aplikacija za praćenje nekih parametara kod dijabetičara
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 Postignuti rezultati u odnosu na složenost zadatka: 2 Jasnoća pismenog izražavanja: 3 Razina samostalnosti: 3
Datum prijedloga ocjene mentora:	22.09.2016.
Datum potvrde ocjene Odbora:	28.09.2016.

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

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28.09.2016.

Ime i prezime studenta:	Luka Kordić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3573, 01.08.2013.
Ephorus podudaranje [%]:	6%

Ovom izjavom izjavljujem da je rad pod nazivom**: Android aplikacija za praćenje nekih parametara kod dijabetičara**

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:

SADRŽAJ:

1. UVOD 1
1. 1. Zadatak završnog rada 2
2. Tehnologije 3
2.1 Komponente aplikacije 3
2.1.1 Aktivnosti 3
2.1.2 Usluge 5
2.1.3 Davatelji sadržaja 6
2.1.4 Namjere 7
2.2.1 SQLiteOpenHelper 10
3. Struktura 12
3.1 Prvo pokretanje aplikacije 12
3.2 Početni zaslon 14
3.3 Podsjetnik 16
3.4 Baza podataka 18
5. ZAKLJUČAK 21
SAŽETAK 22
LITERATURA: 24
ŽIVOTOPIS 25

1. UVOD

Pritajena zdravstvena bomba koja polako otkucava jest dijabetes. Prema novoj studiji provedenoj u SAD-u i Velikoj Britaniji, svaki treći stanovnik zapadnog svijeta, među kojima su i građani Hrvatske, na putu je da razvije ovu bolest koja, u konačnici, može dovesti i do sljepoće ili amputacije udova, ponajprije nogu. Bolesti su posebno sklone osobe koje se ne kreću dovoljno, a ako ne promjene način života, razvoj punog oblika dijabetesa tipa 2 gotovo je neminovan. Dijabetes tipa 2 bolest je suvremenog doba, izazvana ne samo lošom prehranom i nedostatkom kretanja već i stresom. Ne začuđuje stoga velik broj oboljelih. Problem je i to što je dijabetes u svom razvoju gotovo neprimjetan. Bolest se otkriva tek kontrolom šećera u krvi dok opći znaci bolesti, poput učestalog žeđanja, nerijetko prolaze nezapaženo. U borbi sa šećernom bolesti veliku ulogu ima redovito mjerenje šećera, tj. glukoze u krvi uz dodatno praćenje još nekih parametara poput visine, težine, indeksa tjelesne mase i slično.

U današnje vrijeme pametni telefoni postali su svakodnevica. Koriste se u razne svrhe, a najčešće kao pomagala na poslu, za zabavu ili komunikaciju. Najčešće korišteni operacijski sustav na pametnim telefonima danas je Android operacijski sustav. Zadatak ovog završnog rada je izraditi Android aplikaciju pod nazivom Dnevnik dijabetičara. Ova aplikacija omogućava unos najvažnijih parametara za dijabetičare i prikazuje ih u obliku jednostavne statistike kroz određeno vremensko razdoblje. Aplikacija ima i mogućnost postavljanja podsjetnika kako korisnik ne bi zaboravio na redovito dnevno mjerenje razine glukoze u krvi.

Rad je strukturiran u 5 poglavlja. U uvodnom dijelu opisan je dijabetes – kao veliki zdravstveni problem današnjice. Također je dana ideja o izradi aplikacije koja pomaže dijabetičarima u praćenju razine šećera. Zadatak završnog rada je potpoglavlje u uvodnom dijelu koje daje sažet prikaz rješavanja problema opisanog u uvodu. Glavni dio rada sastoji se od dva veća potpoglavlja: tehnologije izrade i struktura aplikacije. Tehnologije izrade podrazumijevaju klase, aktivnosti, poglede, baze podataka, gumbe i ostale elemente iz Android Studio razvojnog okruženja. Struktura aplikacije sadrži pregled i objašnjenja razvojnog procesa sa stajališta korisnika. Zaključak je rezimirajuće poglavlje u kojemu se daje osvrt na postavljene ciljeve u zadatku završnog rada i na postignute rezultate.

1. 1. Zadatak završnog rada

Zadatak ovog završnog rada je izrada Android aplikacije pod nazivom Dnevnik dijabetičara. Cilj ove aplikacije je pomoći korisnicima u borbi sa dijabetesom na način da im omogućava unos parametara, prikazivanje statistike i podsjetnike na uzimanje inzulina i mjerenje razine šećera u krvi.

2. Tehnologije

U ovom poglavlju detaljnije će se objasniti tehnologije izrade aplikacije poput *SQLiteOpenHelper*-a, aktivnosti, *xml* datoteka, pogleda i ostalih komponenata aplikacije.

2.1 Komponente aplikacije

Komponente su osnovni blokovi za izradu aplikacije. Svaka komponenta je drugačija točka kroz koju sustav može djelovati na aplikaciju. Razlikuju se 4 vrste komponenti aplikacije aktivnosti, usluge, davatelji sadržaja i namjere. Svaka od te 4 vrste služi jedinstvenoj svrsi i ima poseban životni ciklus koji definira kako je komponenta stvorena i uništena.

2.1.1 Aktivnosti

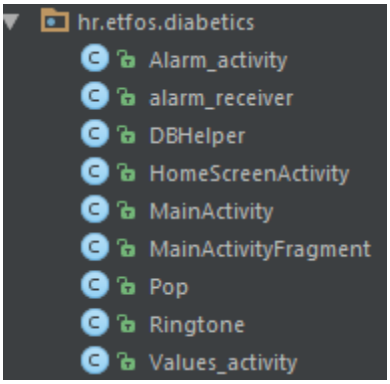
Aktivnosti (eng. *Activites*) predstavljaju zaslone s korisničkim sučeljem. Gotovo sve aktivnosti zahtijevaju interakciju s korisnikom i svaka se od njih u Android studiu prikazuje kao zasebna klasa u kojima se određuje njihova funkcionalnost. Aplikacije najčešće imaju više aktivnosti, a uobičajeno je da se jedna aktivnost, koja se prikazuje korisniku samo pri prvom pokretanju aplikacije, označi kao glavna. Aktivnosti se mogu međusobno pokretati kako bi se odradili različiti zadaci. Svaki puta kada se nova aktivnost pokrene, prethodna se zaustavlja i stavlja na stog (*„the back stack“).* Stog radi na principu *„Last In First Out“*. Dakle kada korisnik pritisne gumb za povratak trenutna aktivnost se prekida i nastavlja se prethodna sa stoga. Aktivnosti je potrebno deklarirati u manifest datoteci kako bi bile dostupne sustavu. Slika 2.1 prikazuje deklaraciju nekoliko aktivnosti iz aplikacije. [1]

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Dnevnik dijabeticara"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="Dnevnik dijabeticara"
            android:theme="@style/AppTheme.NoActionBar">

        </activity>
```

Sl. 2.1: Deklaracija aktivnosti u manifest datoteci



Sl. 2.2: Pogled na aktivnosti i klase u Android studiu

Svaka aktivnost ima svoj životni ciklus. U tu svrhu koriste se povratne metode („*callback methods*“) za pokretanje aktivnosti i metode za zaustavljanje aktivnosti. U metode za pokretanje aktivnosti pripadaju: *onCreate(), onStart(), onRestart()* i *onResume();* dok u metode za zaustavljanje pripadaju: *onPause(), onStop()* i *onDestroy()*. Objašnjena ovih metoda dana su u tablici 2.1.

onCreate()	Poziva se pri prvom pokretanju
onStart()	Poziva se kada aktivnost postane vidljiva korisniku
onRestart()	Poziva se kada se aktivnost ponovno pokrene nakon zaustavljanja
onResume()	Poziva se kada korisnik započne interakciju sa aplikacijom
onPause()	Aktivnost ne prima korisničke naredbe i ne izvodi kod; poziva se kada se trenutna aktivnost zaustavi a sljedeća nastavi
onStop()	Poziva se kada aktivnost više nije vidljiva
onDestroy()	Poziva se prije nego što sustav uništi aktivnost

Tab. 2.1: Pregled povratnih metoda za životni ciklus aktivnosti

2.1.2 Usluge

Usluge su komponente aplikacije koje rade u pozadini kako bi izvršavale dugotrajne operacije ili obavljale radnje za udaljene procese. Rad u pozadini podrazumijeva rad bez korisničke interakcije. Kao primjer takvih radnji može se navesti preuzimanje datoteka s Interneta, reproduciranje glazbe i slično. Usluge ne prikazuju korisničko sučelje pa zbog toga nisu vezane za životni ciklus aktivnosti. Svaka usluga kao i aktivnost treba imati deklaraciju u manifest datoteci, a mogu imati dva stanja: pokrenuta i vezana.

Usluga je pokrenuta kada ju aplikacijska komponenta pokrene pozivajući *startService()* metodu. Kada je pokrenuta, usluga se može izvršavati beskonačno u pozadini, iako je komponenta koja ju je pokrenula uništena. Ovakve usluge najčešće izvode jednu radnju i ne vraćaju rezultat korisniku. Nakon što je radnja obavljena usluga se zaustavlja.

Usluga je vezana kada se aplikacijska komponenta veže za nju pozivajući *bindService()* metodu. Ovakva vrsta usluge nudi klijent-poslužitelj sučelje koje omogućuje komponenti interakciju sa uslugom poput slanja zahtjeva ili dobivanja rezultata. Ovakva usluga izvršava se samo dok je komponenta vezana za nju.

Uz dva navedena pojedinačna načina rada, usluga može biti pokrenuta i u oba načina istovremeno. Neovisno o tome u kojem načinu rada je pokrenuta, svaka komponenta može koristiti uslugu na isti način kao što može koristiti aktivnosti – pokretanjem putem namjera. Ako se želi blokirati pristup usluzi iz drugih aplikacija ona se može deklarirati kao privatna u manifest datoteci. [2] Primjer deklaracije usluge prikazan je na slici 2.3. Kako bi se kreirala usluga potrebno je kreirati podklasu klase *Service*. Također je potrebno pri implementaciji prepisati neke *callback* metode koje obavljaju ključne radnje nad životnim ciklusom usluge. Najvažnije metode koje bi se trebale prepisati su:

- *onStartCommand()* – ova metoda poziva se kada druga komponenta zahtjeva pokretanje usluge,
- *onBind()* – sustav poziva ovu metodu kada se druga komponenta želi povezati sa uslugom. U slučaju da korisnik ne želi dopustiti povezivanje metoda se također mora prepisati, ali vraća vrijednost *null*.
- *onCreate()* – poziva se kada je usluga prvi puta stvorena, kako bi se izvršile procedure za postavljanje usluge. Ukoliko je usluga već pokrenuta, ova metoda se ne poziva.

- *onDestroy()* – sustav ovu metodu poziva kada se usluga više ne koristi i kada ju je potrebno uništiti. Ova metoda također oslobađa sve resurse koje je usluga zauzela.

Android sustav može prisilno prekinuti uslugu u slučaju nedostatka memorije za nastavak njenog izvođenja ili ako je potrebno osloboditi resurse za aktivnost koja je u tom trenutku aktivna.

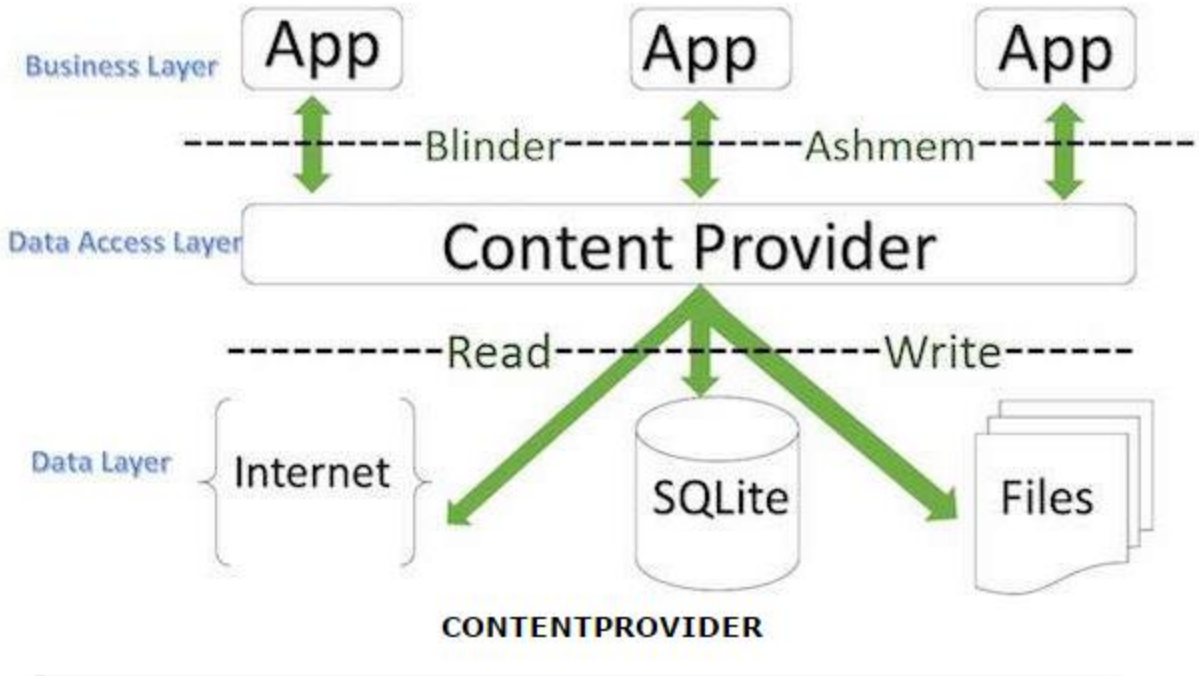
```
    <activity
        android:name=".Pop"
        android:theme="@style/AppTheme.CustomTheme"/>
    <activity
        android:name=".Values_activity"
        android:parentActivityName=".HomeScreenActivity"/>
    <activity android:name=".Alarm_activity"
        android:parentActivityName=".HomeScreenActivity"/>

    <receiver android:name=".alarm_receiver"/>
    <service android:name=".Ringtone"/>
</application>
```

Sl. 2.3 Deklaracija usluge u manifest datoteci

2.1.3 Davatelji sadržaja

Davatelji sadržaja upravljaju skupom aplikacijskih podataka i pružaju mehanizme za sigurnost podataka. Glavno su sučelje koje povezuje podatke u jednom procesu sa kodom koji se izvršava u drugom procesu. Podaci se mogu spremiti na bilo koje trajno mjesto pohrane kojemu aplikacija ima pristup. Kroz davatelj sadržaja ostale aplikacije mogu zatražiti ili mijenjati te podatke. Pristup podacima ostvaruje se pomoću *ContentResolver* objekta. Android uključuje davatelje sadržaja koji upravljaju slikovnim, audio, video i osobnim kontakt podacima. Davatelj sadržaja nije potreban ukoliko se koristi baza podataka samo unutar vlastite aplikacije. U slučaju da je potrebno ponuditi kompleksne podatke ili datoteke drugim aplikacijama ili se želi omogućiti kopiranje takvih podataka potrebno je napraviti vlastiti davatelj sadržaja. [3]



Sl. 2.4: Grafički prikaz davatelja sadržaja (hocthietkeweb.net.vn)

2.1.4 Namjere

Namjera (eng. *Intent*) se mogu uvrstiti među najčešće korištene komponente aplikacije. To su objekti koji se koriste kada je potrebno zatražiti radnju od neke druge komponente aplikacije. Postoje tri osnovna slučaja korištenja namjera:

- za pokretanje aktivnosti – pokreće novu instancu klase *Activity* na način da metodi *startActivity()* predaje kao parametar objekt klase *Intent*. Taj objekt opisuje aktivnost i prenosi potrebne podatke (Sl. 2.5)
- za pokretanje usluga – metodama *startService()* ili *bindService()* predaje kao parametar objekt klase *Intent*. (Sl. 2.6)
- za emitiranje – metodama *sendBroadcast()*, *sendOrderedBroadcast()* ili *sendStickyBroadcast()* predaje kao parametar objekt klase *Intent*. (Sl. 2.7)

```
alarm.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      //pokretanje alarm aktivnosti
        startActivity(new Intent(HomeScreenActivity.this, Alarm_activity.class));
    }
});
```

Sl. 2.5: Primjer eksplicitne namjere za pokretanje alarma

```
//namjera za pozivanje usluge za reprodukciju
Intent service_intent = new Intent(context, Ringtone.class);

//proslijedi extra string iz podsjetnika u ringtonePlayingService
service_intent.putExtra("extra",get_string);

//započni uslugu preko namjere
context.startService(service_intent);
```

Sl. 2.6: Namjera za pokretanje usluge

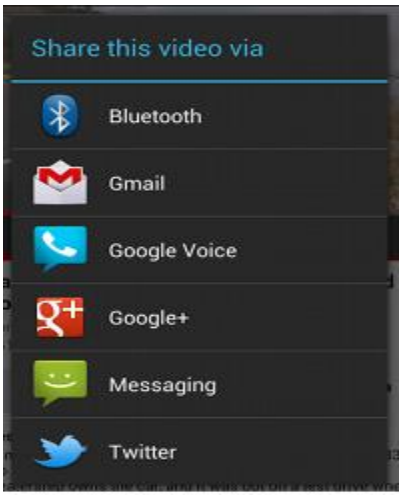
```
//funkcija koja se pokreće pritiskom na gumb za gašenje
assert zaustavi != null;
zaustavi.setOnClickListener((v) -> {
        Toast.makeText(Alarm_activity.this, "Alarm zaustavljen", Toast.LENGTH_SHORT).show();
        alarm_manager.cancel(pending_intent);
        my_intent.putExtra("extra", "Ugašen");
        //zaustavi melodiju
        sendBroadcast(my_intent);
```

Sl. 2.7: Namjera za emitiranje

Također, postoje dvije vrste namjera: eksplicitne i implicitne namjere.

Eksplicitne namjere, kao što ime kaže, eksplicitno određuju ime komponente koja se treba pokrenuti. Uobičajeno je da se ovaj tip koristi kako bi se pokrenula komponenta unutar vlastite aplikacije, jer je poznato ime klase, aktivnosti ili usluge koja se želi pokrenuti. Slika 2.5 prikazuje primjer upotrebe eksplicitne namjere pri izradi ove aplikacije.

Implicitne namjere ne imenuju određenu komponentu, nego umjesto toga definiraju radnju koja se treba izvršiti, što omogućava komponenti iz druge aplikacije da ju obradi. Ako je potrebno podijeliti datoteku, pomoću implicitne namjere može se zatražiti od aplikacije koja ima tu mogućnost da to učini. Slika 2.6 prikazuje prozor koji omogućava odabir aplikacije za izvršavanje određene radnje. Takav prozor može se dobiti korištenjem implicitnih namjera. [4]



Sl. 2.8: Prozor za odabir aplikacije

2.2 SQLite

SQLite je malen ali vrlo moćan mehanizam baza podataka koji je temeljen na maloj programskoj biblioteci napisanoj u C programskom jeziku. Dizajnirao ga je dr. Richard Hipp 2000. godine. Ovaj mehanizam je nedvojbeno najrasprostranjeniji SQL mehanizam na svijetu. Osim u Androidu pojavljuje se i u mnogim drugim platformama poput iPhone, Mozzila Firefox, Skype, Solaris itd. [5]

Tri glavna razloga zašto je tako popularan su:

- Besplatan je
- Mala veličina – trenutna verzija velika je otprilike 150KB, što je za današnje memorije telefona zanemarivo
- Ne zahtijeva instalaciju niti administraciju.

SQLite baza podataka je samo datoteka koju je moguće premještati po želji ili čak kopirati na drugi sustav i radit će izvrsno. U androidu je ta datoteka pohranjena u:

/data/data/packagename/databases directory

Baze podataka idealne su za spremanje ponavljajućih i strukturiranih podataka, kao što su primjerice kontakti, poruke i sl. Za korištenje baza podataka u Androidu potrebna su Android programska sučelja koja su dostupna u *android.database.sqlite* paketu. *SQLiteDatabase* klasa je osnovna klasa za pristup bazama podataka.

Android sakriva dio sintakse ali korisnik i dalje mora poznavati osnove *SQL-a* kako bi ga koristio. Za pristupanje datoteci ne koriste se ulazno/izlazne rutine nego *SQL* naredbe. Tu do izražaja dolaze *helper* klase. Postoje tri osnovne vrste *SQL* naredbi koje korisnik treba poznavati kako bi učinkovito koristio baze podataka, a to su: naredbe za definiranje baze, naredbe za modificiranje i upiti.

2.2.1 SQLiteOpenHelper

SQLiteOpenHelper klasa pomaže pri kreiranju i sadrži metode za upravljanje bazom. Kako bi se koristila *SQLiteOpenHelper* klasa potrebno je kreirati *podklasu* koja ju nasljeđuje i prepisuje *onCreate*() i *onUpgrade*() metode.

Metoda *onCreate()* stvara novu bazu ako zadana baza ne postoji.

Metoda *onUpgrade()* se poziva kada se baza podataka treba nadograditi. To se postiže provjerom vrijednosti zapisane u konstanti *DATABASE_VERSION*.

```
private static class DiabeticsDBHelper extends SQLiteOpenHelper{

    DiabeticsDBHelper(Context ctx){

        super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db){
        //create table when it's ran for the first time
        db.execSQL(CREATE_TABLE_PERSON);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
        db.execSQL("DROP TABLE IF EXISTS " + PERSON_TABLE);
        onCreate(db);
    }
}
```

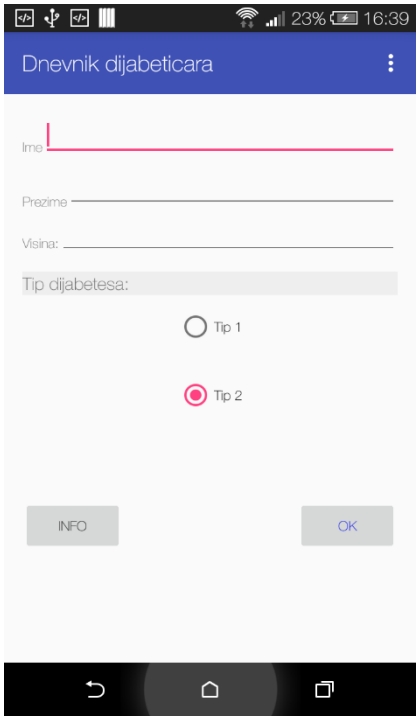
Sl. 2.9: Primjer korištenja *SQLiteOpenHelper*-a u aplikaciji

3. Struktura

Kroz ovo poglavlje daje se detaljan prikaz strukture aplikacije, odnosno uvid u dijelove od kojih je aplikacija sastavljena i postupak njene izrade. Nadalje u ovom poglavlju biti će opsežnije objašnjena implementacija tehnologija koje su se koristile u izradi. Neke važnije tehnologije spomenute su u prethodnom poglavlju.

3.1 Prvo pokretanje aplikacije

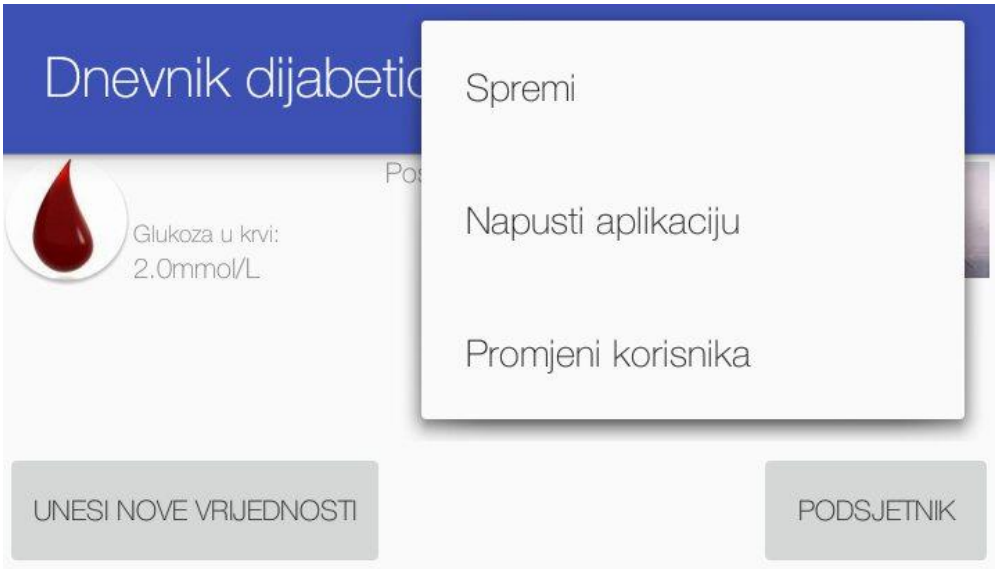
Aplikacija je strukturirana u nekoliko aktivnosti koje se izmjenjuju ovisno o radnjama koje korisnik želi izvršiti. Kako je navedeno u prethodnom poglavlju svaka aplikacija najčešće ima jednu glavnu aktivnost koja se prikazuje samo pri prvom pokretanju aplikacije. Tako i u ovom projektu postoji aktivnost pod nazivom „*MainActivity*“ koja sadrži tekstualna polja za unos imena, prezimena i visine korisnika aplikacije. Također omogućava odabir tipa dijabetesa pomoću dva radio gumba. Ukoliko korisnik nije upoznat sa razlikom između tipova dijabetesa koje treba odabrati pritiskom na gumb „INFO“ otvara se skočni prozor koji sadrži detaljan opis tih dvaju tipova. Ukoliko korisnik ne unese ime, prezime i visinu ili ne označi tip dijabetesa, aktivnost mu ne dopušta prelazak na sljedeći zaslon. To se ostvaruje preko uvjeta da sva tekstualna polja moraju biti popunjena.



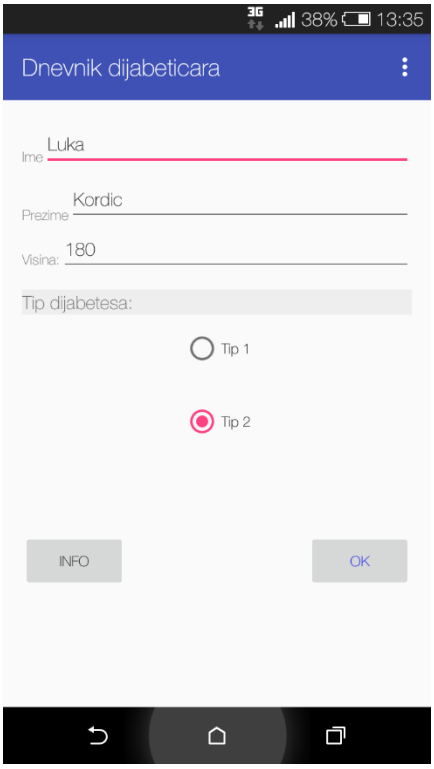
Sl. 3.1: Prikaz glavne aktivnosti

Pritiskom na gumb „OK“ te se vrijednosti pohranjuju u tablicu „Korisnik“ u lokalnoj bazi podataka na uređaju. Kao potvrda o spremanju prikazuje se tzv. *Toast* poruka (sl. 3.2). Prilikom svakog sljedećeg pokretanja aplikacije podaci se čitaju iz tablice u bazi i ta se aktivnost ne prikazuje ukoliko tablica nije prazna. Ovoj aktivnosti u budućnosti se može pristupiti putem postavki, gdje se omogućava promjena korisnika. Pritiskom na gumb postavke > Promjeni korisnika podaci o već postojećem korisniku čitaju se iz baze i ispisuju se u tekstualna polja kao na slici 3.4.

Sl. 3.2: Poruka za potvrdu spremanja podataka



Sl. 3.3: Prikaz izbornika koji omogućava promjenu korisnika



Sl. 3.4: Zaslon za promjenu korisnika

3.2 Početni zaslon

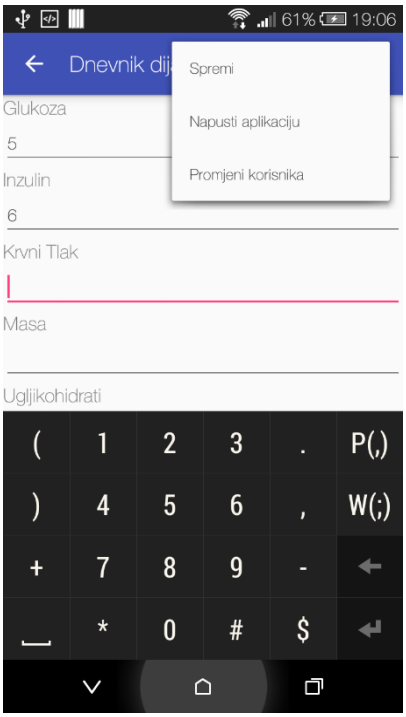
Nakon spremanja podataka u bazu, aplikacija prelazi na početni zaslon koji se naziva *HomeScreenActivity*. Taj prijelaz omogućava se putem namjere. (Sl. 3.5) Kreira se instanca klase *Intent* koja prima dva parametra: aktivnost koja obavlja poziv i aktivnost koju se poziva. Zatim slijedi metoda *startActivity*() kojoj se predaje prethodno kreirani objekt klase *Intent*. U ovom slučaju ime objekta poklapa se s imenom klase, što nije nužno.

```
Intent intent = new Intent(getApplicationContext(), HomeScreenActivity.class);
startActivity(intent);
```

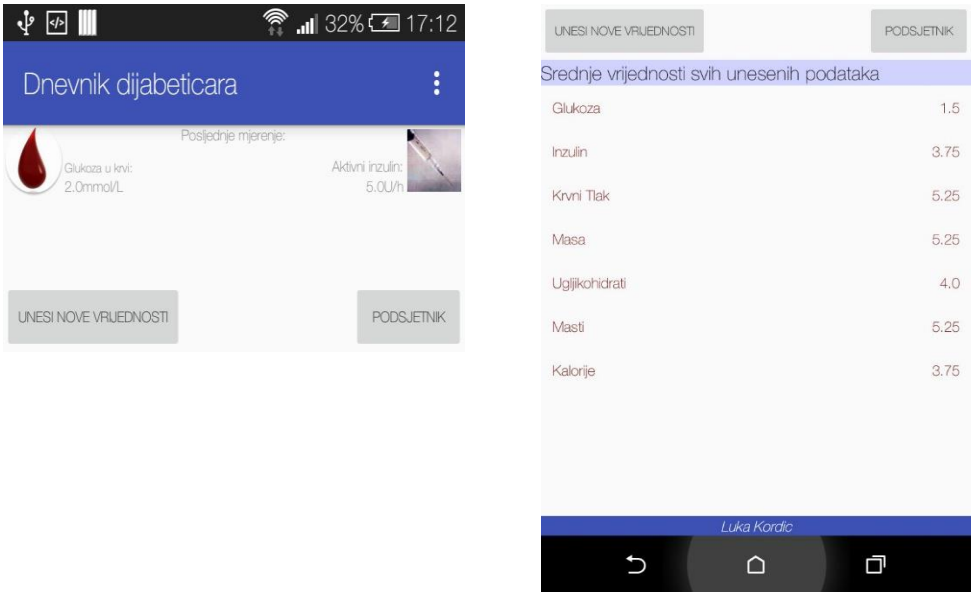
Sl. 3.5: Korištenje namjere za prelazak na „*HomeScreenActivity*“

Pri svakom sljedećem pokretanju aplikacije ova aktivnost provjerava postoje li u bazi uneseni podaci o korisniku. Ukoliko su podaci uneseni prikazuje se zaslon kao na slici 3.7. U suprotnom se pokreće glavna aktivnost koja je objašnjena u prethodnom potpoglavlju. Na taj se način ostvaruje da se glavna aktivnost pokreće samo pri prvom pokretanju aplikacije.

U ovoj se aktivnosti odvija najveći dio radnji koje zahtijevaju interakciju s korisnikom. U gornjem dijelu prozora vidljivi su rezultati posljednjeg mjerenja glukoze i inzulina, a u ostatku prozora prikazane su srednje vrijednosti svih unesenih podataka. Na dnu prozora ispisano je ime trenutnog korisnika. Razina glukoze izražena je u mmol/l (milimola po litri) a razina preostalog aktivnog inzulina u jedinicama po satu (U/h). Klikom na gumb „Unesi nove vrijednosti“ otvara se novi prozor koju pruža mogućnost unosa novih podataka. Polja za unos podataka ograničena su samo na unos cijelih i decimalnih brojeva (Sl.3.6). Gumb podsjetnik otvara novu, istoimenu aktivnost koja omogućava postavljanje podsjetnika.



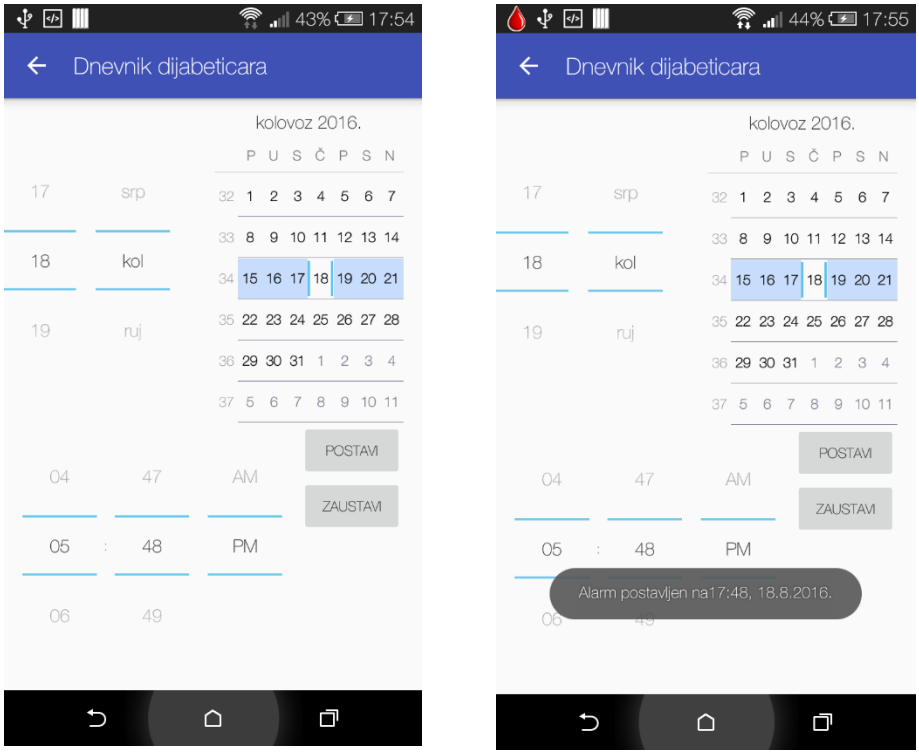
Sl. 3.6: Prozor za unos novih podataka



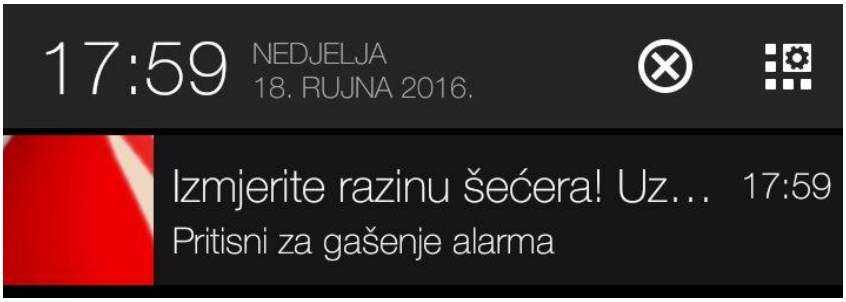
Sl. 3.7: Prikaz vrijednosti posljednjeg mjerenja i ispis srednjih vrijednosti svih unesenih podataka

3.3 Podsjetnik

Još jedna važna komponenta ovog dnevnika je podsjetnik. Podsjetnik je definiran pomoću dva elementa: *„date picker“* i *„time picker“*. Izgled ovog sučelja prikazan je na slici 3.8. Prilikom odabira datuma i vremena podsjetnika potrebno je instancirati objekt klase Kalendar. Tom objektu su zatim postavljene vrijednosti odabrane na „*pickerima“* za sate, minute, dan, mjesec i godinu. Datum može biti odabran preko kalendara ili preko tzv. *Spinnera*. Ta dva elementa međusobno su sinkronizirana, što znači da se promjena na jednom automatsko prenosi na drugi element. Nakon odabira željenog vremena i datuma javljanja podsjetnika potrebno je pritisnuti gumb „Postavi“. Pritiskom na gumb aktivira se namjera koja će se pokrenuti u odabrano vrijeme. To se naziva *broadcast*. Kada se alarm aktivira u statusnoj traci prikazuje se obavijest sa porukom kao na slikama 3.8. i 3.9. Za prekid alarma koristi se gumb „Zaustavi“ koji prekida melodiju, a obavijest ostaje u statusnoj traci sve dok ju korisnik ne deaktivira. Važan dio definiranja podsjetnika je *AlarmManager* klasa koja omogućava pristup uslugama sustava za alarm. Te usluge omogućuju da se aplikacija pokrene u određenom vremenu u budućnosti.



Sl. 3.8: Prikaz aktivnosti za postavljanje podsjetnika i poruke o postavljenom alarmu



Sl. 3.9: Prikaz obavijesti u statusnoj traci

3.4 Baza podataka

Pritiskom na gumb spremi na zaslonu za unos vrijednosti svi uneseni podaci spremaju se u bazu podataka. Baza podataka najvažniji je dio aplikacije, zove se „*Dijabeticari.db*“ i sadrži dvije tablice. Tablica „korisnik“ sadrži podatke o korisniku aplikacije, a sastavljena je od 5 entiteta. Entiteti predstavljaju pojedini parametar koji se upisuje u tablicu. Jedan entitet predstavlja jedan stupac u tablici. Entiteti u tablici „korisnik“ su: ID, ime, prezime, visina i tip dijabetesa. Druga tablica nazvana je „Podaci“ i sastavljena je od 8 entiteta: ID, glukoza, inzulin, krvni tlak, masa, masti, ugljikohidrati i kalorije. Entitet ID predstavlja primarni ključ tablice i postavljen je na *autoincrement*, što znači da je za svaki novi unos ID automatski povećan za jedan. Baza podataka definirana je u java klasi „*DBHelper“*. Na slici 3.10 prikazana je *DBHelper* klasa i deklaracija svih potrebnih entiteta.

```
//----------------------INNER CLASS - HELPER
public class DBHelper extends SQLiteOpenHelper {
    private Context context;


    private static final int DATABASE_VERSION = 7;
    private static final String DATABASE_NAME = "Dijabeticari.db";
    public static final String TABLE_1_NAME = "korisnik";
    public static final String TABLE_2_NAME = "podaci";
    public static final String COLUMN_ID = "_id";
    public static final String COLUMN_NAME = "ime";
    public static final String COLUMN_SURNAME = "prezime";
    public static final String COLUMN_HEIGHT = "visina";
    public static final String COLUMN_GLUCOSE = "glukoza";
    public static final String COLUMN_INSULIN = "inzulin";
    public static final String COLUMN_BP = "krvni_tlak";
    public static final String COLUMN_WEIGHT = "masa";
    public static final String COLUMN_FATS = "masti";
    public static final String COLUMN_CARBS = "ugljikohidrati";
    public static final String COLUMN_CALORIES = "kalorije";
    public static final String COLUMN_RADIOBUTTON1 = "button1";
    public static final String COLUMN_RADIOBUTTON2 = "button2";
```

Sl. 3.10: *DBHelper* klasa i deklaracija entiteta

U metodama „*CREATE_TABLE1*“ i „*CREATE_TABLE2*“ je spremljen dio SQL koda za kreiranje tablica sa pripadnim entitetima. Taj kod se izvodi kada se pozove *onCreate()* metoda kako je prikazano na slici 3.11 i 3.12.

```
private static final String CREATE_TABLE1 =
        "CREATE TABLE " + TABLE_1_NAME + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + COLUMN_NAME + " VARCHAR(255), " + COLUMN_SURNAME + " VARCHAR(15), "
                + COLUMN_HEIGHT + " DOUBLE, " + COLUMN_RADIOBUTTON1  + " INTEGER, " + COLUMN_RADIOBUTTON2 + " INTEGER);";

private static final String CREATE_TABLE2 =
        "CREATE TABLE " + TABLE_2_NAME + " (" + COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + COLUMN_GLUCOSE + " DOUBLE, " + COLUMN_INSULIN + " DOUBLE, "
                + COLUMN_BP + " DOUBLE, " + COLUMN_WEIGHT + " DOUBLE, " + COLUMN_FATS + " DOUBLE, "
                + COLUMN_CARBS + " DOUBLE, " + COLUMN_CALORIES + " DOUBLE);";
```

Sl. 3.11: Metode za kreiranje tablica u bazi podataka

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE1);
    db.execSQL(CREATE_TABLE2);

}
```

Sl. 3.12: *onCreate()* metoda

U *DBHelper* klasi osim same baze, definirane su metode kojima se omogućuje upis podataka u bazu i dohvaćanje pojedinih podataka iz baze. Primjer jedne takve metode prikazan je na slici 3.13.

```
public Cursor getAllData(){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res1 = db.rawQuery("select * from " + TABLE_2_NAME, null);
    return res1;
}
```

Sl. 3.13: Metoda za dohvaćanje svih podataka iz tablice „Podaci“

Prije samog dohvaćanja podataka iz baze ona mora biti otvorena za čitanje. Metodom *getWritableDatabase()* otvara se baza podataka u načinu za čitanje i zapisivanje. Nakon otvaranja izvršava se *sql* upit kojim se dohvaćaju svi podaci iz tablice „Podaci“. Dohvaćeni podaci privremeno se pohranjuju u objekt klase *Cursor*. *Cursor* je posebno sučelje koje pruža mogućnost pisanja i čitanja rezultata vraćenih izvršavanjem upita iz baze podataka. U toj klasi definirane su metode za pomicanje kursora, što omogućuje jednostavan odabir potrebnih stupaca iz tablice i dohvaćanje pojedinih tipova podataka.

5. ZAKLJUČAK

Zadatak ovog završnog rada bio je izraditi aplikaciju koja će pomagati velikom broju ljudi koji danas boluju od dijabetesa. Razvijena je aplikacija pod nazivom Dnevnik dijabetičara koja omogućava korisnicima vođenje dnevnika na njihovim pametnim telefonima, umjesto dosadašnjih ručno pisanih dnevnika.

Pri prvom pokretanju aplikacije od korisnika se traži unos imena, prezimena, visine te odabir tipa šećerne bolesti (tip 1 ili tip 2). Nakon unosa traženih podataka prikazuje se početni zaslon aplikacije. Na njemu korisnik ima mogućnost postavljanja podsjetnika za mjerenje glukoze tokom dana. Kako se potrebe mjerenja glukoze uvelike razlikuju ovisno o tipu dijabetesa i preporukama liječnika, svaki korisnik ima mogućnost postavljanja podsjetnika prema svojim potrebama.

Također na ovom zaslonu odvija se glavni zadatak ove aplikacije a to je unos novih podataka i statistički prikaz već unesenih podataka kroz određeno vremensko razdoblje Prikazuju se podaci poput trenutne razine glukoze u krvi, aktivnog inzulina, indeksa tjelesne mase, težine i sl. Kao dodatak tome može se vidjeti prosječno stanje korisnika, što podrazumijeva prosječan unos i potrošnju kalorija, prosječnu razinu glukoze i inzulina u krvi kao i iznos krvnog tlaka. Svi uneseni podaci sa ovog zaslona spremaju se u bazu podataka na samom telefonu.

Smatram da će ovakve aplikacije uvelike pomoći i liječnicima i pacijentima zbog jednostavnosti njihova korištenja i prikaza informacija. Aplikacija će se u budućnosti nastaviti razvijati i trebala bi imati mogućnost postavljanja svih podataka o pacijentu u „oblak“ kako bi ih njihovi liječnici u svakom trenutku mogli preuzeti u slučaju potrebne pomoći ili savjetovanja.

SAŽETAK

U ovom završnom radu objašnjen je razvoj aplikacije pod nazivom Dnevnik dijabetičara. Aplikacija ima za cilj olakšati borbu sa bolesti oboljelima od dijabetesa na način da im pojednostavi praćenje šećera putem pametnih telefona, umjesto dosadašnjih ručno pisanih dnevnika. Još jedno pojednostavljenje ili prednost ovakvog dnevnika je u tome što korisnik može postaviti podsjetnik za mjerenje šećera prema svojim potrebama. Uneseni podaci mogu se usporediti sa pripadnim optimalnim vrijednostima. Takve usporedbe omogućuju korisnicima da procjene svoje stanje i odluče trebaju li posjetiti liječnika ili je njihovo stanje zadovoljavajuće. Svi navedeni podaci pohranjuju se u bazu podataka na telefonu.

Ključne riječi: android, Android studio, Java, dijabetes, dnevnik dijabetičara, glukoza, inzulin

ABSTRACT

Android application for

monitoring some of the parameters in diabetic patients

This final paper describes development of an application called Diabetes diary. The application's goal is to ease a fight with illness for patients suffering from diabetes in a way that simplifies tracking glucose by using smartphones insted of hand written diaries. Another simplification or advantage of this diary is that the user can set a reminder for tracking glucose based on their personal needs. Entered data can be compared to associated optimal values. These kind of comparisons allow users to asses their current condition and decide whether they need to visit a doctor or their condition is in normal ranges. All of the data is stored in a database on the phone.

Key words: android, Android studio, Java, diabetes, diabetes diary, glucose, insulin

LITERATURA:

[1]https://developer.android.com/guide/components/activities.html
(zadnje posjećeno 20.6.2016.)

[2]https://developer.android.com/guide/components/services.html
(zadnje posjećeno 20.6.2016.)

[3]https://developer.android.com/guide/topics/providers/content-providers.html
(zadnje posjećeno 20.6.2016.)

[4]https://developer.android.com/reference/android/content/Intent.html

(zadnje posjećeno 21.6.2016.)

[5]https://en.wikipedia.org/wiki/SQLite
(zadnje posjećeno 21.6.2016.)

[6] Wei-Meng-Lee, Beginning Android for application development
(zadnje gledano 28.6.2016.)

[7] Ed Burnette, Hello Android – Introducing Google's Mobile Development Platform
(zadnje gledano 29.6.2016.)

ŽIVOTOPIS

Luka Kordić rođen je 14. veljače 1995. godine u Požegi, Hrvatska. Trenutno stanuje na adresi Vinogradska 18, Kutjevo. Osnovno školu pohađa u Kutjevu u razdoblju od 2001. – 2009. godine. U osnovnoj školi sudjeluje na brojnim županijskim natjecanjima iz Hrvatskog jezika, geografije, informatike te sportskim natjecanjima. Značajnije rezultate postigao je iz informatike sa 3. 2. i 1. mjestom 3 godine uzastopno. Nakon osnovne škole, 2009. godine upisuje srednju tehničku školu u Požegi, smjer računalstvo. Na tehničkoj školi ostvaruje vrlo dobar uspjeh i upoznaje se sa razvojem android aplikacija kroz završni rad. Tema završnog rada bila je razvoj jednostavne android aplikacije i rad je ocjenjen odličnim. Nakon završene srednje škole i položene državne mature, 2013 godine upisuje Preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu u Osijeku. Kao posebna znanja i vještine navodi se vrlo dobro poznavanje engleskog jezika, poznavanje i rad u *Microsoft office* programima te poznavanje programskih jezika C, C++, C# i Java.

Vlastoručni potpis

_____