UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Ermin Kentrić

UPORABA PAMETNIH NAPRAV ZA OBDELAVO PODATKOV V PATRONAŽNEM VARSTVU

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Mira Trebar

Ljubljana, 2013

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Ermin Kentrić,

z vpisno številko 63000043

sem avtor diplomskega dela z naslovom:

Uporaba pametnih naprav za obdelavo podatkov v patronažnem varstvu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Mire Trebar
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00541 / 2013
Datum: 15.9.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ERMIN KENTRIČ**

Naslov: **UPORABA PAMETNIH NAPRAV ZA OBDELAVO PODATKOV V PATRONAŽNEM VARSTVU**
THE USE OF SMART DEVICES FOR DATA PROCESSING IN THE HOME CARE SERVICE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Informatizacija zdravstvenih sistemov zajema obsežen nabor podatkov o pacientih, njihovih obiskih zdravnika, specialističnih ambulant in bolnišnic. Ob tem pa se delo v patronažni službi še vedno v veliki meri opravlja na osnovi papirne dokumentacije in naknadnega vnašanja podatkov o opravljenih storitvah na terenu. Kandidat naj v diplomskem delu preuči postopke za zajem podatkov in zasnuje mobilno aplikacijo za operacijski sistem Android, ki omogoča pridobivanje zahtevanih in vnos opravljenih aktivnosti na terenu ter posredovanje podatkov na strežnik. Za predlagano rešitev naj pripravi testne podatke za analizo delovanja aplikacije in ovrednoti njene možnosti za nadaljnjo uporabo v zdravstvu.

Mentor:
doc. dr. Mira Trebar

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Nikolaj Zimic

Zahvala

Rad bi se zahvalil vsem, ki so me vzpodbujali pri študiju. Posebej bi se zahvalil staršem, ki so me, kljub daljšemu študiju, podpirali. Rad bi se zahvalil tudi punci Sari, ki mi je stala ob strani in verjela vame. Mentorici doc. dr. Miri Trebar bi se rad zahvalil za vso strokovno podporo in pomoč pri izdelavi diplomskega dela.

Kazalo vsebine

Seznam kratic in simbolov

3G	*angl. the third generation of mobile telecommunications technology*
ACID	*angl. Atomicity, Consistency, Isolation, Durability*
ADT	*angl. Android Development Tools*
API	*angl.Application Programming Interface*
C	programski jezik C
C#	programski jezik C#
C++	programski jezik C++
DB	*angl. database*
HF	*angl. High Frequency*
HTTP	*angl. HyperText Transfer Protocol*
IDE	*angl. Integrated Development Enviroment*
iOS	kodno ime operacijskega sistema podjetja Apple
JDK	*angl. Java Development Kit*
JDT	*angl. Java Development Tools*
JRE	*angl. Java Runtime Enviroment*
JSON	*angl. JavaScript Object Notation*
NDEF	*angl. NFC Data Exchange Format*
NFC	*angl. Near Field Communication*
OpenGL	*angl. Open Graphics Library*
PDE	*angl. Plug-in Development Enviroment*
PHP	programski jezik PHP
RDBMS	*angl. relational database management system*
RFID	*angl. Radio Frequency Identification*
SDK	*angl. Software Development Kit*
SHA1	*angl. Secure Hashing Algorithm 1*
SOAP	*angl. Simple Object Access Protocol*
SQL	*angl. Structured Query Language*
SQLite	relacijska podatkova baza
WiFi	*angl. Wireless Fidelity, wireless internet*
WSDL	*angl. Web Services Description Language*

Povzetek

Diplomsko delo predstavlja programsko podporo medicinski patronažni službi. Rešitev je namenjena elektronskemu evidentiranju storitev patronažne službe na terenu in predstavlja alternativo dosedanji evidenci v zdravstvenem domu in pa dosedanjemu zamudnemu vnašanju in potrjevanju opravljenih storitev v zdravstvenem domu. Mobilna aplikacija za pametne naprave vsebuje podatke delovnega naloga z zahtevanimi storitvami. Pridobi jih s pomočjo spletne storitve tako, da komunicira s strežnikom zdravstvenega doma. Ti podatki se uporabniku prikažejo na enostaven in razumljiv način in služijo za urejanje in obdelavo delovnega naloga patronažne službe na terenu. Mobilna aplikacija in njen uporabniški vmesnik sta bila razvita v Java programskem jeziku, s pomočjo orodja Eclipse IDE za Android platformo. Implementirana rešitev je bila preverjena s testnimi podatki na pametnem telefonu in predstavlja pilotno izvedbo podpore za delo na terenu. Skupaj s spletno aplikacijo za povezavo v informacijski sistem zdravstvenega doma, ki bo izdelana v drugi diplomski nalogi, bi predlagana rešitev omogočala nadaljnji razvoj celovite rešitve za podporo dela v patronažni službi.

Ključne besede:

Patronažna služba, delovni nalog, spletna storitev, mobilna aplikacija, Android.

Abstract

The thesis represents remote software support for medical home care service. The solution is designed as an electronic system for recording all performed services and it provides an alternative to the current work order processing in the health care center, which can be very time-consuming. Mobile application contains work order data, which is obtained by usage of Web service calls, those services are running on a health care center server. The data is then displayed to the user in an easy and understandable way which can be used to arrange and edit the working order at the remote location. Mobile application and its user interface were developed in the Java programming language, using Eclipse IDE development tools for Android platform. The solution has been verified with the test data on the smartphone. It is available as a prototype and was developed in conjunction with another web application that will be developed and described in the second thesis to serve as a connection to a health center information system. Together they represent the support for a further development of the whole home care solution .

Key words:

Home care service, work order, web service, mobile application, Android.

1. Uvod

Dandanes imajo pametne naprave zelo pomembno vlogo v našem življenju in si velika večina ljudi sploh ne upa predstavljati, kako bi bilo, če jih ne bi imeli. Vse uporabljajo mobilne operacijske sisteme. »Glavni« trije, ki med seboj tekmujejo, so Android, iOS in pa Windows Mobile.

iOS operacijski sistem je operacijski sistem ameriškega giganta Apple, kateri s svojimi napravami velja za nek statusni simbol. Uporaba naprav z operacijskim sistemom Windows Mobile prevladuje v industrijskem okolju, kot so skladišča. Zelo veliko podjetij jih uporablja, ker so zelo robustne. Microsoft je leta 2010 izdal tudi operacijski sistem Windows Phone, vendar je v tem trenutku težko oceniti njegovo uspešnost. Največji delež pa ima operacijski sistem Android. Vzrok za to pa je v odprtokodnosti in ga zato proizvajalci pametnih naprav s pridom uporabljajo. S tem si zmanjšajo stroške, ker jim ni potrebno izdelati lastnega operacijskega sistema, le prilagodijo obstoječi sistem svojim napravam in posledično lahko ponudijo pametne naprave, ki so zelo zmogljive in cenovno ugodne.

V zdravstvu je vedno primanjkovalo finančnih sredstev. Nekatere panoge to občutijo bolj kot druge, zato smo raziskali, kako bi lahko z vse bolj razširjenimi in cenovno ugodnimi pametnimi telefoni, ali pa tudi tabličnimi računalniki olajšali delo zaposlenim. Ugotovili smo, da je najbolj na udaru in z dodatnim delom pogosto obremenjena medicinska patronažna služba, ki mora po opravljenem delu na terenu, opravljati še birokratsko delo v zdravstvenem domu.

Naloge, ki jih opravlja patronažna služba nismo poznali, zato smo se povezali z osebjem v zdravstvenem domu. Spoznali smo procese njihovega dela in probleme s katerimi se srečujejo, tako v zdravstvenem domu in na terenu. Po opravljenem prvem sestanku, smo ugotovili, da je sistem delovanja patronažne službe še vedno takšen kot pred 30 leti in se dejansko ni skoraj nič spremenil. To pomeni veliko papirnate dokumentacije in zamudno iskanje podatkov o delovnih nalogih in naročenih patronažnih storitvah v računalniku. Potrebno je povedati, da zdravstveni domovi v Sloveniji večinoma ne uporabljajo enotnega informacijskega sistema oziroma

programa s katerim obdelujejo podatke. Vsak zdravstveni dom sam izbere informacijski sistem, ki ga bo uporabljal.

Patronažna služba dobi delovni nalog v papirnati obliki, na katerem je veliko podatkov, kot so: podatki o pacientu, podatki o zdravniku, podatki o zavarovanju, podatki o zavarovalnici, podatki o načinu doplačila, itd. Naša ideja je bila, da bi v okviru diplomske naloge izdelali mobilno aplikacijo, ki bi omogočala sprotno obdelavo delovnega naloga patronažne službe na terenu. Pri razvoju rešitve smo se odločili za razvoj aplikacije za operacijski sistem Android, zaradi njegove odprtokodnosti in cenovno ugodnih in zelo zmogljivih pametnih naprav.

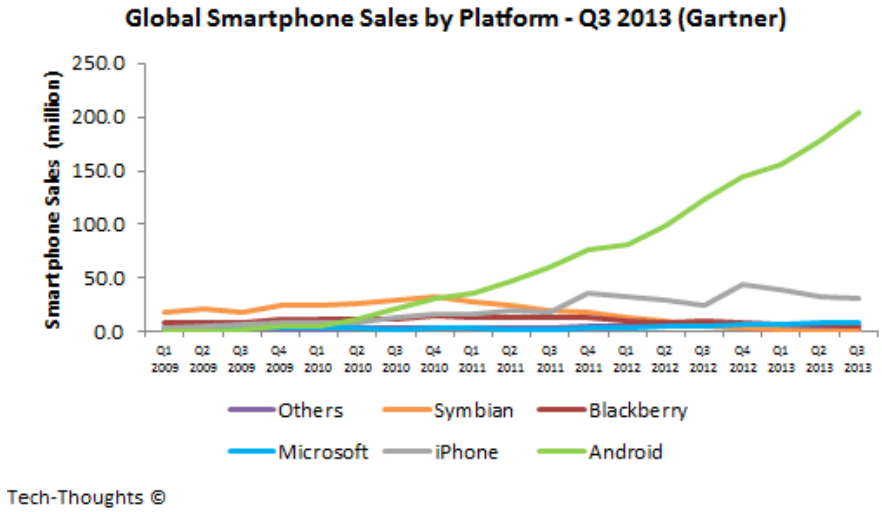
2. Tehnologija in orodja

Na kratko bomo predstavili tehnologije in orodja, ki smo jih uporabljali pri izdelavi diplomske naloge. Najbolj pomemben del rešitve je operacijski sistem Android, ki povezuje vse opisane komponente.

2.1 Operacijski sistem Android

Operacijski sistem Android je s svojim logotipom (Slika 1) trenutno najbolj razpoznaven (Slika 2) in popularen operacijski sistem za pametne naprave (*angl. smart device*). Prva verzija sega v leto 2003, razvijalo ga je podjetje Android, Inc., kasneje jih je prevzelo podjetje Google, ki sedaj skrbi za razvoj sistema.

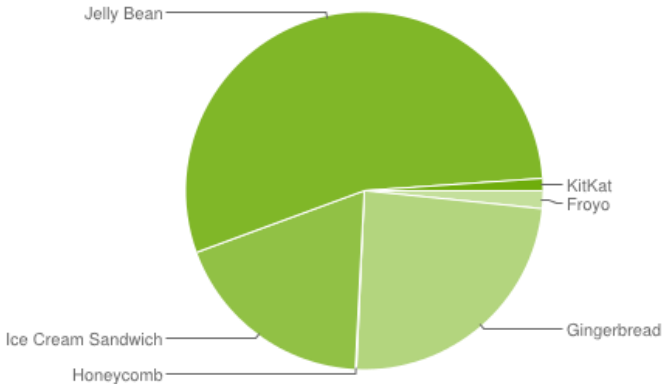
Slika 1: Logotip operacijskega sistema Android



Slika 2: Tržni delež mobilnih operacijskih sistemov [1]

Osnova operacijskega sistema je Linux jedro, ki je močno prilagojeno napravam z na dotik občutljivim zaslonom [2]. Do danes je bilo razvitih 10 verzij operacijskega

sistema, zadnja verzija je Android 4.4., kodno ime KitKat. Trenutno je najbolj razširjen Jelly Bean, v to verzijo spadajo Android 4.1.x, 4.2.x in 4.3 (Slika 3).



Slika 3: Deleži Android verzij na pametnih napravah [3]

Operacijski sistem je sestavljen iz petih komponent: jedra Linux, knjižnic, »Android Runtime«, aplikacijskega ogrodja in aplikacij. Teh pet komponent je potem razdeljeno na štiri sloje (Slika 4).

Jedro Linux (*angl. Linux Kernel*)

Na dnu štirih slojev se nahaja jedro Linux. Skrbi za osnovne funkcije sistema, kot so upravljanje procesov, upravljanje naprav, kot so kamera, tipkovnica, zaslon itd. Jedro poskrbi tudi za omrežje in gonilnike naprav [4].

Knjižnice (*angl. Libraries*)

Stopnjo višje od Linux jedra so knjižnice, ki skrbijo za obdelavo različnih podatkov. Tukaj so knjižnice, ki skrbijo za osnovne funkcije sistema, kot so SQLite za podatkovne baze, WebKit, ki prikazuje spletne vsebine. Napisane so v C in C++ programskih jezikih.

»Android Runtime«

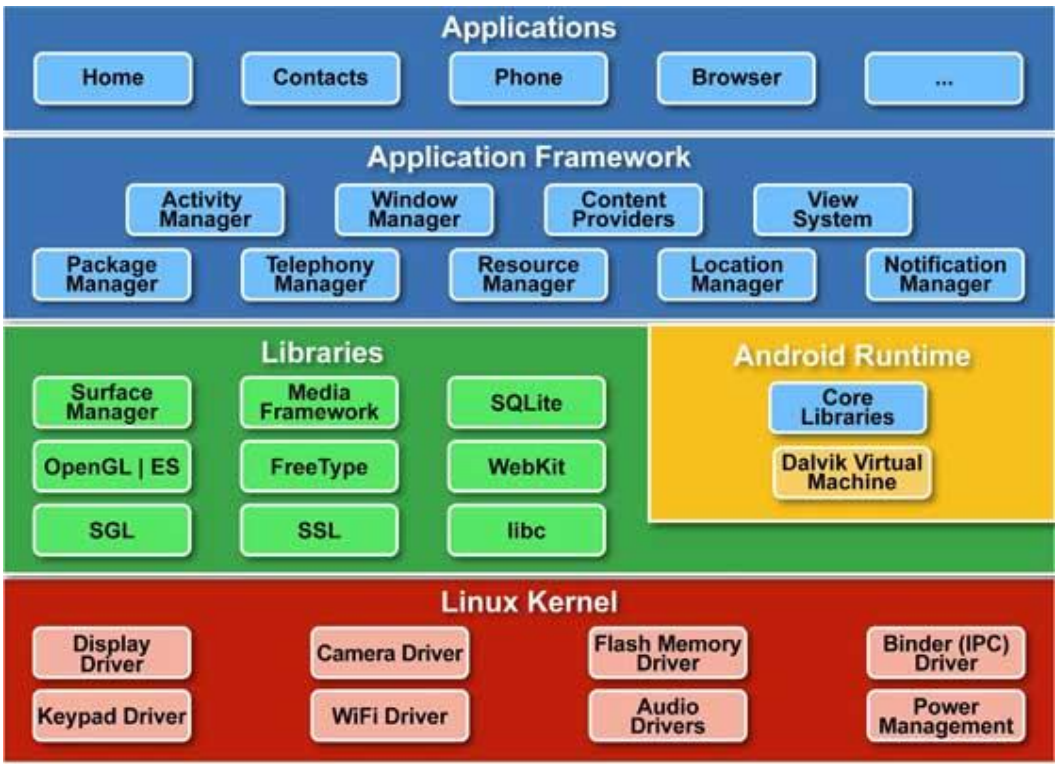
Komponenta si deli isti sloj skupaj s knjižnicami in vsebuje navidezni stroj Dalvik, ki skrbi za optimalno delovanje aplikacij. Tukaj so tudi knjižnice, ki so napisane v Java programskem jeziku.

Aplikacijsko ogrodje (*angl. Application Framework*)

To je komponenta, ki zagotavlja visoko nivojske storitve aplikacijam v obliki Java razredov. Razvijalcem programske opreme je dovoljen dostop do teh storitev.

Aplikacije (*angl. Applications*)

V ta sloj spadajo vse aplikacije, ki so naložene na napravi. Primer aplikacije so igrice, ki se naložijo na Android napravo.



Slika 4: Arhitektura operacijskega sistema

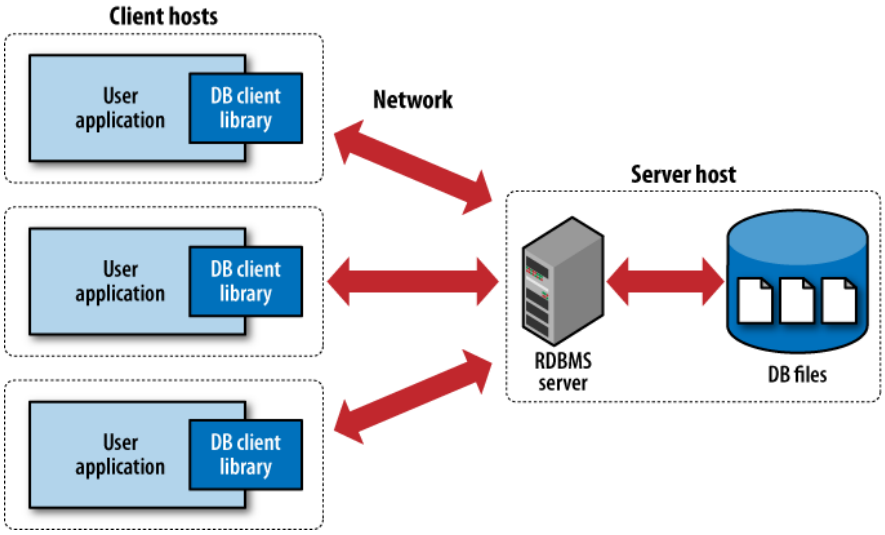
2.2 SQLite

SQLite je odprtokodni programski paket, ki je v javni lasti in je tako prosto dostopen vsem za uporabo v lastnih aplikacijah. Je vgrajena podatkovna baza. Za razliko od ostalih SQL (*angl. SQL – Structured Query Language*) baz (Slika 5), nima strežniškega procesa (Slika 6), ne potrebuje strežnika za delovanje. SQLite bere in zapisuje v navadno datoteko, ki se nahaja na lokalnem disku. Celotna podatkovna baza s tabelami, indeksi, sprožilci in pogledi se nahaja v eni datoteki. Ta je shranjena

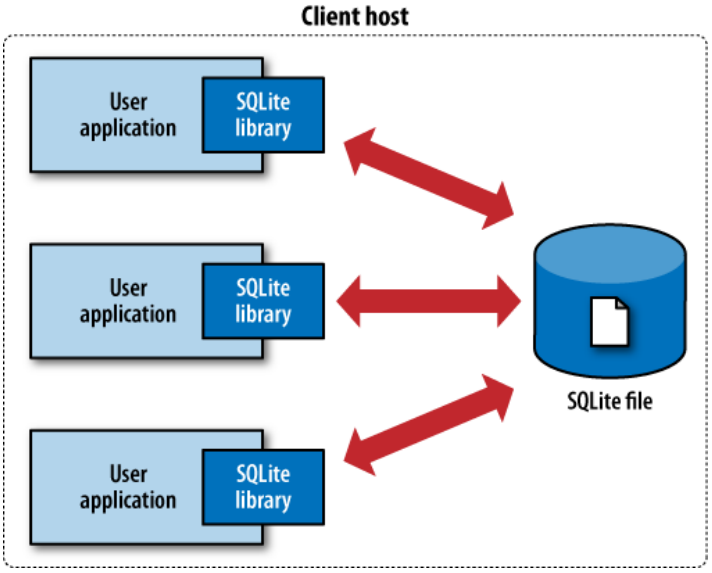
v formatu, ki se lahko uporablja na različnih platformah, ena baza se lahko uporablja tako na 32-bitnih sistemih, kot na 64-bitnih sistemih [5].

Glavne značilnosti SQLite podatkovne baze so: .

- Ničelna konfiguracija, ker ne potrebuje strežnika za delovanje, je kreiranje podatkovne baze enostavno.
- Podpira različne platforme, 32-bitne in 64-bitne.
- Podatkovna baza je v celoti samostojna, kar pomeni, da vsebuje celoten podatkovni sistem, ki je integriran direktno v aplikacijo.
- Ni obremenilna za operacijski sistem, kar pomeni, da pri delovanju, potrebuje samo nekaj megabajtov spomina, osnovni gradnik podatkovne baze je pa manjši od enega megabajta. Z nekaj spremembami je mogoče velikost podatkovne baze in pa porabo spomina za delovanje tudi zmanjšati.
- Transakcijska podatkovna baza, kar pomeni, da je ACID-skladna (*angl. Atomicity, Consistency, Isolation, and Durability*). Omogoča varen dostop do več procesov brez izgube podatkov. Podatki se ne zgubijo, tudi če pride do prekinitve povezave, ostanejo varno shranjeni.
- Podpira večino funkcij iz SQL92(SQL2) standarda.
- Zanesljivost: ekipa, ki razvija SQLite, je zelo zanesljiva, in je zelo dosledna pri svojem delu, kar pomeni, da je testiranje SQLite paketa opravljeno profesionalno [6].



Slika 5: SQL baza z odjemalec-strežnik arhitekturo



Slika 6: SQLite arhitektura brez strežniškega procesa

2.3 JSON

JSON (*angl. JavaScript Object Notation*) je standard za izmenjavo podatkov, ki bazira na tekstu in je zaradi tega enostaven in berljiv. Temelji na programskem jeziku JavaScript, vendar je jezik neodvisen in se ga lahko uporabi povsod, tudi v številnih drugih jezikih, ampak vseeno uporablja konvencije znane programerjem v jezikih C, C++, C#, Javo, JavaScript, Perl, Python in mnoge druge [7].

Značilnosti JSON-a:

- Zasede malo prostora.
- Je hiter in enostaven.
- Omogoča hitro razbiranje podatkov.
- Je kompakten in se ga lahko uporablja tudi v JavaScript-u, podatki se lažje procesirajo.

Slabosti JSON formata:

- Težje berljiv za ljudi.
- Ni primeren za velike količine podatkov.

2.4 Spletna storitev

Spletna storitev (*angl. web service*) je način komunikacije med odjemalcem (*angl. client*) in strežnikom (*angl. server*) preko svetovnega spleta. Izvaja se na nekem spletnem strežniku in je dostopna preko spletnega naslova. Komunikacija poteka s pomočjo standardnih mrežnih protokolov: HTTP, SOAP, WSDL, UDDI. Spletne storitve niso odvisne od operacijskih sistemov in programskih jezikov [8].

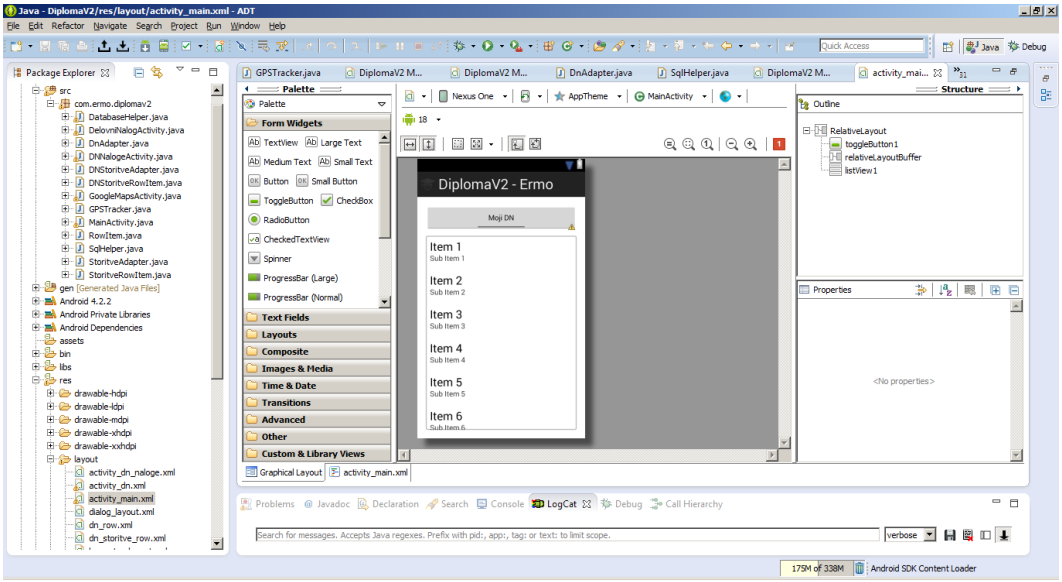
Prednosti spletnih storitev:

- Enostavnejša komunikacija med aplikacijami.
- Lahko jo uporabijo različne aplikacije.
- Enostavnejši prikaz podatkov večim uporabnikom storitve.
- Omogočajo hiter razvoj aplikacij.

2.5 Eclipse IDE

Eclipse IDE (*angl. Integrated Development Enviroment*) je razvojno orodje (Slika 7) Eclipse Foundation skupine, ki je zelo popularno med Java razvijalci programske

opreme. Eclipse je prosto dostopen in ga je mogoče dobiti na uradnem spletnem naslovu[1].



Slika 7: Razvojno orodje Eclipse

Možno ga je prilagajati z vtičniki (*angl. plug-in*), in s pomočjo vtičnikov omogoča tudi razvoj v drugih programskih jezikih, kot so C, C++, JavaScript, PHP, Ruby, Pearl itd [9].

Za pravilno delovanje programa je potrebno namestiti vtičnike in pakete, kot so JDK, JDT, PDE, ADT in pa glavni SDK, ki jih bomo opisali v nadaljevanju.

Vtičnik (*angl. plug-in*) je programski dodatek, ki nekemu programu doda nove funkcije z uporabo lastnih knjižnic (*angl. library*), ki niso del programa v katerega se vtičnik integrira. V našem primeru je ADT vtičnik (*angl. ADT – Android Development Tools plug-in*) razvojnemu orodju Eclipse dodal možnost razvoja aplikacij za Android platformo [10]**.**

JDK paket (*angl. JDK – Java Development Kit*) je razvojno okolje za razvoj aplikacij in ostalih komponent, ki uporabljajo programski jezik Java. JDK paket vsebuje tudi paket JRE (*angl. JRE – Java Runtime Enviroment*), ki je izvajalna platforma za programe, ki so napisani v Javi. Brez paketa JRE ne bi bilo mogoče izvajati javanskih aplikacij [11].

[1] http://www.eclipse.org

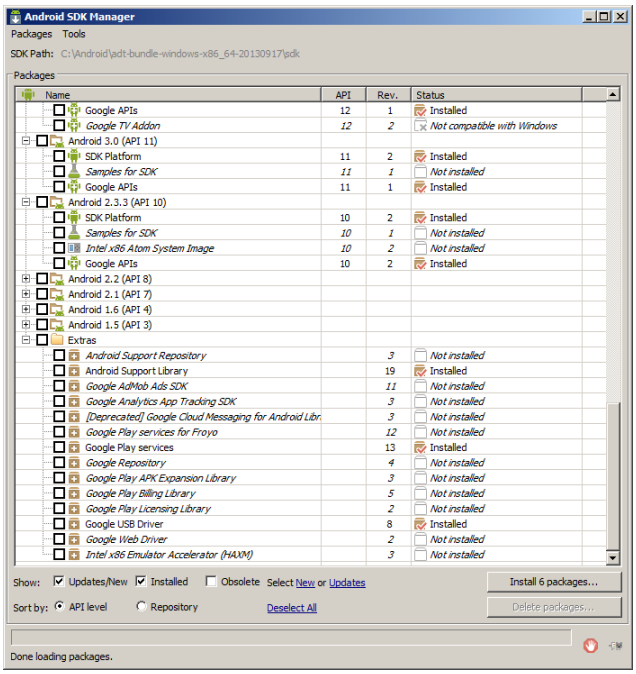
JDT vtičnik (*angl. JDT – Java Development Tools*) je eden od privzetih vtičnikov, ki se namestijo ob postavitvi Eclipse razvojnega okolja, in nudi podporo pri razvijanju v programskem jeziku Java.

PDE vtičnik (*angl. PDE – Plug-in Development Enviroment*) je pa drugi privzeti vtičnik, ki se namesti pri postavitvi Eclipse razvojnega okolja in nudi podporo pri razvoju vtičnikov.

Vtičnika JDT in PDE sta dodatek k razvojni platformi in skupaj s platformo tvorijo javansko razvojno okolje.

ADT vtičnik (*angl. ADT – Android Development Tools*) razširja že tako močno razvojno okolje Eclipse orodja do te mere, da nam omogoča hitro in enostavno izdelavo uporabniškega vmesnika za Android aplikacije, razvoj projektov namenjenih Android sistemu, omogoča dodajanje paketov iz Android ogrodij (*angl. Android Framework API*) in pa izvoz .apk datotek za distribucijo programov [12].

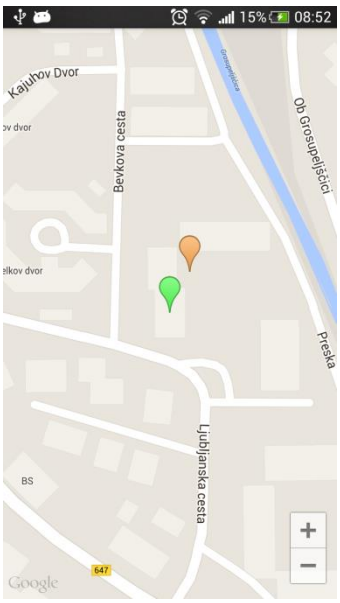
Android SDK (*angl. SDK – Software Development Kit*) je paket orodij za razvoj aplikacij, ki vsebuje primere kode, emulatorje, razhroščevalnike in dokumentacije. Z Android SDK upraviteljem (*angl. Android SDK Manager*) lahko poljubno dodajamo in odstranjujemo razne komponente (Slika 8**).**



Slika 8: Android SDK upravitelj

2.6 Google Maps

Google Maps (Slika 9) je storitev podjetja Google in je na voljo kot del storitev Google Play. Da bi jo lahko uporabili v aplikaciji, je potrebno na razvojnem računalniku imeti naložen Google Play services SDK paket, ki ga prenesemo s pomočjo Android SDK upravitelja (Slika 8), po prenosu je pa potrebno paket vključiti v projekt aplikacije [13].



Slika 9: Google Maps zemljevid

Obstaja več verzij Google maps API paketov, Google Maps API V1, Google maps API V2 in pa Google Maps V3. V naši rešitvi smo uporabili drugo verzijo, ker je ta privzeta verzija za Android sistem, ki omogoča najboljše rezultate pri uporabi na Android platformi. Verzija V3 je namenjena spletnemu razvoju aplikacij. Lahko se jo uporabi tudi pri razvoju Android aplikacij, vendar je pri tem potrebno uporabljati »WebView« kontrolnik (komponenta v razvojnem okolju Eclipse), ki se uporablja za prikaz spletnih strani.

Po uspešnem prenosu in vključitvi paketa v projekt aplikacije je potrebno poskrbeti za dostop do Google Maps strežnikov. Pridobiti moramo Google Maps API ključ (*angl. Google Maps API Key*) s pomočjo Google API konzole (*angl. Google API Console*), ki se nahaja na spletnem naslovu[2]. Ključ pa dobimo na podlagi digitalnega ključa (*angl. SHA1 fingerprint*) [14] s katerim je podpisana naša aplikacija. Za dostop do Google API konzole je potrebno imeti odprt Google račun. Po uspešno pridobljenem

[2] https://code.google.com/apis/console/

Google Maps API ključu, ga je potrebno implementirati v našo aplikacijo. To storimo tako, da v AndroidManifest.xml datoteko dodamo pridobljeni ključ in dodamo podporo knjižnici OpenGL, ki jo uporablja Google maps.

Google maps v aplikacijo dodamo kot drobec (*angl. fragment*). S tem poskrbimo za vso interakcijo z zemljevidom, kar pomeni, da je omogočeno približevanje, rotiranje in premikanje zemljevida. Preko vmesnika lahko poljubno dodajamo lokacije na zemljevid, izrisujemo poti med dodanimi lokacijami, API celo izračuna približno oddaljenost od ene lokacije do druge, časovno in metrično. To pa so samo nekateri primeri uporabe Google Maps API-ja, vmesnik omogoča še veliko več.

2.7 Pametna naprava

Pametna naprava je vsaka naprava (Slika 10), ki se lahko povezuje z drugimi napravami ali omrežji preko različnih protokolov, kot so Bluetooth, NFC, WiFi, 3G in tako naprej in deluje do neke mere samostojno, brez interakcije uporabnika [15].

Slika 10: Pametne naprave

Današnje pametne naprave (pametni telefoni, tablični računalniki) so zelo zmogljivi računalniki, ki so sposobni kompleksnih opravil.

Slika 11: Pametna naprava HTC One X

HTC One X (Slika 11) je pametna naprava proizvajalca HTC, z zaslonom občutljivim na dotik. Napravo poganja operacijski sistem Android 4.2.2, Jelly Bean, ima vgrajeno NFC tehnologijo, ki omogoča branje in pisanje NFC kartic [16].

HTC One X naprava je bila omenjena, ker bosta tako razvoj in testiranje mobilne rešitve izpeljana na njej.

2.8 NFC

NFC (*angl. NFC – Near Field Communication*) je nadgradnja RFID tehnologije (*angl. Radio Frequency Identification*). Je brezžični način komunikacije s pomočjo radijskih valov, ki deluje pri HF frekvenci (angl. HF – High Frequency), 13.56 MHz, območje delovanja NFC je do 10 cm.

Slika 12 prikazuje razlike med NFC in RFID tehnologijo [17].

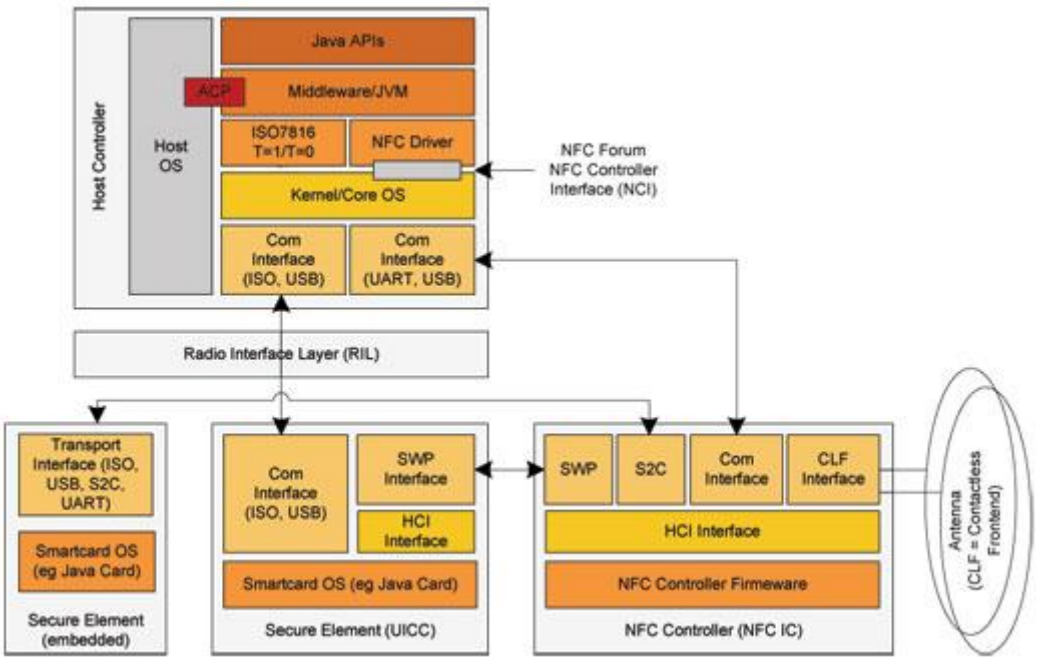
	HF RFID	NFC
Operating Frequency	13.56 MHz	13.56 MHz
Communication	One way	Two way
Standards	ISO 14443, 15693, 18000	ISO 14443
Scan Distance	Up to 1 m	Up to 10 cm
Scan Tags Simultaneously	Yes	No

Slika 12: Razlika NFC in RFID

Prednost NFC tehnologije je, da lahko NFC napravo uporabimo kot sprejemnik in oddajnik, z drugimi besedami NFC naprava lahko postane tudi NFC kartica, ki jo lahko uporabimo za razna opravila. Primer: uporaba NFC naprave (mobilni telefon), kot plačilno sredstvo ali bančna kartica. Veliko pametnih naprav danes že ima vgrajeno NFC funkcionalnost, istočasno pa proizvajalci napovedujejo njeno vključitev v vse novejše naprave (Slika 13).

NFC naprava se lahko uporablja na tri načine:

- Deluje kot čitalnik ali zapisovalnik NFC kartic
- Nastavimo jo tako, da deluje kot NFC kartica in jo uporabimo kot plačilno sredstvo, bančna kartica.
- P2P komunikacija (*angl. peer-to-peer*), to pa pomeni, da lahko povežemo dve napravi in prenašamo podatke iz ene naprave na drugo. Pri prenosu podatkov se uporablja NDEF format prenosa podatkov (angl. Near Field Communcation Data Exchange Format) [18].



Slika 13: Integracijo NFC tehnologije v pametno napravo

3. Patronažna služba

Je ena od zdravstvenih služb, ki izvaja dejavnost osnovnega zdravstvenega varstva. Organizirana je v sklopu dejavnosti zdravstvenega doma. Patronažna enota je samostojna, vodi jo vodja patronažne službe, ki ga potrdi direktor zdravstvenega doma. Izvajalec patronažne službe se imenuje patronažna medicinska sestra. Patronažno varstvo je sestavljeno iz dveh vrst dejavnosti: preventivna dejavnost in kurativna dejavnost [19].

Kurativna dejavnost je zdravstveno-socialna obravnava posameznika, družine in skupnosti in pa zdravstveno nego nosečnice in novorojenčka na domu.

Preventivna dejavnost predstavlja obravnavo bolnika na domu, ki jo patronažna služba izvaja na podlagi izdanega delovnega naloga, ki ga napiše zdravnik.

3.1 Postopki obdelave podatkov o pacientu

Trenutni proces obdelave pacienta v patronažni službi poteka tako, da zdravnik odpre nov delovni nalog in doda storitve, ki jih je potrebno opraviti. Po potrditvi se pošlje v obdelavo patronažni službi.

Patronažna služba dobi delovni nalog (Slika 14, Slika 15) v papirnati obliki. Na podlagi zapisa lahko vidijo katerega pacienta je potrebno obiskati in katere storitve je potrebno opraviti. Vidijo tudi, kdo je naročil storitev (kateri zdravnik), vzrok za napotitev, šifro zavarovalnice, itd.

Po opravljeni storitvi na terenu, in vrnitvi v zdravstveni dom, mora medicinska sestra poiskati delovni nalog v računalniku in vsako storitev označiti kot opravljeno. Nato pa zaključi delovni nalog.

Pomanjkljivost takega procesa je naknadno vnašanje podatkov po opravljeni storitvi in pa neažurno obveščanje o novo odprtih delovnih nalogih, katere patronažna sestra lahko dobi samo takrat, ko je prisotna v zdravstvenem domu (papirnata oblika).

3.2 Načrtovanje aplikacije za delo na terenu

Cilj diplomske naloge je pohitriti procese medicinske patronažne službe na terenu in omogočiti enostavnejši dostop do delovnih nalogov. Naša rešitev bo poizkušala poenostaviti obdelavo pacienta na terenu do te mere, da patronažni službi ne bo potrebno naknadno vnašati podatke v računalnik, istočasno pa bi imeli dostop do vseh potrebnih podatkov na terenu. Cel proces bi se izvedel že ob sinhronizaciji podatkov s strežnikom.

Razvoja rešitve smo se lotili tako, da smo se po večkratnih sestankih s patronažno službo natančno dogovorili, kateri podatki so dejansko potrebni za delo na terenu. Pri izbiri operacijskega sistema in pametne naprave za aplikacijo smo upoštevali tudi finančni vidik in se odločili za Android platformo, ker za relativno nizko ceno lahko dobimo izredno zmogljivo napravo.

Aplikacijo smo zasnovali tako, da s pomočjo spletne storitve pridobi vse potrebne podatke (podatki o uporabnikih, podatki o storitvah, podatki o delovnih nalogih, podatki o pacientu in pa podatki zdravniku, ki odpre delovni nalog) in jih uporabniku razumljivo predstavi.

Omogočati bi morala opcijo prijave uporabnika. Uporabnik bi se lahko prijavil na dva načina, ali s tipkovnico (vnos uporabniškega imena in gesla) ali pa z NFC kartico. Na koncu smo se odločili, da bomo uporabnika vezali na unikatno številko naprave. Tako bi že ob zagonu aplikacije prikazali delovne naloge, ki so vezani na lastnika pametne naprave. S tem smo se pa izognili nerodnemu vnašanju uporabniškega imena z navidezno tipkovnico. Dodali pa smo možnost prijave začasnega uporabnika z NFC kartico. Ob branju kartice bi aplikacija preverila še istovetnost uporabnika in njegov obstoj v bazi uporabnikov.

Ker je patronažna služba vezana na teren, smo se odločili dodati Google Maps storitev v aplikacijo, ki bi uporabniku pomagala pri iskanju naslova pacienta. Pomanjkljivost storitve je v tem, da je odvisna od internetne povezave.

Obdelava pacienta bi potekala tako, da bi patronažna sestra s pomočjo aplikacije dobila vse potrebne podatke o pacientu. Po opravljenem delu bi storitve z nekaj kliki potrdila in označila delovni nalog kot opravljen. Ta bi se nato »izbrisal« iz seznama

delovnih nalogov in ob naslednji sinhronizaciji podatkov poslal na strežnik v obdelavo.

Brisanje delovnega naloga iz naprave bi se dejansko opravilo takrat, ko bi strežnik potrdil prejem delovnega naloga. Naknadnega vnašanja in spreminjanja v zdravstvenem domu naj ne bi bilo, razen v primeru, če se pokaže potreba po tem.

4. Razvoj aplikacije

V tem poglavju bodo predstavljene glavne komponente, ki tvorijo implementacijo zasnovane rešitve, ki je bila zahtevana za diplomsko nalogo. Aplikacija je bila razvita v razvojnem okolju Eclipse za Android platformo in je bila testirana na pametni napravi HTC One X, na kateri je nameščen operacijski sistem Android 4.2.2, Jelly Bean.

V prvi fazi razvoja je bilo potrebno definirati podatkovno bazo in določiti, katere podatke bomo prenašali oziroma shranjevali v bazo, ta faza razvoja je vzela največ časa, ker se je bilo potrebno uskladiti z zahtevami patronažne službe, ki so bile včasih nejasne vendar smo se na koncu uskladili in prišli do skupne rešitve. V drugi fazi razvoja smo se osredotočili na razvoj aplikacijskih funkcij, komponent in implementiranje dodatnih knjižnic (Google Maps). V tretji fazi smo poskrbeli za izgled uporabniškega vmesnika in preizkušanje aplikacije na testnih podatkih.

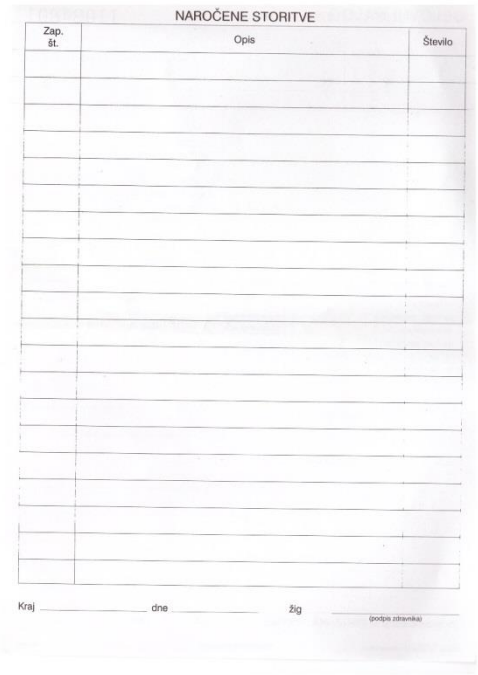
Pred samim začetkom razvoja je bilo potrebno narediti raziskavo komponent, ki smo jih nato uporabljali v naši rešitvi. Na podlagi raziskave smo se potem odločili katere komponente moramo implementirati v našo aplikacijo in pa katere komponente bi bilo dobro dodati zaradi dodane vrednosti aplikacije in morda kasnejše razširitve aplikacije.

4.1 Podatkovna baza

Za podatkovno bazo smo uporabili SQLite programski paket, ki smo ga podrobneje opisali v poglavju 2.2, zato se bomo v tem poglavju osredotočili na vsebino podatkovne baze in vire iz katerih smo sestavljali bazo, ter podrobneje opisali komponente (tabele, stolpci in SQL poizvedbe).

Pri definiranju tabel smo uporabili podatke, ki so nujno potrebni za delo patronažne službe na terenu in so zapisani na delovnem nalogu (Slika 14 in Slika 15).

Slika 14: Sprednja stran delovnega naloga



Slika 15: Zadnja stran delovnega naloga

V naslednjih točkah bodo podrobno predstavljene tabele in stolpci, ki so bile uporabljene v naši podatkovni bazi:

[SifrantUporabniki] – je tabela uporabniških računov in je sestavljena iz naslednjih stolpcev:

- [ID] – predstavlja indeks v tabeli in se povečuje inkrementalno +1 in služi kot pomožna spremenljivka
- [Username] – uporabniško ime oziroma številka uporabnika
- [Password] – geslo uporabnika za prijavo
- [Naziv] – opisni naziv uporabnika, ime in priimek

[SifrantStoritve] – je šifrant storitev, ki se kasneje lahko dodajajo na delovni nalog, sestavljen je iz naslednjih stolpcev:

- [ID] – pomožna spremenljivka
- [IDstoritve] – šifra storitve
- [OpisStoritve] – opis storitve
- [TipStoritve] – 1 - Brez doplačila, 2 - Zavarovana oseba, 3 – Zavarovalnica; na koncu smo se odločili, da tega stolpca ne bomo upoštevali, vendar smo ga ohranili za nadaljnji razvoj aplikacije

[DN_Storitve] – je tabela storitev, v katero se vpisujejo storitve, ki morajo biti opravljene na terenu, sestavljena je iz naslednjih stolpcev:

- [ID] – pomožna spremenljivka
- [DNid] – številka delovnega naloga na katerega je vezana storitev
- [IDstoritve] – šifra storitve
- [OpisStoritve] – opis storitve
- [Kolicina] – količina opravljene storitve, po prenosu podatkov se nastavi na 0
- [NarocenaKolicina] – količina, ki je bila naročena (primer: 1x preveza)
- [TipStoritve] – 1 - Brez doplačila, 2 - Zavarovana oseba, 3 – Zavarovalnica; na koncu smo se odločili, da tega stolpca ne bomo upoštevali, vendar smo ga ohranili za nadaljnji razvoj aplikacije

[DelovniNalogi] – je tabela v kateri so shranjeni podatki o delovnem nalogu, sestavljena je iz naslednjih stolpcev:

- [ID] – pomožna spremenljivka
- [Dnid] – številka delovnega naloga
- [Datum] – datum delovnega naloga
- [Username] – šifra uporabnika, ki opravlja delovni nalog
- [StZdravnika] – številka zdravnika, ki je izdal delovni nalog
- [NazivZdravnika] – ime in priimek zdravnika, ki je izdal delovni nalog
- [TelefonZdravnika] – telefonska številka zdravnika, ki je na voljo v slučaju, če patronažna služba potrebuje dodatna navodila na terenu
- [StZavarovanca] – številka pacienta
- [DatumRojstva] – datum rojstva pacienta
- [PriimekZavarovanca] – priimek pacienta
- [ImeZavarovanca] – ime pacienta
- [UlicaZavarovanca] – naslov pacienta
- [PostaZavarovanca] – poštna številka kraja pacienta
- [KrajZavarovanca] – naziv kraja pacienta
- [TipZavarovanja] – pomembno pri dodajanju dodatnih storitev na delovni nalog, podatek o tem ali je storitev krita ali ne

- [TipNaloga] – 0/1 (0 - enkraten obisk/ 1- delovni nalog za neko obdobje)
- [OpomnikDatum] – opomnik za delovne naloge tipa 1, ki ga uporabnik sam vnese
- [Komentar] – komentar, ki ga lahko patronažna sestra sama vnese
- [DNZakljucen] – polje, ki nam pove ali je nalog zaključen(1- zaključen, 0 – ni zaključen); če je zaključen, se nam ne prikazuje v seznamu delovnih nalogov; pri naslednji sinhronizaciji se prenese na strežnik in izbriše iz lokalne baze
- [SyncToken] – je nek ključ, ki strežniku pomaga pri pošiljanju in sprejemanju podatkov

Spodaj lahko vidimo, kako ustvarimo zgoraj navedene tabele.

```
// Definicija nazivov tabel
private static final String TABLE_UPORABNIKI = "SifrantUporabniki";
private static final String TABLE_STORITVE = "SifrantStoritve";
private static final String TABLE_DN_STORITVE = "DN_Storitve";
private static final String TABLE_DELOVNINALOGI = "DelovniNalogi";

private static final String CREATE_TABLE_UPORABNIKI =
"CREATE TABLE "+ TABLE_UPORABNIKI +
"(ID INTEGER PRIMARY KEY AUTOINCREMENT  NOT NULL, Username TEXT, Password"
+ "TEXT, Naziv TEXT)";

private static final String CREATE_TABLE_STORITVE = "CREATE TABLE "+
TABLE_STORITVE +
"(ID INTEGER PRIMARY KEY AUTOINCREMENT  NOT NULL, IDstoritve TEXT, " +
"OpisStoritve TEXT, TipStoritve TEXT)";

private static final String CREATE_TABLE_DN_STORITVE = "CREATE TABLE "+
TABLE_DN_STORITVE +
"(ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,DNid TEXT, " +
"IDstoritve TEXT,OpisStoritve TEXT,NarocenaKolicina INTEGER,Kolicina "+
"INTEGER, TipStoritve TEXT)";

private static final String CREATE_TABLE_DELOVNINALOGI = "CREATE TABLE "+
TABLE_DELOVNINALOGI + "(ID INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL ,
"+ "DNid TEXT, Datum DATETIME, Username TEXT, " +
"StZdravnika TEXT, NazivZdravnika TEXT, TelefonZdravnika TEXT, " +
  "StZavarovanca TEXT, DatumRojstva TEXT, PriimekZavarovanca TEXT, " +
  "ImeZavarovanca TEXT, UlicaZavarovanca TEXT, " +
"PostaZavarovanca TEXT, KrajZavarovanca TEXT, " +
      "TipZavarovanja TEXT, TipNaloga TEXT, OpomnikDatum DATETIME,Komentar
  TEXT, " + "DNZakljucen BOOL, SyncToken TEXT)";
```

```
db.execSQL(CREATE_TABLE_UPORABNIKI);
db.execSQL(CREATE_TABLE_STORITVE);
db.execSQL(CREATE_TABLE_DN_STORITVE);
db.execSQL(CREATE_TABLE_DELOVNINALOGI);
```

4.2 Funkcije in komponente aplikacije

Aplikacija je bila razvita v razvojnem orodju Eclipse, v programskem jeziku Java, s pomočjo ADT vtičnika. V tem poglavju bomo prikazali nekatere pomembnejše komponente, ki smo jih uporabili v sami kodi.

Filtriranje, prikaz in vnos podatkov

Izvaja se s pomočjo vmesnika Cursor (*angl. Cursor interface*), s poizvedbami SQL pa dostopamo do želenih podatkov. Prav tako s tem vmesnikom vnašamo, brišemo in posodabljamo podatke.

Funkcija za prikaz delovnega naloga za prijavljenega uporabnika:

```
private void PrikaziSeznamDNUporabnik(String uporabnik)
{
	DatabaseHelper dbHelper =
		new DatabaseHelper(getApplicationContext());
	SQLiteDatabase DB = dbHelper.getWritableDatabase();
	ArrayList<Map<String, Object>> data =
		new ArrayList<Map<String,Object>>();

	Cursor cur = DB.rawQuery("SELECT DNid, PriimekZavarovanca, " +
	ImeZavarovanca " +
	"FROM DelovniNalogi where Username='"+uporabnik+"' AND  " +
	"DNZakljucen='0'", null);
	while(cur.moveToNext())
	{
		Map<String, Object> rowOfCells = new HashMap<String,
		Object>();
		rowOfCells.put("DNid",
		cur.getString(cur.getColumnIndexOrThrow("DNid")));

		rowOfCells.put("priimek",
		cur.getString(cur.getColumnIndexOrThrow("PriimekZavarovanca"
		)));
		rowOfCells.put("ime",
		cur.getString(cur.getColumnIndexOrThrow("ImeZavarovanca")));
```

```
                data.add(rowOfCells);
            }
            addResults(data, (ListView)findViewById(R.id.listView1));
        }
```

Na podoben način vnašamo oziroma posodabljamo podatke v tabelah. Spodaj imamo primer s katerim posodobimo nek zapis v tabeli (zapis opombe v tabelo DelovniNalogi):

```
    private void DodajKomentar(String dnid,String komentar)
    {
            DatabaseHelper dbHelper = new
            DatabaseHelper(getApplicationContext());
            SQLiteDatabase DB = dbHelper.getWritableDatabase();
            {
                DB.execSQL("UPDATE DelovniNalogi SET Komentar='"
                +komentar+"'"+
                      "WHERE DNid='"+dnid+"'");
            }
    }
```

Razred DatabaseHelper.java

DatabaseHelper je potrebno omeniti, ker je to podrazred razreda SqlLiteOpenHelper, s pomočjo katerega upravljamo kreiranje podatkovne baze. Spodaj lahko vidimo primer kode s katero ustvarimo naše tabele v podatkovni bazi.

```
@Override
public void onCreate(SQLiteDatabase db) {

  db.execSQL(CREATE_TABLE_UPORABNIKI);
  db.execSQL(CREATE_TABLE_STORITVE);
  db.execSQL(CREATE_TABLE_DN_STORITVE);
  db.execSQL(CREATE_TABLE_DELOVNINALOGI);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  db.execSQL("DROP TABLE IF EXISTS " + TABLE_UPORABNIKI);
  db.execSQL("DROP TABLE IF EXISTS " + TABLE_STORITVE);
  db.execSQL("DROP TABLE IF EXISTS " + TABLE_DN_STORITVE);
  db.execSQL("DROP TABLE IF EXISTS " + TABLE_DELOVNINALOGI);

  onCreate(db);
}
```

NFC prijava

V našo aplikacijo smo integrirali tudi možnost prijave s pomočjo NFC kartic. To naredimo tako, da kličemo NFC komponento (*angl. Intent*) Android sistema. Spodnji primer kode prikazuje, kako jo kličemo.

```
@Override
public void onNewIntent(Intent intent) {
      if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
      detectedTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
      setIntent(intent);
      final String zacasniUporabnik = readFromTag(getIntent());
      if(zacasniUporabnik.length()>0)
      {
            AlertDialog.Builder builder = new
            AlertDialog.Builder(MainActivity.this);
            builder.setMessage("Prijavi začasnega uporabnika "+
            zacasniUporabnik +"?");
            builder.setPositiveButton("Da", new
            DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
               uporabnik = zacasniUporabnik;
               PrikaziSeznamDNUporabnik(uporabnik);
                  Toast.makeText(getApplicationContext(), "Prijavljen
                  začasni uporabnik:\n " + uporabnik,
                  Toast.LENGTH_LONG).show();
               }
            }).setNegativeButton("Ne", new
            DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int id) {
               dialog.cancel();
               }
           })
           .show();
      }
      else
      Toast.makeText(getApplicationContext(), "Napaka pri branju
      kartice!", Toast.LENGTH_LONG).show();
      }
}
```

Google Maps API V2

V aplikaciji smo uporabili Google Maps API V2 storitev za katero je bilo potrebno pridobiti ključ za dostopanje do Google Maps strežnikov, tega smo pa dobili na

podlagi digitalnega ključa (*angl. SHA1 fingerprint*) s katerim je podpisana naša aplikacija. Na Slika 16 vidimo primer ukaza, kako dobimo digitalni ključ, ki ga izvedemo v ukazni vrstici.

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -
storepass android -keypass android
```

Slika 16: Ukaz za pridobitev digitalnega ključa

V Google API konzoli[3] smo pa s pomočjo pridobljenega digitalnega ključa, pridobili Google Maps API ključ.

Google Maps se v aplikacijo doda kot fragment, ki nam omogoča poln nadzor nad storitvijo. Spodaj vidimo primer inicializacije storitve in prikaza zemljevida.

```
FragmentManager fragmentManager = getFragmentManager();
MapFragment mapFragment =
(MapFragment)fragmentManager.findFragmentById(R.id.map);
googleMap = mapFragment.getMap();
```

V našo rešitev smo dodali možnost prikaza naše trenutne lokacije in pa prikaz lokacije pacienta. Naša lokacija se določa s pomočjo GPS (*angl. Global Positioning System*) lokatorja naprave. Če lokacije ni mogoče določiti na tak način, naprava poizkusi določanje lokacije s pomočjo brezžičnega omrežja. Za prikaz naše lokacije je bilo potrebno aplikaciji omogočiti dostop do lokatorjev položaja, to pa smo storili tako, da smo dodali dovoljenja v AndroidManifest.xml datoteko. Spodaj vidimo, kako dodamo aplikaciji dostop do lokacijskih storitev.

```
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

[3] https://code.google.com/apis/console/

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

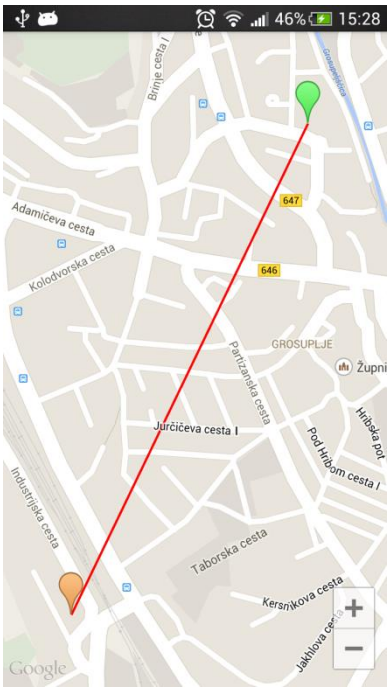
Google Maps storitev lokacije prikazuje s pomočjo geografske širine in dolžine. Spodaj vidimo primer kode s katero izrišemo lokacijo na zemljevidu:

```
        LatLng sfLatLng = new LatLng(addressLat, addressLng);
        googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        googleMap.addMarker(new MarkerOptions()
                .position(sfLatLng)
                .title("Naslov pacienta")
.snippet(naslov).icon(BitmapDescriptorFactory.defaultMarker
(BitmapDescriptorFactory.HUE_GREEN)));
```

Lokacijo pacienta pa določimo s pomočjo Geocoder razreda [20], ki nam vrne geografsko širino in dolžino opisnega naslova, spodaj lahko vidimo primer uporabe:

```
Geocoder coder = new Geocoder(this);
List<Address> address;
String naslov = "Bevkova cesta 1,1290 Grosuplje,Slovenija";
address = coder.getFromLocationName(naslov,5);
Address location = address.get(0);
addressLat = location.getLatitude();
addressLng = location.getLongitude();
```

Ko smo določili geografsko širino in dolžino za obe lokaciji (našo in pacientovo) izrišemo črto med njima za lažje orientiranje na zemljevidu (Slika 17).



Slika 17: Izris črte med lokacijami na zemljevidu

Primer kode s katero izrisujemo črte med lokacijami:

```
Polyline line = googleMap.addPolyline(new PolylineOptions()
.add(new LatLng(x,y), new LatLng(addressLat,   addressLng))
.width(5)
.color(Color.RED).geodesic(true));
```

4.3 Uvoz in izvoz podatkov

Sinhronizacija se izvaja s pomočjo spletnih storitev. Podatki se bodo sinhronizirali večkrat na dan z uporabo brezžične povezave. Prva sinhronizacija se izvede že ob zagonu aplikacije, druga pa pri izhodu aplikacije. Vmesno preverjanje pa se izvede vsakič, ko zapremo delovni nalog v aplikaciji. Postopek prenosa podatkov je tak, da se ob zagonu aplikacije pokliče spletna storitev, ki vrne podatke vseh delovnih nalogov patronažne službe za tisti datum.

Podatki se prenašajo v JSON formatu. Ko aplikacija prejme podatke spletne storitve, jih s pomočjo JSON razčlenjevalnika (*angl. parser*) razčleni v sebi razumljivo obliko, in jih doda v svojo lokalno podatkovno bazo.

Uvažanje podatkov se izvede s pomočjo SQL poizvedb (*angl. SQL query*). Vsak na novo dodan delovni nalog se v bazo doda z oznako 0 (stolpec DnZakljucen v tabeli DelovniNalogi). Ta podatek nam pove, da je delovni nalog nezaključen in se bo prikazal na seznamu delovnih nalogov.

Ko nalog zaključimo se v stolpec DnZakljucen vpiše vrednost 1, s tem podatkom aplikaciji povemo, kateri delovni nalogi so pripravljeni za prenos na strežnik in se ob naslednji sinhronizaciji se prenesejo na strežnik. S tem pa izvozimo podatke na strežnik.

Potrebno je povedati, da smo v aplikaciji zgoraj opisan postopek simulirali, ker bo spletna storitev, s pomočjo katere sinhroniziramo podatke, naknadno narejena v drugi diplomski nalogi.

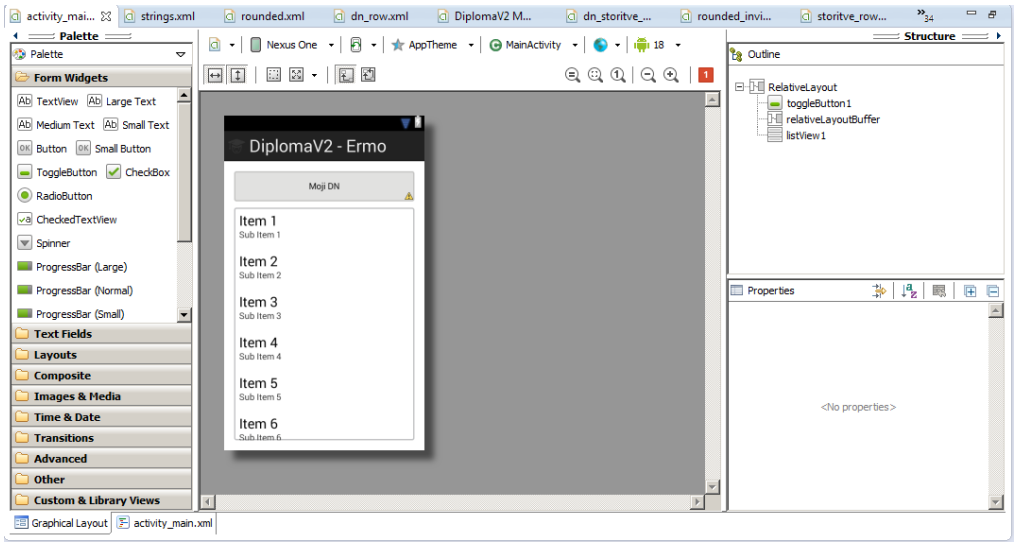
4.4 Uporabniški vmesnik

Uporabniški vmesnik za Android aplikacijo je potrebno narediti s pomočjo XML »layout« datoteke. V XML (*angl. Extensible Markup Language*) datoteki definiramo katere komponente bomo uporabljali, in kako bodo komponente postavljene. Datoteko lahko urejamo grafično (Slika 18) in tekstovno. Grafično postavljanje komponent je nekoliko nerodno, ker grafični vmesnik ne deluje tako, kot bi moral, zato je najbolje uporabljati tekstovno urejanje, ki je dokaj enostavno.

Vsaki komponenti je nato potrebno določiti parametre, kot so velikost, besedilo, lokacije komponente, itd. Spodaj lahko vidimo primer XML datoteke v kateri se nahajajo definicije komponent: preklopni gumb (*angl. ToggleButton*), »RelativeLayout«, ki je dodan kot vmesna pregrada med dvema komponentama in pa seznam elementov (*angl. ListView*).

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

      <ToggleButton
            android:id="@+id/toggleButton1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="ToggleButton"
            android:textOff="Moji DN"
            android:textOn="Vsi DN"
            android:background="@layout/rounded_invis_button"/>

      <RelativeLayout
          android:id="@+id/relativeLayoutBuffer"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:paddingBottom="10dp"
          android:layout_below="@id/toggleButton1" >

      </RelativeLayout>

    <ListView
```

```
        android:id="@+id/listView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/relativeLayoutBuffer"
        android:background="@layout/rounded_invis" >

    </ListView>

</RelativeLayout>
```



Slika 18: Urejanje uporabniškega vmesnika

5. Uporaba mobilne aplikacije

V tem poglavju bomo opisali uporabniški vmesnik in način uporabe aplikacije. Aplikacijo je potrebno namesti na mobilno napravo, ki jo poganja operacijski sistem Android. Minimalna verzija sistema za uporabo mora biti 2.3., kodno ime Gingerbread. Aplikacijo se namesti s pomočjo namestitvene datoteke »*.apk«*. Deluje tako, da s pomočjo spletnih storitev prenaša podatke na in iz strežnika zdravstvenega doma. Prva sinhronizacija se izvede že ob zagonu aplikacije, naslednje sinhronizacije se izvajajo takrat, ko zapiramo delovne naloge, izvede se tudi takrat, ko zapustimo program. Podatki se sinhronizirajo z uporabo brezžične povezave (internetno ali mobilno omrežje), tako, da se sinhronizacija izvaja tudi na terenu. Edini pogoj za sinhronizacijo je dostop aplikacije do omrežja, ki je povezan v svetovni splet. Če je bila sinhronizacija neuspešna, nas aplikacija opozori, da je prišlo do napake in nas zaprosi za vklop ene od tehnologij za povezavo v splet.

5.1 Namen mobilne aplikacije

Namen aplikacije je pohitriti in olajšati delo medicinske patronažne službe tako, da bi poskusili zmanjšati dodatno delo z vnašanjem podatkov po opravljeni patronažni storitvi na terenu, ob enem pa omogočiti hitro obveščanje o novih nalogah. Patronažna služba lahko z uporabo aplikacije dobi ažurne podatke o delovnih nalogih in nalogah na terenu.

5.2 Opis uporabniškega vmesnika

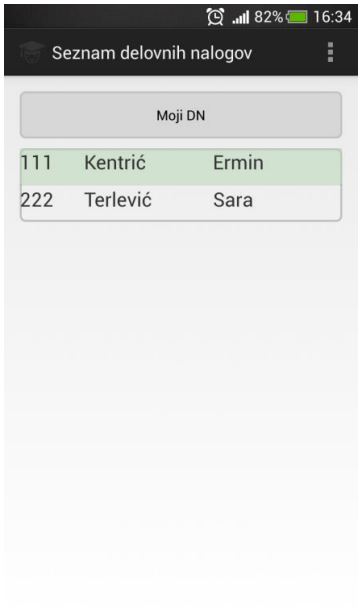
Tukaj bomo podrobno predstavili uporabniški vmesnik in opisali kako se uporablja aplikacija.

Seznam delovnih nalogov

Pri zagonu aplikacije se izvede avtomatska prijava na uporabnika, na katerega je vezana naprava. Po prijavi sledi klic spletne storitve in sinhronizacija podatkov. Če je

bil postopek uspešen, bi se nam morali prikazati delovni nalogi prijavljenega uporabnika (Slika 19).

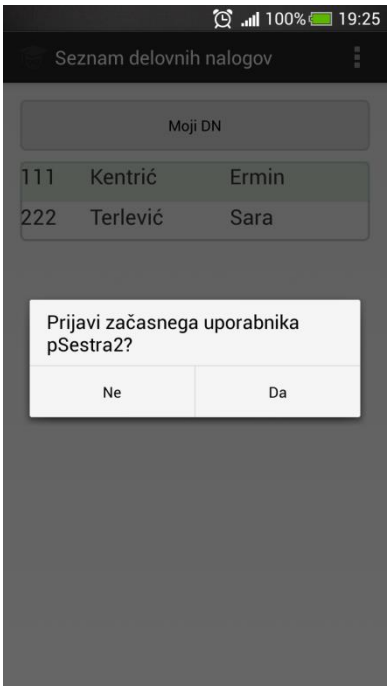
V primeru neuspešne sinhronizacije (neodzivna spletna storitev, omejen dostop do interneta) nam aplikacija javi, da je prišlo do napake. Do naslednje uspešne sinhronizacije se uporabljajo obstoječi podatki.



Slika 19: Obrazec seznama delovnih nalogov (Moji DN)

Slika 20: Obrazec seznama delovnih nalogov (Vsi DN)

Če želimo videti vse delovne naloge, potem kliknemo na gumb Moji DN in prikaže se celoten seznam (Slika 20). Tu se je mogoče prijaviti tudi z začasnim uporabniškim imenom s pomočjo NFC tehnologije. To je omogočeno v primeru, če bo napravo uporabljal nekdo drug, čeprav je vezana na določenega uporabnika. NFC prijavo izvedemo tako, da NFC kartico približamo napravi, ki bo ob zaznavi kartice prikazala pogovorno okno (Slika 21), in s pritiskom na gumb »Da« izvedemo začasno prijavo na terminal, ob prijavi se nam samodejno prikažejo delovni nalogi, ki so vezani na začasno prijavljenega uporabnika.



Slika 21: NFC Prijava začasnega uporabnika

Obrazec seznama delovnih nalogov (Slika 19), je sestavljen iz naslednjih komponent:

- Preklopnega gumba (»*Moji DN*«,«*Vsi DN*«), s katerim preklapljamo med delovnimi nalogi prijavljenega uporabnika in delovnimi nalogi ne glede na uporabniško ime.
- Seznama objektov, v katerem se prikazujejo podatki delovnih nalogov, kjer je prikazana samo številka delovnega naloga in pa priimek in ime pacienta.

Delovni nalog izberemo tako, da ga na seznamu z dotikom izberemo, po izbiri se prestavimo na naslednjo formo, »delovni nalog«.

V primeru, da želimo zaključiti z delom in zapustiti aplikacijo to storimo z gumbom »nazaj« (na telefonu). S tem dejanjem prikličemo pogovorno okno (Slika 22), ki zahteva potrditev za izhod.

Slika 22: Izhod iz aplikacije

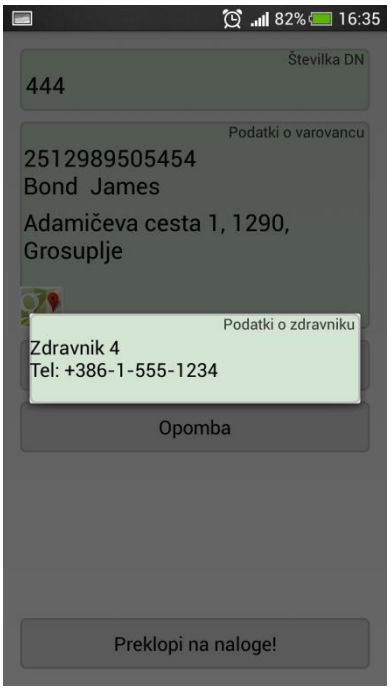
Delovni nalog

Slika 23 prikazuje podatke izbranega delovnega naloga.

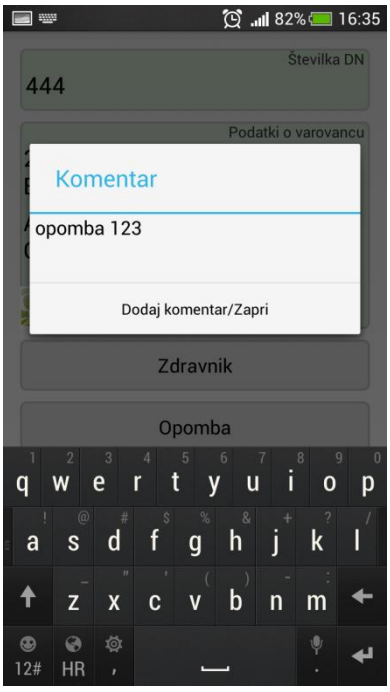
Slika 23: Obrazec delovnega naloga

Sestavljen je iz naslednjih komponent:

- Številka delovnega naloga: v tem polju se nam izpiše številka delovnega naloga, ki ga prikazuje obrazec.
- Podatki o varovancu: to polje prikazuje podatke o pacientu, na katerega je vezan delovni nalog, prikazana je številka pacienta, naziv ter naslov pacienta; potrebno je povedati, da se nam ob kliku na naslov pacienta zažene Google maps storitev (Slika 17), ki nam prikaže lokacijo naslova na Google maps zemljevidu.
- Gumb »Zdravnik«: s pritiskom na ta gumb se nam prikažejo podatki (Slika 24) o zdravniku, ki je bil zadolžen za odprtje tega delovnega naloga. Prikažeta se naziv zdravnika in pa telefonska številka zdravnika; ti podatki nam pridejo prav, če patronažna služba potrebuje dodatna navodila zdravnika. Z dotikom na telefonsko številko ga lahko tudi pokličemo.
- Gumb »Opomba«: gumb opomba nam odpre pogovorno okno (Slika 25) v katerega lahko zapisujemo poljubne zapise o pacientu ali kakšni drugi posebnosti na terenu, ki je vezana na odprti delovni nalog.
- Gumb »Preklopi na naloge!«: s pritiskom na ta gumb se premaknemo na seznam nalog, ki jih je potrebno opraviti na terenu.



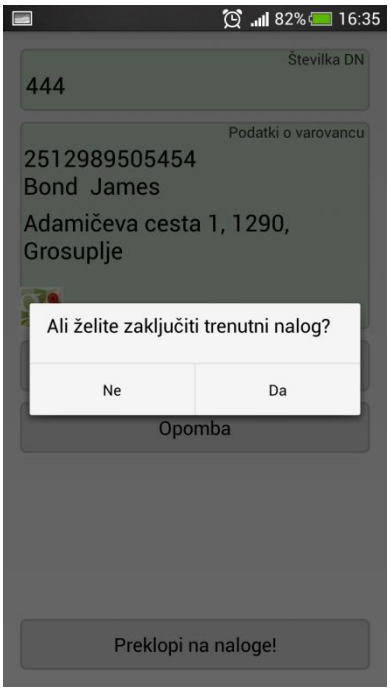
Slika 24: Podatki o zdravniku



Slika 25: Dodajanje opomb

Potrebno je omeniti, da se lahko med obrazcem »Delovni nalog« in obrazcem »Seznam nalog« premikamo s pritiskom na gumb »Preklopi na naloge!« ali pa z dotikom, tako, da se s prstom na zaslonu premaknemo v levo (Slika 23).

Za vrnitev na seznam nalogov je potrebno pritisniti na napravi gumb »nazaj«. Prikaže se nam pogovorno okno (Slika 26), ki nas vpraša ali želimo trenutni nalog zaključiti. S pritiskom na gumb »Da« ali »Ne« se vrnemo na seznam nalogov, s tem, da ob pritisku na »Da«, zapremo delovni nalog, ki se nam izbriše iz seznama nalogov in je pripravljen za oddajo na strežnik, ob pritisku na gumb »Ne«, ne zapremo naloga, vendar vse spremembe, ki so bile narejene na nalogu, ostanejo shranjene. V primeru, da trenutno odprtega delovnega naloga ne želimo zapustiti, moramo še enkrat pritisniti gumb nazaj, ki zapre pogovorno okno.



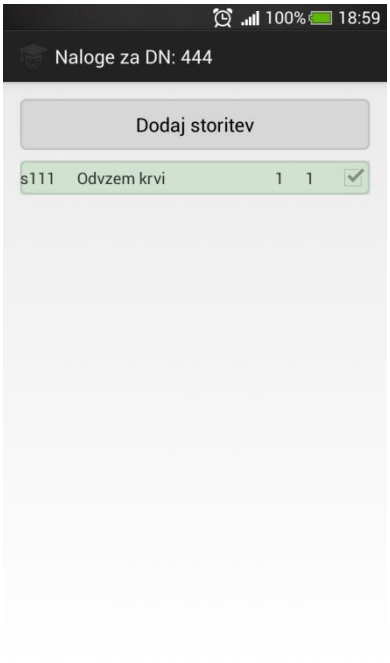
Slika 26: Izhod iz delovnega naloga

Seznam nalog

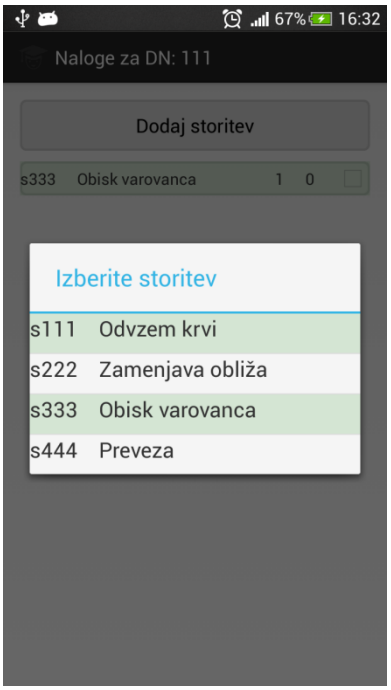
Obrazec seznama nalog (Slika 27) prikazuje storitve, ki jih mora patronažna služba oziroma prijavljen uporabnik opraviti na terenu. Na tem obrazcu lahko vidimo številko delovnega naloga in seznam nalog. Sestavljata ga:

- Gumb »Dodaj storitev«: s pritiskom na ta gumb se nam odpre pogovorno okno (Slika 28), ki prikaže šifrant storitev. Lahko jih dodajamo, vendar v večini primerov to ni potrebno.

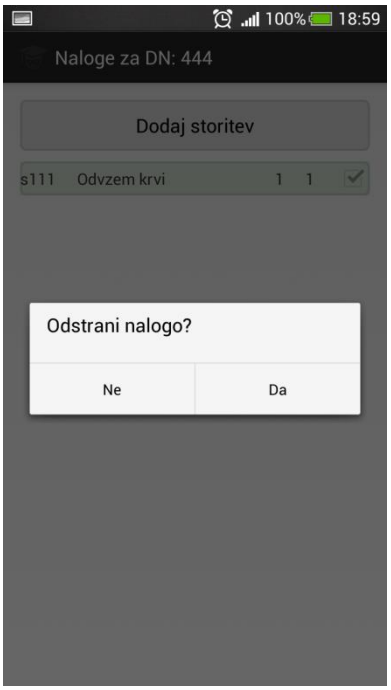
- Seznam storitev: seznam prikazuje storitve, ki jih mora opraviti patronažna služba.



Slika 27: Obrazec seznama nalog



Slika 28: Šifrant storitev



Slika 29: Odstranjevanje naloge

Seznam storitev je sestavljen iz štirih stolpcev:

- Prvi stolpec predstavlja identifikacijsko številko storitve.
- Drugi stolpec predstavlja naziv storitve.
- Tretji stolpec nam prikazuje količino storitve, ki mora biti opravljena.
- Četrti stolpec je količina storitve, ki je dejansko opravljena. Ko sta ta dva stolpca enaka, je naloga označena kot opravljena.

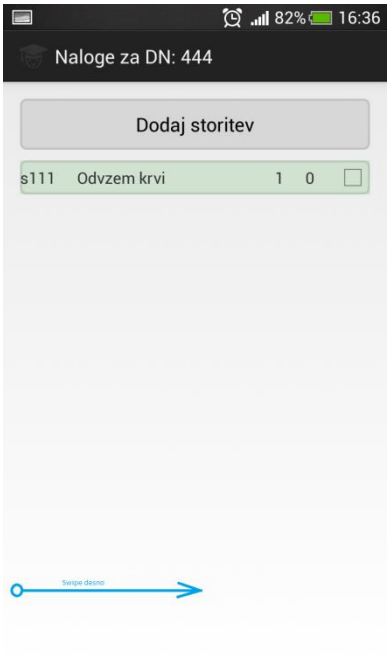
Potrjevanje opravljenih nalog:

- Nalogo lahko označimo kot potrjeno tako, da v seznamu storitev, kliknemo na izbrano storitev in se količina opravljene se storitve poveča za 1.
- Nalogo lahko potrdimo tudi tako, da odpremo šifrant storitev (Slika 28) in iz seznama izberemo isto storitev, kot je navedena v seznamu nalog.

Odstranjevanje storitve se načeloma ne uporablja, vendar včasih lahko pride do pomote, zato je bila dodana opcija za odstranitev. Storitev lahko odstranimo tako, da v seznamu storitev, z dolgim klikom na storitev, ki jo želimo odstraniti, prikličemo

pogovorno okno (Slika 29), ki nas vpraša, če želimo storitev odstraniti oziroma zmanjšati za 1.

Za izhod iz obrazca storitev je potrebno pritisniti na napravi gumb nazaj, to pa nas premakne nazaj na obrazec delovnega naloga. Isto lahko storimo tudi drug način in sicer tako, da se s prstom po zaslonu premaknemo v desno (Slika 30).



Slika 30: Izhod iz seznama nalog

6. Sklepne ugotovitve

Kot posebnost razvoja lahko navedem to, da je pri načrtovanju rešitve sodelovala tudi patronažna služba v Zdravstvenem domu Grosuplje, ki je bila neko vodilo pri razvoju celotne rešitve. Poskušal sem se držati tistega, kar so mi povedali, in aplikacijo prilagoditi njihovim potrebam in možnostim, ki jih vidim kot razvijalec.

Pri razvoju aplikacije sem spoznal delo patronažne službe, se veliko naučil o razvojnem okolju Eclipse, in ga sedaj poznam mnogo bolje, kot pred začetkom tega diplomskega dela, čeprav vem, da orodje omogoča še veliko več. Veliko sem se tudi naučil o razvoju aplikacij za Android platformo. Pred diplomsko nalogo sem se zanimal tudi za razvoj aplikacij za Android, ker se tudi v podjetju, kjer sem zaposlen ukvarjam s programiranjem v Microsoft Visual Studio, v programskem jeziku C#, vendar zaradi delovnih obveznosti nisem imel veliko več časa. S tem želim povedati, da me je diplomsko delo na nek način »prisililo« v učenje novih tehnologij.

Težave, s katerimi sem se srečal, so bile povezane z nepoznavanjem procesov razvoja v razvojnem okolju Eclipse in s tem tudi nepoznavanjem razvoja aplikacij za Android platformo. Največ sivih las mi delalo to, da je pri razvoju aplikacije za Android potrebno vsako komponento, ki jo dodamo, posebej definirat in določiti, kaj bo komponenta izvedla, tudi čisto navaden klik. Vendar sedaj, ko bolje poznam okolje, se mi zdi dokaj enostavno.

Mobilna aplikacija, ki je bila razvita v tem diplomskem delu, bi po mojem mnenju lahko poenostavila delo na terenu in ima potencial. Z nadaljnjim razvojem bi lahko postala obvezna oprema vsake patronažne službe, vendar samo ob predpostavki, da se poenoti informacijski sistem v zdravstvenih domovih.

7. Literatura

[1] „tech-thoughts,“ 2013. [Elektronski]. Dostopno na: http://www.tech-thoughts.net/2013/11/smartphone-market-share-by-country-q3-2013.html#.UqdA2PTuL_E.

[2] „Android, the world's most popular mobile platform,“ 2013. [Elektronski]. Dostopno na: http://developer.android.com/about/index.html.

[3] „BGR,“ 2013. [Elektronski]. Available: http://bgr.com/2013/12/03/android-kitkat-market-share/.

[4] „Android Architecture,“ 2013. [Elektronski]. Dostopno na: http://www.tutorialspoint.com/android/android_architecture.htm.

[5] „About SQLite,“ 2013. [Elektronski]. Dostopno na: http://www.sqlite.org/about.html.

[6] J. A. Kreibich, „Using SQLite,“ v *Using SQLite*, O’Reilly Media, Inc., 2010, pp. 1-7.

[7] „Introducing JSON,“ 2013. [Elektronski]. Dostopno na: http://json.org/.

[8] „Codeproject,“ 2013. [Elektronski]. Dostopno na: http://www.codeproject.com/Articles/17908/What-is-Web-Service.

[9] „Eclipse Newcomers FAQ,“ 2013. [Elektronski]. Dostopno na: http://www.eclipse.org/home/newcomers.php.

[10] „Vtičniki,“ 2013. [Elektronski]. Dostopno na: https://support.google.com/chrome/answer/142064.

[11] „Java JRE,“ 2013. [Elektronski]. Dostopno na: http://www.s-sers.mb.edus.si/gradiva/w3/elek_v1/Kazala/kazalo_navodila/4_korak.html.

[12] „ADT Plugin,“ 2013. [Elektronski]. Dostopno na: http://developer.android.com/tools/sdk/eclipse-adt.html.

[13] Google, „Google Maps Android API v2,“ 2013. [Elektronski]. Dostopno na: https://developers.google.com/maps/documentation/android/start#overview.

[14] „Andorid Google Maps Tutorial,“ 2013. [Elektronski]. Dostopno na: http://www.androidhive.info/2013/08/android-working-with-google-maps-v2/.

[15] „Collins,“ 2013. [Elektronski]. Dostopno na: http://www.collinsdictionary.com/submission/11527/Smart%20Device.

[16] „HTC,“ 2013. [Elektronski]. Dostopno na: http://www.htc.com/.

[17] „RapidNFC,“ 2013. [Elektronski]. Dostopno na: http://rapidnfc.com/blog/72/the_difference_between_nfc_and_rfid_explained.

[18] „NFC,“ 2013. [Elektronski]. Dostopno na: http://www.nfc.cc/technology/nfc/.

[19] Z. d. Grosuplje, Pravilnik patronažne službe, PRA 903 - 2, 2011.

[20] „Geocoder,“ 2013. [Elektronski]. Dostopno na: http://developer.android.com/reference/android/location/Geocoder.html.