Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matjaž Krašovec

Mobilna aplikacija za hranjenje in prikaz lokacijskih podatkov

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00433 / 2013
Datum: 15.4.2013

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

Kandidat: **MATJAŽ KRAŠOVEC**

Naslov: **MOBILNA APLIKACIJA ZA HRANJENJE IN PRIKAZ LOKACIJSKIH PODATKOV**
MOBILE APPLICATION FOR STORING AND PRESENTING LOCATION DATA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Uporabnost mobilnih aplikacij pride še zlasti do izraza, ko te prikazujejo lokacijske podatke. V diplomski nalogi sistematično prikažite razvoj tovrstne mobilne aplikacije, ki omogoča učinkovit zajem, hrambo in prikaz množice lokacijskih podatkov. Nalogo zaključite z analizo uporabnosti aplikacije v praksi.

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Mentor: Dekan:

viš. pred. dr. Igor Rožanc prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matjaž Krašovec, z vpisno številko **63070280**, sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za hranjenje in prikaz lokacijskih podatkov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- 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 15. februarja 2014 Podpis avtorja:

Zahvaljujem se moji dragi Niki, ki me je ves čas priganjala k delu in podpirala med študijem.

Zahvalil bi se rad družini, predvsem mami Ireni in očetu Janezu, ki sta me vedno podpirala tako finančno kot tudi moralno.

Zahvaliti se moram tudi obema babicama, Fridi in Heleni, dedku Stanetu in pokojnemu dedku Stanetu, pa tudi stricu Tomažu in tetam Heleni, Nataliji in Tini.

Iskreno se zahvaljujem vsem prijateljem, sošolcem in profesorjem, ki so kakorkoli pripomogli na moji poti do tako pomembnega mejnika v življenju.

Posebna zahvala gre viš. pred. dr. Igorju Rožancu za kakovostno in potrpežljivo mentorstvo.

Zahvala gre tudi Martini za hitro in kakovostno lektoriranje.

To diplomsko delo posvečam dekletu Niki in pokojnemu dedku Stanetu.

Kazalo

Seznam uporabljenih kratic in simbolov

ADT	Android Developer Tools	razvojna orodja za Android
API	Application Programming Interface	vmesnik uporabniškega programa
GPS	Global Positioning System	navigacijski sistem
HTML	Hyper Text Markup Language	jezik za označevanje hiperteksta
IDE	Integrated Development Environment	integrirano razvojno okolje
Java EE	Java Enterprise Edition	Java izdaja za srednja in velika podjetja
Java ME	Java Micro Edition	Java izdaja za majhne in mobilne naprave
Java SE	Java Standard Edition	Java izdaja za osebne računalnike
JVM	Java Virtual Machine	Java navidezni stroj
KML	Keyhole Markup Language	označevalni jezik uporabljen za zapis geografskih podatkov
KMZ	Keyhole Markup Language Zipped	stisnjen fo¸rmat KML
SDK	Software Development Kit	paket programskih orodij za razvoj
SQL	Structured Query Language	sestavljeni jezik za poizvedbe
XML	Extensible Markup Language	razširljivi označevalni jezik

XP	Extreme Programming	Ekstremno programiranje

Povzetek

Cilj tega diplomskega dela je opisati in izdelati aplikacijo za mobilno platformo Android, ki nam omogoča hranjenje in prikaz lokacijskih podatkov. Prvi del diplomskega dela je namenjen predstavitvi orodij in tehnologij, ki jih uporabljamo pri razvoju mobilnih aplikacij. Pri predstavitvi orodij smo se osredotočili na orodja, ki smo jih uporabljali pri razvoju aplikacije. Preostala orodja so opisana manj podrobno.

V drugem delu pa smo predstavili idejo za aplikacijo, potek izdelave aplikacije, nekaj zanimivih delov kode v programskem jeziku Java, težave pri izvedbi in možne izboljšave. Predstavili pa smo tudi podobne aplikacije, ki so na voljo za platformo Android.

Rezultat diplomske naloge je delujoča aplikacija, ki je napisana v programskem jeziku Java. Aplikacija se s storitvijo Google zemljevidi poveže s pomočjo ustreznih knjižnic, ki med drugim omogočajo označevanje točk na zemljevidu, ki so predstavljene kot koordinate zemljepisne dolžine in širine.

Z izdelano aplikacijo lahko na zemljevidu označimo točke, za katere lahko vpišemo različne opise, npr. naslov lokacije ali opis lokacije. Aplikacija omogoča tudi prikaz našega trenutnega položaja na zemljevidu, prikaz najbližjih in vseh točk, ki smo jih predtem vnesli ali uvozili. Podprli smo uvoz datotek KML (angl. Keyhole Markup Language). Aplikacija se lahko uporablja na vseh mobilnih napravah, ki imajo nameščen operacijski sistem Android.

Pri izdelavi diplomskega dela smo osvojili veliko novih znanj s področja razvoja aplikacij za platformo Android. Naš glavni cilj, razviti rešitev za

hranjenje in prikaz lokacijskih podatkov je bil torej dosežen.

Ključne besede: Android, aplikacije Android, Google zemljevidi, zemljepisna dolžina, zemljepisna širina.

Abstract

The aim of this thesis is to describe and develop an application for the Android mobile platform that will be able to store and display location data. The first part of the thesis is devoted to the presentation of tools and technologies that are used in the development of mobile applications. In presenting tools, we focused on those used in our application. Other tools are described in less detail.

In the second part, we presented an idea for the application, the process of its development, some interesting pieces of code in the Java programming language, implementation problems and possible improvements. We also presented similar applications available for the Android platform.

The result of the thesis is an application that is written in the Java programming language. The application utilizes special libraries to connect with Google Maps, allowing us, among other things, to mark points on a map presented as coordinates of longitude and latitude.

The developed application allows us to mark points on the map, for which we can then enter descriptions, for example the address or description of the location. The application also allows us to view our current position on the map and select the nearest and all points that have been previously entered or imported. The application supports the importing of KML files (Keyhole Markup Language). It can be used on all mobile devices that are running the Android operating system.

During writing this thesis, we have gained considerable expertise in the field of application development for the Android platform. Our main goal to

develop a solution for storing and displaying location data has been achieved.

Keywords: Android, Android applications, Google maps, longitude, latitude.

Poglavje 1

Uvod

Živimo v obdobju, ko imajo mobilne naprave veliko vlogo v našem življenju. Samo pomislimo, kdo od nas še nima tabličnega računalnika ali vsaj pametnega telefona.

Obstaja veliko storitev za pametne telefone, ki nas nenehno zasipajo z raznimi podatki in informacijami. Zmogljivost teh vrst naprav je dandanes že skoraj primerljiva z zmogljivostjo osebnih računalnikov. Največja prednost aplikacij za pametne telefone je ta, da ga imamo skoraj vedno s sabo, zato nam lahko velikokrat olajša različne odločitve. Redke naprave nimajo vgrajenega sprejemnika GPS (angl. Global Positioning System), vmesnika za povezavo z brezžičnim omrežjem, vmesnika Bluetooth in drugih podobnih tehnologij. Vse to nam omogoča, da uporabimo mobilno napravo za veliko različnih namenov.

Trenutno najbolj razširjen mobilni operacijski sistem je Android. Aplikacije Android so večinoma razvite z uporabo programskega jezika Java. Tudi sami uporabljamo pametni mobilni telefon, na katerega je nameščen ta sistem.

To je bil poleg tega, da nas zanima programiranje v programskem jeziku Java, eden glavnih razlogov za razvoj mobilne aplikacije na platformi Android. Razlog pa je bil tudi ta, da podobne aplikacije ne omogočajo nekaterih funkcionalnosti, ki si ji želimo. Čeprav smo imeli zelo malo predhodnjih

izkušenj z razvojem aplikacij Android, smo se kljub temu odločili, da bo glavni cilj diplomskega dela prikaz razvoja mobilne aplikacije na platformi Android.

Danes obstaja veliko storitev, ki za svoje delovanje uporabljajo geografske podatke. Ena najbolj znanih je Google Maps, ki zna prikazati zemljevid določenega kraja. Podjetje Google je svoje storitve tudi ponudilo na voljo v obliki Google Maps API, ki je dejansko vmesnik uporabniškega programa, s pomočjo katerega lahko uporabljamo zemljevide in označujemo točke, poti in še množico drugih stvari, ki pa za nas pri diplomskem delu niso bile relevantne.

Seznanili smo se z razvojem programske opreme, ki smo ga do zdaj poznali le s teoretičnega vidika. Še prej pa smo preučili literaturo, ki obravnava operacijski sistem Android, ter razvoj aplikacij Android. Nato smo si namestili ustrezna orodja in začeli razvijati aplikacijo po posameznih fazah. Rezultat našega dela je delujoča mobilna aplikacija Maps, ki omogoča številne funkcionalnosti ter deluje na mobilnih napravah, ki imajo nameščen operacijski sistem Android, povezavo do svetovnega spleta in sprejemnik GPS.

Poglavje 2

Predstavitev uporabljenih metodologij, tehnologij in orodij

V tem poglavju bomo predstavili metodologije, tehnologije in orodja, ki smo jih uporabljali med razvojem mobilne aplikacije. Podrobneje so opisane stvari, ki smo jih pri našem delu uporabili, ostale pa so le omenjene.

2.1 Metodologije

2.1.1 Kaj je metodologija

Metodologija [4] je skupek metod, postopkov in standardov, ki sestavljajo zaključeno celoto pri izvajanju inženirskih pristopov k razvoju produkta. Pri razvoju programske opreme to pomeni postopen način razvoja izdelka, ki vključuje uporabo različnih tehnik in orodij.

Metodologija je prežeta z idejami ter načeli organizacije in njenih članov, kar še posebej poudari njeno sociološko komponento. Ne more nastati neodvisna od ljudi oz. organizacije. Čeprav so nekatere metodologije formalno določene, si jih posamezna podjetja prilagajajo po svoje, da ustrezajo njiho-

vemu načinu dela in njihovi domeni. Z uporabo metodologije si uporabniki pridobivajo izkušnje, s tem pa se bogati tudi metodologija sama [6].

2.1.2 Vrste metodologij

Če metodologije delomo glede na njihovo utežitev, jih lahko delimo na spredaj utežene, zadaj utežene in uravnotežene [8].

Spredaj utežene metodologije (angl. frontloaded) dajejo poudarek postopkom analize in načrtovanja, ki jih v veliki meri izvedemo vnaprej. Rezultat izvajanja teh postopkov so do podrobnosti opredeljene zahteve za sistem, izdelani natančni načrti za sistem in podobno. Nastali načrt se pozneje bistveno ne spreminja, izdelava večinoma pomeni samo "rutinsko kodiranje metod", ki so podrobno opredeljene z načrtom.
Take metodologije so primerne za razvoj:

- sistemov s stabilnimi zahtevami. Ker je poudarek na analizi in načrtovanju, ki ju večinoma izvedemo vnaprej, morajo biti zahteve dobro specificirane in stabilne. Pozneje so večje spremembe v načrtu drage, pogosto pa tudi podirajo dobro arhitekturo prvotno načrtovanega sistema;

- kritičnih sistemov. Pri kritičnih sistemih (npr. življenjsko kritični sistemi) sta dobra analiza in načrtovanje ključnega pomena. Pri tem moramo predvideti tudi čim več možnih alternativnih scenarijev dogajanja (alternativni dogodki, napake), ki jih zajamemo že v načrtu;

- obsežnih in zahtevnih sistemov z velikimi razvojnimi skupinami. Kadar imamo opravka z veliko razvojno skupino, so potrebni bolj podrobni načrti, ki omogočajo usklajevanje in razdeljevanje dela med člani.

Med spredaj utežene metodologije lahko uvrstimo večino klasičnih metodologij. Glavna kritika takih metodologij je, da preveč časa posvečajo arhitekturi in načrtovanju, rezultat pa je načrt, ki zaradi hitrega spreminjanja zahtev ni uporaben. Pomembna kritika je tudi, da zvesto sledenje takim metodologijam lahko pomeni podrobno definiranje stvari, ki jih še ne poznamo dobro

oziroma ki jih bomo spoznali šele v poznejših fazah razvoja (npr. šele v izvedbi).

Zadaj utežene metodologije (angl. backloaded) dajejo prednost postopkom izvedbe in testiranja. Analiza in načrtovanje sta potisnjena v ozadje, izdelani so le osnovni načrti, ki se ne ukvarjajo s podrobnostmi. Razvijalci se takoj začnejo ukvarjati s kodiranjem, pri čemer analizirajo zahteve in načrtujejo sistem kar sproti. Z obsežnim testiranjem in popravljanjem kode lahko pri uporabi zadaj uteženih metodologij nastane relativno stabilen izdelek, vendar je arhitektura takega izdelka navadno kljub temu manj robustna in razširljiva.
Zadaj utežene metodologije so primerne za razvoj:

- manjših in bolj enostavnih sistemov, z majhnimi in izkušenimi razvojnimi skupinami, hkrati pa podrobnosti načrta nadomeščta znanje in izkušenost članov projektne skupine;

- sistemov s slabo določenimi zahtevamimi, ki pa se bodo najverjetneje še spreminjale. Kadar zahtev ni mogoče dobro definirati vnaprej je možen pristop uporaba zadaj utežene metodologije;

- nekritičnih sistemov. Posledica manj podrobne analize in načrtovanja bo tudi manj stabilna arhitektura, ki ni primerna za kritične sisteme. Hkrati je take sisteme pozneje težje nadgrajevati in vzdrževati;

- sistemov, ki uporabljajo tehnologijo, s katero nismo seznanjeni. V primeru, da ne poznamo tehnologije (npr. pogost primer v zadnjem času je razvoj aplikacij za splet), lahko z uporabo zadaj utežene metodologije tehnologijo preizkusimo in spoznamo.

Med take pristope bi lahko uvrstili metodologije, ki temeljijo na prototipiranju kot življenjskem ciklu razvoja. Hkrati pa bi med te metodologije uvrstili tudi večino agilnih metodologij [5]. Agilne metodologije imajo skupna načela, ki omogočajo hiter razvoj kvalitetne programske opreme. Glavna kritika teh

metodologij je, da niso primerne za razvoj obsežnih sistemov ter da so pogosto opravičilo za razvoj tipa kodiraj in popravi, katerega rezultat sta slaba koda in nestabilen sistem.

Uravnotežene metodologije (angl. balanced) dopuščajo, da za tiste dele sistema, za katere so zahteve dobro znane in stabilne, izvedemo podrobnejšo analizo. Za dele, ki jih še ne poznamo dovolj, začnemo kodirati in testirati (npr. izdelava prototipov), to pa nam pomaga pri podrobnejšem določanju zahtev. Gre torej za to, da nekatere aktivnosti izvajamo na spredaj uravnotežen način, druge pa na zadaj uravnotežen način.

Četudi se morda na prvi pogled zdi, da so uravnotežene metodologije idealna izbira za vse primere, pa to ne drži. Treba se je zavedati, da uteženi procesi nimajo niti vseh značilnosti spredaj, niti vseh značilnosti zadaj uteženih procesov.

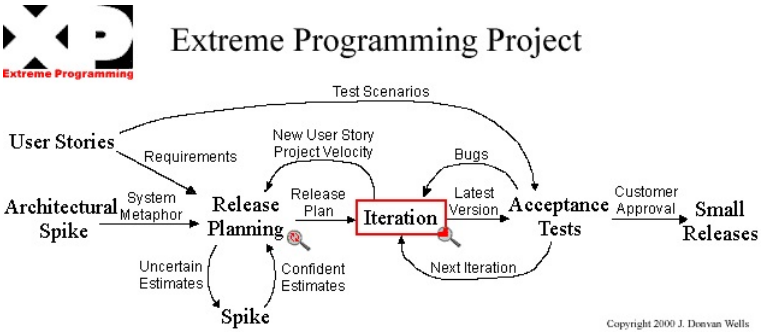
Tako je na primer uporaba spredaj uteženega procesa pri kritičnemu sistemu z dobro definiranimi zahtevami bolj ustrezna kot uporaba uravnoteženega [8].

2.1.3 Metodologija ekstremno programiranje – XP

Metodologija ekstremno programiranje [9] (angl. Extreme Programming) se je razvila kot odgovor na težave, ki so nastale zaradi dolgih razvojnih ciklov, kot so jih poznali drugi razvojni modeli. Cikel razvoja po metodologiji XP lahko vidimo na sliki 2.1. Nastala je iz potreb in praks, ki so bile učinkovite v procesu razvoja. Po več preizkušnjah v praksi je nastala teoretična oblika metodologije ekstremnega programiranja na podlagi ključnih principov in praks.

Čeprav posamezne prakse uporabljene, v metodi, niso nove, so bile zbrane in povezane tako, da oblikujejo novo metodologijo razvoja programske opreme. Izraz ekstremno izhaja iz dejstva, da metodologija uporablja smiselne in že obstoječe prakse, principe na nov in ekstremen način.

Ekstremno programiranje bi uvrstili med zadaj utežene ali tako imenovane agilne metodologije. Je učinkovit, nizko tvegan in zabaven način analize,



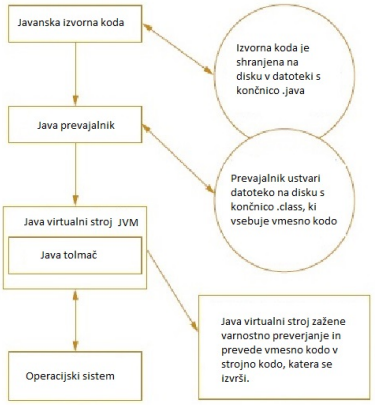
Slika 2.1: Projekt razvoja po metodologiji XP [9].

načrtovanja in izdelave programske opreme [3]. Štiri vrednote XP so:

- komunikacija,
- enostavnost,
- povratne informacije,
- pogum.

XP je primeren za majhne sisteme, kjer se zahteve spreminjajo in hitro potrebujemo učinkovite rešitve. Uporaba ekstremnega programiranja ni primerna:

- za prevelike, predolge ali preveč zahtevne projekte,
- takrat, ko ga razvijalci niso pripravljeni sprejeti,
- ko je prekomerno in prepogosto prekoračen urnik 40 ur dela na teden,
- ob uporabi določene tehnologije, ki eksponentno viša stroške,
- ob prepočasni povratni informaciji naročnikov,
- ko nimamo ustreznega delovnega okolja,
- ko so obdobja med ponovitvami daljša od 5 mesecev.



Slika 2.2: Izvajanje javanskega programa [1].

2.2 Programski jezik Java

Java [1] je bila razvita v podjetju Sun Microsystems (pozneje ga je prevzelo podjetje Oracle) kot objektno usmerjen programski jezik za namene poslovnih in spletnih aplikacij. Med prednostmi, ki so naredile Javo tako priljubljeno v zadnjih letih so njena varnost in dejstvo, da je neodvisna od platforme, kar pomeni, da lahko uporabimo program, napisan v Javi, na katerikoli platformi oziroma operacijskem sistemu.

Javansko aplikacijo lahko poženemo na različnih platformah, ker se aplikacija ne izvede neposredno na platformi, ampak se izvede na tako imenovanem virtualnem stroju Java (angl. Java Virtual Machine). Programska koda, napisana v visokonivojskem programskem jeziku, se imenuje izvorna koda (angl. source code).

Ko pišemo javansko aplikacijo, moramo izvorno kodo najprej napisati z urejevalnikom besedila, npr. Beležnico. Koda je shranjena v datoteki, potem jo prevajalnik (angl. compiler) prevede v vmesno kodo (angl. bytecode). Nato se vmesna koda s pomočjo tolmača Java (angl. Java Interpreter) preveri in izvaja na JVM. Postopek prikazuje slika 2.2. Ker se program izvaja na JVM in ne na operacijskem sistemu, je izoliran tudi glede na strojno opremo, na kateri teče. JVM tako zagotavlja varnost pred vsiljivci (angl. intruders),

ki bi želeli dostopati do strojne opreme skozi operacijski sistem. Druga velika prednost JVM, je manj dela za programerje, saj morajo drugače proizvesti več različic istega programa za različne platforme [1].

Java je prav tako preprostejša od večine drugih objektno usmerjenih programskih jezikov, saj:

- ni aritmetike s kazalci,
- pretvorbo med različnimi tipi preverja prevajalnik,
- samodejno upravlja s pomnilnikom in
- uporablja angleške izraze, kar nam olajša branje, razumevanje in pisanje izvorne kode.

Obstajajo 3 različne izdaje, glede na to kakšen produkt razvijamo:

- tehnologija Java v majhnih in mobilnih napravah (Java ME),
- tehnologija Java v osebnih računalnikih (Java SE),
- tehnologija Java v srednjem in velikem poslovnem okolju (Java EE).

V našem primeru smo uporabljali verzijo Java SE, ki smo jo namestili z uradne spletne strani podjetja Oracle.

2.3 Platforma Android

Android [26] je odprtokodni operacijski sistem za pametne telefone in druge prenosne naprave. Zgrajen je na Linux [1] jedru in prvotno namenjen mobilnim napravam na dotik, npr. pametnim telefonom, tabličnim računalnikom. Prednosti uporabe platforme Android so:

[1] Linux je prost operacijski sistem s prosto dostopno izvorno kodo, zaščiteno s splošnim dovoljenjem GNU [24].

- odprtokodnost, ki omogoča cenejše in lažje razvijanje programov. Prednost občutimo tudi uporabniki, saj so programi za ta operacijski sistem večinoma zastonjn;
- omogoča cenejše, lažje in hitrejše razvijanje pametnih telefonov, saj proizvajalcem ni več treba razvijati operacijskih sistemov;
- je enostaven, odziven in omogoča večopravilnost in
- samodejno se sinhronizira z Googlovimi storitvami.

2.3.1 Aplikacije

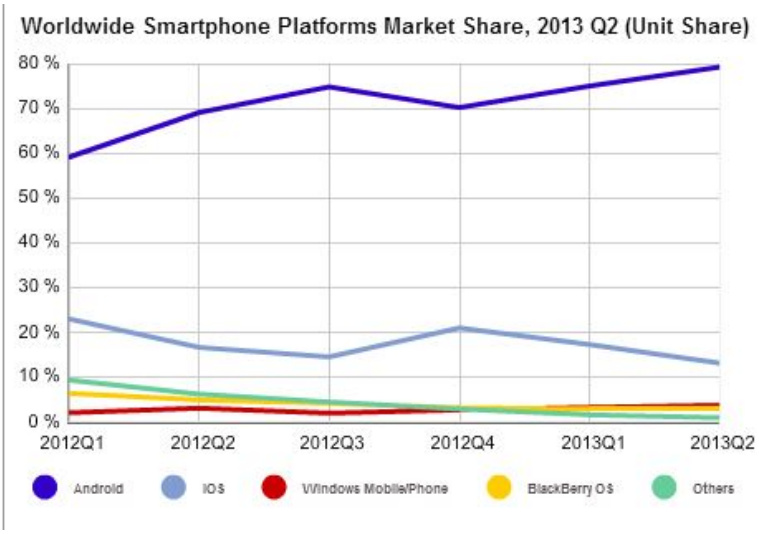
Vse aplikacije so napisane v programskem jeziku Java. Aplikacije so postavljene v Android paket s končnico ***.apk***. Vsaka aplikacija se požene v svojem Linux procesu. Operacijski sistem požene proces takrat, ko mu je poslana zahteva za izvajanje aplikacije. Ko izvajanje aplikacije zaključimo, se proces zapre.

To omogoča porabo pomnilnika tudi drugim aplikacijam. Vsak posamezni proces se prevede posebej, kar omogoči izolacijo aplikacij, da delujejo med sabo neodvisno. Za vsako aplikacijo se ob zagonu ustvari lastna identifikacijska koda, tej pa se nato dodajo pravice za uporabo strojne opreme.

2.3.2 Aplikacijsko ogrodje

V aplikacijskem ogrodju se nahajajo vse sistemske aplikacije, ki se uporabljajo za kodiranje aplikacij. Te aplikacije so:

- upravljalec aktivnosti,
- upravljalec pomnilnika,
- upravljalec lokacij in
- upravljalec obvestil.



Slika 2.3: Delež OS za pametne telefone [7].

2.3.3 Google Play

Google Play [21] je licenčna aplikacija podjetja Google, ki se uporablja za prenos oziroma nalaganje aplikacij. To se izvršuje na dva načina. Lahko jih nalagamo neposredno prek aplikacije ali pa jih naložimo prek enolične kode, ki jo s prenosno napravo preberemo.

Pri branju nato Google Play sam poišče aplikacijo na njihovem strežniku, jo prenese ter namesti na prenosno napravo. Poleg aplikacij so na voljo tudi filmi, glasba, knjige, revije in naprave – te sicer v Sloveniji še niso na voljo.

2.3.4 Delež OS na trgu pametnih telefonov

Kot vidimo na sliki 2.3 [7] je delež OS Android daleč pred vsemi in še narašča. Sklepamo, da je platforma tako priljubljena, ker je brezplačna, odprtokodna in ker je na voljo veliko različic, s katerimi lahko proizvajalci pametnih mobilnih telefonov opremijo naprave.

Vidimo, da naprave z OS Android, zavzemajo pribižno 80 odstotkov trga. Sledi mu sistem iOS podjetja Apple z nekaj več kot 10 odstotki, opazimo pa

tudi, da delež naprav s sistemom Windows MobilePhone narašča.

2.4 SQLite

SQLite [38] je programska knjižnica, ki implementira:

- samovsebujoč (angl. self - contained),
- brezstrežniški (angl. serverless),
- transakcijski (angl. transactional) in
- brezkonfiguracijski (angl. zero configuration)
- SQL [2] pogon podatkovne baze (angl. database engine).

V nasprotju z drugimi podatkovna baza SQLite, nima ločenega strežniškega procesa, ampak se vse branje in pisanje izvede neposredno na disk. Celotna podatkovna baza SQL z več tabelami, indeksi, sprožilci je shranjena kot ena datoteka na disku.

Datotečni format SQLite ni vezan na samo eno specifično platformo. Podatkovna baza, ki je pravzaprav datoteka, napisana na enem računalniku, se lahko enostavno prenese na računalnik z drugo arhitekturo/platformo. Vsi računalniki uporabljajo enak datotečni format.

Z vsemi funkcijami, ki jih omogoča, je knjižnica zelo majhne velikost (cca. 500 Kb), odvisno od ciljne platforme in nastavitev prevajalnika. SQLite je prav zaradi svoje majhnosti velikokrat uporabljena na mobilnih telefonih, MP3 predvajalnikih, ki so omejeni s spominom. Baza deluje bolje v sistemih z veliko pomnilnika, vendar je njena zmogljivost zelo dobra tudi v sistemih z malo pomnilniškega prosotora.

[2] SQL (angl. Structured Query Language) je strukturirani povpraševalni jezik za delo s podatkovnimi bazami. Je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi zbirkami, s programskimi stavki, ki posnemajo ukaze v naravnem jeziku [37].

SQLite je velikokrat testirana in ima sloves zelo zanesljive podatkovne baze. Večino izvorne kode je javno objavljeno in namenjeno testiranju in preverjanju učinkovitosti. Pri bazi je poskrbljeno tudi za napake pri dodeljevanju pomnilnika in vhodno/izhodne napake diska.

SQlite je transakcijska podatkovna baza. To pomeni, da se vse spremembe in poizvedbe pojavljajo kot atomske, konsistentne, izolirane in trajne (angl. Acid, Consistent, Isolated, Durable - ACID). To velja tudi, če je transakcija prekinjena zaradi napake v programu, operacijskem sistemu ali pa če računalnik izgubi električni tok. Vsaka transakcija se izvede v celoti ali pa se sploh ne izvede.

2.5 Razčlenjevalniki

Razčlenjevanje (angl. Parsing) [31] je dejanje razčlenitve informacij na dele. Razčlenjevalniki so programi (del kode ali API, na katerega se lahko sklicujemo znotraj svojih programov), ki analizirajo datoteke za identifikacijo posameznih delov. Vse aplikacije, ki berejo vhodne podatke, imajo neke vrste razčlenjevalnik, saj v nasprotnem primeru ne bi bili sposobni razbrati, kaj določena informacija pomeni. Microsoft Word vsebuje razčlenjevalnik, ki se zažene, ko odpremo datoteko, da preveri vse skrite kode. Če odpremo poškodovano datoteko (manjka del kode), nam program pošlje sporočilo o napaki.

Aplikacije za delo z datotekami XML [40] so enake. Vsebujejo razčlenjevalnik, ki prebere dokument XML, in preveri ter identificira vrsto podatkov, ki jih nato shrani v spomin in tako preda na voljo preostalemu delu programa.

Med branjem XML dokumenta razčlenjevalnik preveri strukturo dokumenta (značke itd.) ter sporoči napako, če je ta prisotna. V programskem jeziku Java poznamo naslednje razčlenjevalnike:

- razčlenjevalnik DOM (Document Object Model) [32],
- razčlenjevalnik SAX (Simple API for XML) [33] in

- razčlenjevalnik StAX(Streaming API for XML) [34].

2.5.1 Razčlenjevalnik SAX

Razčlenjevalnik SAX se od drugih razlikuje po tem, da je obdeluje dele dokumenta zaporedno. Razčlenjevalnik DOM deluje ravno nasprotno, saj v pomnilnik naloži celoten dokument. Zato je SAX uporaben v okoljih z malo pomnilnika.

Na voljo imamo več metod, ki jih nato prilagodimo za obdelavo našega dokumenta. Metode, ki jih uporabljamo, so naslednje:

- ***public void startDocument ()*** – določimo lahko, kaj se zgodi ob začetku dokumenta, npr. definiramo ustrezne podatkovne strukture,
- ***public void endDocument ()*** – določimo lahko, kaj se zgodi ob koncu dokumenta,
- ***public void startElement (String uri, String name, String qName, Attributes atts)*** – določimo, kaj se zgodi, ko razčlenjevalnik naleti na začetno značko z imenom ali vrednostjo, in
- ***public void endElement (String uri, String name, String qName)*** – določimo, kaj se zgodi, ko razčlenjevalnik naleti na končno značko z imenom ali vrednostjo.

2.6 Standardni opis podatkov

Standardni opis podatkov pomeni, da na standardni način opišemo podatke. Tega se poslužujemo zato, da so opisi standardni oziroma enaki pri neki vrsti podatkov, kar nam omogoča hitrejše razumevanje in tudi združjivost enake vrste podatkov. V računalništvu to pomeni, da lahko podatke uporabimo v več različnih aplikacijah oziroma za več različnih namenov. Ponavadi se uporabi opisni jezik XML, v katerem podatke razčlenimo.

2.6.1 XML

XML [40] je razširljiv označevalni jezik, ki ga pogosto srečamo, če brskamo po internetu in med komunikacijo aplikacije s strežnikom. Je preprost, podoben HTML, ki nam omogoča format za opisovanje podatkov. Mogoče ga je tudi razširiti, saj ima namreč to možnost, da si lahko sami izmislimo imena oznak (angl. tag). XML je razdeljen na 3 dele:

- podatkovni (vanj shranimo podatke v neki obliki z željenimi oznakami),
- deklarativni (skrbi za to, da lahko pri dodajanju novih podatkov vidimo kaj kakšna oznaka pomeni) in
- predstavitveni del (z njim oblikujemo izpis podatkov).

XML je pravilen, če je ustrezno oblikovan (izpolnjuje vsa sintaksna pravila) in veljaven. Je generično ogrodje za hrambo katerekoli dolžine teksta ali podatkov, katerih struktura je lahko prikazana tudi v obliki drevesa. Primer dokumenta lahko vidimo na sliki 2.4. Sintaksna pravila določajo, da:

- morajo imeti vsi elementi XML končno oznako,
- oznake so občutljive za velike in male črke,
- elementi morajo biti pravilno ugnezdeni ter
- dokument mora imeti en element, ki bo korenski (angl. root).

Sintaksnih pravil je veliko več, kot smo jih predstavili, vendar za predstavitev jezika XML bralcu ni treba poznati vseh. Veljaven dokument XML izpoljnjuje določena semantična pravila. Ta pravila lahko definira uporabnik ali pa so definirana v shemi XML. Če dokument vsebuje nedefiniran element, je celoten dokument neveljaven.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Slika 2.4: Primer XML dokumenta.

2.6.2 KML

KML (angl. Keyhole Markup Language) [25] je notacija XML za zapis geografskih podatkov v aplikacijah za prikaz lokacij, kot sta naprimer: Google Zemlja (angl. Google Earth) [18] in Google Zemljevidi (angl. Google Maps) [19]. KML uporablja strukturo ugnezdenih elementov z oznakami in atributov, ki so v skladu s standardi XML. Vse oznake morajo biti v skladu z dokumentacijo KML, v kateri je predvideno, katere oznake so potrebne in katere so opcijske. Pri elementu je pomembno, da se podatki navedejo v istem vrstnem redu, kot je navedeno v dokumentaciji. Podatki, ki jih vsebuje KML, so:

- latitude ali geografska dolžina,
- longitude ali geografska širina,
- altitude ali nadmorska višina,
- name ali ime oznake in
- description ali opis.

in še mnogo drugih. V dokumentu lahko označimo npr. tudi pot, 3D model, vendar smo se osredotočil le na zadeve, ki smo jih uporabljali pri našem delu.

```
<Placemark>
    <name>Oznaka2</name>
            <description>dasfasa</description>
    <LookAt>
        <longitude>15.35970945399048</longitude>
        <latitude>45.66004037610991</latitude>
        <altitude>0</altitude>
        <heading>-5.569090979563353</heading>
```

Slika 2.5: Primer KML dokumenta.

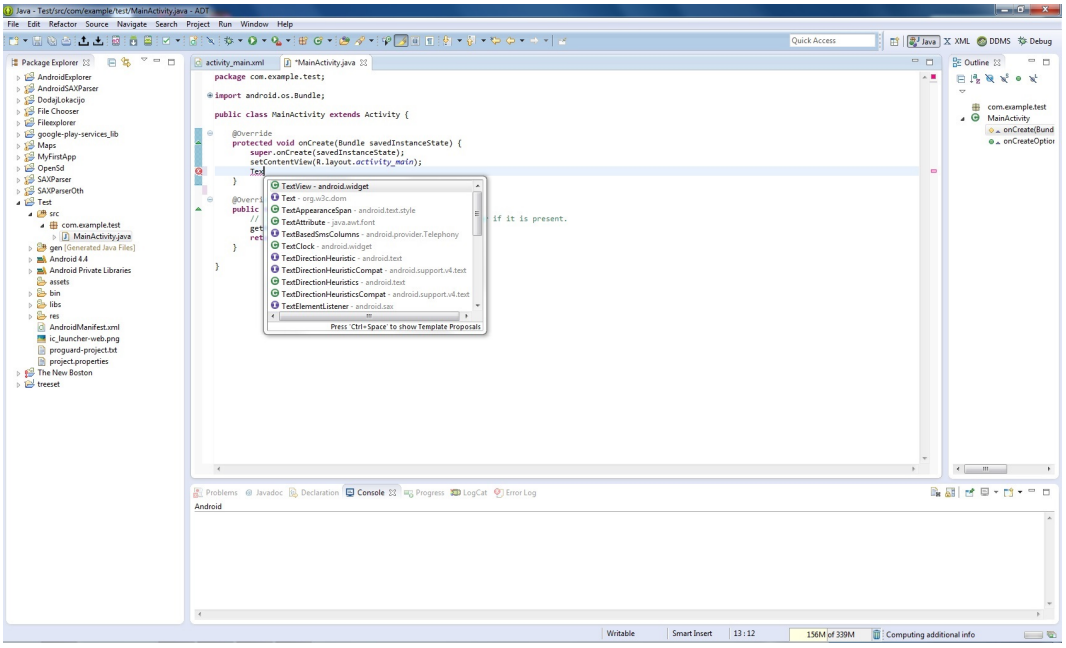
2.7 Razvojno orodje Eclipse

Z razvojnimi orodji je razvoj programov oz. aplikacij mnogo lažji. Aplikacije bi lahko razvijali tudi z enostavno aplikacijo kot je Beležka v operacijskem sistemu Windows ali urejevalnik Vi v Unix, vendar nam razvojna orodja nudijo nekatere prednosti:

- avtomatiziranje ponavljajočih se opravil,
- prevajanje kode,
- faktorizacijo kode,
- dokončanje kode ob pisanju (angl. completion),
- formatiranje kode,
- preverjanje kode med pisanjem,
- razhroščevanje kode (angl. debugging) ter
- prikaz pomoči in definicije metod in razredov.

Eclipse [36] je odprtokodno razvojno orodje. Na voljo so naslednje različice oziroma programski paketi:

- Eclipse IDE for Java Developers – razvoj aplikacij v programskem jeziku Java,



Slika 2.6: Grafični vmesnik razvojnega okolja Eclipse.

- Eclipse IDE for Java EE Developers – razvoj aplokacij v programskem jeziku Java za srednja in velika podjetja ter

- Eclipse IDE for C/C++ Developers – razvoj aplikacija v programskih jezikih C in C++.

Obstaja še veliko drugih programskih paketov, ki pa nas pri našem projektu niso zanimali. Eclipse med drugim omogoča tudi dodajanje vtičnikov (angl. plugin), da lahko razširimo funkcionalnost orodja.

Na desni strani slike 2.6 lahko vidimo razrede, ki so uporabljeni v aktivnem projektu, ter metode, ki pripadajo razredu.

Razrede in metode lahko razvrstimo po abecedi ali po vrstnem redu, kot se nahajajo v izvorni kodi. Na levi strani so prikazani obstoječi projekti. Del, kjer pišemo kodo, vidimo na sredini okna. Prav tako je vidno podokno za dokončevanje kode. Na spodnji strani urejevalnika se izpisujejo stanje izvajanja, razhroščevanja, napake.

2.7.1 Android SDK

Android SDK [11] je zbirka programskih knjižnic in orodij, ki jih potrebujemo za programiranje ali napredno upravljanje našega Android telefona. Predstavlja osnovo za programiranje, razhroščevanje in testiranje aplikacij Android, prinaša pa tudi nekaj orodij, ki omogočajo napredno upravljanje z našim Android telefonom. Pri rešitvi našega problema smo uporabili paket ADT Bundle for Windows [10], ki vključuje:

- razvojno orodje Eclipse in vtičnik ADT,
- orodja Android SDK,
- najnovejšo platformo Android ter
- sliko najnovejše platforme Android za emulator.

Paket smo uporabili zato, ker ga je mogoče enostavno namestiti in ker vsebuje vsa orodja, ki smo jih potrebovali za rešitev problema.

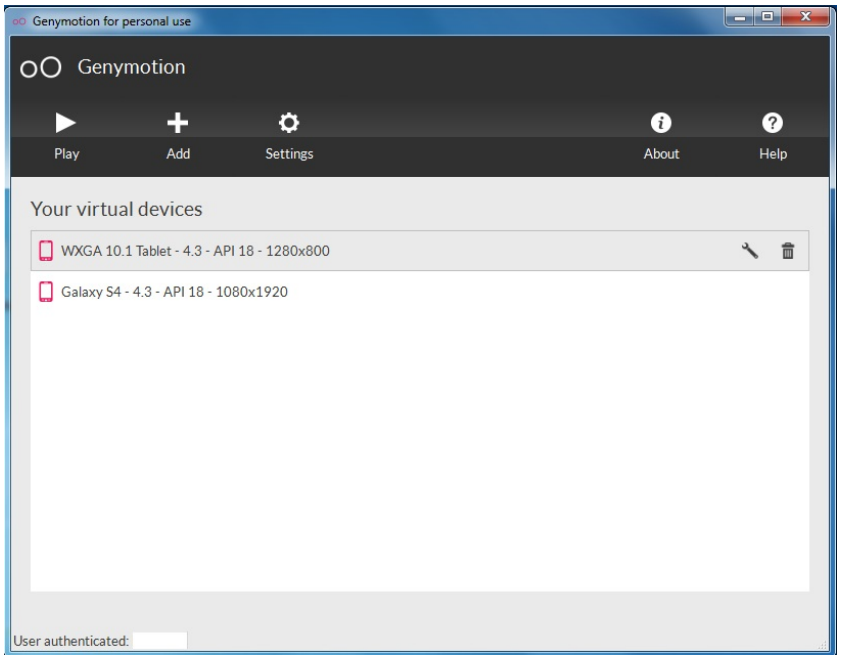
2.7.2 Emulator Genymotion

Za testiranje aplikacije smo prvotno uporabljali vtičnik za razvojno orodje Eclipse z imenom Genymotion [17], kasneje pa tudi Android telefon Samsung Galaxy Core.

Za vtičnik Genymotion namesto vgrajenega emulatorja smo se odločili zaradi boljše odzivnosti, hitrejšega delovanja in lažje konfiguracije za uporabo zemljevida. Prednosti vtičnika Genymotion so, da je:

- zelo hiter,
- enostaven za uporabo in konfiguracijo ter
- nudi napredne razhroščevalne možnosti (angl. dabug features).

Kot vidimo na sliki 2.7, lahko za emulator uporabimo vrsto pametnih telefonov z OS Android v našem primeru Samsung Galaxy S4, in tablični računalnik Nexus, na katerih smo tudi testirali aplikacijo v začetni fazi.



Slika 2.7: Emulator Genymotion, izbira virtualne naprave.

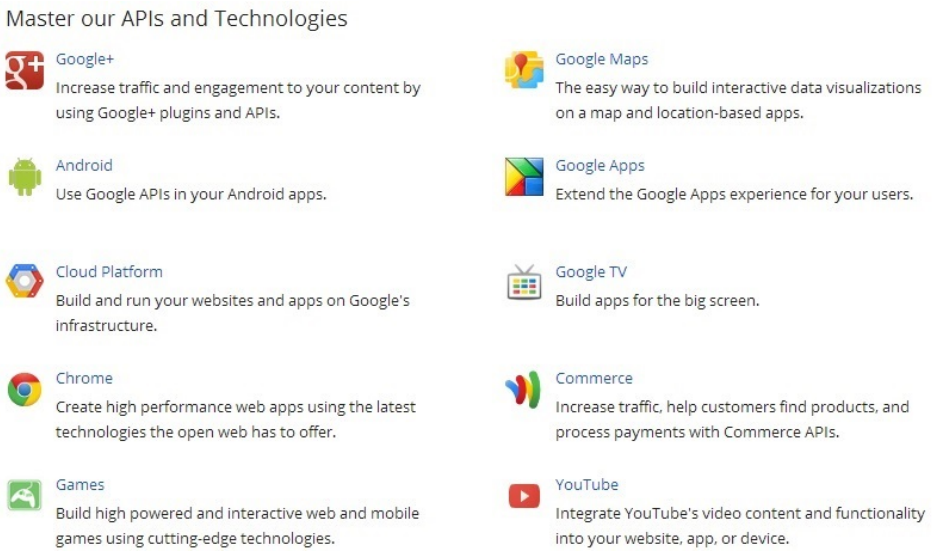
2.8 Google Maps API

Podjetje Google ponuja veliko spletnih storitev končnim uporabnikom. Njegove najbolj znane storitve so Youtube, Gmail, Maps itd.

Razvijalci, ki želijo povezati svoje storitve s storitvami podjetja Google, imajo, kot vidimo na sliki 2.8, na voljo veliko možnosti.

Z Google Maps API [20] lahko dodajamo zemljevide, ki temeljijo na podatkih Google Maps v aplikacijo. API se samodejno poveže s strežniki, prenese podatke in zemljevide, ki so potrebni za prikaz. Omogoča nam tudi dodajanje:

- ikon pripetih določenemu položaju na zemljevidu (angl. Markers),
- zaporednih daljic, ki označujejo pot (angl. Polylines),
- zaprtih zaporednih daljic, s katerimi lahko označimo predel ali del (angl. Polygon) in



Slika 2.8: Prikaz Googlovih tehnologij in API-jev.

- slik za določen položaj na zemljevidu (angl. Ground overlay).

Pri razvoju naše aplikacije smo uporabljali Google Maps API verzije 2, ki so ga medtem že nadgradili v verzijo 3. Google ponuja storitev na platformah Android [27], iOS [28] ter Javascript [29].

2.9 Geografski koordinatni sistem

Geografski koordinatni sistem [22] je dvorazsežni sferni koordinatni sistem, ki je poravnan z vrtilno osjo Zemlje.

Zemlja določa dva kota, merjena od središča Zemlje, njegovega koordinatnega izhodišča. Prvi kot, imenovan zemljepisna širina (angl. latitude), podaja kot med poljubno točko in ekvatorjem. Drugi del, imenovan zemljepisna dolžina (angl. longitude), pa podaja kot vzdolž ekvatorja od poljubne točke na zemlji. V večjem delu sveta so za ničto zemljepisno dolžino sprejeli Greenwich v Angliji.

S pomočjo teh dveh kotov lahko določimo lego poljubnega kraja na Zemlji. Baltimore v Marylandu, ZDA, ima na primer zemljepisno širino 39,3 stopinje

severno in zemljepisno dolžino 76,6 stopinje zahodno. Če potegnemo vektor iz središča Zemlje v točko 39,3 stopinje severno od ekvatorja in 76,6 stopinje zahodno od Greenwicha, bo potekal skozi Baltimore.

Ekvator je očitno pomemben del tega koordinatnega sistema in predstavlja ničto točko za širinski kot in polovično točko med tečajema (poloma). Ekvator je osnovna ravnina za geografski koordinatni sistem. Takšna osnovna ravnina je določena v vseh sfernih koordinatnih sistemih.

Poleg zemljepisne širine in dolžine spada h geografskemu koordinatnemu sistemu tudi nadmorska višina (angl. altitude), ki skupaj s predhodnima koordinatama enolično določa lego poljubnega kraja. Nadmorska višina se običajno meri od morske gladine, in ne od središča Zemlje.

2.9.1 Računanje razdalje med točkama

Razdalja je dolžina poti med dvema točkama. Je numerični opis, kako daleč v prostoru so telesa [35].

Za računanje razdalj med točkama, ki so določene z zemljepisno dolžino in zemljepisno širino na zemljevidu, imamo na voljo več izračunov oziroma formul (http://www.movable-type.co.uk/scripts/latlong.html). Pri naši rešitvi smo uporabili Haversinovo formulo [23], ki jo prikazuje enačba 2.1.

$$\begin{aligned} a &= \sin^2(\Delta\phi/2) + \cos(\phi_1) * \cos(\phi_2) * \sin^2(\Delta\lambda/2) \\ c &= 2 * a \tan 2(\sqrt{a}, \sqrt{(1-a)}) \\ d &= R * c \end{aligned} \tag{2.1}$$

R = radij zemlje (6,371km), ϕ = zemljepisna širina, λ = zemljepisna dolžina, koti morajo biti podani v radianih.

Poglavje 3

Zasnova in izvedba rešitve

V tem poglavju bomo opisali kako smo zasnovali rešitev, od ideje do delujoče aplikacije. Najpomembnejša se nam je zdela ideja, katera je bila na začetki enostavna, kasneje pa smo jo razširili.

3.1 Ideja

Dandanes imamo že (skoraj) vsi pametne telefone, ki nam lahko v mnogo pogledih olajšajo življenje, telefoni imajo vgrajenih veliko vmesnikov in tehnologij, npr. GPS, Bluetooth, wifi itd. Glavna ideja je bila narediti aplikacijo za pametni telefon z operacijskim sistemom Android, ker je ta najbolj razširjen. Aplikacija naj bi omogočala dodajanje, urejanje in brisanje oznak na zemljevidu, prikaz trenutne lokacije in prikaz najbližjih oznak in oznak, ki smo ji dodali oziroma uvozili. Pri pregledu podobnih aplikacij smo ugotovili, da je večina rešitev narejena s pomočjo Google Maps API, za katerega smo se odločili tudi sami, da ga uporabimo. V mislih smo imeli aplikacijo, ki prikaže lokacije najbližjih defibrilatorjev glede na našo trenutno lokacijo in lokacije defibrilatorjev nasploh, prikaže pa tudi podatke, kje se naprava nahaja (naslov, zgradba). Kasneje smo idejo razširili na nekakšno univerzalno aplikacijo za prikaz lokacij z različnimi podatki na zemljevidu, ki jih lahko urejamo, brišemo in dodajamo.

3.2 Zajem zahtev in osnovni pogoji

Aplikacija naj bi bila uporabna za iskanje najbližje lokacije glede na naš trenutni položaj ali vseh lokacij, ki smo jo predtem vnesli ali uvozili v aplikacijo. Aplikacija naj bi bila enostavna za uporabo in čim bolj hitra. Glavne funkcionalnosti aplikacije naj bi bile:

- prikaz trenutnega položaja na zemljevidu,
- vnos, urejanje in brisanje podatkov o lokacijah,
- uvoz KML formata lokacij iz pomnilnika mobilne naprave in
- prikaz vseh in najbližjih lokacij in drugih podatkov na zemljevidu.

Za uporabo aplikacije za prikaz lokacij na zemljevidu s pomočjo storitve Google Maps API so osnovni pogoji naslednji:

- pametni telefon ali mobilna naprava z operacijskim sistemom Android,
- dovolj zmogljiv pametni telefon ali druga mobilna naprava,
- internetna povezava in sprejemnika GPS ter
- osnovno poznavanje geografskih pojmov in osnovne uporabe naprave.

3.3 Razvoj v skladu z metodologijo XP

Pri izbiri primerne metodologije za rešitev našega problema se je kot najbolj primerna metodologija izkazala metodologija ekstremnega programiranja. Upoštevali smo, da je pomembno izdelati delujočo rešitev, ne pa popolne dokumentacije. Za to metodologijo smo se odločili tudi zato, ker smo imeli za diplomsko nalogo na voljo omejen čas in omejena sredstva. Druge metodologije zahtevajo več osebja in dolgoročnejše planiranje izvedbe celotnega projekta.

Pri izdelavi diplomske naloge smo pazili, da nismo prekoračili 40 urnega tedenskega delovnika, držali smo se standardov kodiranja in testirali delovanje po vsaki končani iteraciji. Testiranje je bilo izvedeno s funkcionalnimi testi.

Upoštevajoči metodologijo XP, smo najprej razvili funkcionalnosti, ki so najbolj potrebne. V našem primeru smo razvili naslednje izdaje:

- prvo izdajo aplikacije, ki je omogočala dodajanje oznak in podatkov (angl. Marker) na zemljevid,
- drugo izdajo aplikacije, ki je omogočala hrambo in urejanje podatkov v podatkovi bazi, in
- tretjo izdajo aplikacije, ki je omogočala uvoz podatkov iz pomnilnika naprave.

3.4 Namestitev in uporaba Google Maps Android API

Preden začnemo uporabljati API, ga je treba prenesti in pridobiti ključ Google Maps API. Oba sta na voljo brezplačno. Preden ga začnemo uporabljati, je treba narediti naslednje korake:

- namestitev zbirke orodij Android SDK,
- prenos in konfiguracija Google Play services, ki vključuje Google Maps Android API,
- pridobitev ključa API; da pridobimo ključ API, se moramo registrirati v Google APIs Console, v katerega vnesemo certifikat za aplikacijo,
- zahtevane nastavitve dodamo v ***AndroidManifest.txt*** datoteko [1]

[1] Vsaka Android aplikacija mora imeti datoteko ***AndroidManifest.xml*** [12]. V datoteki so zapisani bistveni podatki aplikacije (dovoljenja, aktivnosti ...), ki so potrebni za zagon in izvajanje.

```
<permission
    android:name="com.example.maps.permission.MAPS_RECEIVE"
    android:protectionLevel="signature" />

<uses-permission android:name="com.example.maps.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<!--
     The following two permissions are not required to use
     Google Maps Android API v2, but are recommended.
-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

     <meta-data
        android:name="com.google.android.maps.v2.API KEY"
        android:value="AIzaSyAOPOUIjAwN\                    />


    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />

    <uses-library android:name="com.google.android.maps" />
```

Slika 3.1: ***AndroidManifest.txt*** datoteka aplikacije Maps.

naše aplikacije ter

- dodamo zemljevid v aplikacijo.

Kot vidimo na sliki 3.1, moramo v ***AndroidManifest.txt*** datoteko aplikacije vključiti nekatera dovoljenja.

Najbolj pomembna so dovoljenja za dostop do interneta, prenosa podatkov, sprejem signala GPS in dovoljenje za branje zunanjega spomina za potrebe uvoza datotek. S tem dejanjem dovolimo aplikaciji uporabljati prej omenjena sredstva.

V ***AndroidManifest.txt*** moramo dodati tudi vrstico s ključem, je videti približno takole:
"AIzaSyBdVl-cTICSwYKrZ95SuvNw7dbMuDt1KG0".

Ključ je potreben zato, da lahko dostopamo do strežnikov Google Maps z API. Pomembno je tudi, da pred izdelavo certifikata navedemo pravo ime projekta, ki ga bomo uporabljali, saj v nasprotnem primeru ključ ne bo deloval pravilno in posledično tudi aplikacija ne.

3.5 Podatki

Za uporabo aplikacije potrebujemo naslednje podatke:

- zemljepisno širino,
- zemljepisno dolžino,
- naslov oznake in
- besedilo oznake.

Podatki v aplikaciji so prikazani na sliki 3.2. Zemljepisne koordinate so uporabljene, da lokacijo prikažemo na zemljevidu. Naslov oznake je prikazan zgoraj odebeljeno, besedilo pa spodaj v navadni pisavi.

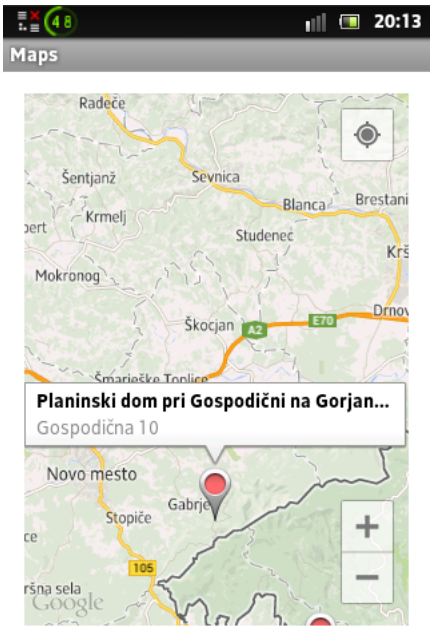
Za lažjo uporabo in obdelavo podatkov smo napisali javanski razred ***Lokacija*** z naslednjimi atributi:

- ***naziv***, ki je naslov oznake,
- ***lokacija***, ki je besedilo oznake,
- ***tLat***, ki je zemljepisna širina,
- ***tLang***, ki je zemljepisna dolžina in
- ***razdalja***, ki je razdalja od naše trenutne lokacije do izbrane lokacije, izračunana po formuli 2.1.

Podatki polj ***naziv*** in ***lokacija*** so lahko poljubni, pomembno pa je, da so podatki ***tLat*** in ***tLang*** res podatki, ki pomenijo zemljepisno dolžino in širino.

V aplikaciji smo uporabili podatkovno strukturo ***TreeSet***, ki nam omogoča urejeno dodajanje podatkov tipa ***Lokacija*** v strukturo. V ta namen smo tudi redefinirali metodo ***compareTo*** [2]. Vedno, kadar se podatki preberejo

[2] Metoda compareTo v programskem jeziku Java vrača negativno vrednost, če je objekt manjši, število 0, če je enak, in pozitivno število, če je objekt večji od objekta, s katerim ga primerjamo.



Slika 3.2: Prikaz podatkov na zemljevidu.

iz podatkovne baze in uporabijo za prikaz na zemljevidu, te podatke tipa Lokacija urejeno vstavimo v podatkovno strukturo ***TreeSet***. To nam omogoča lažje in hitrejše iskanje in izpis najbližjih točk, če to zahtevamo od aplikacije.

3.6 Podatkovna baza

Da bi rešili naš problem, smo se odločili, da se morajo podatki hraniti lokalno v podatkovni bazi. Vrsto podatkovne baze smo izbrali glede na enostavnost uporabe ter omejenost pomnilnika in hitrost zaradi okolja uporabe. Podatkovna baza SQLite se je v našem primeru izkazala za dobro rešitev.

V podatkovni bazi oziroma tabeli so naslednji podatki:

- ***_id*** – primarni ključ v tabeli [3],

[3] Primarni ključ tabele pri relacijskih podatkovnih bazah je stolpec ali kombinacija stolpcev, ki enolično določa vrstico. To pomeni, da se za katerikoli dve vrstici v tabeli osnovna vrednost primarnih ključev razlikuje. Primer primarnega ključa je EMŠO [30].

```
public void createEntry(String tNaziv, String tLokacija, Double tLat,
        Double tLang) {
    // TODO Auto-generated method stub
    ContentValues cv = new ContentValues();
    cv.put(KEY_TITLE, tNaziv);
    cv.put(KEY_SNIPPET, tLokacija);
    cv.put(KEY_LOCLAT, tLat);
    cv.put(KEY_LOCLANG, tLang);
    ourDatabase.insert(DATABASE_TABLE, null, cv);
}
```

Slika 3.3: Funkcija, ki doda lokacijo in njene podatke v podatkovno bazo.

- ***title*** – naslov oznake,
- ***snippet*** – besedilo oznake,
- ***loc_lat*** – zemljepisna širina in
- ***loc_lang*** – zemljepisna dolžina.

Podatke zapisujemo v podatkovno bazo SQLite s funkcijo, ki jo prikazuje slika 3.3. Pri tem moramo paziti, da so podatki pravilnega podatkovnega tipa:

- ***tNaziv*** – podatkovnega tipa [4] niz znakov (angl. String),
- ***tLokacija*** – podatkovnega tipa niz znakov (angl. String),
- ***tLat*** – podatkovnega tipa realno število (angl. double) ter
- ***tLang*** – podatkovnega tipa realno število (angl. double).

Podatkovna baza SQLite se je sprva izvajala na računalniku, na kateremu smo poganjali in testirali aplikacijo. V poznejših izdajah smo zaradi uporabe

[4] Uporabni programi obdelujejo spremenljive podatke. Shranjujemo jih v spremenljivkah. Spremenljivke (angl. variables) so v dobi predmetno usmerjenih programskih jezikih dokaj raztegljiv pojem, saj predstavljajo tudi predmet, ki vsebuje posebne metode in lastnosti, te pa so lahko nov predmet in tako naprej. Vsebino spremenljivke določa njen tip [2].

```
<Placemark>
    <name>PGD Dragatuš</name>
            <description>Dragatuš 19</description>
    <LookAt>
        <longitude>15.40534876070076</longitude>
        <latitude>45.66314130389565</latitude>
        <altitude>0</altitude>
        <heading>-5.536449604109281</heading>
        <tilt>0</tilt>
        <range>24798.7342917895</range>
        <gx:altitudeMode>relativeToSeaFloor</gx:altitudeMode>
    </LookAt>
    <styleUrl>#m_ylw-pushpin</styleUrl>
    <Point>
        <gx:drawOrder>1</gx:drawOrder>
        <coordinates>15.36824587261599,45.67162270586882,0</coordinates>
    </Point>
</Placemark>
```

Slika 3.4: Struktura KML, oznaka ***Placemark***, ki vsebuje za nas pomembne podatke.

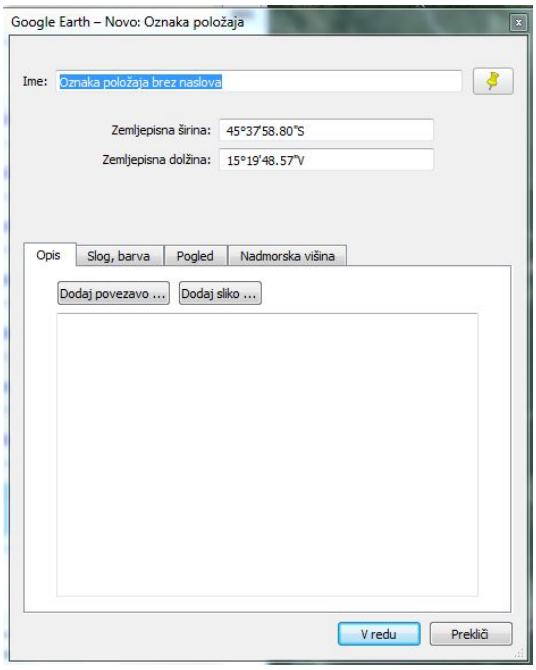
signala GPS aplikacijo testirali na dveh mobilnih telefonih, prvi je bil Samsung Galaxy Core z OS Android različica 4.1.2 (Jelly Bean), drugi pa Sony Xperia Go z OS Android različica 2.3 (Gingerbread).

Za operacije in komunikacijo s podatkovno bazo (urejanje, dodajanje, brisanje lokacij) smo napisali dva javanska razreda, ***Database*** in ***DbHelper***, ki sta nam delo ustrezno olajšala.

3.7 Uvoz formata KML

Za uvoz podatkov v formatu KML, ki je podrobneje opisan v poglavju 2.6.2, smo uporabili razčlenjevalnik XML. Podatki so strukturno prikazani na sliki 3.4. Za nas pomembni podatki so ugnezdeni znotraj značke ***Placemark***, ki predstavlja eno lokacijo z ostalimi podatki. Pomen značk je naslednji:

- ***name*** – naziv,
- ***description*** – lokacija,



Slika 3.5: Google Earth okno za vnos podatkov.

- ***longitude*** – zemljepisna dolžina in
- ***latitude*** – zemljepisna širina.

Vse značke vsebujejo angleške izraze, zato smo jih lahko v trenutku razumeli in razbrali pomen podatkov dokumenta.

Pri testiranju aplikacije smo izdelali dokument ***KML*** z aplikacijo Google Earth, v katerem smo različne kraje na zemljevidu označili in vpisali potrebne podatke, v našem primeru naziv in lokacijo.

Program pri shranjevanju oznak izdela stisnjeno datoteko formata KMZ. Ko to datoteko razširimo s programom za razširjanje datotek (uporabili smo WinRar [5]), dobimo ven datoteko s končnico KML, ki smo jo kasneje uvozili v našo aplikacijo.

[5] WinRar [39] je brezplačen program za stiskanje in razširjanje datotek, ki podpira številne formate.

Na sliki 3.5 je prikazano okno za vnos podatkov o lokaciji. Google Earth nam ponuja še veliko drugih možnosti, npr. dodajanje slik položaju in podobne stvari, ki pa nas pri rešitvi našega problema niso zanimale.

Poglavje 4

Prikaz in analiza

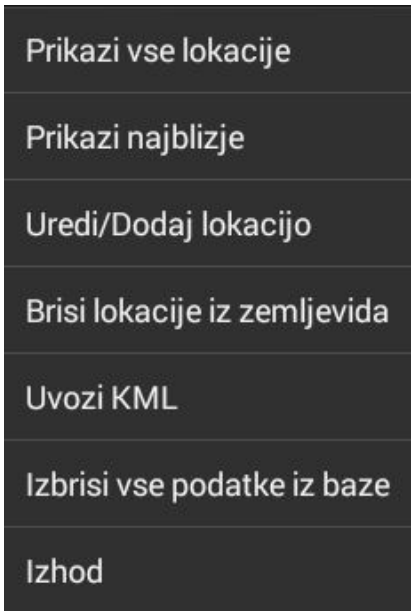
V tem poglavju bomo predstavili uporabniški vmesnik aplikacije, težave, ki smo jih imeli pri izvedbi in analizo aplikacije v praksi.

4.1 Prikaz aplikacije

Pri izdelavi grafičnega vmesnika aplikacije smo se trudili, da bi bil ta čim bolj enostaven, prijazen za uporabo in samoumeven. Kot smo omenili že v poglavju 3.2, mora imeti uporabnik aplikacije osnovno znanje zemljepisa in osnovno znanje uporabe mobilne naprave.

Ko zaženemo aplikacijo, se nam prikaže zemljevid sveta. Na voljo imamo gumb, s katerim približamo trenutno lokacijo, na kateri se nahajamo. Aplikacija tudi osvežuje lokacijo, npr. če se premikamo, pokaže smer, kam gremo. S pritiskom na gumb mobilne naprave, ki predstavlja meni, se nam odpre meni, ki ga prikazuje slika 4.1. Funkcije smo razdelili na logične sklope, ki omogočajo urejanje, prikaz in brisanje lokacij ter izhod iz aplikacije.
Na voljo imamo naslednje funkcionalnosti:

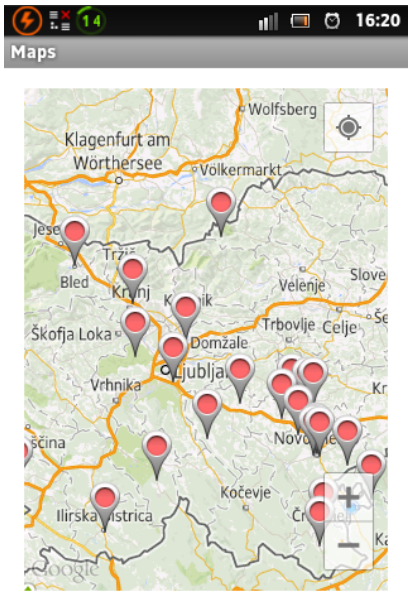
- ***Prikazi vse lokacije*** – omogoča prikaz vseh lokacij na zemljevidu (slika 4.2),
- ***Prikazi najblizje*** – omogoča prikaz treh najbližjih lokacij glede na našo trenutno lokacijo,



Slika 4.1: Prikaz menija, če pritisnemo na gumb z menijem.

- ***Uredi/Dodaj lokacijo*** – omogoča vnos nove lokacije ali ureditev že obstoječe,
- ***Brisi lokacije z zemljevida*** – omogoča brisanje prikazanih lokacij z zemljevida,
- ***Uvozi KML*** – omogoča uvoz datoteke formata KML,
- ***Izbrisi vse podatke iz baze*** – omogoča izbris vseh podatkov, ki so v podatkovni bazi, in
- ***Izhod*** – omogoča izhod iz aplikacije.

Za meni smo se odločili zato, da ne bi bilo nepotrebnih gumbov na zemljevidu, skratka, da je aplikacija čim bolj pregledna in enostavna.



Slika 4.2: Aplikacija Maps, prikaz lokacij na zemljevidu.

4.2 Odziv uporabnikov

Aplikacijo smo dali v uporabo dvema osebama. Vnaprej smo jima nekoliko razložili njen pomen in jima na kratko dali navodila za uporabo. Ker sta imela oba pametni telefon z operacijskim sistemom Android, s platformo ni bilo težav. Testirala sta jo na telefonih Samsung Galaxy S2 z operacijskim sistemom Android 4.3 (Jelly Bean) in Samsung Galaxy S4 Mini z operacijskim sistemom Android verzije 4.2.2 (Jelly Bean).

Oba sta zatrdila, da je aplikacija uporabna in enostavna za uporabo, kar je bil tudi namen rešitve našega problema. Oba sta preizkusila tako vnos kot tudi urejanje in seveda prikaz lokacij.

Uporabnik s telefonom Samsung Galaxy S2 je imel težavo pri iskanju mesta trenutne lokacije. Ob zagonu aplikacije se mu ni prikazal gumb za navigacijo do prikaza trenutne lokacije. Težavo je odpravil z izhodom iz aplikacije in ponovnim zagonom aplikacije. Uporabnik je tudi predlagal razširitev aplikacije z izvozom podatkov v obliki KML, da bi jih lahko poslali npr. prek

tehnologije Bluetooth prijatelju za uporabo na njegovi mobilni napravi v naši aplikaciji. Ta razširitev se nam je zdela zelo smiselna, in ko bomo aplikacijo razvijali naprej, bo na prvem mestu podprtje te funkcionalnosti. Predlagal je tudi, da bi aplikacijo naredili bolj robustno, odporno na napake v vpisu lokacij, npr. da bi preverili, da smo res vpisali koordinate, in podobno.

Uporabnik mobilne naprave znamke Samsung Galaxy S4 Mini je imel pripombo glede osveževanja trenutnega položaja.

Ker je aplikacijo testiral med hojo, se položaj ni osveževal dovolj pogosto.

To smo odpravili tako, da smo v kodi popravili čas, ki je potreben, da se osveži položaj. To se nastavi s pomočjo razreda LocationManager, ki je podprt v API. Objektu lahko z metodo ***public void requestLocationUpdates (String provider, long minTime, float minDistance, PendingIntent intent)*** nastavimo parametra čas (v milisekundah) in razdalja (v metrih), ki je potrebna za osveževanje. Tako se lahko položaj osvežuje glede na čas ali razdaljo.

Tako smo zaključili, da je aplikacija uporabna in dovolj enostavna za uporabo.

Uporabnika sta nam predlagala še nekaj izboljšav, ki smo jih prav tako upoštevali in določene tudi implementirali v kodo.

Veliko idej za možne izboljšave pa smo dobili tudi med pisanjem kode.

4.3 Analiza

4.3.1 Podobne aplikacije

Na spletu oziroma na spletni strani Google Play smo našli veliko aplikacij, ki omogočajo urejanje in prikaz geografskih podatkov na zemljevidu.
Spodaj so naštete nekatere od njih:

- Map Coordinates [16],
- Coordinates [13],

- GPS Coordinates to Map [14],
- GPS Map [15] ter
- Google Earth [18].

Najbolj izstopa rešitev podjetja Google z imenom Google Earth. Aplikacijo smo tudi preizkusili na lokalnem računalniku, in ne na mobilni napravi. Rešitev ima veliko več funkcionalnosti, kot smo jih podprli v naši rešitvi. Vendar smo mi implementirali funkcionalnosti na svoj način, ki je preprost za uporabo.

Pri ostalih aplikacijah nismo opazili uvoza podatkov iz drugih virov. Omogočale pa so več nastavitev map, npr. različne možnosti pogleda, česar naša aplikacija ne vsebuje.

Mislimo, da smo z aplikacijo naredili nekaj novega, omogočili pregleden in enostaven način prikaza geografskih lokacij. Njena uporabnost je široka, saj jo lahko uporabimo za vrsto namenov. Eden od njih je tudi prvotni, za prikaz lokacij defibrilatorjev po Sloveniji.

4.3.2 Težave pri izvedbi

Posebnih težav pri razvoju nismo imeli, saj smo prej preučili nekaj literature o aplikacijah Android, pri tem pa smo večino literature oziroma vso našli na internetu. Nekaj aplikacij Android pa smo izdelali že med študijem v okviru različnih predmetov.

Uporabili smo tudi znanje programskega jezika Java, ki smo ga pridobili pri dosedanjem študiju na fakulteti.

Od težav je najbolj izstopala izvedba razčlenjevalnika, za namen razčlenjevanja dokumenta KML. Zaradi nepoznavanja delovanja in vrste razčlenjevalnikov smo sorazmeroma veliko časa porabili za izbiro in izdelavo razčlenjevalnika. Razčlenjevalnik smo sprva napisali za programski jezik Java in ga testirali na osebnem računalniku. Izbrali smo razčlenjevalnik DOM, ki se je kasneje izkazal za napačno izbiro, saj porabi preveč pomnilnika. Ob testiranju razčlenjevalnika na mobilni napravi se je aplikacija prenehala odzivati.

Pozneje smo preučili vrste razčlenjevalnikov bolj podrobno in se odločili za razčlenjevalnik SAX, ki je bolj podrobno predstavljen v poglavju 2.5.1. Napisali smo razred ***MySaxHandler***, ki nam je delo z razčlenjevalnikom ustrezno olajšal. Razlika je bila očitna, aplikacija je bila odzivna in razčlenjevanje je delovalo.

4.3.3 Analiza uporabe aplikacije v praksi

Aplikacijo smo uporabili za predstavitev podatkov o lokacijah defibrilatorjev po Sloveniji. Vnesli smo 35 lokacij defibrilatorjev, ki se nahajajo na območju Slovenije. Za takšno bazo podatkov bi bilo morda bolje, če bi jo lahko prenesli s strežnika in da bi bili podatki redno posodobljeni. Kot sklepamo se vsaj vsak mesec na novi lokaciji pojavi nov defibrilator. Funkcionalnosti ne bi bilo težavno dodati, vendar bi morali dobiti posodobljene podatke v primerni obliki. Najbolj bi nam ustrezalo v formatu KML.

Uporabili smo jo tudi na potepanju po Portorožu, kjer smo si v aplikacijo uvozili podatke z znamenitostmi, ki smo si jih želeli ogledati v kraju in okolici. Ugotovili smo, da je zelo uporabna funkcionalnost ***Prikazi najblizje***, s katero lahko z enim ukazom najdemo lokacije v bližini in si tako prihranimo marsikatero sitnost oziroma vračanje nazaj, ker smo zgrešili cilj.

Ob uporabi aplikacije smo uporabljali mobilni prenos podatkov, ki nam je omogočil, da so se zemljevidi naložili v aplikacijo in sprejemnik GPS, zaradi prikaza in osveževanja trenutne lokacije. Moramo poudariti, da je aplikacija zelo potratna z energijo, kar se je poznalo na stanju baterije, sklepamo da zaradi prenosa podatkov in sprejemnika GPS.

Z uporabo aplikacije smo dobili veliko idej za razširitev funkcionalnosti in za izboljšave.

Nekatere smo tudi implementirali v rešitev. Menimo, da aplikacija služi svojemu namenu in da smo dosegli pričakovanja tako glede uporabnosti kot glede enostavnosti.

Poglavje 5

Zaključek

V zadnjih nekaj letih so mobilne naprave postajale vse manjše, bolj estetsko oblikovane in hkrati zmogljivejše, bolj uporabne. To, kar naredi neko napravo uporabno, je to, kar lahko z njo počnemo. Danes so možnosti skorajda neomejene, saj ima skoraj vsak, res skoraj vsak človek pametni telefon, ki ima vgrajenih veliko senzorjev in različnih tehnologij (GPS, WI-Fi, Bluetooth ...). Tako lahko telefon ali drugo mobilno napravo uporabimo še za vse drugo kot le za telefoniranje. Razvoj aplikacij za mobilne naprave je v velikem porastu. Vse večje storitve ali del teh so že ponujeni kot mobilna aplikacija. Lahko pogledamo samo trgovine, npr. Hofer, Lidl, Spar, Petrol ..., vsaka ima svojo aplikacijo – za ugodnosti, spremljanje raznih prodajnih akcij in podobno.

Velika prednost mobilnih naprav je, da jih imamo pri sebi ves čas, zato jih veliko tudi uporabljamo. Uporaba se je razširila praktično na vsa področja, od navadnega telefoniranja pa do elektronskega poslovanja z banko, uporabe interneta, spremljanja televizijskih programov, osebnega trenerstva ... Velikokrat moramo samo poiskati kakšno informacijo (npr. delovni čas trgovine) na internetu, kar nam lahko olajša mobilna naprava, s pomočjo katere v trenutku to izvemo, če imamo na voljo internetno povezavo.

Največji delež na trgu operacijskih sistemov za pametne telefone in mobilne naprave ima operacijski sistem Android podjetja Google, zato so aplikacije zanj zelo iskane. Aplikacije so večinoma napisane v programskem jeziku

Java, kar še olajša prehod od kodiranja programske opreme za računalnike na kodiranje programske opreme za mobilne naprave.

Aplikacije, namenjene platformi Android, so objavljene v spletni aplikaciji Google Play, kjer lahko najdemo različne vrste aplikacij, od iger pa do različnih programov. Imamo na voljo brezplačne in plačljive aplikacije. Praksa je največkrat taka, da v brezplačni različici dobimo okrnjeno aplikacijo, kar pomeni, da ne omogoča vseh funkcionalnosti, kot jih omogoča plačljiva različica. Dobra stran tega je, da lahko brezplačno preverimo, ali nam izbrani produkt ustreza, in se potem laže odločimo za njegov nakup. Druga možnost pa je, da lahko uporabljamo vse funkcionalnosti samo določen čas, ponavadi 30 dni.

V prvem delu naloge smo na kratko povzeli tehnologije in orodja, ki smo jih uporabili pri razvoju aplikacije. Sledita del z zasnovo in prikazom rešitve, v katerih smo prikazali pot razvoja in težave, na katere smo naleteli med razvojem.

Končni rezultat tega diplomskega dela je delujoča mobilna aplikacija za hranjenje in prikaz lokacijskih podatkov. Aplikacijo smo razvili v programskem jeziku Java, namenjena pa je platformi Android. Na trgu je veliko podobnih aplikacij, ki pa ne omogočajo vseh funkcionalnosti, ki smo jih podprli. Trudili smo se, da bi bil uporabniški vmesnik enostaven in prijazen za uporabo in da bi podprli funkcionalnosti, ki jih potrebujemo pri delu z lokacijskimi podatki. Aplikacijo smo intenzivno uporabljali in testirali približno mesec in pri tem smo odkrili nekaj težav, ki smo jih tudi uspešno odpravili.

Možnih izboljšav naše aplikacije je nešteto mnogo. Veliko smo jih opazili že med razvojem, nekaj predlogov pa smo dobili tudi od uporabnikov, ki smo jim aplikacijo namestili na mobilnik in ki so jo tudi testirali. Če bi aplikacijo razvijali naprej, bi podprli naslednje možnosti:

- izvoz datoteke KML, s katero bi omogočili deljenje lokacij med prijatelji ali ustvarjanje varnostne kopije,
- spreminjanje možnosti zemljevida (promet, satelit, relief), ki je podprto

tudi v Google Maps API, katerega smo uporabili pri razvoju, in

- dodajanje točk na zemljevid s klikom na točko.

in še kaj bi lahko našli. To so le nekatere funkcionalnosti, na katere smo pomislili, da bi bilo dobro, če bi jih vsebovala tudi naša aplikacija.

Literatura

[1] Joyce Farrell, JAVA Programming, Fifth Edition, 2009.

[2] U. Mesojedec, B. Fabjan, Java 2: Temelji programiranja, Ljubljana 2004, str. 64 - 77.

[3] Igor Rožanc, Prosojnice pradavanj pri predmetu Razvoj programskih sistemov 2: Agilne metodologije razvoja programske opreme. Fakulteta za računalništvo in informatiko, Ljubljana, 2008.

[4] Damjan Vavpotič. Prosojnice predavanj pri predmetu Razvoj informacijskih sistemov: Kaj je metodologija razvoja IS?. Fakulteta za računalništvo in informatiko, Ljubljana, 2010.

[5] (2014) Agilne metodologije. Dostopno na: http://sl.wikipedia.org/wiki/Agilne_metode_razvoja_programske_opreme

[6] (2014) M. Bajec, M. Krisper, Agilne metodologije razvoja informacijskih sistemov. Dostopno na: http://bajecm.fri.uni-lj.si/CRP2001/clanki/agilne%20metodologije%20razvoja%20is.pdf

[7] (2014) Delež pametnih telefonov z Android OS. Dostopno na: http://www.idc.com/getdoc.jsp?containerId=prUS24257413

[8] (2014) Delitev metodologij. Dostopno na: http://www2.gov.si/mju/emris.nsf/0/A67CD3BA7292D3B2C1256E9E003E4933?-OpenDocument

[9] (2014) XP metodologija. Dostopno na:
http://www.extremeprogramming.org/

[10] (2014) ADT Bundle for Windows. Dostopno na:
http://developer.android.com/sdk/index.html

[11] (2014) Android SDK. Dostopno na:
http://www.zmaga.com/content.php?id=4716

[12] (2014) AndroidManifest datoteka. Dostopno na
http://developer.android.com/guide/topics/manifest/manifest-intro.html

[13] (2014) Aplikacija Coordinates. Dostopno na:
https://play.google.com/store/apps/details?id=com.orbitalmotion.coordenadas

[14] (2014) Aplikacija GPS Coordinates to Map. Dostopno na:
https://play.google.com/store/apps/details?id=reversegeocode.paq

[15] (2014) Aplikacija GPS Map. Dostopno na:
https://play.google.com/store/apps/details?id=com.appspot.wrightrocket.GPSMap

[16] (2014) Aplikacija Map Coordinates. Dostopno na:
https://play.google.com/store/apps/details?id=sands.mapCoordinates.android

[17] (2014) Emulator Genymotion: Dostopno na:
http://genymotion.en.softonic.com/

[18] (2014) Google Earth. Dostopno na:
http://www.google.com/earth/

[19] (2014) Google Maps. Dostopno na:
https://www.google.com/maps/preview

[20] (2014) Google Maps API. Dostopno na:
https://developers.google.com/maps/documentation/android/intro

[21] (2014) Google Play aplikacija. Dostopno na:
http://en.wikipedia.org/wiki/Google_Play

[22] (2014) Geografski koordinatni sistem. Dostopno na:
http://sl.wikipedia.org/wiki/Geografski_koordinatni_sistem

[23] (2014) Haversinova formula. Dostopno na:
http://www.movable-type.co.uk/scripts/latlong.html

[24] (2014) Kaj je Linux. Dostopno na:
http://sl.wikipedia.org/wiki/Linux

[25] (2014) KML format. Dostopno na:
https://developers.google.com/kml/documentation/kml

[26] (2014) Operacijski sistem Android. Dostopno na:
http://sl.wikipedia.org/wiki/Android_*operacijski_sistem*

[27] (2014) Platforma Android. Dostopno na:
http://android.fri.uni-lj.si/index.php/Platforma

[28] (2014) Platforma iOS. Dostopno na:
http://sl.wikipedia.org/wiki/IOS

[29] (2014) Platforma Javascript. Dostopno na:
http://sl.wikipedia.org/wiki/JavaScript

[30] (2014) Primarni ključ. Dostopno na:
http://wiki.fmf.uni-lj.si/wiki/Primarni_ključ

[31] (2014) Razčlenjevalniki, na splošno. Dostopno na:
http://xml.silmaril.ie/parsers.html

[32] (2014) Razčlenjevalnik DOM. Dostopno na:
http://en.wikipedia.org/wiki/Document_Object_Model

[33] (2014) Razčlenjevalnik SAX. Dostopno na:
http://en.wikipedia.org/wiki/Simple_API_for_XML

[34] (2014) Razčlenjevalnik StAX. Dostopno na:
http://stax.codehaus.org/

[35] (2014) Razdalja med dvema točkama. Dostopno na:
http://sl.wikipedia.org/wiki/Razdalja

[36] (2014) Razvojno orodje Eclipse. Dostopno na:
http://en.wikipedia.org/wiki/Eclipse_(software)

[37] (2014) SQL. Dostopno na:
http://sl.wikipedia.org/wiki/SQL

[38] (2014) SQLite podatkovna baza. Dostopno na:
http://www.sqlite.org/about.html

[39] (2014) WinRar. Dostopno na:
http://en.wikipedia.org/wiki/WinRAR

[40] (2014) XML. Dostopno na:
http://en.wikipedia.org/wiki/XML