UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Majzelj

Mobilna aplikacija za pregled informacij o prometu v Sloveniji

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Ljubljana, 2011

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: 00059/2011
Datum: 05.01.2011

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

Kandidat: **MIHA MAJZELJ**

Naslov: **MOBILNA APLIKACIJA ZA PREGLED INFORMACIJ O PROMETU V SLOVENIJI**

MOBILE APPLICATION FOR TRAFFIC INFORMATION IN SLOVENIA

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Na Android platformi razvijte mobilno aplikacijo, ki omogoča pregled informaciji o razmerah na slovenskih cestah. Kot vir uporabite vse razpoložljive vire na slovenskem internetu. Aplikacija naj podaja informacijo tudi preko prikaza ikon na zemljevidu Slovenije in slike iz spletnih kamer.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

za

prof. dr. Nikolaj Zimic

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

diplomskega dela

Spodaj podpisani/-a Miha Majzelj,

z vpisno številko 63030421,

sem avtor/-ica diplomskega dela z naslovom:

Mobilna aplikacija za pregled informacij o prometu v Sloveniji

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) doc. dr. Rok Rupnik
- 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 5.7.2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju doc. dr. Roku Rupniku za nasvete in strokovno pomoč pri izdelavi diplomskega dela.

Zahvaljujem se staršem in sorodnikom za omogočen študij in podporo.

Zahvaljujem se tudi vsem ostalim sošolcem, prijateljem in sodelavcem, ki so me spodbujali ali kakorkoli pomagali pri izdelavi te naloge. Še posebej pa Nuši in Gašperju.

Kazalo

Seznam uporabljenih kratic in simbolov

GPS – Global Positioning System; globalni sistem za pozicioniranje

SDK – Software Developmet Kit; programsko razvojno ogrodje

UMTS - Universal Mobile Telecommunications System; univerzalni mobilni telekomunikacijski sistem

API – Application Programming Interface; programski vmesnik

APP – Application; aplikacija

AVD – Android Virtual Device; navidezna Android naprava oziroma emulator

JIT – Just In Time; tip prevajalnika za programsko kodo

ADT – Android Development Tools; skupek orodij, knjižnic in primerov za razvoj na Android platformi

SMS – Short Message Service; kratko tekstovno sporočilo

WLAN – Wireless Local Area Network; brezžično lokalno omrežje

WiFi – ime za brezžično lokalno računalniško mrežo (WLAN)

JSON – Javascript Object Notation; Javascript objektni zapis

RAD – Rapid Application Development; metodologija razvoja aplikacij

PIC – prometno informacijski center

SVN – Subversion; sistem za kontrolo nad verzijami

USB – Universal Serial Bus; univerzalno serijsko vodilo

Povzetek

V diplomskem delu je predstavljen razvoj aplikacije SloPromet za spremljanje trenutnih prometnih in hidrometeoroloških podatkov, narejen za mobilno platformo Android. V delu je poleg uvodnega poglavja še kratka predstavitev ključnih komponent razvijalskega ogrodja Android SDK v navezavi z orodjem za razvoj, Eclipse. Razhroščevalnik, emulator in urejevalnik kode so glavne usmeritve. V nadaljevanju so podrobno opisani primeri uporabe aplikacije, arhitektura, podatkovni model, uporabljeni viri podatkov, programski razredi, uporabniški vmesnik. Končni izdelek je delujoča aplikacija za pametne telefone, ki omogoča pregled trenutnih zastojev na slovenskih cestah v obliki seznama ali na zemljevidu. Podatke lahko uporabnik poljubno ureja po prioriteti, datumu in oddaljenosti od lokacije, kjer se trenutno nahaja. Aplikacija omogoča tudi pregled slik iz spletnih kamer, pregled vodostajev rek in vremenskih podatkov za posamezne kraje. Vse to so pomembni podatki, ko smo na poti. V zaključnem poglavju pa so opisane možnosti za izboljšave, dodelave in nadaljni razvoj aplikacije.

Ključne besede:

Android, Eclipse, prometne informacije, vreme, SloPromet

Abstract

The thesis describes development of mobile application for current traffic report and weather conditions, SloPromet. Beside the introduction there is a chapter about key components of Android software delopment kit and rapid developmet tool Eclipse. Debuging, emulator and code editor are the main focuses. Further on there are more detailed description of use case scenarios, arhitecture, data model and date sources used by the aplication, code classes and user interface. End product of this thesis is a working, fully functional mobile application for smartphones using Android platform. It enables overview of current traffic jam and other traffic information in Slovenia using simple list or map. Data can be sorted by priority, date or distance from current location by the user. Application also enables view od images from webcams, waterleverls of rivers and weather information. Since all of this is important data when traveling on the road. In the final chapeter there are some possibilities for further development and improvements.

Key words:

Android, Eclipse, traffic report, weather, SloPromet

1 Uvod

V zadnjih nekaj letih smo priča intenzivnemu razvoju na področju mobilnih naprav, občutljivih na dotik. To nam potrjujejo podatki o prodanih napravah, podatki o številu aplikacij na tržnicah [10] ali na primer podatki o obisku spletnih strani [9]. Vsi omenjeni podatki kažejo hiter razvoj tako imenovanih pametnih telefonov in naprav občutljivih na dotik, kamor prištevamo tudi tablične računalnike. Še posebej velik uspeh pa, kot je razvidno iz slike 1.1, beleži platforma Android.

Top Smartphone Platforms 3 Month Avg. Ending Feb. 2011 vs. 3 Month Avg. Ending Nov. 2010 Total U.S. Smartphone Subscribers Ages 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Nov-10	Feb-11	Point Change
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
Google	26.0%	33.0%	7.0
RIM	33.5%	28.9%	-4.6
Apple	25.0%	25.2%	0.2
Microsoft	9.0%	7.7%	-1.3
Palm	3.9%	2.8%	-1.1

Slika 1.1: Primerjava trga pametnih telefonov za tromesečje v letih 2010 in 2011 v Združenih državah Amerike.

Vzrokov za to je več. Na eni strani gre vsekakor za povečano pretočnost mobilnih omrežij, ki jih ponujajo operaterji, na drugi strani pa za tehnološki razvoj na področju mobilnih naprav, predvsem z zaslonom na dotik. Pojem mobilni telefon tako že nekaj časa opredeljuje ne le napravo, s katero pokličemo prijatelja, ampak napravo, ki omogoča dostop do interneta, predvajanje filmov, uporabo raznovrstnih aplikacij, ki so lahko poslovne, zabavne ali informativne. Lahko rečemo, da se je s pojavom teh tehnologij informacija približala posamezniku na doseg roke ali žepa. Mobilni telefon počasi, a vztrajano pridobiva vse funkcionalnosti prenosnega osebnega računalnika.

Skupaj z razvojem stojne opreme se revolucija dogaja tudi na programskem področju. Dosedanji mobilni operacijski sistemi niso več dorasli zmogljivostim novih procesorjev,

količini pomnilnika, ki je na voljo in množici podatkov za najrazličnejšo obdelavo ali prikaz. Na trgu se je tako pojavilo kar nekaj možnosti (Bada, iOS, Android, Windows Phone 7), katere proizvajalci oziroma založniki (Samsung, Apple, Open Handset Alliance, Microsoft) na različne načine trudijo uspešno razviti in povečati svoj tržni delež. Med seboj se razlikujejo predvsem po tržnih prijemih, strategijah, podpori različnim strojnim konfiguracijam, možnostih za razvoj in odprtosti platforme za zunanje razvijalce. Skupno vsem navedenim operacijskim sistemom je njihova odprtost za razvoj s strani samostojnih razvijalcev in podjetij. To je privedlo do živahnega dogajanja v razvijalskih skupnostih na področju razvoja mobilnih aplikacij, tako imenovanih App-ov. Aplikacije so postale nekakšno središče dogajanja za posamezno platformo. Objava, bodisi brezplačna ali plačljiva, je odvisna od platforme in praviloma poteka na uradni tržnici oziroma Marketu, kjer je na voljo vsem uporabnikom določene platforme. Aplikacije imajo tako zagotovljen velik doseg pri potencialnih uporabnikih, razvijalci pa zagotovljen trg. Po podatkih za maj 2011 je na Android Market-u objavljenih 200.000 aplikacij, ob tem, da je bilo prijavljenih 100 miljonov naprav, na katere je bilo nameščenih 4.500 bilijonov aplikacij [10].

V tem diplomskem delu je predstavljen razvoj aplikacije SloPromet za operacijski sistem Android. Kot pove že ime, je aplikacija namenjena predvsem spremljanju informacij o stanju na slovenskih cestah, omogoča pa tudi druge funkcionalnosti. Po uvodnem poglavju sledi poglavje, v katerem je bolj natančen opis aplikacije. Ker je platforma Android razmeroma mlada tehnologija, sledi kratek teoretični uvod v samo platformo, arihtekturo in koncepte ter predstavitev orodij za delo. V četrtem poglavju je opisan razvojni cikel aplikacije, ki sestoji iz načrtovanja, arhitekture, same obdelave podatkov in njihove vizualne predstavitve na različne načine. Na koncu je povzetek, kjer so predstavljene možnosti za dodelavo ali drugačno implementacijo nekaterih problemov.

2 SloPromet

2.1 Ideja

Podatki o stanju na cestah so zelo pomembni za vse udeležence v prometu, in sicer pred odhodom ali ko smo že na poti. Ponavadi jih slišimo preko radija ali preverimo na spletni strani. Vendar v obeh primerih uporabniška izkušnja ni najboljša. Podatki so lahko zastareli, netočni ali pa preprosto ne vemo na kateri odsek ceste se informacija nanaša. Mobilna aplikacija za spremljanje prometnih podatkov se zdi najboljša izbira, ki je zmožna elegantnega prikaza dogodkov na cesti na zahtevo uporabnika in to na njegovem telefonu. Glavne prednosti so predvsem razpoložljivost, dobra, učinkovita prezentacija podatkov in dodatne informacije, ki so na voljo. Želja je bila izdelati aplikacijo, ki bo imela za uporabnika veliko uporabno vrednost. Bo hitra, enostavna in dostopna tudi takrat, ko nimamo možnosti komunikacije z omrežjem.

2.2 Funkcionalnosti

Aplikacija mora omogočati pregled trenutnega stanju na cestah. Med različne dogodke se šteje zastoj, delo na cesti, zaprta cesta, prometna nesreča, sneg, poledica, veter in drugi izredni dogodki. Spremljanje stanja je omogočeno v obliki seznama, ki se lahko poljubno ureja po tipu dogodka, oddaljenosti od trenutne lokacije in času veljavnosti ali času vnosa. Druga možnost pa je pregled na zemljevidu, ki nam še bolj nazorno prikaže mesto dogodka in oddaljenost od naše lokacije.

Z aplikacijo naj bo možen tudi vpogled v stanja na cestah s pomočjo DARS-ovih spletnih kamer, ki so nameščene predvsem na avtocestnem križu. Uporabnik prejme slikovno datoteko s podrobno časovno oznako. Na zahtevo pa se le-ta osveži. Ta funkcionalnost ima za omejitev pogoj, da ima uporabnik med uporabo omogočen dostop do interneta.

Še ena od prednosti mobilnih naprav je zavedanje o trenutni lokaciji bodisi s pomočjo lokacije baznih postaj, brezžičnih omrežij ali v napravo vgrajenega GPS modula. Dodatna funkcionalnost je urejanje dogodkov po oddaljenosti od lokacije, kjer se uporabnik trenutno nahaja in posodabljanje teh podatkov glede na gibanje uporabnika.

Sama naprava nam nudi tudi možnost, da znotraj aplikacije enostavno in hitro pošljemo opis dogodka prek SMS-a in tako obvestimo druge udeležence v prometu.

Na razmere na cesti v veliki meri vpliva tudi vreme, zato je zahteva za SloPromet tudi pregled trenutnih vremenskih podatkov, kar se zdi priročno in učinkovito. Dodatna možnost pa so podatki o vodostaju, pretoku in temperaturi rek.

2.3 Viri

Ažurni in natančni podatki so ključni za aplikacijo, ki obvešča o stanju na cestah. Pri uporabi virov za izdelavo prometne aplikacije se zanašamo na javno pridobljene podatke, ki so nam na voljo in pa druge storitve, ki jih lahko uporabimo.

Podatki

Za stanje na slovenskih cestah skrbi Prometno-informacijsi center za državne ceste (PIC), ki podatke objavlja na svoji spletni strani. Podatki so javno dostopni in tako na voljo za uporabo. Vremenski podatki in podatki iz avtomatskih merilnih postaj, ki zajemajo vodostaj, temperaturo vode in pretok, so zbrani s strani Agencije Republike Slovenije za vreme (ARSO) in so prav tako javno objavljeni na njihovi spletni strani. Informativne slike s spletnih kamer zagotavlja Direkcija za avtoceste Republike Slovenije (DARS).

Spletne storitve (*angl.* Online Services)

Aplikacija uporablja tudi nekatere spletne storitve. Zemljevid je del storitve Maps podjetja Google, ki omogoča priročne načine za interakcijo, postavljanje točk oziroma markerjev na njem, računanje razdalje med točkami na zemljevidu. Omogoča tudi geolokacijske storitve, na primer pretvorbo z naslova v GPS koordinate lokacije in obratno. Knjižnice, ki omogočajo uporabo teh storitev, niso sestavni del Android SDK-ja, temveč so del dodatka vmesnika Google API.

Drugi viri

Poleg podatkovnih virov so pomemben del celotne podobe aplikacije tudi grafični gradniki aplikacije, ker pa za samostojno izdelavo teh ni ostalo veliko časa, sem v projektu uporabil slikovno gradivo z odprtokodno licenco, ki ob uporabi zahteva navedbo avtorja.

3 Orodja in tehnologije

3.1 Platforma Android

3.1.1 Splošno o Androidu

Android je odprtokodni operacijski sistem za mobilne telefone, ki temelji na Linux jedru. Vključuje sam operacijski sitem, vmesni sloj in aplikacije za končne uporabnike. Leta 2005 ga je podjetje Google odkupilo od prvotnih snovalcev. Kasneje pa je prišel pod domeno združenja Open Handset Allinace, ki sestoji iz 80 podjetij, ki se ukvarjajo s programsko ali strojno opremo, kot tudi mobilnih operaterjev, ki so se zavezali k razvoju odprtih standardov na področju mobilne telefonije [1]. V zadnjih dveh letih smo priča veliki rasti deleža te platforme med vsemi prodanimi telefoni v svetovnem obsegu. To dokazujejo tudi podatki vodilnega podjetja za raziskave in svetovanje na področju informacijske tehnologije Gartner.

Company	1Q11 Units	1Q11 Market Share (%)	1Q10 Units	1Q10 Market Share (%)
Android	36,267.8	36.0	5,226.6	9.6
Symbian	27,598.5	27.4	24,067.7	44.2
iOS	16,883.2	16.8	8,359.7	15.3
Research In Motion	13,004.0	12.9	10,752.5	19.7
Microsoft	3,658.7	3.6	3,696.2	6.8
Other OS	3,357.2	3.3	2,402.9	4.4
Total	**100,769.3**	**100.0**	**54,505.5**	**100.0**

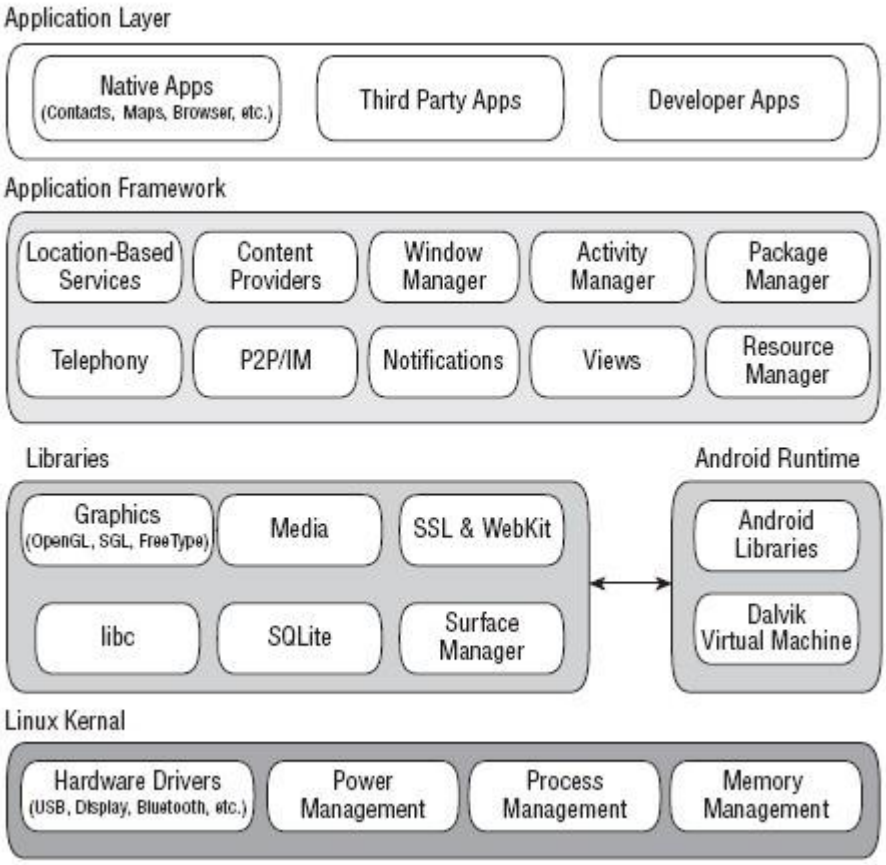
Slika 3.1: Delež prodanih pametnih telefonov po operacijskih sistemih v prvi četrtini leta 2011 v primerjavi s prvo četrtino 2010 (številke so v tisočih enot)

Glavni gradniki pratforme so objektno-orientirano Javansko okolje, ki sestoji iz osnovnih Javanskih knjižnic, ki pa se izvajajo znotraj Dalvik virtualnega stroja, ki deluje kot prevajalnik JIT. Ostale komponente so še OpenCore medijska platforma, SQLite podatkovna baza, Open GL in SGL grafični gradni, WebKit standardi za brskalnik. Osnova sistema je jedro Linux, katerega osnova je koda napisana v programskem jeziku C, C++, Java in XML-u skupaj približno 12 milijonov vrstic programske kode. [1]

Velika prednost pred ostalimi mobilnimi operacijskimi sistemi je dejstvo, da proizvajalci telefonov, ki želijo na svoje naprave namestiti Android, ne plačujejo licenčnin za njegovo uporabo. To je zagotovo tudi eden od razlogov za tak uspeh.

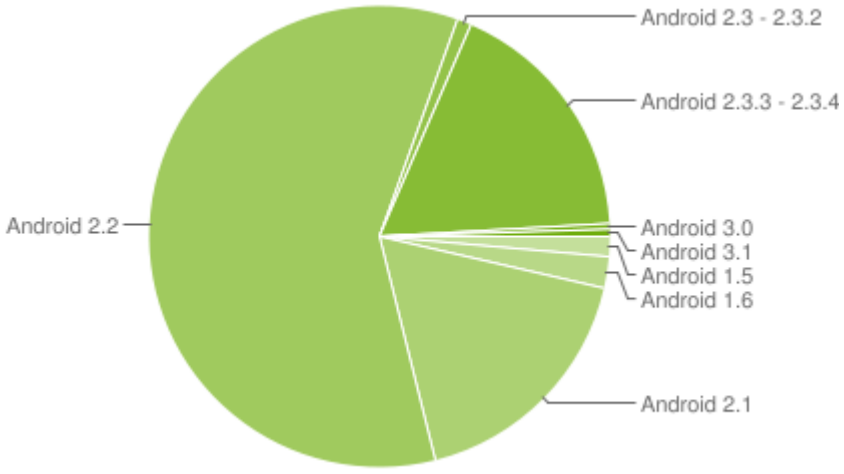
3.1.2 Arhitektura

Celotno programsko arhitekturo operacijskega sistema Android najbolje ponazori spodnja slika (Slika 3.2). Na osnovnem nivoju je jedro Linux različice 2.6.29, ki je odgovorno za gonilnike naprave, dostop do virov, upravljanje s porabo energije, upravljanje s pomnilnikom in ostale funkcije sistema. Vsebuje gonilniško podporo za naprave kot so zaslon, kamera, tipkovnica, WiFi, zvočna kartica in pomnilnik Flash. Na vrhu osnovnega nivoja so programske knižnice C in C++ kot so OpenCORE (medijska knjižnica), OpenGL (grafična knjižnica), WebKit (brskalnik), FreeType (tipografija), SSL (varni komunikacijski kanal), libc (sistemske knjižnice jezka C), SQLite (relacijska podatkovna baza). Večina teh knjižnic dostopa do virov jedra prek navideznega stroja Dalvik, ki deluje kot prehod (*angl*. gateway). Ta je narejen tako, da se za vsako zahtevo iz drugih nivojev ustvari nov virtualni stroj. V naslednjem nivoju sledijo aplikacijska ogrodja, ki pokrivajo področja klicanja (*angl.* Telephony), lokacijske storitve (*angl.* Location Based Services), upravljalnik oken (*angl.* Window Manager), upravljalnik aktivnosti (*angl.* Activity Manager), upravljanik paketov (*angl.* PacketManager), ki je odgovoren za namestitve in varnost. Najvišji nivo predstavlja aplikacijska raven, kamor se uvrščajo že vključene aplikacije (brskalnik, telefon, seznam kontaktov in druge) in aplikacije, ki jih uporabnik poljubno namesti.



Slika 3.2: Arhitektura operacijskega sistema Android

3.1.3 Android SDK



Slika 3.3: Delež posameznih verzij operacijskega sistema Android na napravah. Podatki so pridobljeni na podlagi obiska Android Market-a v 14 dneh s koncem 15.6.2011 [13].

Ogrodje Android SKD vsebuje različna orodja za razvoj aplikacij. Glavni deli so razhroščevalnik, programske knjižnice, emulator naprav, dokumentacija, primeri programske kode in vodiči za razvoj. Trenutno so podprti opercacijski sitemi Linux, Mac OS X 10.4.9 ali kasnejši in Windows XP ali kasnejši. [8]. Vsako novo verzijo operacijskega sistema tako spremlja tudi njen razvijalski del. Verziji platforme Android 1.1 je bila dodeljena številska vrednost 2, ki predstavlja identifikator, ki je shranjen znotraj sistema. Imenuje se API Level. To omogoča sistemu, da še pred namestitvijo pravilno določi ali je aplikacija kompatibilna s sistemom na katerem naj bi se izvajala [2]. Trenutno je zadnja različica API Level 12, in sicer za Android platformo 3.1. Slika 3.3 prikazuje deleže posameznih verzij operacijskega sistema Android nameščenega na napravah. Delež naprav pred verzijo Android 2.1 je samo 3,6%, največ pa je naprav z nameščeno verzijo Android 2.2 in sicer 59,4%. Pomembni so tudi najnovejši podatki o deležu glede na velikost in zgoščenost (*angl.* density) zaslona naprav. Vse to mora razvijalec upoštevati pri načrtovanju aplikacije. Razvojni cikel izdajanja novih verzij je bil do sedaj zelo hiter. Trenutno se letno izda dve novi verziji, po besedah vodilnih pa se bo ta cikel podaljšal na eno izdajo letno [3]. Vsaka nova različica prinese dodatne funkcionalnosti in podporo. Tako je na primer različica Android 3.0 namenjena predvsem za tablične naprave z večjimi zasloni, medtem ko je različica Android 3.1 prinesla podporo za naprave USB in omogoča priklop tretjih naprav. Za razvijalca je pomembno, da svoje aplikacije testira in po potrebi prilagodi novi različici, saj s tem zagotovi dobro uporabniško izkušnjo na najnovejših napravah, ki so na trgu. Vsako ogrodje ima kopico konceptov, ki jih moramo poznati, preden se lahko lotimo razvoja. V nadaljevanju so predstavljeni ključni programski razredi ogrodja, konfiguracijska datoteka Manifest in ostale komponente.

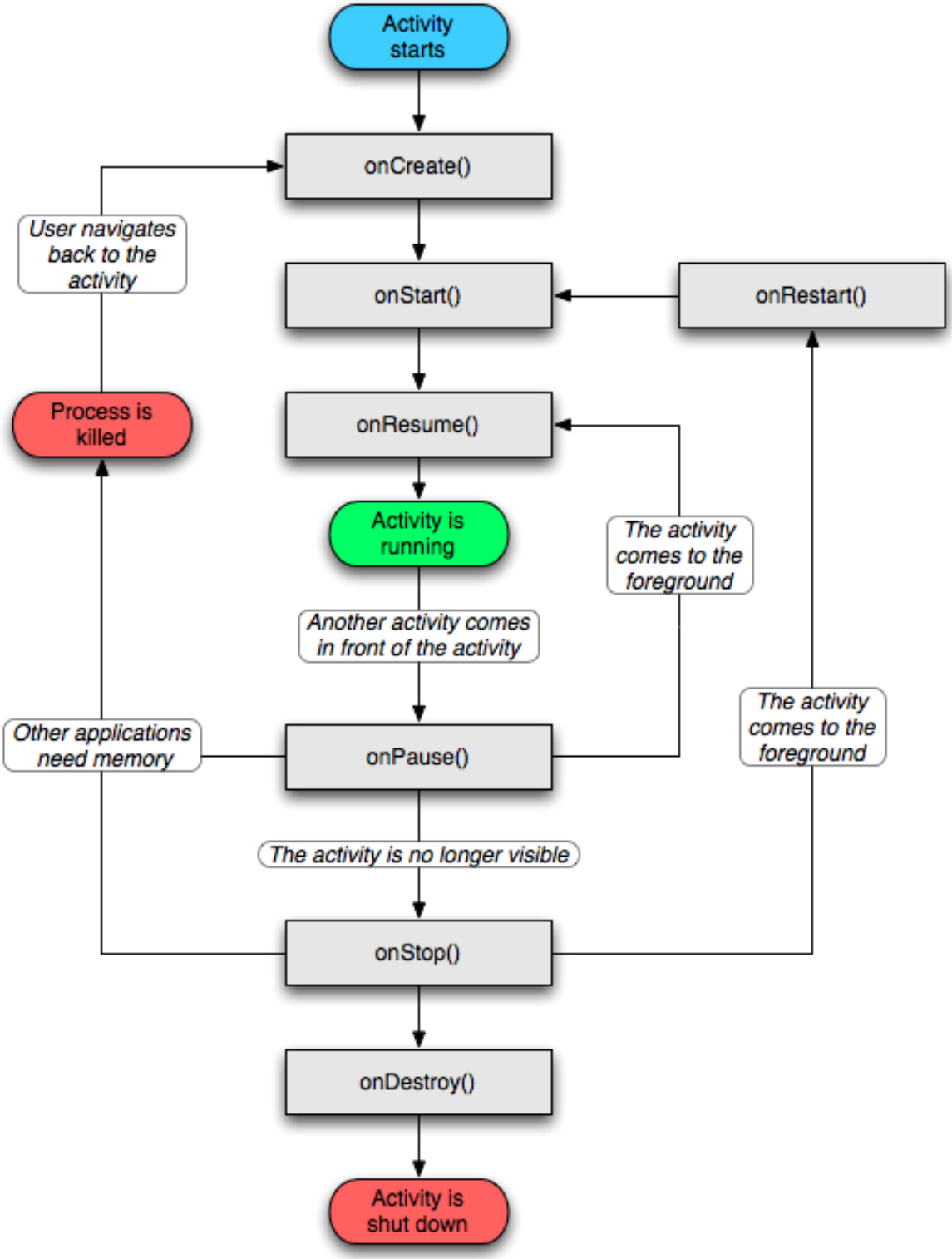
Pogled (*angl*. View)

Pogledi so elementi vmesnika, ki tvorijo osnovne gradnike uporabniškega vmesnika. Pogledi so hierarhično urejeni. Pomembno je, da se znajo sami prikazati oziroma izrisati na zaslonu. Primer pogleda je gumb (*angl.* Button) ali vnosno polje (*angl*. TextField). Poglede lahko primerjamo z elementi v Javanski knjižnici Swing [5].

Aktivnost (*angl.* Activity)

Aktivnost je koncept grafičnega vmesnika. Razred Aktivnost skrbi za novo okno ali zaslonsko masko na zaslonu, znotraj katere lahko postavljamo Poglede. Navadno so to celozaslonska

okna, vendar se lahko uporabijo tudi kot lebdeča okna ali so vsebovana znotraj druge Aktivnosti [6]. Pomemben za uporabo in razumevanje je življenski cikel aktivnosti.



Slika 3.4: Diagram življenskega cikla stanj razreda Aktivnost. Kvadrati na sliki predstavljajo metode, ki jih lahko implementiramo, ko Aktivnost spreminja stanje. Barvni ovali pa so končna stanja, v katerih je lahko Aktivnost [6].

Namera (*angl.* Intent)

Kot že ime pove, namere določajo namen, ki kliče k akciji. Namere se uporabljajo za naslednje opravila:

- oddajanje sporočila (*angl*. broadcast a message);
- zagon storitve;
- zagon aktivnosti;
- prikaz spletne strani ali zbirke kontaktov;
- telefonski klic ali sprejem le-tega;

Namere so lahko implicitne ali eksplicitne. Če želimo le prikazati nek spletni naslov, se sistem sam odloči, katera komponenta bo namero izpolnila. Lahko pa sami podamo informacijo o tem, kaj naj se na namero odzove. Kot primer je lahko uprabnikov klik na bližnjico v glavnem oknu aplikacije SloPromet. To ima za posledico kreiranje nove namere (*new Intent(Context c, Activity a)*), ki ima že podano Aktivnost, ki bo odgovorila.

Podatkovni vir (*angl*. Content Provider)

Deljenje podatkovnih virov med aplikacijami je pogost koncept. Android ponuja standardni mehanizem, ki opravlja prav to. Prek podatkovnih virov lahko podatke iz svoje aplikacije nudite drugim in prav tako sami uporabljate podatke drugih, ne da bi pri tem razkrili strukuturo, logiko ali implementacijo.

Storitev (*angl*. Service)

Storitve so neke vrste procesi v ozadju, ki jih poznamo iz operacijskih sistemov za namizne računalnike. Njihova glavna značilnost je, da lahko delujejo zelo dolgo časa. Poznamo dve vrsti storiteve.

1. lokalne storitve: so dostopne samo aplikaciji, katere del so;

2. oddaljene storitve: so dostopne vsem aplikacijam.

Android Manifest

Vsaka aplikacija mora vsebovati datoteko AndroidManifest.xml (natančno s tem imenom) v njenem korenskem direktoriju. Datoteka vsebuje osnovne informacije o aplikaciji, namenjene opracijskemu sistemu; brez teh sistem ne more pognati kode same aplikacije [7]. V manifestu, razen podprtih elementov, ne moremo dodajati lastnih. Strukura datoteke zglede tako:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <uses-permission_/>
    <permission_/>
    <permission-tree_/>
    <permission-group_/>
    <instrumentation_/>
    <uses-sdk_/>
    <uses-configuration_/>
    <uses-feature_/>
    <supports-screens_/>
    <compatible-screens_/>
    <supports-gl-texture_/>
    <application>
        <activity>
            <intent-filter>
                <action_/>
                <category_/>
                <data_/>
            </intent-filter>
            <meta-data_/>
        </activity>
        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data_/>
        </activity-alias>
        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>
        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data_/>
        </receiver>
        <provider>
            <grant-uri-permission_/>
            <meta-data_/>
        </provider>
        <uses-library_/>
    </application>
</manifest>
```

Njene poglavitne naloge pa so:

- poimenovanje Javanskega paketa aplikacije, ki služi kot enolični identifikator;
- opis komponent aplikacije (aktivnosti, storitev, podatkovnih virov,..);
- določitev procesov, ki bodo gostili komponente aplikacije;
- dovoljenja, ki jih aplikacija potrebuje za dostop do nekaterih delov vmesnika API in interakcije z drugimi aplikacijami;
- podatek o minimalnih zahtevah glede verzije vmesnika API za delovanje aplikacije;
- seznam knjižnic, ki jih aplikacija potrebuje za svoje delovanje.

3.2 Razvojno okolje

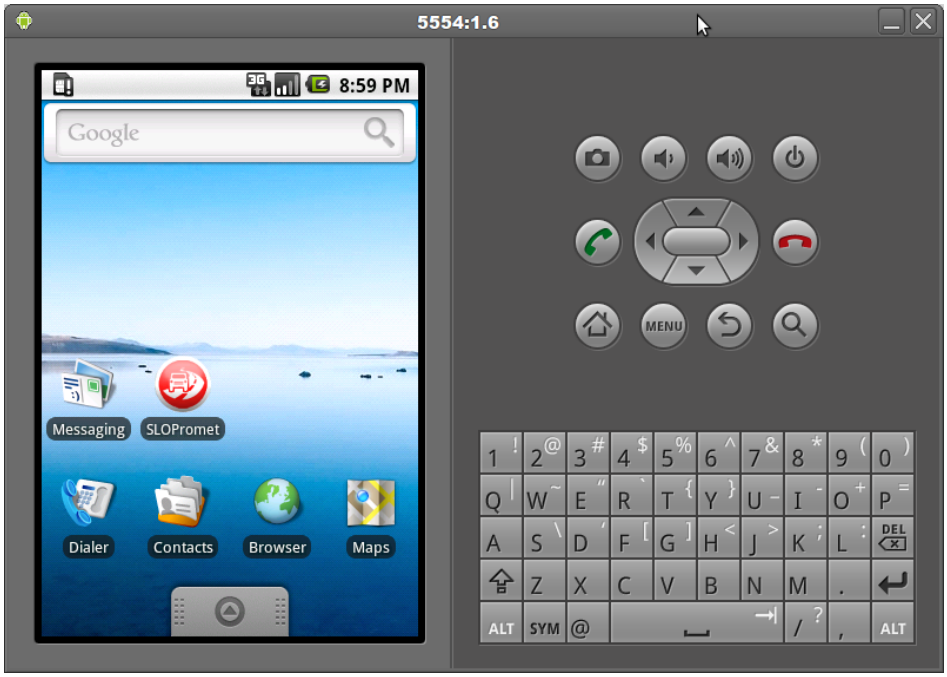
3.2.1 Eclipse

Eclipse je integrirano okolje za ravoj programske opreme (*angl*. integrated development environment). Razvito je v Javi in se uporablja za razvoj v Javi ali drugih programskih jezikih s pomočjo vtičnikov za le-te. Mednje spadajo C, Python, Ruby, PHP, Pearl in ostali. Platforma Android uradno podpira Eclipse (verzija 3.5 in 3.6) [8]. Z uporabo Android Development Tools vtičnika za Eclipse imamo dostop do vseh sestavnih delov Android SDK-ja. Znotraj okolja se tako poganjata razhroščevalnik in emulator. Služi pa seveda tudi kot urejevalnik za programsko kodo, XML datoteke, slike v projektu in vse potrebne knjižnice.

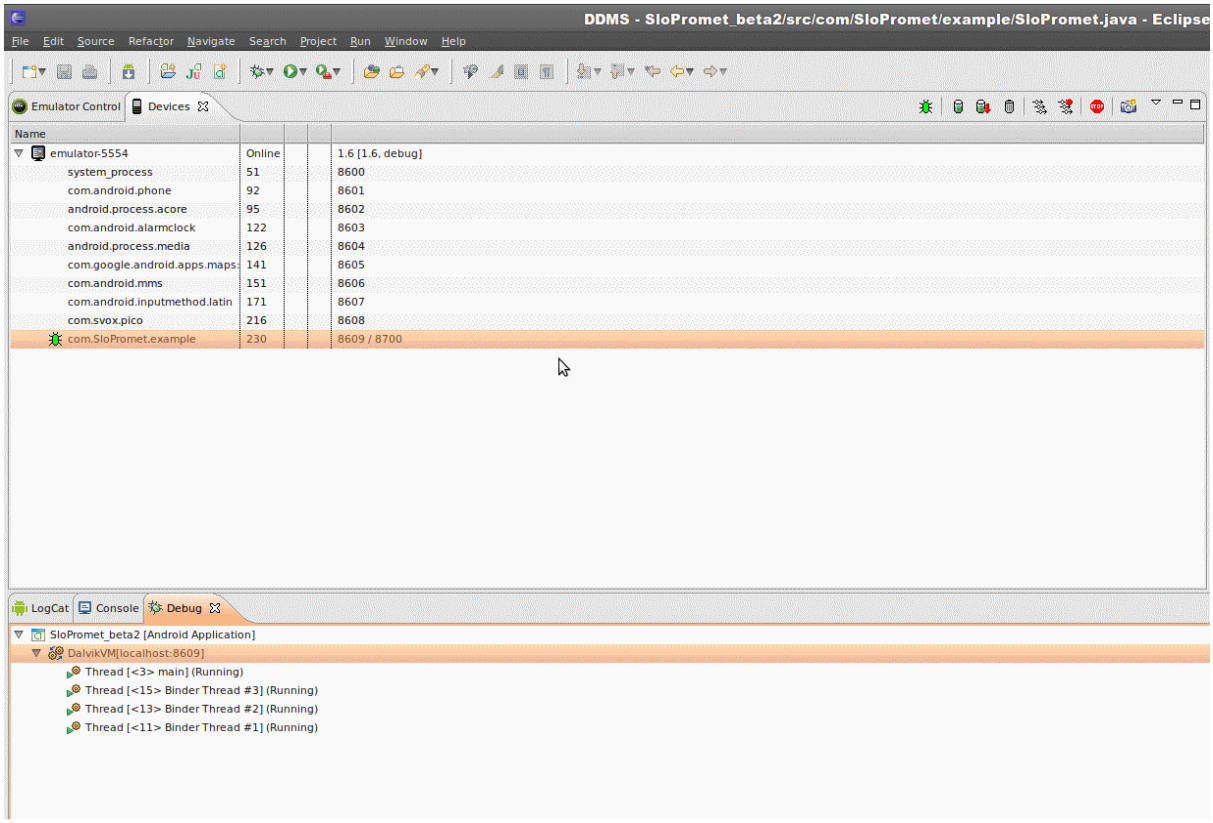
3.2.2 Emulator, razhroščevalnik

Virtualna naprava Android ali emulator je del ogrodja Android. Omogoča nam testiranje, izdelavo prototipov in razvoj aplikacij brez uporabe fizične naprave. Emulator simulira obnašanje, programske in strojne lastnosti naprave. Poljubno mu lahko nastavimo velikost zaslona, različico operacijskega sistema, ki jo bo uporabljal in druge nastvavitve, kot so omrežne nastavitve in nastavitve uporabniškega vmesnika. Primer emulatorja je prikazan na sliki 3.5.

Razhroščevalnik znotraj okolja Eclipse, ki je del ogrodja Android SDK se imenuje Dalvik Debug Monitor Server. Omogoča nam zajemanje zaslona na napravi, vpogled v sklad in niti programa, nudi podatke o logiranju, procesih, dohodnih klicih, lahko pa tudi opravi navidezni klic ali pošlje navidezni SMS (spoofing SMS and calls) in več [12].



Slika 3.5: Navidezna naprava AVD s testnimi nastavitvami Android 1.6



Slika 3.6: Razhroščevalnik znotraj orodja Eclipse. V zgornjem oknu vidimo procese, ki tečejo na navidezni napravi, med katerimi je tudi SloPromet. Spodnji zavihki nam omogočajo različne izpise vrednosti spremenljivk ali pregled nad kontrolnimi izpisi prek razreda Log.

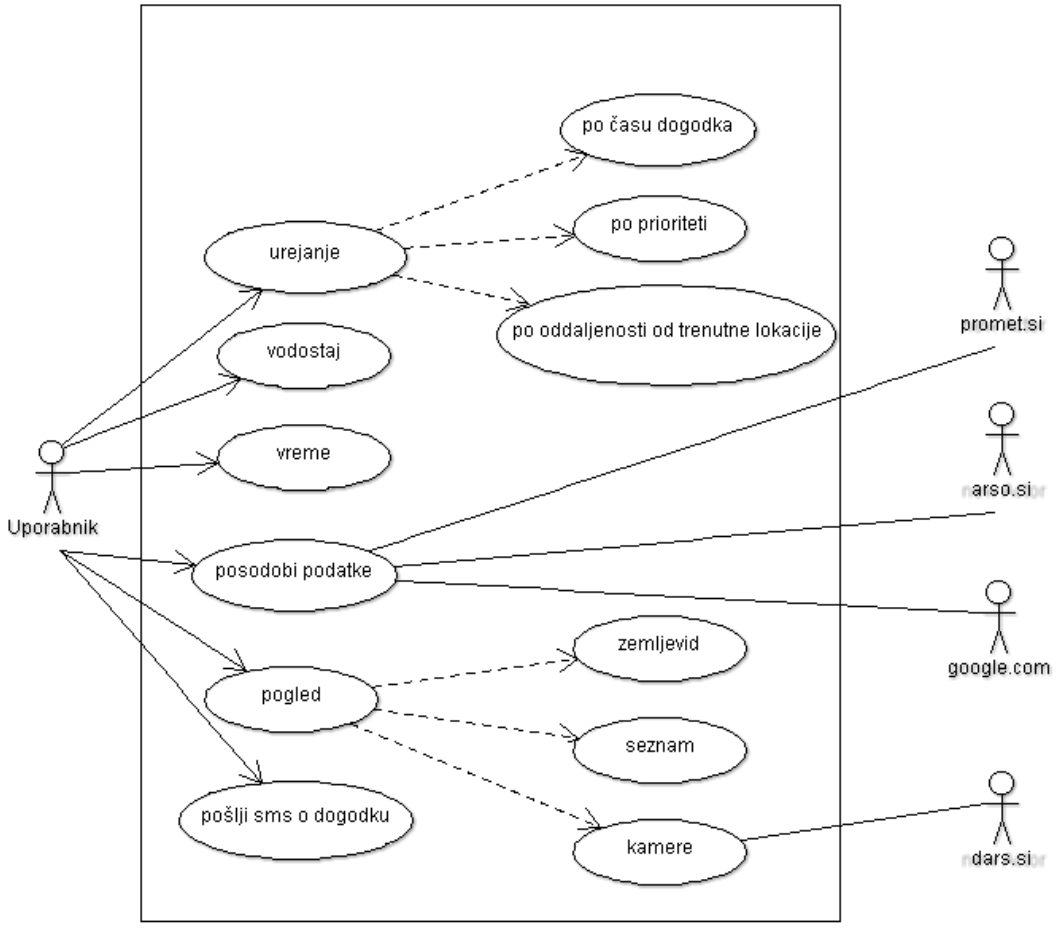
4 Razvoj

4.1 Metodologija razvoja

Razvoj se navadno začne z nalogo, določitvijo zahtev in specifikacijam za izvedbo določenega projekta. Glede na vse značilnosti projekta in na drugi strani razvijalsko ekipo, programske arhitekte, upravljalce podatkovnih baz, testne uporabnike in vse ostale, se določi metodologijo razvoja. V zadnjem času prevladujejo agilne metodologije in tehnike, kot so metodologije RAD, ekstremno programiranje (*angl.* extreme programing) in druge, ki stremijo k hitrem razvoju, izdelavi prototipa, sprotnemu testiranju, hitrim razvojnim ciklom, sodelovanju med vsemi vpletenimi in vključevanju naročnika v proces razvoja na vseh nivojih. Pri mojem projektu sem sledil tem ciljem. Delo je potekalo v orodju Eclipse z dodatki (*angl.* plug-ins) ADT, Google API in Subclipse. Glede na to, da predhodnih izkušenj s platformo nisem imel, je bil cilj na začetku izdelati delujoč prototip, ki se kasneje postopno nadgrajuje, tako s strani grafičnega vmesnika, kot podprtih funkcionalnostih. Z uporabo repozitorija SVN je bila zagotavljena vedno zadnja delujoča različica, ki je vedno na voljo za testiranje in zgodovina vseh sprememb skozi celoten cikel razvoja. Pomemben del skozi celoten razvoj pa je bilo tudi testiranje na napravah.

4.2 Diagram uporabe

Diagram primerov uporabe (*angl.* use case diagram) na sliki, poleg osnovnih primerov uporabe in nekaterih relacij, prikaže tudi meje aplikacije in akterje, ki sodelujejo. Jasno je razvidno, katere funkcionalnosti je potrebno podpreti z aplikacijo. Pomembne so tudi meje aplikacije in interakcija z zunanjimi viri podatkov. Razvidno je, da so to spletni viri podatkov in spletne storitve.



Slika 4.1: Diagram primerov uporabe, izdelan z odprtokodnim orodjem ArgoUML

4.3 Podatkovna baza

Podatkovna baza je vedno pomemben del sistema. V aplikaciji ni primerov uporabe, ki bi predvidevali zapletenejšo zgradbo podatkovnega modela. V aplikaciji služi izključno hranjenju trenutno aktualnih podatkov, pridobljenih pri zadnji posodobitvi podatkov znotraj aplikacije. Glede na to, da podatkovni model ne vsebuje nobene relacije, ampak samo štiri tabele, entitete s pripadajočimi atributi določenih podatkovnih tipov, aplikacija ne shrani podatkov za kasnejše analize ali druge obdelave. Celoten model je sestavljen iz sledečih tabel

in njihovih opisov. Podčrtani atributi so primarni ključi tabel, tujih ključev pa, kot sem že omenil, ne potrebujemo.

Dogodek

Dogodek vsebuje ključne podatke o nekem dogodku na cesti. Poznamo osem vrst takih dogodkov, zato se postavi vprašanje, ali uporabiti dodatno tabelo, ki bi služila kot šifrant. Povezavo med tabelami pa bi določil tuji ključ. Ker pa v našem modelu tega šifranta ne potrebujemo nikjer drugje in še vedno lahko implementiramo urejanje po tem atributu, se za to nisem odločil.

Dogodek [integer id, text kratek_opis, text opis, double x_koordinata, double y_koordinata, integer prioriteta, text datum]

Vreme

Tabela Vreme se delno napolni že ob kreiranju podatkovne baze. Gre za podatke o naprej določenih mestih opravljanja meritev, za katera v naprej vemo, kje so in imamo njihov opis.

Vreme [integer id, double x_koordinata, double y_koordinata, text mesto, text temeratura, text opis, text veter]

Kamera

Prav tako kot Vreme je tudi tabela Kamera že naprej določena in se ne spreminja. Ker uporaba kamer brez povezave ni mogoča, je celotna vsebina znana v naprej, vključno z URL naslovom slike.

Kamera [integer id, text opis, double x_koordinata, double y_koordinata, bool jeMejniPrehod, bool jeAvtocesta, text url]

Vodostaj

Vodostaj, je podobno kot Vreme, delno predizpolnjen, delno se podatki vpišejo ob posodobitvah podatkov znotraj aplikacije.

Vodostaj[id, datum, opis, pretok, temperatura, url, x_koordinata, y_koordinata, lokacija_opis]

Nekateri podatki se zapišejo v podatkovno bazo ob njenem kreiranju, druge pa vnesemo šele, ko so na voljo. Podatki o lokaciji in opisu vremenskih postaj se ne bodo spreminjali, zato jih

vnesemo takoj, lahko tudi pred izmenjavo podatkov s strežniki. Temperaturo ali nivo vode pa lahko vnesemo šele, ko imamo ustrezne podatke iz virov.

4.4 Uporabniški vmesnik

Uporabniški vmesnik se načrtuje s tako imenovanimi orodji Mockup (App Inventor za Android, DroidDreaw, Balsamiq), s katerimi lahko predvidimo razporeditev gradnikov na zaslonu. Ta orodja nam pomagajo dobiti občutek, kako bo videti končni izdelek še preden ga začnemo izdelovati. Merilo za uporabniški vmesnik so vedno jasne, intuitivne zaslonske maske, ki same kličejo k akciji. Uporabniku morajo pričarati občutek preprostosti, uporabnosti in odzivnosti.

Android za izdelavo zaslonskih mask aktivnosti uporablja XML datoteke. S temi določimo grafične gradnike, njihov položaj in lastnosti. Na podlagi teh se znotraj okolja Eclipse samodejno generira datoteka (R.java) z referenco za vsak element. Tako imamo v programski kodi prek tega razreda referenco na grafične elemente v XML-u. Android pozna štiri glavne tipe razporeditev elementov na zaslonu:

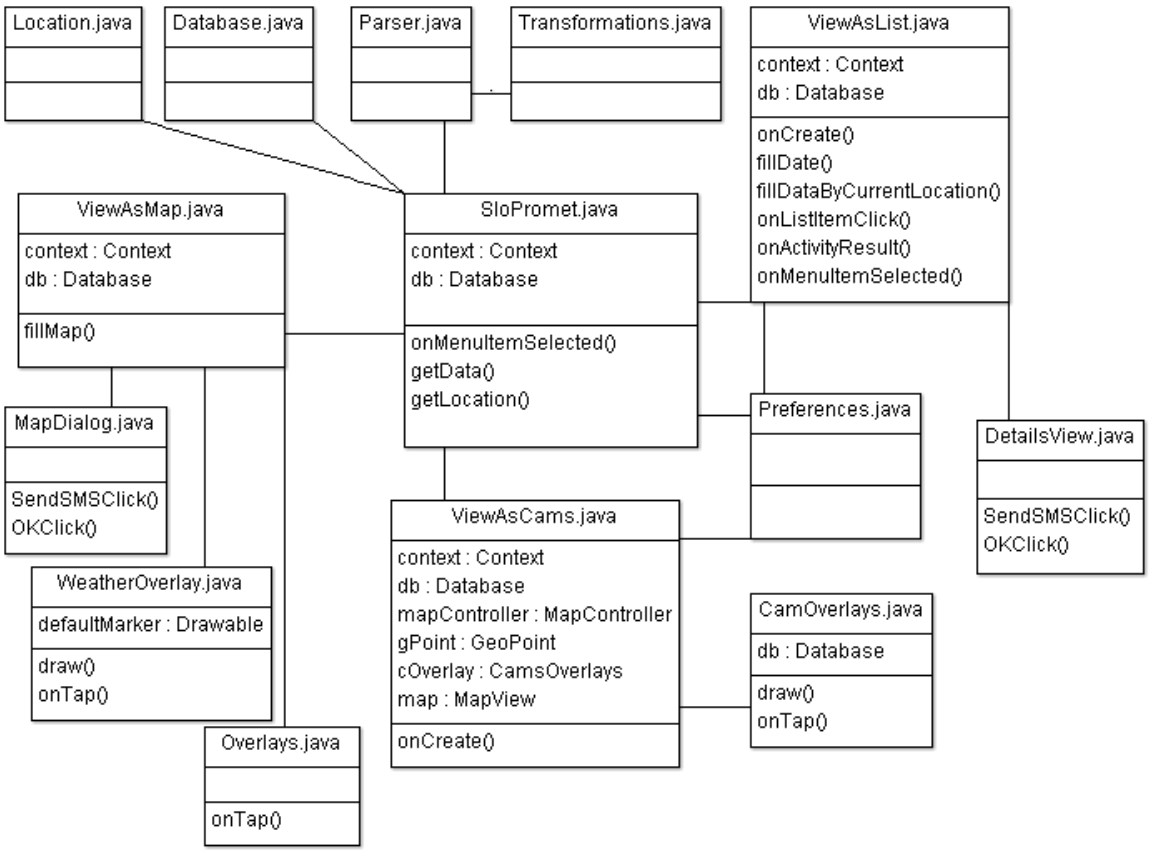
- FrameLayout;
- LinearLayout;
- TableLayout;
- RelativeLayout.

Sam sem uporabil LinearLayout, RelativeLayout in TableView. Primer linearne razporeditve glavnega okna aplikacije SloPromet na sliki:

Slika 4.2: Glavno okno aplikacije SloPromet

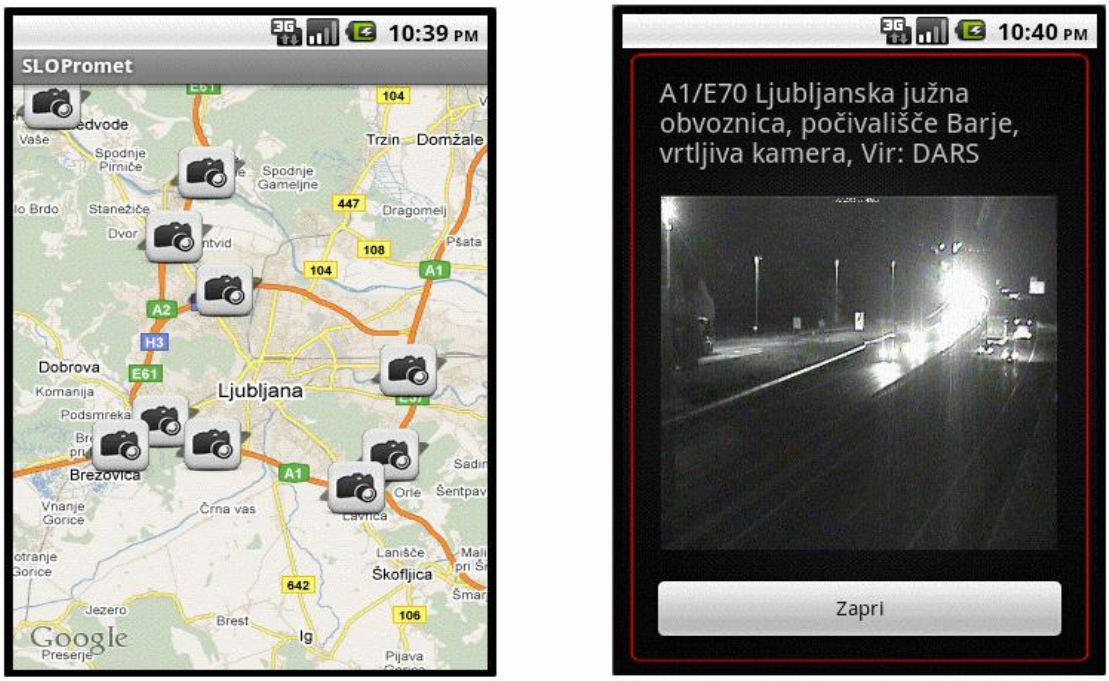
Poleg gradnikov za razporeditev elementov obstajajo še ostali grafični gradniki, ki jih lahko uporabimo znoraj izbrane razporeditve. Ostali gradniki, uporabljeni v aplikaciji so še ImageView za prikaz slik, ImageButton za prikaz gumba s sliko v ozadju, MapView za prikaz zemljevida, TextView za izpis besedila, ListView za izris seznama, ScrolView za izris vsebine, ki presega velikost zaslona.

4.5 Programski razredi in logika



Slika 4.3: Razredni diagram aplikacije SloPromet. Prikazani so vsi razredi, njihove najpomembnejše metode in atributi. Potrebno je povedati, da pri vseh razredih, ki so razširitve katere od oblik razreda Aktivnost, manjkajo metode onRestart(), onStart(), onResume(), onPause() in onDestroy(). Diagram je narejen z orodjem ArgoUML.

Glavni ali osnovni programski razred (*angl*. main class) je razred SloPromet, ki razširja razred Aktivnost. Vključuje konstruktor, privatne spremenljivke in deklaracije razredov ter konstant. Omenjeni razred ob zagonu zažene ločeno nit pridobivanja uporabnikove lokacije, če je le-ta na voljo. Bolj natančen opis pridobivanja lokacije je v poglavju 4.5. V razredu so implementirani še dogodki za posodobitev podatkov in inicializacija razreda za delo s podatkovno bazo Database. Tudi posodobitev podatkov poteka v ločeni niti. Tako je delovanje glavnega programa nemoteno. Z metodo onMenuItemSelected() se glede na uporabnikovo izbiro kreira nova Aktivnost. ViewAsList razširja (*angl.* extends) ListActivity, ViewAsMap in ViewCams pa MapActivity.



Slika 4.4: Pregled spletnih kamer na zemljevidu (razred ViewAsMap) levo in podroben pogled (razred MapDialog) desno

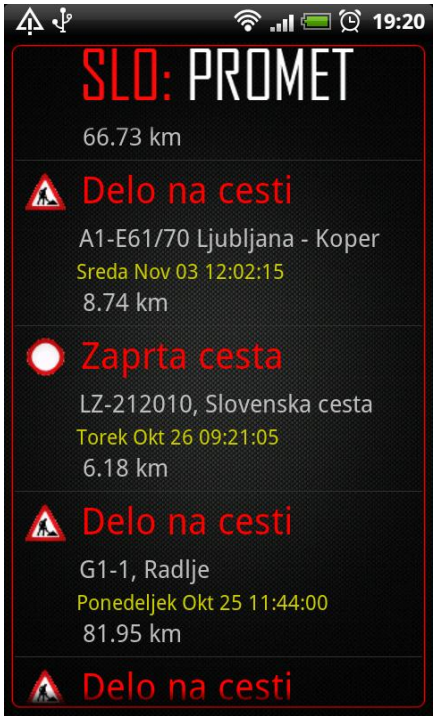
Pomembni razredi za aplikacijo so pomožni razredi, ki se uporabljajo za posodobitev podatkov. Tako sta v razredu Transformations implementirani dve metodi, ki pretvarjata iz Gauss-Krugerjevih koordinat, ki jih dobimo s spletne strani PIC, v geografske koordinate, ki jih potrebujemo za izrisovanje na zemljevidu. Razred Parser med drugim poskrbi za pretvorbo podatkov o vremenu in vodostaju, ki sta na spletni strani v obliki datoteke HTML, v bolj prijazno obliko, nakar se podatki shranijo v podatkovno bazo, za kar skrbijo metode v razredu Database. Primer teh metod so: createEvent(), ki ustvari nov dogodek (*angl.* insert), fetchAllEventsByVzrok(), ki vrne samo dogodke določenega tipa, fetchAllEventsByOddaljenost(), ki vrne dogodke, urejene po oddaljenosti od trenutne lokacije. Omenil bi še metode za posodabljanje podatkov v podatkovni bazi (*angl.* update) updateWeather() in updateWaterLevel(). Vsa interakcija s podatkovno bazo temelji na razredu SQLLiteDatabse in njegovih metodah insert(), update() in query(), s pomočjo katerih lahko izvajamo operacije nad podatkovno bazo.

Slika 4.5: Pregled dogodkov na zemljevidu

Slika 4.6: Nastavitve aplikacije in okno z informacijami o aplikaciji

Za shranjevanje nastavitev znotraj aplikacije, Android ponuja razred SharedPreferences, ki deluje na principu ključ/vrednost [11]. Shranjujemo lahko osnovne vrednosti za podatkovne tipe, ki so v aplikaciji vedno dosegljivi. Za shranjevanje nastavitev in privzetih vrednosti sem uporabil ta mehanizem. Pregled nastavitev je prikazan na sliki 4.6. Pri zaslonski maski s

podatki o aplikaciji pa je uporabljen razred AlertDialog, ki je namenjen prikazovanju kratkih sporočil na zaslon znotraj Aktivnosti.



Slika: 4.7: Seznam dogodkov, podatek o kilometrih predstavlja oddaljenost od trenutne lokacije.

Za računanje razdalje od trenutne lokacije sem uporabil razred Location. Le-ta je del vmesnika Google API in omogoča geolokacijske pretvorbe. Naj omenim metodo distanceBetween(), ki izračuna približno razdaljo med dvema točkama, katere koordinate sprejme kot vhodne parametre. Obstajajo še metode distanceTo(), convert(), ki omogočajo pretvarjanje med koordinatami in naslovi [12].

4.6 Lokacija uporabnika

Določanje lokacije uporabnika je lahko zahtevno. Razlog za to je več virov lokacijskih podatkov na uporabnikovi napravi. Zavedati pa se moramo tudi, da se lahko lokacija spreminja in je lahko v vsakem trenutku nenatančna. Za dostop do lokacijskih podatkov in njihovih posodobitev moramo v datoteki manifest navesti ustezne pravice, ki jih mora uporabnik potrditi pri nalaganju aplikacije [4]. V ta namen sta dodani spodnji vrstici:

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

Viri lokacijskih podatkov

GPS, informacija o celici (*angl*. Cell ID) ali WiFi. Vsi navedeni viri lahko nudijo informacijo o lokaciji. Od tega, katerega bomo izbrali pa je odvisna natančnost, hitrosti in poraba baterije na napravi.

Gibanje uprabnika

Gibanje uporabnika moramo pri določanju natančne lokacije vzeti v zakup in nenehno preverjati podatke o spremembi.

Spremenljiva natančnost

Lokacijski podatki iz različnih virov imajo različno natančnost. Tako je lahko podatek izpred desetih sekund, pridobljen iz nekega vira, bolj natančen od trenutnega podatka, pridobljenega iz drugega vira.

Podatke o lokaciji se znotraj ogrodja SDK pridobi s pomočjo razreda LocationManager. S pomočjo metode isProviderEnable(), ki sprejme za atribut tip providerja (GPS_PROVIDER ali NETWORK_PROVIDER), preverimo, ali lahko dostopamo do lokacije uporabnika. Če dostopa nimamo, metoda vrne izjemo. V nasprotnem primeru lahko dostopamo do podatkov z metodo requestLocationUpdates(). V trenutku, ko to zahtevamo, se sproži števec z namenom ponovne posodobitev trenutne lokacije. Za to sem uporabil razred LocationListener in metodo onLocationChanged, ki se izvede na vsaki dve sekundi. Opisane metode so del razreda Lokacija, ki se inicializira takoj ob zagonu aplikacije.

4.7 Testiranje

Testiranje prototipov in razvojnih verzij je potekalo znotraj okolja Eclipse, v emulatorju z naslednjimi nastavitvami navidezne naprave AVD:

- A Test (resolucija zaslona , 256 MB pomnilnika, Android 1.6);
- B Test (resolucija zaslona , 512 MB pomnilnika, Android 1.8).

Testiranje je potekalo tudi na fizičnih napravah. Pri tem so bili na voljo trije modeli telefonov:

- HTC Desire (resolucija zaslona 480x800 , 512 MB notranjega pomnilnika, procesor 1 GHz, Android 2.2);
- HTC Desire HD (resolucija zaslona 480x800, 768 MB notranjega pomnilnika, procesor 1 GHz, Android 2.2);
- HTC Wildfire (resolucija zaslona 240x320, 384 MB notranjega pomnilnika, procesor 528 MHz Android 2.0).

Testiranje je pokazalo, da je emulator sicer uporaben za delo. Deluje dokaj stabilno, vendar je v nekaterih primerih počasen in zahteva veliko sistemskih virov (pomnilnika). Velik minus takega načina testiranja je tudi razlika med zaslonom občutljivim na dotik in klikanjem z računalniško miško. Prav tako je zapleteno spreminjati lokacijo napravam AVD, čeprav je to omogočeno. Tako sem večji del testiranja opravil kar na fizičnih napravah. Zavedam se, da bi moral aplikacijo preizkusiti tudi na napravah drugih proizvajalcev, vendar to žal ni bilo mogoče.

5 Sklep in smernica za ravoj

Pred začetkom razvoja je bil cilj izdelati mobilno aplikacijo za Android za spremnljanje stanja na slovenskih cestah. To mi je v celoti tudi uspelo. Izpolnjene so bile vse zahteve glede uporabe, delovanja in grafičnega vmesnika. Že med razvojem pa se je pojavilo kar nekaj idej in možnosti za izboljšave, ki sem jih razdelil v dva sklopa.

Strežniški del aplikacije

Ena glavnih možnosti za izboljšavo je izdelava strežniškega dela aplikacije. To bi zelo povečalo možnosti za dodatne funkcionalnosti. Obdelava podatkov bi se izvajala na strežniku, kar prihrani nekaj časa in virov naprave. Posodobitve podatkov bi bile hitrejše in bolj učinkovite. Uporabniki bi na primer lahko dodajali svoje dogodke. Le-tem bi lahko pripenjali slike ali pa bi omogočil preprosto komentiranje posameznih dogodkov, možnost dodajanja slik k obstojčim dogodkom. Tako bi imela aplikacija še večjo uporabno vrednost. Tudi delovanje bi bilo hitrejše. Prenašalo bi se manj podatkov, kar je zelo pomembno takrat, ko uporabnik uporablja mobilni internet svojega ponudnika. Slaba oziroma druga stran vsega naštetega pa je zahteva po zanesljivih strežniških kapacitetah in potreba po administraciji. Ker bi uporabniki lahko dodajali poljubne vnose in komentarje, bi bilo do neke mere potrebno bdenje na temi vnosi, da ne bi prihajalo do napačnih podatkov.

Uporabniški vmesnik

Izboljšave na področju interakcije z uporabnikom so vedno možne. Morda se jih kot razvijalec še premalo zavedam. Cilj je bil vsekakor izdelati jasen in uporaben uporabniški vmesnik.

Dodatne storitve

Druga možnost za popestritev in še večjo dodano vrednost aplikacije, so prikaz dodatnih podatkov, ki so v pomoč, kadar smo na poti. Ena od teh bi bil prikaz zastojev na določeni poti. Uporabnik bi lahko izbral destinacijo na zemljevidu, aplikacija pa bi na podlagi tega prikazala samo zastoje na poti od trenutne do željene destinacije. Ostale uporabne informacij bi lahko bile lokacije in delovni časi bencinskih postaj, stacionarnih radarjev, počivališč, bolnišnic ali večjih turističnih znamenitosti. V dodatne storitve bi vključil tudi možnost delovanja same aplikacije kot storitev (*angl.* Service), ki bi glede na trenutno lokacijo z zvočnimi signali samodejno opozarjala na izredne dogodke na cesti.

V zvezi z dodatnimi storitvami vidim tudi možnost trženja aplikacije, kot platforme za oglaševanje in dodatno storitev podjetjem, ki želijo svojim strankam ponuditi nekaj več. Zaenkrat aplikacija še ni objavljena na Android Market-u. Bila pa je uspešno predstavljena na natečaju za najboljšo aplikacijo družbe Mobitel, ki je potekal v drugi polovici 2010 in začetku leta 2011. Izmed 54 prijavljenih aplikacij je aplikacija glede na število prenosov dosegla petnajsto mesto. Projekta pa še zdaleč ni konec. Možnosti je vsekakor še veliko. Stvari bom skušal še izboljšati, tako da bodo še bolj prijazne končnemu uporabniku. Glede na svetovne trende in raziskave je trenutna pot Android platforme obrnjena strmo navzgor. Menim, da z modelom, ki so ga vzpostavili, lahko prevzamejo vodilno mesto med vsemi operacijskimi sistemi na mobilnih napravah. Nas, kot uporabnike in razvijalce, pa čaka še veliko izivov pri uporabi in razvoju te platforme.

Slike

Literatura

[1] Wikipedia: Android (operating system)

Dostopno na http://en.wikipedia.org/wiki/Android_operating_system (Zadnji obisk: 10.5.2011)

[2] Android 1.1

Dostopno na http://developer.android.com/sdk/android-1.1.html (Zadnji obisk 15.6.2011)

[3] Intervju z A. Rubinom, odgovornim za Android

Dostopno na http://techcrunch.com/2010/06/01/android-chief-andy-rubin-updates-will-eventually-come-once-a-year/ (Zadnji obisk: 10.6.2011)

[4] Pridobivanje lokacije uporabnika

Dostopno na http://developer.android.com/guide/topics/location/obtaining-user-location.html (Zadnji obisk: 11.5.2011)

[5] S. Hashimi, S. Komatineni, D. MacLean; Pro Android 2, Apress, 2010

[6] Wikipedia: Activity

Dostopno na http://developer.android.com/reference/android/app/Activity.html (Zadnji obisk: 10.5.2011)

[7] Wikipedia: Android Manifest

Dostopno na http://developer.android.com/guide/topics/manifest/manifest-intro.html (Zadnji obisk: 10.5.2011)

[8] Wikipedia: Android SDK

Dostopno na http://en.wikipedia.org/wiki/Android_SDK#Android_SDK (Zadnji obisk: 10.6.2011)

[9] Uporaba interneta na mobilnih napravah

Dostopno na: http://www.mobilegroove.com/smartphone-market-drives-600-growth-in-mobile-web-usage (Zadnji obisk: 1.7.2011)

[10] Uradni Googlo-ov blog

Dostopno na: http://googleblog.blogspot.com/2011/05/android-momentum-mobile-and-more-at.html (Zadnji obisk: 1.7.2011)

[11] R.Maier, Profesional Android 2, Appliacation development, Wiley Publishing, 2010, pogl. 6

[12] Lokacija

Dostopno na: http://developer.android.com/reference/android/location/Location.html (Zadnji obisk: 1.7.2011)

[13] Različice operacijskega sistema Android

Dostopno na http://developer.android.com/resources/dashboard/platform-versions.html (Zadnji obisk 5.7.2011)