Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Mitja Kastrevc

Geografsko lociranje na platformi Android z uporabo spletnih storitev

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana, 2013

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: 00059/2012
Datum: 05.11.2012

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

Kandidat: **MITJA KASTREVC**

Naslov: **GEOGRAFSKO LOCIRANJE NA PLATFORMI ANDROID Z UPORABO SPLETNIH STORITEV**

GEO-LOCATING ON ANDROID PLATFORM USING WEB SERVICES

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Spoznajte koncepte in pristope aplikacij za mobilne naprave ter opišite zaznavanje podatkov na mobilnih platformah, geografsko lociranje in pridobivanje informacij o baznih postajah. Analizirajte platformo Android. Identificirajte gradnike aplikacije, ki so namenjeni pridobivanju podatkov o bazni postaji, lokaciji mobilne naprave, povezavi aplikacije Android s spletnim strežnikom in prikazovanju zemljevida. Izdelajte aplikacijo, ki prikazuje podatke o bazni postaji, na katero je prijavljena mobilna naprava in izriše lokacijo mobilne naprave in bazne postaje na zemljevidu z uporabo RESTful spletnih storitev.

Mentor: prof. dr. Matjaž B. Jurič

Dekan: prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Mitja Kastrevc, z vpisno številko **63060105**, sem avtor diplomskega dela z naslovom:

Geografsko lociranje na platformi Android z uporabo spletnih storitev

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Matjaža Branka Juriča,
- 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. januar 2013 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Matjažu Branku Juriču za vso izkazano pomoč ob nastajanju tega diplomskega dela. Za praktične nasvete se še posebej zahvaljujem prijatelju Mateju Slapšaku. Posebna zahvala pa gre tudi moji družini, ki me je vsa leta spodbujala in mi stala ob strani.

Svojim dragim staršem.

Kazalo

Seznam uporabljenih kratic

SIM	kartica, ki omogoča prijavo na mobilno omrežje
IMEI	enolična serijska številka telefonske naprave
GSM	standard mobilnih komunikacij druge generacije
UMTS	standard mobilnih komunikacij tretje generacije
LTE	standard mobilnih komunikacij četrte generacije
BTS	bazna postaja s tehnologijo GSM
NodeB	bazna postaja s tehnologijo UMTS
MCC	mednarodna identifikacijska številka države
MNC	s kodo MCC tvorita mednarodno številko operaterja
Cell ID	identifikacijska številka celice na bazni postaji
LAC	regijska identifikacijska koda
Latitude	zemljepisna širina
Longitude	zemljepisna dolžina
API	programski vmesnik aplikacije
HTTP	protokol za prenos informacij na spletu
POST	metoda za prenos podatkov preko protokola HTTP
Wi-Fi	lokalno brezžično omrežje

Povzetek

Cilj diplomske naloge je spoznati koncepte in pristope aplikacij za mobilne naprave. V prvem delu diplomska naloga opisuje zgodovino mobilne telefonije in mobilna omrežja v Sloveniji. Grobo je opisano zaznavanje podatkov na mobilnih platformah, geografsko lociranje in pridobivanje informacij o bazni postaji. Predstavljeno je orodje Eclipse in programski jeziki Java, PHP, XML in JSON, ki omogočajo razvoj aplikacije na operacijskem sistemu Android. Pomembna povezava med strežnikom in aplikacijo poteka po spletni storitvi RESTful. V drugem delu diplomske naloge so realizirani pomembni koncepti gradnikov aplikacije. Ti so namenjeni pridobivanju podatkov o bazni postaji, lokaciji mobilne naprave, povezavi aplikacije Android s spletnim strežnikom in prikazovanju zemljevida. Na koncu smo ustvarili aplikacijo Lokator, ki vsebuje vse te gradnike. Aplikacija prikazuje podatke o bazni postaji, na katero je prijavljena mobilna naprava in izriše lokacijo mobilne naprave in bazne postaje na zemljevidu.

Ključne besede: Android, bazna postaja, mobilna telefonija, Java, PHP, XML, REST, Eclipse

Abstract

The objective of this graduate work is to get familiar with concepts and approaches of the mobile devices applications. In the first part one can find a description of the history of mobile telephone systems and networks in Slovenia. There is a rough description of data registration on mobile platforms, geographical locating and obtaining information regarding base stations. The Eclipse tool and programming languages Java, PHP, XML and JSON are introduced - these enable application development in the Android operation system. An important connection between the server and the application is enabled via RESTful web service. In the second part of the graduate work important concepts of application part are realized. Their function is to obtain information about the base station, mobile device location, connection of the Android application to web server, and map displaying. At the end the Lokator application was created, which contains all these parts. The application displays information regarding the base station to which the mobile device is registered, and displays the location of the mobile device and the base station on the map.

Keywords: Android, base station, mobile telephony, Java, PHP, XML, REST, Eclipse

Poglavje 1

Uvod

V času, ki ga živimo, v času izredne mobilnosti vsak človek na tem planetu uporablja najmanj eno mobilno napravo. Naprava služi pošiljanju sporočil, klicanju in navsezadnje tudi brskanju po spletu. Osnovne mobilne telefone postopoma zamenjujejo pametni telefoni in še zmogljivejši tablični računalniki. Ti poleg ostalih funkcij omogočajo uporabo raznih aplikacij. Tržni delež teh se postopoma povečuje. Kot osnovni telefoni so tudi ti povezani na mobilno omrežje preko bazne postaje. Te so postavljene po Sloveniji glede na zgoščenost potencialnih uporabnikov v posamezni pokrajini. Vsaka bazna postaja je enolično poimenovana.

Mnoge uporabnike teh naprav zanima, katero bazno postajo trenutno uporabljajo oziroma na katero so prijavljeni. V ta namen se nam je v okviru diplomskega dela porajala ideja razviti zanesljivo mobilno aplikacijo, ki bo uporabniku grafično posredovala te podatke. Naloga je bila težavna, ker pred nami aplikacije za prikazovanje baznih postaj ni razvil še nihče. V delu se je bilo potrebno seznaniti z razvojem aplikacij Android, načinom geografskega lociranja in pridobivanja podatkov iz bazne postaje na mobilno napravo. Iz teh podatkov smo razvili algoritem, ki je zmožen izračunati ime bazne postaje različnih mobilnih operaterjev. Odločili smo se ustvariti spletni strežnik, kjer bomo shranjevali vse podatke o baznih postajah v Sloveniji (opis, lokacija, slika). Smo edini, ki hranimo vse te podatke. Iz tega strežnika

smo, ob izračunanem imenu bazne postaje, želeli pridobiti ostale podatke o bazni postaji in jih s pomočjo aplikacije grafično prikazati uporabniku. Za grafični prikaz smo razvili aplikacijo Android. Zato je bilo potrebno osvojiti programske jezike Java, XML, JSON in PHP ter spletno storitev RESTful.

Poglavje 2

Mobilno omrežje

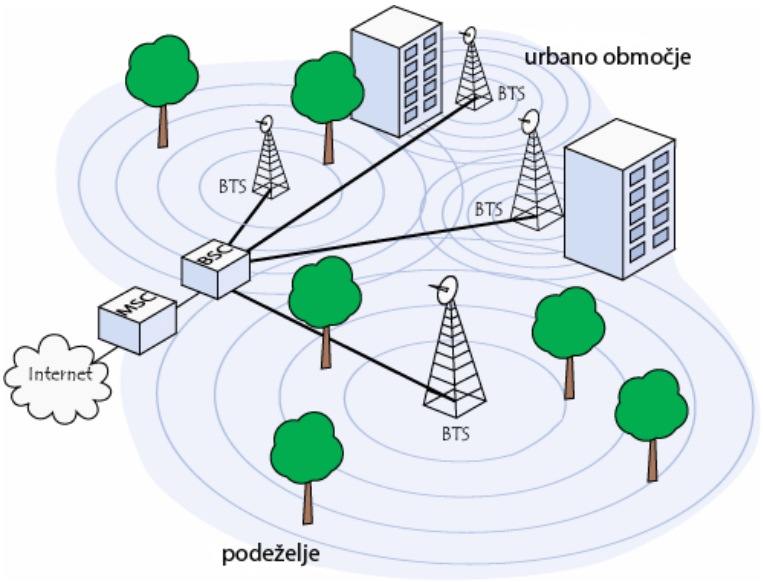
Mobilne naprave so dandanes nepogrešljiv element v našem življenju. Poleg klicanja in pisanja sporočil SMS nam omogočajo tudi brskanje po spletu. Za povezavo z omrežjem morajo mobilne naprave pošiljati in prejemati radijske signale iz poljubne celice na bazni postaji. V Sloveniji trenutne bazne postaje oddajajo tehnologije GSM, UMTS ali LTE na frekvencah 900, 1800 in 2100 MHz. Običajno so postavljene na stolpih ali strehah zgradb. Od uporabnikov običajno niso oddaljene več kot 8 do 13 km. Bazne postaje so priključene na žično komunikacijsko omrežje. Če bazne postaje ni mogoče povezati neposredno v žično komunikacijsko omrežje, mora biti ta povezana z drugo bazno postajo preko relejne povezave. Na Sliki 2.1 je prikazana Tušmobilova bazna postaja postavljena v Sodišincih. Na levi strani je prikazana bazna postaja od zunaj, na desni pa pogled v notranjost [23].

Na bazne postaje s tehnologijo GSM se prijavljajo le mobilne naprave, ki imajo vstavljeno kartico SIM (*Subscriber Identity Module*). Ta kartica enolično določa identiteto uporabnika in je lahko zavarovana s štirimestno kodo PIN (*Personal Identification Number*). Vsaka mobilna naprava vsebuje petnajstmestno identifikacijsko številko, imenovano IMEI (*International Mobile Equipment Identity*). Kartica SIM omogoča identifikacijo uporabnika ne glede na uporabljeno napravo. Komunikacijo med napravo in celico na bazni postaji imenujemo brezžični vmesnik. Vsaka bazna postaja BTS (*Base Tran-*

Slika 2.1: Tušmobilova bazna postaja NE1585 v Sodišincih

sceiver Station) je povezana na krmilnik baznih postaj BSC (*Base Station Controller*), ki je odgovoren za porazdelitev sredstev. Sistem ima krmilnik za bazne postaje in vključuje bazne postaje v sistem. Vsi krmilniki baznih postaj BSC so fizično povezani v center MSC (*Mobile Switching Centre*). Ta je povezan v javno telefonsko omrežje ter internet. Sistem MSC vključuje tudi podsistem NSS (*Network Station Subsystem*), ki je odgovoren za upravljanje uporabnikovih identitet, njihove lokacije in vzpostavitev povezave z ostalimi uporabniki. Princip delovanja mobilnega omrežja je predstavljen na Sliki 2.2. Na center MSC so ponavadi priključene štiri baze [8]:

- **Home Location Register (HLR):** Baza vključuje lokacijo in administrativne podatke registriranega domačega uporabnika.
- **Visitor Location Register (VLR):** Baza vključuje uporabnike, ki trenutno gostujejo v tem omrežju. Ti se iz baze izbrišejo v primeru neaktivnosti ali odjave iz omrežja.



Slika 2.2: Delovanje mobilnega omrežja GSM [8]

- **Equipment Identify Register (EIR):** Baza mobilnih naprav.
- **Authentication Centre (AUC):** Odgovorna za preverjanje identitete uporabnikov.

Omrežje zagotavlja neprekinjeno zvezo ob prehajanju med celicami baznih postaj. Princip delovanja sistema GSM je podoben tudi za sistema UMTS in LTE [8].

2.1 Zgodovina mobilne telefonije

Prvi mobilni telefon so izdelali v Združenih državah Amerike v podjetju Bell Labs leta 1946. Telefon je bil nameščen v avtomobilu. Podjetje AT&T ga je začelo tržiti leta 1947 v 100 mestih v ZDA pod imenom MTS (*Mobile Telephone Service*). Klic se je vzpostavil s strani izvajalca storitve. Uporabnik sistema je imel gumb. Ko je bil ta pritisnjen, je uporabnik lahko sogovorniku govoril, v nasprotnem primeru je lahko le poslušal. Naprava je tehtala okoli

36 kilogramov, sistem je bil omejen. Tehnologija spada v pred generacijo mobilnih telekomunikacij. V to generacijo spadajo tudi IMTS, RCC, A-Netz, B-Netz [20].

Prva generacija mobilnih telekomunikacij (1G) je bila predstavljena v osemdesetih letih prejšnjega stoletja. V tej generaciji se je uporabljalo analogne radijske signale. Signalizacija za prijavo na bazno postajo je potekala digitalno. Hitrost je bila med 28 in 56 kbit/s. Prvo komercialno avtomatsko celično omrežje so predstavili v ZDA leta 1978. Razvili so jo v podjetju Bell Labs in se je imenovala AMPS (*Advanced Mobile Phone System*). Leta 1981 so na Danskem, Finskem, Norveškem in Švedskem zagnali sistem NMT (*Nordic Mobile Telephone*). Ta je prvi podpiral gostovanje v drugih omrežjih [13].

V devetdesetih letih prejšnjega stoletja se je pojavila druga generacija mobilnih telekomunikacij (2G). Od prejšnje generacije se je razlikovala po uporabi digitalnih radijskih signalov ter uvedbi sporočil SMS. Evropski standard GSM (*Global System for Mobile communications*) in ameriški CDMA (*Code Division Multiple Access*) sta za prevlado tekmovala na svetovnem trgu. Najbolj je to uspelo standardu GSM, saj se uporablja skoraj v vseh državah sveta. Prvo omrežje GSM je postavil Radiolinja na Finskem leta 1991. Pogosto so omrežja GSM implementirana na 900 in 1800 MHz, v Ameriki pa na 850 in 1900 MHz. Prvo nadgradnjo je standard GSM dobil s tehnologijo GPRS (2,5G). Ta omogoča paketni prenos podatkov do 82,4 kbit/s. Druga nadgradnja je s tehnologijo EDGE (2,75G), ki omogoča paketni prenos podatkov do 236,8 kbit/s [14].

Pomanjkljivost druge generacije mobilnih telekomunikacij je bil počasen prenos podatkov pri uporabi telefona v vsakdanjem življenju za brskanje po spletu. Zato se je začela razvijati tretja generacija mobilnih komunikacij (3G), ki je poskrbela za hitrejši prenos podatkov po mobilnem omrežju. Prvo omrežje UMTS (*Universal Mobile Telecommunications System*) je komercialno zagnal NTT DoCoMo na Japonskem leta 2001. Tehnologija UMTS omogoča prenos podatkov do 384 kbit/s. Prav tako je bil uveden videoklic. Večina omrežij oddaja na frekvenci 2100 MHz, na ameriškem kontinentu pa

na 850 in 1900 MHz. Evropska unija je leta 2009 dovolila uporabo tehnologije UMTS tudi na frekvencah GSM. Tako nekateri evropski operaterji uporabljajo spekter 900 MHz za tehnologijo UMTS. Prva posodobitev je bila HSDPA, ki teoretično omogoča 14,4 Mbit/s prenosa podatkov k uporabniku ter nato HSUPA, ki omogoča 5,76 Mbit/s prenosa od uporabnika. Skupek HSDPA in HSUPA imenujemo tudi HSPA (3,5G). Zadnja nadgradnja HSPA+ (3,75G) omogoča teoretično prenos podatkov 168 Mbit/s k uporabniku in 22 Mbit/s od uporabnika [15].

Četrta generacija (4G) omogoča širokopasovni dostop na internet, spremenjen način dostopa do spleta, IP telefonijo, igralne storitve, televizijo v ločljivosti HD, video konference in televizijo v 3D. Velika prednost je tudi v večji internetni hitrosti prenosa podatkov od že obstoječih mobilnih storitev. Kot kandidata za četrto generacijo sta se pojavila dva standarda: LTE (*Long Term Evolution*) in mobile WIMAX. Mobilni operaterji so večinoma izbrali standard LTE. Prvo omrežje LTE so zagnali v Skandinaviji leta 2009. Prva različica dosega teoretično 300 Mbit/s prenosa podatkov k uporabniku in 75 Mbit/s prenosa od uporabnika. Čeprav LTE ne dosega vseh lastnosti četrte generacije, ga vseeno štejemo v to generacijo. Najpogostejše uporabljene frekvence za LTE bodo 700, 800, 1800 in 2600 MHz. Zaradi trenutno nedostopnosti frekvence 800 MHz mobilni operaterji v Evropi komercialna omrežja LTE zaženejo na frekvenci 1800 MHz. Nadgradnja LTE se bo imenovala LTE Advanced in bo omogočala 3 Gbit/s prenosa podatkov k uporabniku ter 1,5 Gbit/s prenosa od uporabnika [16].

2.2 Informacije o bazni postaji

Vsak mobilni telefon, ki je povezan na mobilno omrežje, prejema podatke o bazni postaji. Iz določenih parametrov lahko pridobimo nekatere informacije o celici, na kateri smo prijavljeni [12]:

- **Mobile Country Code (MCC):** Je kratica za unikatno trimestno identifikacijsko številko države. MCC Slovenije je 293.

- **Mobile Network Code (MNC):** S kodo MCC se določa identifikacijo mobilnega omrežja. Koda je lahko dvo ali trimestna.

- **BTS (GSM) / NodeB (UMTS):** Obe kodi predstavljata identifikacijsko številko bazne postaje. Ta je lahko med 0 in 65535, operater jo lahko določi poljubno. Ponavadi jo določi glede na regijo, mesto, tip tehnologije.

- **RncId (UMTS):** RNC je odgovoren za krmiljenje baznih postaj (NodeB). Način delovanja je podoben krmilniku baznih postaj BSC pri tehnologiji GSM (glej Sliko 2.2). RncId nam vrne identifikator RNC-ja. Osnovna nastavitev le-tega je vrednost 1. Če je omrežje večjih dimenzij in uporabljamo več RNC-jev, potem jih lahko poljubno poimenujemo glede na mesta ali na različno obdobje implementacije.

- **Sector Identity (SectorID):** Je enomestno število, ki nam poda poimenovanje celice (sektorja) na bazni postaji (BTS, NodeB). Navadno ima bazna postaja tri celice, vsako v svoji smeri. Nekatere bazne postaje imajo le eno celico, ki oddaja v vse smeri in ima običajno vrednost poimenovanja 0.

- **Cell Identiity (Cell ID):** Je unikatno poimenovanje celice na bazni postaji. Pri tehnologiji GSM je to število običajno sestavljeno iz vrednosti BTS in SectorID (2.1). Medtem ko je pri tehnologiji UMTS to število sestavljeno iz RNC števila, presledka in skupka NodeB, SectorID (2.2) [1].

$$GSM\ Cell\ ID = (BTS * 10) + SectorID \tag{2.1}$$

$$\begin{aligned} UMTS\ Cell\ ID &= RncID\ ((BTS * 10) + SectorID) \\ &= (RncID * 65536) + ((BTS * 10) + SectorID) \end{aligned} \tag{2.2}$$

[1] Zaradi lažjega razumevanja se navadno za vrednost Cell ID na UMTS-u uporablja prva notacija v enačbi (2.2), kjer RncID ločimo od preostale vrednosti.

- **Location Area Code (LAC):** Je lokacijsko identifikacijsko število posamezne regije.
- **xARFCN:** Je število radijsko-frekvenčnega kanala. To število nam podaja frekvenco (v MHz), na kateri je telefon povezan z bazno postajo. Za telefonijo GSM veljajo ARFCN kode, za UMTS pa UARFCN. Kode so že vnaprej definirane. Tako na primer vrednost 2950 v UARFCN pomeni frekvenčni pas 900 MHz.

2.3 Mobilna omrežja v Sloveniji

V Sloveniji imamo trenutno štiri infrastrukturne mobilne operaterje. 30. junija 2012 smo imeli na slovenskem trgu okoli 2.189.000 uporabnikov mobilne telefonije, od tega približno 71 % naročnikov. Slovenija beleži 106,3 % penetracijo mobilne telefonije. Seznam operaterjev z infrastrukturo najdete v Tabeli 2.1 [4].

Lastnik	**Blagovna znamka**	**MCC**	**MNC**
Telekom Slovenije, d.d.	Mobitel	293	41
Si.mobil, d.d.	Si.mobil, bob	293	40
Tušmobil, d.o.o.	Tušmobil	293	70
T-2, d.o.o.	T-2	293	64

Tabela 2.1: Lastniki operaterjev s svojimi mednarodnimi oznakami

2.3.1 Telekom Slovenije, d.d.

Telekom Slovenije, d.d. ima blagovno znamko Mobitel. Na njegovem omrežju gostujeta ponudnika storitev Debitel in Izimobil, omogoča pa tudi nacionalno gostovanje (roaming) Tušmobilu. Leta 1991 je zagnal omrežje NMT, ki je bilo ukinjeno ob koncu leta 2005. Novembra leta 1995 je začelo poskusno delovati omrežje GSM in decembra leta 2003 omrežje UMTS. Leta 2013 načrtuje

zagon omrežja LTE na 1800 MHz [10].

Pokritost prebivalstva s signalom na dan 31. december 2011:

- **GSM 900/1800 MHz:** 99,7 % s 1026-imi baznimi postajami,
- **UMTS 2100 MHz:** 88,1 % s 792-imi baznimi postajami.

Imena baznih postaj

Telekom Slovenije je bazne postaje razdelil v 10 regij, kot je prikazano v Tabeli 2.2.

regija	**ime regije**	**Cell ID**
A	Ljubljana	x0xxx
L	Ljubljana (okolica)	x1xxx
M	Maribor	x2xxx
C	Celje	x3xxx
K	Kranj	x4xxx
G	Nova Gorica	x5xxx
P	Portorož	x6xxx
T	Trbovlje	x7xxx
N	Novo mesto	x8xxx
S	Murska Sobota	x9xxx

Tabela 2.2: Regije Telekoma Slovenije s pripadajočimi Cell ID-ji

Ime Telekomovih baznih postaj se izračuna po enačbi (2.3). Regije se določi s pomočjo Tabele 2.2.

$$f(x) = \begin{cases} regija + \lfloor x/10 \rfloor, & \text{if } GSM; \\ regija + \lfloor ((x \mod 65535) - 20000)/10 \rfloor, & \text{if } UMTS; \\ error, & \text{sicer.} \end{cases} \tag{2.3}$$

Tako na primer CellID 8421 po enačbi dobi vrednost `N842`. Če je število v imenu manjše od treh številk, se levo zapolni z vrednostmi nič (na primer `A020`).

2.3.2 Si.mobil, d.d.

Si.mobil, d.d. trži blagovni znamki Si.mobil in Bob. Na njegovem omrežju preprodajata njegove pakete MMobil in AmisMobil, omogoča pa tudi nacionalno gostovanje podjetju T-2. Leta 1999 je zagnal omrežje GSM, leta 2007 pa še omrežje UMTS. Julija 2012 je kot prvi v Sloveniji zagnal omrežje LTE 1800 MHz [10].

Pokritost prebivalstva s signalom na dan 31. december 2010:

- **GSM 900/1800 MHz:** 99,6 % s 696-imi baznimi postajami,
- **UMTS 900/2100 MHz:** 69,0 % z 265-imi baznimi postajami.

Imena baznih postaj

Imena baznih postaj so dodeljena po mestih v Sloveniji, kot prikazuje Tabela 2.3. Tako se ime bazne postaje določi po enačbi (2.4).

regija	ime regije	Cell ID
LJ	Ljubljana	0xxxx
MB	Maribor	1xxxx
KR	Kranj	2xxxx
NM	Novo mesto	3xxxx
NG	Nova Gorica	4xxxx
KP	Koper	5xxxx
KK	Krško	6xxxx

Tabela 2.3: Regije Si.mobila s pripadajočimi Cell ID-ji

$$f(x) = \begin{cases} regija + \lfloor (x \mod 10000)/10 \rfloor, & \text{if } GSM; \\ regija + \lfloor (((x \mod 65535) \mod 10000) - 5000)/10 \rfloor, & \text{if } UMTS; \\ error, & \text{sicer.} \end{cases} \tag{2.4}$$

2.3.3 Tušmobil, d.o.o.

Tušmobil je svoje omrežje zagnal 31. oktobra 2007, leto kasneje pa še UMTS 2100 MHz. V Sloveniji je leta 2010 kot prvi ponudil UMTS na frekvenci 900 MHz. Je prvi mobilni operater v Sloveniji, ki je postavil bazno postajo v obliki drevesa (Slika 2.3). Tušmobil gostuje tudi na omrežju Telekoma Slovenije, na njegovem omrežju pa gostuje ponudnik storitev Telemach [10].

Slika 2.3: Tušmobilova bazna postaja "drevo"NE1562 v vasi Grad

Pokritost prebivalstva s signalom na dan 31. december 2011:

- **GSM 900/1800 MHz:** 98,0 % s 480-imi baznimi postajami,
- **UMTS 900/2100 MHz:** 77,5 % z 248-imi baznimi postajami.

Imena baznih postaj

Tušmobil je poimenoval regije malo drugače, in sicer kot je prikazano v Tabeli 2.4. Imena baznih postaj je določil po enačbi (2.5), podobno kot je to storil Telekom Slovenije.

regija	**ime regije**	**Cell ID**
LJ	Ljubljana	x1xxx, x2xxx
MA	Maribor	x3xxx
SW	South West	x4xxx
NE	North East	x5xxx
SE	South East	x6xxx
NW	North West	x7xxx
KO	Koper	x8xxx

Tabela 2.4: Regije Tušmobila s pripadajočimi Cell ID-ji

$$f(x) = \begin{cases} regija + \lfloor x/10 \rfloor, & \text{if } GSM; \\ regija + \lfloor ((x \mod 65535) - 30000)/10 \rfloor, & \text{if } UMTS; \\ error, & \text{sicer.} \end{cases} \tag{2.5}$$

2.3.4 T-2, d.o.o.

Kot najmlajši operater je T-2 1. junija 2008 zagnal omrežje UMTS na frekvenci 2100 MHz. Na začetku je gostoval na omrežju Telekoma Slovenije, od aprila 2012 pa gostuje na omrežju Simobila [10].

Pokritost prebivalstva s signalom na dan 31. december 2011:

- **UMTS 2100 MHz:** 35,0 % s 114-imi baznimi postajami.

Imena baznih postaj

Kot večina operaterjev je tudi T-2 poimenoval regije po večjih krajih v Sloveniji, kot je prikazano v Tabeli 2.5. Imena baznih postaj je določil po enačbi (2.6).

regija	**ime regije**	**Cell ID**
LJ	Ljubljana	1xxxx
MB	Maribor	20xxx, 21xxx, 22xxx, 23xxx, 24xxx
SG	Slovenj Gradec	25xxx, 26xxx
MS	Murska Sobota	27xxx, 28xxx, 29xxx
CE	Celje	3xxxx
KR	Kranj	4xxxx
NM	Novo mesto	5xxxx
KP	Koper	60xxx, 61xxx, 62xxx
GO	Nova Gorica	63xxx, 64xxx

Tabela 2.5: Regije T-2-ja s pripadajočimi Cell ID-ji

$$f(x) = \begin{cases} regija + \lfloor (x \mod 65535)/10 \rfloor, & \text{if } UMTS; \\ error, & \text{sicer.} \end{cases} \tag{2.6}$$

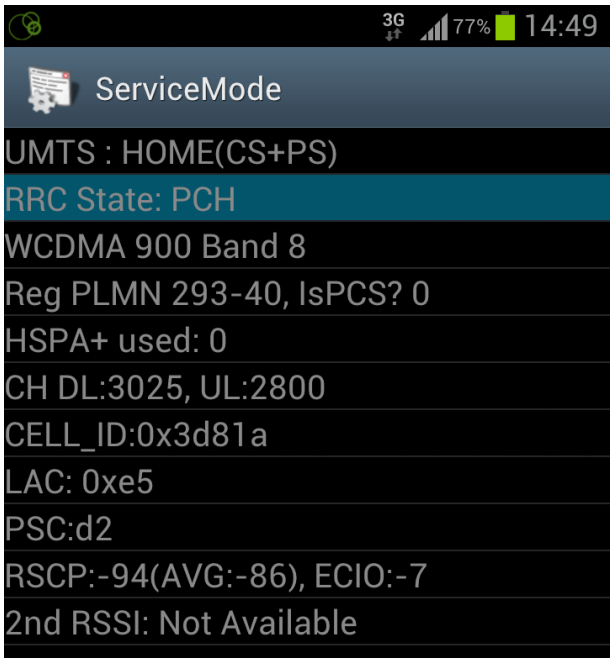
2.4 Zaznava podatkov na mobilnih platformah

Poznamo različne mobilne platforme. Trenutno so med najbolj priljubljenimi Android, iOS in Windows Phone. Vsaka platforma ima različne postopke

pridobivanja podatkov o trenutno prijavljeni bazni postaji. V nadaljevanju bomo opisali postopke pridobivanja teh podatkov.

2.4.1 Android

Android je odprtokodni operacijski sistem za pametne telefone in ostale prenosne naprave, ki ga je razvil Google. Zgrajen je na Linuxovem jedru. Ta operacijski sistem nima posebnega ukaznega niza za pridobitev informacij o telefonu. Nekateri proizvajalci mobilnih telefonov, ki uporabljajo ta operacijski sistem, razvijejo svoj program za pridobivanje teh podatkov. Ponavadi take programe poimenujejo `Field Test`. Pri podjetju Samsung so za pridobivanje podatkov ustvarili ukazni niz `*#0011#` (Slika 2.4).



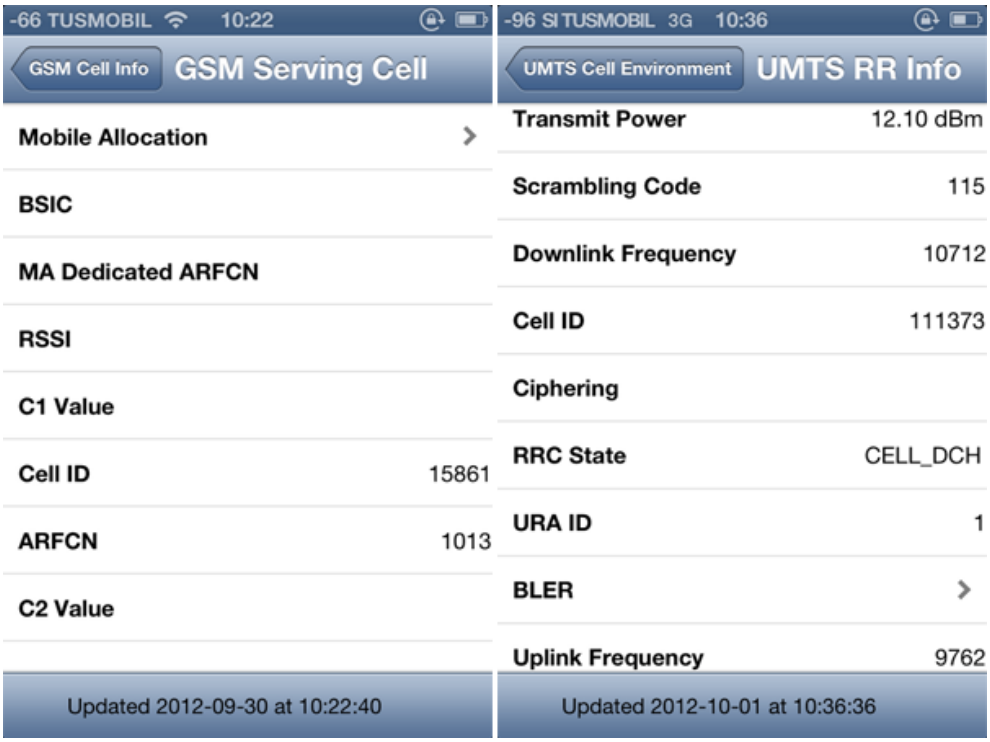
Slika 2.4: Samsungova aplikacija ServiceMode na telefonih Android

Za razvijalce aplikacij Android je Google implementiral knjižnico `Telephony Manager`, ki omogoča pridobitev podatkov o mobilnem telefonu. S pomočjo te knjižnice je mogoče pridobiti vse ključne podatke o prijavljeni bazni postaji

(MCC, MNC, CellID), prav tako je mogoče pridobiti jakost signala.

2.4.2 iOS

Ta operacijski sistem je naložen na telefonih iPhone in tabličnih računalnikih iPad podjetja Apple. Podatke je mogoče pridobiti z ukaznim nizom `*3001#12345#*` (Slika 2.5). Odpre se program `Field Test`. Glavni podatki te aplikacije se nahajajo v menijih MM Info (LAC), UMTS Cell Environment (CellID, UARFCN) in GSM Cell Environment (CellID, ARFCN).



Slika 2.5: Applova aplikacija Field Test na telefonih iPhone in iPad

Slabost operacijskega sistema je, da Apple razvijalcem aplikacij ne omogoča pridobivanja podatkov o telefonu. Prav tako te aplikacije prepoveduje v svoji trgovini iTunes.

2.4.3 Windows Phone

Windows Phone je razvilo podjetje Microsoft. V različici Windows Phone 7 lahko podatke o telefonu pridobimo z ukaznim nizom `##3282#`. Pri tem so pomembni meniji GSM (LAC), GPRS/E-GPRS (CellID, ARFCN) in WCDMA (RncID, CellID, UARFCN).

Za razvijalce aplikacij telefonov različice Windows Phone 7 v uradnih dokumentih ni mogoče najti knjižnice za pridobitev podatkov o telefonu. Za razliko v verziji Windows Mobile 6.5 na spletu obstaja postopek za pridobitev le-teh.

Poglavje 3

Geografsko lociranje

Zadnja leta se zvišuje trend aplikacij na mobilnih telefonih z lokalnimi informacijami in vsebino. Tovrstne aplikacije potrebujejo za pridobitev teh vsebin lokacijo mobilne naprave. Do tega podatka pridemo s pomočjo metod, ki posredno ali neposredno uporabljajo triangulacijo in trilateracijo.

Geografsko lociranje je pomembno tudi pri odkrivanju baznih postaj. S poznavanjem geografske lokacije celic na baznih postajah in hkrati identifikacijo baznih postaj preko celic, lahko zelo natančno določimo lokacije bazne postaje. Za iskanje točne lokacije bazne postaje imamo nato dve možnosti. Lahko vemo, na katerem področju je ali imamo seznam potencialnih baznih postaj. Tu z geografsko lokacijo sevanja celic na bazni postaji lahko ugotovimo ime bazne postaje. Ko poznamo lokacijo bazne postaje, lahko s pomočjo lokacije mobilne naprave izračunamo oddaljenost do bazne postaje.

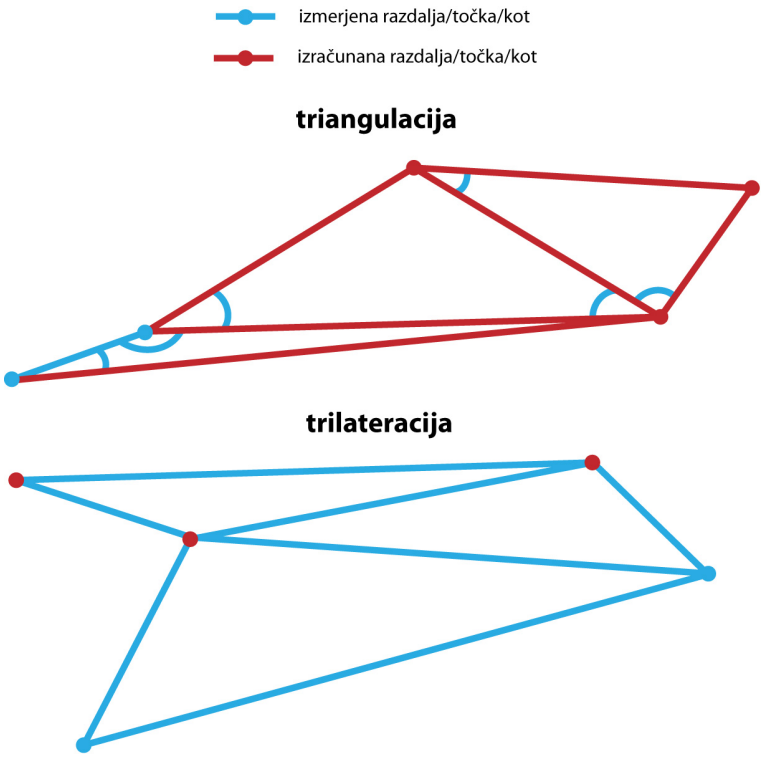
V nadaljevanju si bomo pogledali splošen opis triangulacije in trilateracije ter metode pridobivanja lokacije iz satelitske navigacije in mobilnega omrežja.

3.1 Triangulacija in trilateracija

Triangulacija in trilateracija sta postopka, ki ju uporabljamo za določitev lokacije s pomočjo trikotniških pravil in dveh točk z znanima koordinatama. Triangulacija se uporablja za delo s koti, trilateracija pa delo z razda-

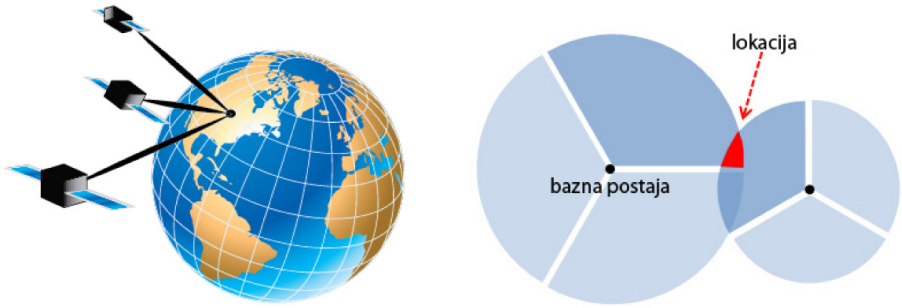
ljami. Kot vidimo na Sliki 3.1, s triangulacijo izračunamo točko in razdaljo s pomočjo kotov, s trilateracijo pa točke izračunamo s pomočjo razdalje med točkami. Postopka se med seboj dopolnjujeta, zato ju v nekateri literaturi med sabo ne ločujejo in ju imenujejo kar triangulacija. Postopka uporabljamo za navigacijo, v naravoslovju, v astrometriji in pri modeliranju smeri vojaških raket [27].

Triangulacija in trilateracija sta zelo pomembni pri definiranju položaja



Slika 3.1: Primer uporabe triangulacije in trilateracije

mobilnega telefona (Slika 3.2). To še nista metodi lokalizacije položaja, ampak le postopka določitve natančnega položaja. Z njuno pomočjo si lahko izračunamo lokacijo pri uporabi satelitske navigacije (Poglavje 3.2) ter metodami na mobilnem omrežju (Poglavje 3.3).



Slika 3.2: Triangulacija in trilateracija se uporabljata tudi pri satelitski navigaciji (levo) in v metodah mobilnega omrežja (desno)

3.2 Satelitska navigacija

Satelitska navigacija je sistem satelitov, ki zagotavlja geografski položaj s svetovno pokritostjo. Sateliti krožijo okoli 20.000 kilometrov visoko nad Zemljo po tirnicah tako, da je vsaka točka na Zemlji pokrita z vsaj tremi sateliti. Za pridobitev podatkov o zemljepisni širini, dolžini, višini ter točnem času sprejemna enota potrebuje signale štirih satelitov. Preko različnih časov prejema in oddaje signala določimo razdaljo med sprejemno enoto in satelitom [26].

Globalni navigacijski sistemi [26]:

- **GPS (*Global Positioning System*):** Je sistem Združenih držav Amerike in je sestavljen iz najmanj 24 satelitov po šestih tirnicah. Prvi satelit so v orbito poslali leta 1978. Sistem GPS je na voljo od leta 1994 (Slika 3.3). Je najbolj razširjena satelitska navigacija na svetu, tudi v Sloveniji.

- **GLONASS (*Global'naya Navigatsionnaya Sputnikovaya Sistema*):** Je sistem Rusije oziroma nekdaj bivše Sovjetske zveze. Leta 1980 so bili zgrajeni prvi sateliti, vendar so bili zaradi tehničnih težav uporabni šele od leta 1982 naprej, ko so jih začeli izstreljevati v orbito. Leta 1995 so z izstrelitvijo štiriindvajsetega satelita omogočili polno na-

Slika 3.3: Satelit GPS [26]

tančnost sistema. Sateliti prve generacije so bili manj zanesljivi, zato je število uporabnih satelitov upadalo in je bilo leta 2001 uporabnih le še šest satelitov. Leta 2011 je bil sistem obnovljen in ponovno delujoč s štiriindvajsetimi sateliti.

- **COMPASS:** Kitajska se je odločila, da svoj regionalni navigacijski sistem Beidou razširi v globalnega do leta 2020 z imenom COMPASS. Po dokončanju sistema naj bi vseboval petintrideset satelitov. Uradno ni znano, koliko satelitov je trenutno v orbiti. Ocenjuje se, da jih je trenutno dvanajst.

- **Galileo:** Je projekt Evropske unije in Evropske vesoljske agencije (ESA). Predstavljen je bil leta 2002. Prvotno je bil zagon sistema s tridesetimi sateliti predviden v letu 2010, vendar je danes predviden šele v letu 2014. Galileo naj bi bil združljiv z ameriškim sistemom GPS. Tako bodo sprejemniki zmožni združiti signale iz Galilea in GPS-a za močno povečanje natančnosti. Ni pričakovati, da bo sistem v polnem

obratovanju do leta 2020.

3.3 Metode na mobilnem omrežju

Za določanje položaja mobilne postaje je razvitih kar nekaj metod, ki temeljijo na različnih principih in lastnostih, bodisi radijskega signala ali samega mobilnega omrežja. Glede na njihove lastnosti jih lahko razvrstimo v različne kategorije [9]:

- **Cell ID:** Je najbolj osnovna metoda, ki določa lokacijo na osnovi podatka identifikacije celice na bazni postaji (Cell ID). Lokacija je določena glede na pozicijo bazne lokacije, ki oddaja to celico ter območje, ki ga pokriva le-ta. Natančnost je odvisna od velikosti pokrivanja celice in se giblje med sto metri in nekaj tisoč metri.

- **Timing Advance (TA):** S pomočjo te metode pridobimo čas TA, ki ga potrebujemo ob pošiljanju iz mobilne naprave na bazno postajo. Vrednost TA določa, koliko prej mora mobilna naprava poslati podatke bazni postaji, da jih bo ta sprejela. Nove vrednosti TA se izračunajo sproti. Omogoča sinhronizacijo mobilne naprave do 35 kilometrov daleč od bazne postaje. Z vrednostjo TA lahko določimo, koliko je mobilna naprava oddaljena od trenutno prijavljene bazne postaje.

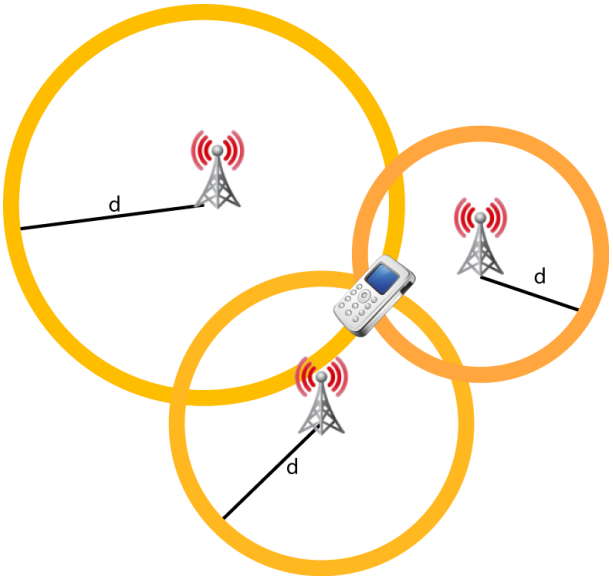
- **Time Of Arrival (TOA):** Metoda temelji na omrežju, ki ugotavlja lokacijo mobilne naprave s časom prihoda signala t na bazno postajo. S produktom časa in hitrosti širjenja radijskega valovanja c lahko izračunamo razdaljo med mobilno napravo in bazno postajo. Kot rezultat dobimo krožnico z radijem d okoli bazne postaje. Če vsaj tri bazne postaje prejmejo signal iz mobilne naprave, lahko s presečiščem treh krožnic izračunamo točno lokacijo (Slika 3.4).

- **Time Difference Of Arrival (TDOA):** Metoda je zelo podobna metodi TOA. Metoda TOA uporablja absolutne čase prihodov signala iz

mobilne naprave na bazne postaje, medtem ko metoda TDOA uporablja razlike v časih prihodov med pari baznih postaj. Vsak par baznih postaj ima za rezultat hiperbolo z gorišči v baznih postajah. Za določanje natančnejšega položaja sta potrebni dve hiperboli, torej tri bazne postaje.

- **Angle of Arrival (AOA):** Metoda izračuna kot, pod katerim bazna postaja pošilja in sprejema signale iz mobilne naprave. Potrebne so bazne postaje, ki oddajajo signal na ozkem območju. Za natančno lokacijo sta potrebni vsaj dve bazni postaji. Problem te metode je, da je vidljivost med bazno postajo in mobilno napravo nujna.

- **Sprejeta moč radijskega signala:** Ena od možnosti je tudi uporaba metod, ki slonijo na merjenju moči radijskega signala. Za določitev lokacije je potrebna meritev jakosti signala mobilne naprave, kot jih vidijo bazne postaje. Iz teh podatkov se izračuna lokacija mobilne naprave.



Slika 3.4: Time Of Arrival (TOA)

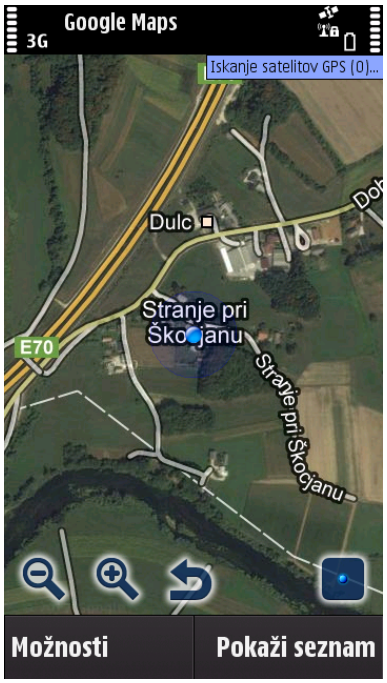
Problem teh metod je, da jih lahko uporabijo le lastniki mobilnega omrežja. Le oni vedo točne lokacije baznih postaj in konfiguracijo vsake celice na bazni postaji. Lokacija mobilne naprave torej ni javno dostopna, vendar obstajajo izjeme. Na primer, center za sprejem klicev v sili 112 avtomatsko pridobi poleg klicateljeve telefonske številke tudi lokacijo s pomočjo navedenih metod.

3.4 Baze mobilnih omrežij

Uporabniki imajo navadno na svojih mobilnih napravah izključeno satelitsko navigacijo (na primer GPS), ker ta ob vključitvi na mobilni napravi porabi veliko baterije in ne deluje v zaprtih prostorih. Potrebno je bilo najti metodo, ki bo pridobila lokacijo naprave z uporabo mobilnega omrežja brez točnega poznavanja konfiguracije tega omrežja in dovoljenja lastnika. Nekatera podjetja so problem rešila v bazah mobilnih omrežij. V te baze se shranjujejo podatki o celici na bazni postaji (MCC, MNC, Cell ID, LAC) ter geografska lokacija, kjer je bila ta zaznana. Medtem ko mobilna naprava želi pridobiti lokacijo iz teh baz, ta vrne izračunano geografsko lokacijo iz že pridobljenih podatkov. Nekatere baze vrnejo tudi natančnost lokacije. Najbolj sta v uporabi baza podjetja Google in odprto kodna baza Open Cell ID. Podatki iz baze niso tako natančni kot tisti iz satelitske navigacije, so pa dovolj natančni za prikaz določenih lokalnih vsebin (na primer prikaz najbližje bencinske postaje, prikaz prireditev v vašem kraju ...).

3.4.1 Google

Google je bazo mobilnih omrežij ustvaril zaradi svojega operacijskega sistema Android ter njegovih aplikacij (na primer Google Maps, ki je prikazan na Sliki 3.5). S to bazo lahko lokalizira podatke in da svojim aplikacijam dodano vrednost. Poleg zbiranja lokacij celic na baznih postajah baza hrani tudi lokacije brezžičnih omrežij Wi-Fi. Baza je namenjena le storitvam podjetja Google in ni na voljo drugim. Za svoje aplikacije Android je možno pridobiti podatke iz te baze s pomočjo razreda `LocationManager`.



Slika 3.5: Aplikacija Google Maps na telefonu Nokia N97

Ker Google ni nikoli uradno objavil načina zbiranja podatkov v to bazo, o tem ni znano veliko. Predvideva se, da se podatki zbirajo s pomočjo sistemskih podatkov iz telefona in sistema GPS. Branje podatkov poteka preko URL naslova `http://www.google.com/glm/mmap`. Po preteklih izkušnjah se lokacija nove celice na bazni postaji ponavadi prikaže teden kasneje, kot je bila bazna postaja vključena. Težava se pojavlja tudi na območjih, kjer država meji z drugimi državami, ker se mobilnim napravam, ki so priklopljene na te celice na baznih postajah ne prikaže lokacija. To pa zato, ker Google ne dovoli lokacij celic baznih postaj slovenskih operaterjev izven Slovenije.

3.4.2 Open Cell ID

To je odprtokodni projekt, katerega cilj je ustvariti popolno zbirko Cell ID-jev s svojimi lokacijami. Vsi podatki tega projekta so brezplačno dostopni preko uvoza v Google Earth in že vnaprej definiranih API-jih. Za dodajanje

podatkov je nujen uporabniški ključ `apikey`, ki ga pridobimo z brezplačno registracijo [11].

Branje

Za branje lokacije potrebujemo svoj uporabniški ključ, informacijo celice (MCC, MNC, CellID, LAC). Kličemo naslov URL `http://www.opencellid.org/cell/get` s parametri oblike GET. Vrnjeni podatki z lokacijo so v obliki formata XML. Vrnjena lokacija je približna in je izračunana s povprečjem vseh vnesenih podatkov o izbrani celici. Na primer ob klicu naslova `http://www.opencellid.org/cell/get?key=&mcc=293&mnc=70&cellid=6321&lac=60` pridobimo podatke kot prikazuje Koda 3.1 [11].

Koda 3.1: Primer vrnjene kode ob klicu danega naslova URL

```
1 <rsp stat="ok">
2         <cell lat="45.881274" mcc="293" lon="15.30815" cellId
              ="6321" nbSamples="1" mnc="70" lac="60" range="
              6000"/>
3 </rsp>
```

Dodajanje

za dodajanje znane lokacije za določeno celico na bazni postaji so ključni naslednji podatki: uporabniški ključ, informacija celice (MCC, MNC, Cell ID, LAC) ter lokacija (Latitude, Longitude). Pošljemo jih na naslov URL `http://www.opencellid.org/measure/add`. Kot odgovor na uspešno poslane podatke prejmemo *ok* ali *nok*. Primer poslanega naslova URL za dodajanje podatka v bazo: `http://www.opencellid.org/measure/add?key=myapikey&mnc=1&mcc=2&lac=200&cellid=234&lat=3.42&lon=3.12` [11].

Poglavje 4

Platforma Android in spletne tehnologije

V tem poglavju bomo predstavili operacijski sistem Android in Android SDK ter opisali vsa potrebna orodja, ki jih potrebujemo za razvoj aplikacije Android. Najprej bomo predstavili spletne tehnologije, ki jih potrebujemo pri izgradnji stila spletne arhitekture REST ter razvojna orodja Android SDK in Eclipse.

4.1 Spletne tehnologije

4.1.1 REST

REST (*Representational State Transfer*) je stil spletne arhitekture, ki je bil prvič predstavljen v doktorski disertaciji Roya Fieldinga leta 2000. Je enostavnejša alternativa razvoja SOAP/WSDL spletnih storitev. Je neodvisen od platforme in izkorišča protokol HTTP in tipe sporočil HTTP (GET, POST, PUT, DELETE). REST je brez stanja in temelji na komunikaciji med strežnikom in odjemalcem. Uporablja se za razvoj, omogočanje in uporabo spletnih storitev. Odjemalec sproži zahtevo na spletni strežnik. Ta vrne ustrezen odgovor v že znanem formatu [25].

Spletno storitev, ki uporablja pristope REST, imenujemo RESTful. Za njo so potrebni [25]:

- Identifikator URI spletne storitve (`http://primer.com/resources/`).
- Označevalni jezik za prenašanje podatkov med spletno storitvijo. Ponavadi je to jezik XML, vendar je lahko uporabljen tudi drug format (na primer JSON).
- Niz operacij spletnih storitev, ki uporabljajo metode HTTP (GET, POST, PUT, DELETE).

4.1.2 PHP

PHP je skriptni programski jezik na spletnih strežnikih, namenjen izdelavi dinamičnih spletnih strani. Je prvi skriptni jezik, ki omogoča vstavljanje kode v vir HTML. S kodo PHP razpolaga spletni strežnik, ki jo pretvori v spletno stran. Začetki tega jezika segajo v leto 1994, ko je programer Rasmus Lerdorf ustvaril paket skript Perl, imenovanih `Personal Home Page Tools` za prikaz svojega življenjepisa in prikaz prometa na spletni strani. Te skripte je predelal v skupine CGI programov, razvitih v programskem jeziku C in jim dodal možnost izdelave spletnih obrazcev ter omogočil komunikacijo z bazo. Te je izdal pod imenom `Personal Home Page/Form Interpreter (PHP/FI)` leta 1995. Do sedaj je bilo izdanih že pet glavnih različic, zadnja leta 2004. Trenutna verzija ima oznako 5.4.7. Verzija 6.0 je bila predvidena za leto 2010, vendar je zaradi iskanja boljšega načina obdelave podatkov Unicode preložena. Za izdelavo dinamičnih spletnih strani, poleg skriptnega jezika PHP, obstajajo še jeziki ASP, VBScript, JScript in JSP [24].

4.1.3 XML

Extensible Markup Language (XML) je označevalni jezik, ki definira seznam pravil, zakodiranih v dokumentu, in je lahko berljiv tako človeku kot stroju

(računalniku). S tem jezikom je mogoče opisati strukturirane podatke ali arhitekturo za prenos podatkov in njihovo izmenjavo med omrežji. Pomembna lastnost tega jezika je, da popolnoma podpira Unicode za podporo vseh jezikov sveta. Razvila ga je organizacija W3C, ki je mednarodni inštitut za razvoj spletnih standardov. Mnogo programskih vmesnikov (API) je bilo razvitih za procesiranje podatkov XML. Za XML so pomembne tudi sheme XML, ki natančno opišejo, kako naj bo sestavljen dokument XML. Razvitih je bilo mnogo jezikov, baziranih na jeziku XML, kot so RSS, Atom, SOAP in XHTML. XML najdemo tudi v komunikacijskih protokolih, na primer v protokolu XMPP [28].

4.1.4 JSON

JavaScript Object Notation (JSON) je format za izmenjavo podatkov, podoben jeziku XML. Namenjen je prikazu preprostih podatkovnih struktur in objektov. Izpeljan je iz skriptnega jezika JavaScript. Programski jezik je razvil Douglas Crockford. JSON je naprej začelo uporabljati podjetje Crockforda leta 2001, leto kasneje je Crockford izdal še spletno stran `json.org`. Yahoo je leta 2005 začel uporabljati jezik za nekatere njihove spletne storitve. Prav tako ga je Google vključil v svoj spletni protokol GData leta 2006. Čeprav je jezik JSON izpeljan iz jezika JavaScript in je največkrat uporabljen v tem jeziku, je neodvisen format. Razčlenjevanje in ustvarjanje kode JSON je na voljo mnogim programskim jezikom. Format JSON se pogosto uporablja za prenos strukturiranih podatkov preko omrežne povezave. Uporablja se predvsem za prenos podatkov med strežnikom in spletnimi aplikacijami ter služi kot alternativa XML-ju [22].

4.2 Operacijski sistem Android

Android je odprtokodni operacijski sistem, zgrajen na Linuxovem jedru, namenjen predvsem pametnim telefonom in tabličnim računalnikom. Razvit je bil s strani podjetja Google, ki je v ta namen ustanovilo poslovno združenje

Slika 4.1: Uradni logo operacijskega sistema Android [17]

več podjetij, imenovano Open Handset Alliance. To združenje je bilo ustanovljeno in predstavljeno 5. novembra 2007 z uradnim logotipom prikazanim na Sliki 4.1. Njihov prvi izdelek je bil telefon HTC Dream izdan 22. oktobra 2008. Danes ima Android nekaj čez 60 % tržnega deleža pri prodaji pametnih telefonov. Njihov operacijski sistem Android danes uporabljajo številni proizvajalci mobilnih telefonov, med njimi Samsung, HTC, Motorola, LG in Sony. Vsak dan je na novo aktiviranih približno 1,3 milijona naprav Android. Mobilne aplikacije je mogoče prenašati prek spletne trgovine Google Play. Do oktobra 2012 je bilo prenesenih 25 milijard mobilnih aplikacij. Nekatere izmed njih so brezplačne, druge žal plačljive [17].

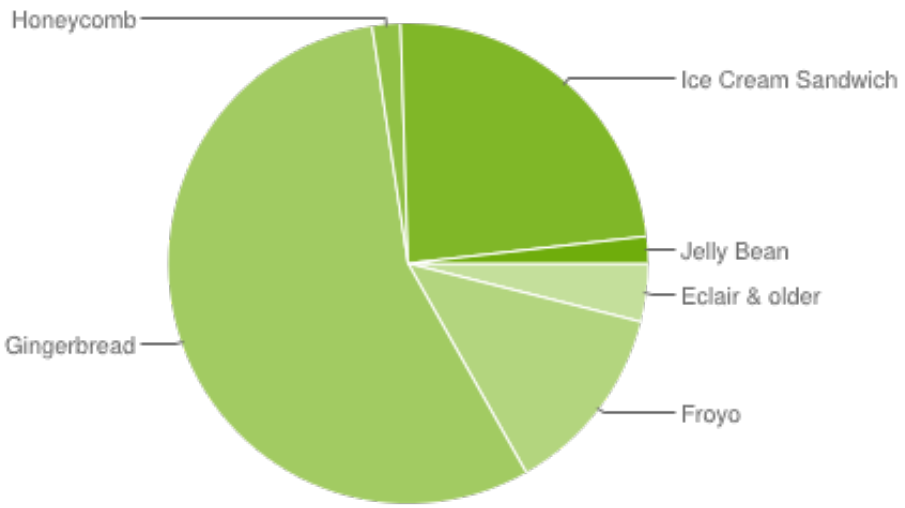
Prednosti:

- odprtokodni operacijski sistem omogoča cenejše in lažje razvijanje programov, posledično so programi za ta operacijski sistem večinoma brezplačni,
- proizvajalcem strojne opreme ni več potrebno razvijati svojega operacijskega sistema, ampak razvijajo le posamezne komponente sistema,
- operacijski sistem podpira večopravilnost (*Multitasking*), z vsako novo verzijo je enostavnejši in odzivnejši,

- samodejna sinhronizacija z Googlovimi storitvami,
- aplikacije so pogosto razvite v jeziku Java, Native Development Kit pa podpira razvoj tudi v drugih jezikih (na primer C in C++),
- projekt Nexus, kjer Google sam poskrbi za implementacijo operacijskega sistema na mobilne naprave.

Slabosti:

- fragmentacije različic operacijskega sistema, ker proizvajalci telefonov ne nadgrajujejo svojih telefonov na novejše verzije,
- proizvajalci strojne opreme s svojimi dodatnimi programi upočasnjujejo delovanje operacijskega sistema.



Slika 4.2: Tržni delež različic operacijskega sistema Android na dan 2. oktober 2012 [17]

Zadnjo različico Android 4.2 Jelly Bean je Google naznanil 29. oktobra 2012. Ta ponuja povsem novo izkušnjo z digitalnim fotoaparatom, prenovljen in učinkovit način pisanja sporočil ter daljšo avtonomijo delovanja. Zgrajena

je na jedru Linux 3.0. Prejšnje različice, naštete od starejše proti novejši, so [17]:

- 1.5 Cupcake (30. april 2009)
- 1.6 Donut (15. september 2009)
- 2.0, 2.1 Eclair (26. oktober 2009)
- 2.2 Froyo (20. maj 2010)
- 2.3 Gingerbread (6. december 2010)
- 3.0, 3.1, 3.2 Honeycomb (22. februar 2011)
- 4.0 Ice Cream Sandwich (19. oktober 2011)
- 4.1 Jelly Bean (27. junij 2012)
- 4.2 Jelly Bean (29. oktober 2012)

Na Sliki 4.2 je prikazana uporaba različic operacijskega sistema Android.

4.2.1 Arhitektura operacijska sistema

Operacijski sistem Android je sestavljen iz petih elementov (Slika 4.3) [17]:

- **Aplikacije:** Aplikacije so napisane s programskima jezikoma Java in XML. Končni projekt se izvozi v izvršljivo datoteko `.apk`, ki je namenjena nameščanju aplikacij Android. Posamezna aplikacija se zažene v svojem Linux procesu. Vsak proces se prevede posebej, kar omogoči izoliranje aplikacij, da delujejo med seboj neodvisno.
- **Aplikacijsko ogrodje:** Sestavljeno je iz sistemskih aplikacij, ki podpirajo koordiniranje aplikacij (upravljavec aktivnosti, pomnilnika, lokacij, obvestil).
- **Knjižnice:** Na voljo so razvijalcu Android aplikacij za dostop do strojnih komponent naprave.

Slika 4.3: Arhitektura operacijskega sistema Android [17]

- **Prevajalnik:** Operacijski sistem Android za prevajanje kod aplikacij uporablja prevajalnik `Just in time compiler` (JIT), kar omogoča prenos aplikacij na različne prenosne naprave, brez spreminjanja izvorne kode.

- **Jedro Linux:** Do verzije Android 4.0 se je uporabljala verzija 2.6. Za novejše verzije pa se uporablja verzija 3.

4.2.2 Java

James Gosling s sodelavci je v podjetju Sun Microsystems razvil objektno usmerjeni programski jezik Java. Prva različica je bila izdana leta 1995. Danes programski jezik Java spada pod okrilje podjetja Oracle Corporaction,

zaradi prevzema podjetja Sun Microsystems leta 2010. Ustvarjen je bil kot alternativa programskima jezikoma C in C++. Java aplikacije se večinoma prevedejo v binarno obliko in delujejo na Java Virtual Machine (JVM), ne glede na arhitekturo računalnika [21].

Podjetje Google je za programiranje aplikacij v operacijskem sistemu Android zaradi priljubljenosti programskega jezika in dostopnosti na vseh platformah izbral programski jezik Java. Lastnik jezika Oracle je leta 2010 tožil Google zaradi kršitev avtorskih pravic in patentov, povezanih s programskim jezikom Java, za 6,1 milijard dolarjev. Maja 2012 je sodišče odločilo, da Google ni kršil patentov podjetja Oracle in s tem tožbo zavrnilo [17].

4.3 Android SDK

Programsko razvojno orodje Android vključuje celovit nabor orodij za razvoj. Vključuje iskanje in zaznavanje napak, knjižnice, emulator, dokumentacijo, že izdelane primere in navodila za razvoj. Android SDK je mogoče namestiti na operacijske sisteme Windows, Mac OS X in Linux. Ob namestitvi razvojnega orodja dobimo nameščena dva programa: Android SDK Manager in Android AVD Manager. S prvim prenašamo modularne pakete, ki jih potrebujemo za izbrano različico platforme Android. To so na primer SDK Tools, dokumentacija, slika sistema, primeri kode [5]. Pomembno je omeniti, da lahko prenašamo vse različice od Androida 1.5 naprej. Drugi program, imenovan Android Virtual Device Manager, omogoča prikaz operacijskega sistema Android na emulatorju. Seveda le tiste različice, ki jih prenesemo preko Android SDK Managerja [18].

4.3.1 Datoteka Manifest

Vsaka aplikacija Android mora imeti datoteko `Manifest.xml` v korenskem imeniku. V datoteki so predstavljeni podatki, ki jih mora imeti sistem Android pred zagonom aplikacije. Ti so predstavljeni v formatu XML. Med drugim so v njem navedeni [3]:

- Ime paketa, ki služi kot enoličen identifikator aplikacije.
- Komponente aplikacije: aktivnosti, storitve, sprejemnik sporočil znotraj aplikacije. Navedeni so razredi, ki točno določajo izvajanje izbrane komponente aplikacije. Taka deklaracija da sistemu vedeti, katere komponente ima aplikacija in pod katerimi pogoji jih lahko izvedemo.
- Procesi, ki bodo gostovali na programski komponenti.
- Dovoljenja za dostop do sistemskih knjižnic Androida, ki so potrebna za izvajanje aplikacije.
- Seznam instrumentov za razrede omogoča izdelavo profilov in drugih informacij med izvajanjem aplikacije. Te so navedene le takrat, ko aplikacijo razvijamo in jo testiramo.
- Najmanjša verzija Androida, ki je s strani aplikacije še podprta.
- Seznam knjižnic, ki jih aplikacija potrebuje za izvajanje.

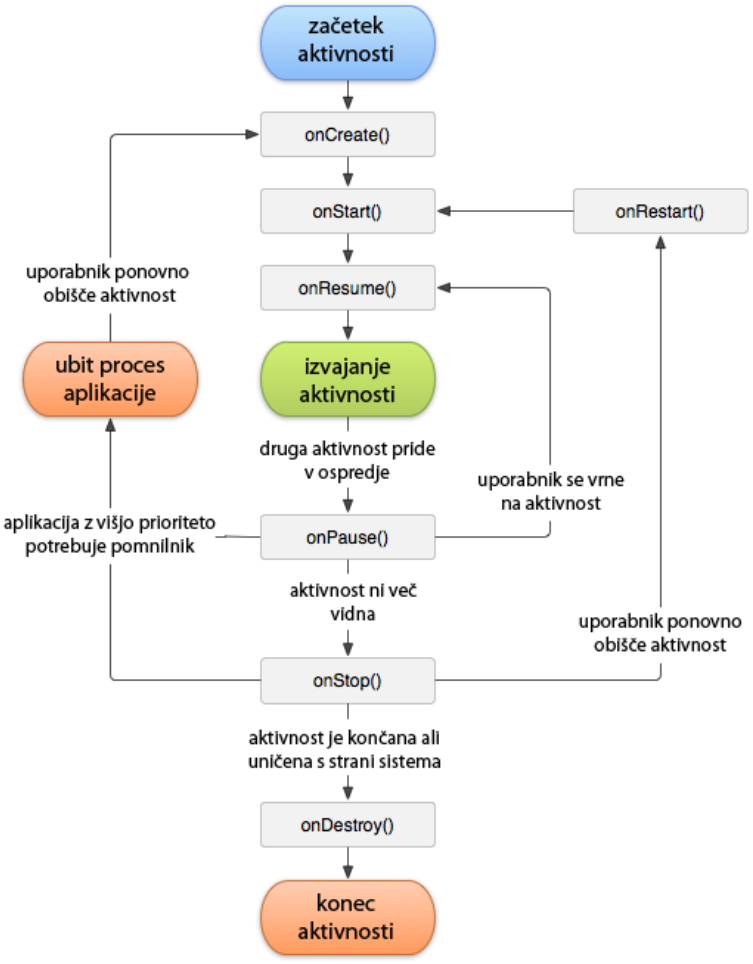
Poleg datoteke Manifest mora imeti vsak projekt aplikacije Android še vključene datoteke:

- `src`: koda aplikacije napisana v jeziku Java,
- `gen`: koda napisana v Javi, ki jo ustvari Android SDK,
- `asset`: hrani začasne datoteke, ki so potrebne pri izvajanju aplikacije,
- `bin`: Android SDK v to mapo ustvari datoteke `.class` in `.apk`,
- `res`: potrebne datoteke za aplikacijo: slike, besedila, meniji, postavitve.

4.3.2 Delovanje aplikacije

Aktivnost (*Activity*) je komponenta aplikacije za zagotavljanje grafičnega vmesnika, na katerem lahko uporabniki izvajajo interakcije (klicanje, fotografiranje, pregled zemljevidov ...). Za vsako aktivnost je treba ustvariti

grafični vmesnik, tako da se aktivnost prilega celotnemu zaslonu. Aplikacija je običajno sestavljena iz številnih aktivnosti, ki so med seboj slabo povezane. Vedno obstaja glavna aktivnost, ki se zažene ob zagonu aplikacije in je definirana v datoteki Manifest. Vsaka aktivnost lahko zažene novo aktivnost, s tem pa doda staro aktivnost na sklad in jo ustavi. Ko se nova aktivnost zapre, se iz sklada obnovi stara aktivnost po sistemu LIFO (*Last in, First out*) [3].



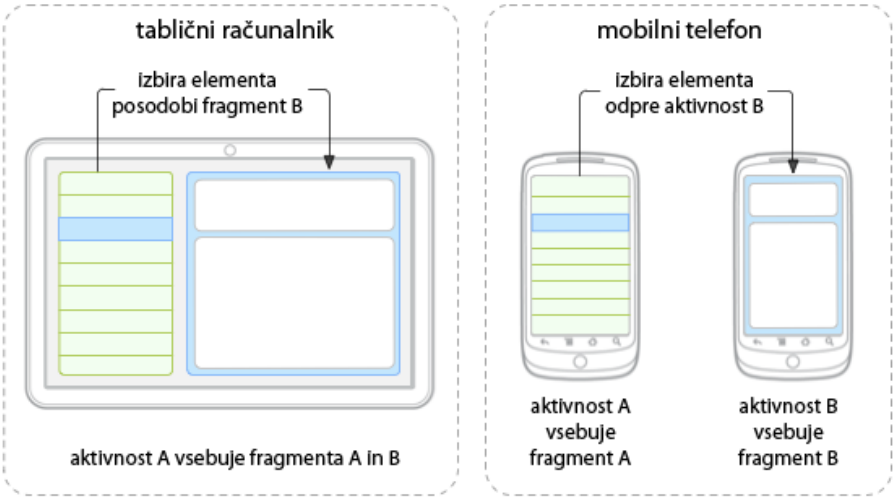
Slika 4.4: Življenjski cikel aktivnosti [3]

Vsaka aktivnost v aplikaciji ima svoj življenjski cikel, kot je prikazano na Sliki 4.4. Če se izvede sprememba stanja ob kreiranju, zaustavljanju, nadaljevanju ali uničenju aktivnosti, se izvedejo določene metode, ki zagotavljajo izvedbo kode med prehajanji stanj. Med zaustavljanjem aktivnosti se na primer lahko prekine mrežna ali povezava z bazo, ob obnovitvi pa ponovna vzpostavitev le-teh. Za kreiranje aktivnosti je potrebno implementirati razred in ga razširiti z razredom `Activity`. V njem moramo nujno implementirati nekatere metode, kot je na primer `onCreate()`, v kateri se izvede ustvarjanje aktivnosti. Po tej metodi se izvedeta metodi `onStart` in `onResume()` in tako se aktivnost lahko prične izvajati. Ko pride druga aktivnost v ospredje, se izvede metoda `onPause()`. Če se uporabnik vrne v to aktivnost, se izvede metoda `onResume()`, če aktivnost ni več na voljo, pa metoda `onStop()`. Ko uporabnik ponovno odpre aplikacijo v stanju `onStop()`, se izvede metoda `onRestart()` ter nato še `onStart()`. Če druga aplikacija z višjo prioriteto potrebuje pomnilnik, se pri metodah `onPause()` in `onStop()` celotna aplikacija zapre. Ob uporabnikovem ponovnem odprtju aplikacije se izvede metoda `onCreate()`. Če se aktivnost konča, se poleg metode `onStop()` izvede še metoda `onDestroy()`. S tem se aktivnost konča [3].

Fragmenti

Fragmenti (*Fragments*) so se pojavili v različici Android 3.0 in omogočajo večdelni uporabniški vmesnik. Fragmenti niso nič drugega kot moduli aktivnosti in imajo vsak svoj življenjski cikel. Fragmenti morajo biti vedno vključeni v aktivnost. Življenjski cikel aktivnosti neposredno vpliva na življenjski cikel fragmentov. Na primer, ko je aktivnost začasno zaustavljena, so začasno zaustavljeni tudi vsi fragmenti aktivnosti. Prav tako se ob uničenju aktivnosti uničijo vsi fragmenti. Kljub temu da aktivnost poteka, lahko z vsakim fragmentom manipuliramo in jih dodajamo ali uničimo. Največja prednost fragmentov je v različnem oblikovanju grafičnega vmesnika za različne velikosti ali položaj zaslonov. Kot prikazuje Slika 4.5 lahko za telefone in tablične računalnike na drugi strani prikažemo drugačno grafično podobo.

Na tabličnem računalniku ob kliku na meni (fragment A) se odpre zahtevan fragment B na desni strani. Na mobilnem telefonu pa se ob kliku na izbran meni (fragment A) odpre v drugem oknu in s tem se zažene nova aktivnost s fragmentom B [3].



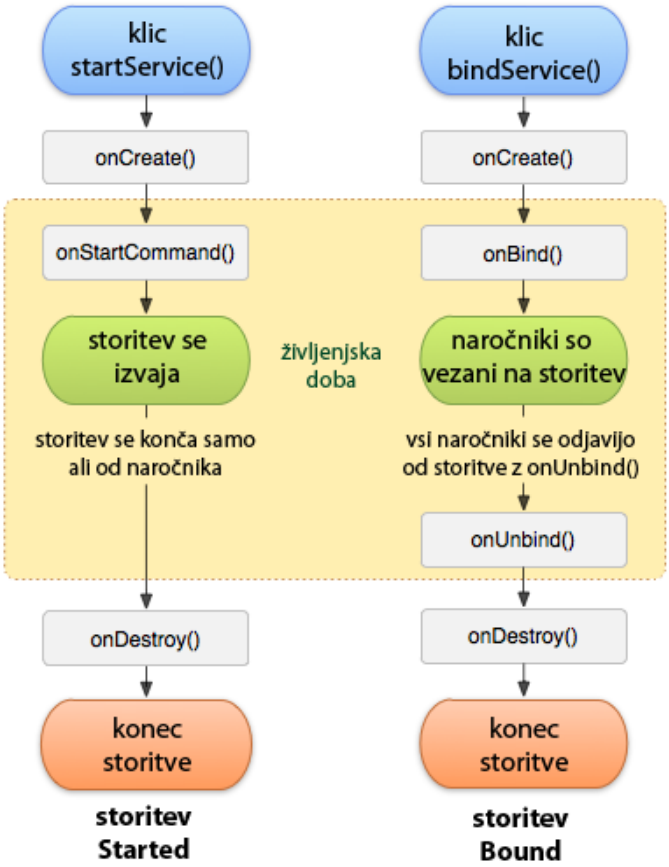
Slika 4.5: Primer prikaza aktivnosti s pomočjo fragmentov [3]

4.3.3 Delovanje pripomočka

Pripomočki (*App Widgets*) so miniaturne aplikacije, ki so lahko prikazane v drugih aplikacijah (na primer začetni zaslon telefona) in sprejemajo posodobitve v intervalih. Pomembno je omeniti, da mora biti interval posodobitve najmanj trideset minut. Tudi pripomočke je potrebno obvezno navesti v datoteki Manifest. Za prikaz pripomočka ga je potrebno grafično oblikovati, definirati mere in oblikovati razred, ki je razširljiv iz razreda `AppWidgetProvider`. V tem razredu je najbolj pomembna metoda `onUpdate()`, ki glede na interval posodobi pripomoček. Priporočljivo je, da se posodobitve pripomočkov izvedejo prek storitev [3].

Storitve

Storitev (*Service*) je komponenta aplikacije za zagotavljanje dolgo izvajajoče operacije, medtem ko uporabniškega oziroma grafičnega vmesnika ne omogoča. Druga komponenta aplikacije lahko zažene storitev in je zagnana v ozadju tudi, če uporabnik uporablja drugo aplikacijo. Poleg tega zagotavlja medprocesno komunikacijo (*interprocess communication*). Na primer storitev lahko ureja transakcije omrežja, predvaja glasbo, zapisuje ali bere datoteke iz ozadja [3].



Slika 4.6: Življenjski cikel storitve [3]

Poznamo dva tipa storitev:

- **Started:** Je storitev, ki jo komponenta aplikacije (aktivnost, pripomoček) zažene s klicem `startService()`. Storitev se v ozadju lahko izvaja nedoločen čas, čeprav je element, ki ga je začel, uničen. Po končani storitvi se ta samodejno uniči.

- **Bound:** Storitev zaženemo s klicem `bindService()`. Ta zagotavlja vmesnik med klientom in strežnikom, ki dovoljuje komponentam iteracijo s storitvijo, pošiljanje zahtev, pridobitev rezultatov. Storitev je lahko le enkrat vezana na komponento aplikacije. Ko se vse komponente izklopijo od storitve, se ta samodejno uniči.

Življenjski cikel obeh vrst tipov storitev si lahko ogledate na Sliki 4.6.

4.3.4 Lokalizacija

Operacijski sistem Android podpira mnogo naprav in prav tako podpira mnogo regij. Za doseganje večjega števila uporabnikov moramo razvijalci aplikacij besedilo, zvok, številke, valute in grafično oblikovanje čim bolj lokalizirati, kjer bo aplikacija uporabljena. V projektu Android obstaja datoteka `res`, kjer so shranjene slike (datoteka `drawable`), postavitve (datoteka `layout`), meniji (datoteka `menu`) in besedila (datoteka `values`). Vse te vrednosti lahko lokaliziramo glede na jezik, postavitev in velikost. Tako na primer pri telefonih, kjer je izbran slovenski jezik, se poleg datoteke `values` ustvari datoteka `values-sl`. V primeru, da telefon uporablja angleški ali katerikoli drug jezik, pa je jezik aplikacije privzeto angleški. Besedila, ki so shranjena v datoteki `values-sl`, bodo uporabljena, če bo telefon prikazoval slovenski jezik, sicer bo aplikacija pridobila besedilo iz osnovne datoteke. V osnovni datoteki morajo biti vsa besedila, ki so uporabljena v aplikaciji, v ostalih datotekah pa to ni pogoj. V primeru, da besedilo ne obstaja v datoteki `values-sl`, se ta prebere iz datoteke `values` [3].

4.3.5 Branje in pisanje podatkov za aplikacijo

Podatki aplikacije se lahko shranjujejo na dva načina. Prvi način je z razredom `SharedPreferences`, ki omogoča zapis globalnih nastavitev aplikacije. Razvijalca aplikacije ne zanima, kje so shranjene nastavitve. Potrebno je le vedeti identifikacijsko ime nastavitve, ki jo želimo spremeniti ali prebrati. Drugi način je splošen in omogoča branje ter pisanje datotek na notranji ali razširljiv pomnilnik telefona. Za branje datotek se uporablja standardni javanski razred `FileReader`, za pisanje pa `FileWriter` [3].

4.3.6 Dodatek Google APIs

Google APIs je dodatek razvojnega okolja Android SDK. Ta omogoča razvoj aplikacij, ki vključujejo nabor aplikacij, knjižnic in storitev podjetja Google. Najpomembnejša je zunanja knjižnica za zemljevide, ki omogoča prikaz zemljevidov na razvijalčevi aplikaciji Android [6].

Dodatek Google APIs vključuje [6]:

- zunanjo knjižnico za zemljevide (Google Maps Android API),
- knjižnico za upravljanje s priključkom USB,
- slika sistema Android z dodatnimi knjižnicami Google za emulator,
- demo aplikacije,
- dokumentacijo za zemljevide.

Google Maps Android API

Dodatna knjižnica `com.google.android.maps` za zemljevide je namenjena lažjemu oblikovanju aplikacij z zemljevidi. S pomočjo razreda `MapView` je mogoče prikazati podatke in upravljavska orodja na zemljevidu, premikanje ter povečavo zemljevida in dodajanje točk. Ta razred ne dela nič drugega kot manipulira s podatki, v okviru razrednih metod in omogoča delo s podatki iz

zemljevida. Pri uporabi zemljevidov za namen aplikacije Android je potreben API ključ, ki je odvisen od certifikata aplikacije [6].

4.3.7 Android Developer Tools (ADT) za Eclipse

Pri razvoju aplikacije Android je zelo pomemben ADT vtičnik za Eclipse. Ta nam omogoča integracijo Android SDK s programom Eclipse, kar pripomore k razvijanju aplikacij Android. Vtičnik ADT nam omogoča grafični dostop do številnih orodij, ki bi jih sicer morali vnašati preko ukazne vrstice orodij SDK. Z njegovo pomočjo je lažje oblikovanje grafičnega vmesnika in omogoča hitro izdelavo prototipov [2].

Glavne prednosti uporabe vtičnika ADT [2]:

- **razvoj aplikacije v Eclipsu:** Vtičnik ADT omogoča, da se ob kreiranju novega projekta Android v Eclipsu zgradi projekt z vsemi potrebnimi datotekami. Prav tako ponuja prevajanje, namestitve in razhroščevanje nove aplikacije.

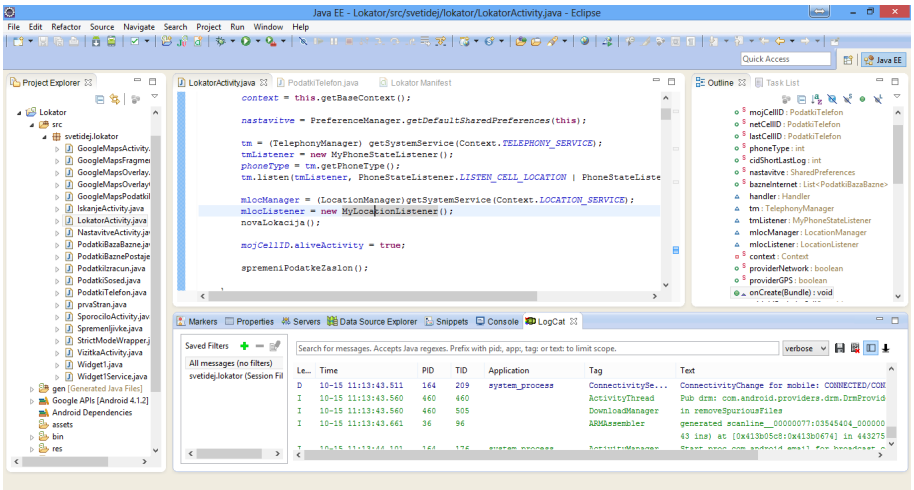
- **integracija pripomočkov SDK:** SDK Tools integrirajo različna orodja v Eclipse. V program se dodajo bližnjice SDK in AVD Managerja, ki na primer omogočajo pogled `LogCat` za pomoč pri razhroščevanju.

- **programski jezik Java in urejevalnik za XML**: Urejevalnik Java že vključuje skupne značilnosti IDE, kot so preverjanje sintakse, integracija dokumentacije, prevajanje kode. ADT zagotavlja poseben urejevalnik XML z možnostjo pretvorbe datotek XML v grafično obliko. Prav tako podpira grafično grajenje datotek XML z vmesnikom povleci in spusti.

- **vgrajena pomoč za knjižnice Android**: Omogočen pregled dokumentacije s premikom miške na razred, metodo ali spremenljivko.

4.4 Eclipse

Eclipse je razvojno orodje, ki vsebuje integrirano razvojno okolje (IDE) in podpira razširljive vtičnike. Okolje je narejeno predvsem za programiranje v Javi, vendar s pomočjo vtičnikov lahko programiramo v jezikih Ada, C, C++, COBOL, Fortran, haskell, Perl, PHP, Phyton, R, Ruby, Scala, Clojure, Groovy, Android. Projekt Eclipse je leta 2001 ustanovilo podjetje IBM. Ker je projekt podprl konzorcij ponudnikov programske opreme, so leta 2004 ustanovili organizacijo Eclipse Foundation, ki skrbi za nadaljnji razvoj orodja Eclipse [1]. Zadnja verzija tega programa je bila izdana 27. junija 2012 pod kodnim imenom Juno 4.2 in je prikazan tudi na Sliki 4.7. Pri razvoju aplikacije smo ga uporabili tudi mi. Razvojno okolje Eclipse vključuje Java Development Tools (JDT) za Javo, CDT za C/C++, Eclipse PDT za PHP in tako naprej. Vsa ta orodja je mogoče namestiti s pomočjo vtičnika [19].



Slika 4.7: Razvojno okolje Eclipse Juno

Poglavje 5

Pomembni gradniki aplikacije

Za izdelavo aplikacije Android potrebujemo gradnike, ki nam omogočajo pridobitev in prikaz potrebnih podatkov. Zato bomo v tem poglavju najprej predelali razred `TelephonyManager`, ki nam omogoča branje sistemskih podatkov telefona. Ogledali si bomo, kako pridobimo lokacijo uporabnika iz razreda `LocationManager` in pridobivanje podatkov iz spletnega strežnika s spletno storitvijo RESTful. Na koncu pa si bomo ogledali, kako pridobljene podatke prikažemo na zemljevidu Google Maps.

5.1 TelephonyManager

Razred `TelephonyManager` zagotavlja dostop do telefonskih storitev na napravi. Razreda ni mogoče deklarirati neposredno, ampak kot je opisano v Kodi 5.1. Pomembno je vedeti, da so nekateri podatki zavarovani z dovoljenji. Aplikacija Android ne more dostopati do zaščitenih informacij, če ta dovoljenja niso navedena v datoteki Manifest [3].

Koda 5.1: Inicializacija razreda TelephonyManager

```
1 TelephonyManager tm = (TelephonyManager) getSystemService(
    Context.TELEPHONY_SERVICE);
```

5.1.1 Metode

Razred vsebuje več metod, ki omogočajo pridobitev in nastavljanje podatkov o bazni postaji, napravi, operaterju. V nadaljevanju bomo predstavili metode, ki se nanašajo na bazno postajo. V Manifestu za dostop potrebujemo dovoljenja, kot kaže Koda 5.2.

Koda 5.2: Potrebna dovoljenja v datoteki Manifest

```
1 <uses-permission android:name="android.permission.
    ACCESS_COARSE_LOCATION"></uses-permission>
2 <uses-permission android:name="android.permission.
    ACCESS_COARSE_UPDATES"></uses-permission>
3 <uses-permission android:name="android.permission.
    ACCESS_FINE_LOCATION" ></uses-permission>
4 <uses-permission android:name="android.permission.
    READ_PHONE_STATE"></uses-permission>
```

getPhoneType()

Metoda nam vrne podatek, ali telefon bazira na tehnologiji GSM ali na CDMA.

getCellLocation()

Vrne nam podatke o celici na bazni postaji glede na tip telefona. Če telefon bazira na tehnologiji GSM, pridobimo podatke o CellID in LAC. Ta metoda nam žal ne vrne podatkov o frekvenčnem pasu (ARFCN, UARFCN).

getNetworkType()

Metoda nam vrne radijsko tehnologijo, ki jo naprava trenutno uporablja za prenašanje podatkov. Kadar je naprava povezana na omrežje GSM, nam metoda vrne GPRS ali EDGE. Če pa je naprava povezana na UMTS, dobimo UMTS, HSDPA, HSUPA, HSPA ali HSPA+.

getNetworkOperator()

Metoda nam poda številčno vrednost MCC in MNC trenutno registriranega operaterja.

getNetworkOperatorName()

Metoda nam vrne ime trenutno registriranega operaterja.

getNeighboringCellInfo()

S to metodo pridobimo seznam sosednjih celic. Za vsako celico pridobimo Cell ID, LAC, jakost signala in uporabljeno tehnologijo sosednje celice. Moja ugotovitev je, da te metode proizvajalci telefonov ne implementirajo pravilno, kar posledično ne vrača podatka o sosednji celici. Podjetje Samsung metode ni definiralo pravilno, medtem ko proizvajalec telefonov Sony Ericsson prikazuje podatek le na omrežju GSM.

5.1.2 PhoneStateListener

S pomočjo razreda `PhoneStateListener` sledimo spremembam stanj na napravi, vključno s spremembo jakosti signala, prejetjem klica, sporočila, itn. Osredotočili smo se na spremembo trenutno uporabljene celice in jakosti signala. Del Kode 5.3 prikazuje inicializacijo povedanega.

Koda 5.3: Inicializacija poslušalca

```
1 MyPhoneStateListener tmListener = new MyPhoneStateListener();
2 tm.listen(tmListener, PhoneStateListener.LISTEN_CELL_LOCATION
      | PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
```

Med spremembo trenutno uporabljene celice se v razredu `PhoneStateListener` izvede metoda `onCellLocationChanged()`. Ob spremembi jakosti signala pa se izvede metoda `onSignalStrengthsChanged()`. Ta razred je predstavljen v Kodi 5.4.

Koda 5.4: Razred MyPhoneStateListener

```
class MyPhoneStateListener extends PhoneStateListener {

        @Override
        public void onCellLocationChanged(CellLocation
            location)
        {
                super.onCellLocationChanged(location);
                // koda
        }

        @Override
        public void onSignalStrengthsChanged(SignalStrength
            signalStrength) {
                super.onSignalStrengthsChanged(signalStrength
                    );
                // koda
        }
}
```

Pri Androidu verzije 3.0 na tabličnem računalniku Samsung Galaxy Tab sprememba jakosti signala ni delovala, zato smo počakali na nadgradnjo verzije 4.0, kjer teh težav nismo zaznali. Pri branju podatkov o trenutni celici na bazni postaji smo naleteli na težavo, da se je del podatkov posodobil, drugi del pa je ostal od prejšnje celice. Težava je vidna predvsem na mestu, kjer naprava prehaja med omrežji mobilnih operaterjev.

5.2 LocationManager

Razred omogoča pridobivanje geografske lokacije telefona preko GPS ali brezžičnega omrežja (mobilno omrežje ali Wi-Fi). Inicializacija poteka po Kodi 5.5 [3].

Koda 5.5: Inicializacija razreda LocationManager

```
LocationManager mlocManager = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
```

5.2.1 Metode

Za pridobivanje podatkov o geografski lokaciji potrebujemo dovoljenja v datoteki Manifest, ki so našteta v kodi 5.6.

Koda 5.6: Potrebna dovoljenja v datoteki Manifest

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

V nadaljevanju so opisane metode, s katerimi pridobivamo geografsko lokacijo.

getAllProviders

Vrne vse znane ponudnike, ki omogočajo pridobitev geografske lokacije.

getProvider

Vrne ime ponudnika, ki trenutno zagotavlja informacijo o lokaciji. Če nihče ne zagotavlja te informacije, vrne vrednost `null`.

isProviderEnabled

Metoda nam vrne primerno logično vrednost (ponudnik vključen ali ne).

getLastKnownLocation

Metoda vrne zadnjo znano lokacijo izbranega ponudnika. Vrednost je mogoče pridobiti tudi, če je ponudnik neaktiven. To pomeni, da podatki o lokaciji niso nujno posodobljeni.

requestLocationUpdates

S to metodo inicializiramo, kateri ponudnik bo pridobival geografsko lokacijo. Na voljo sta dve možnosti, pridobitev preko GPS ali brezžičnega omrežja.

Zadnji pridobiva podatke od vrednosti Cell ID na mobilnem omrežju ali preko imena lokalnega brezžičnega omrežja. Te podatke omogoča Google prek svoje baze. Pridobi jih z indeksiranjem omrežja, ko uporabniki storitev Google koristijo njegove aplikacije (Google Maps) ali operacijski sistem. Določimo tudi minimalni časovni interval preverjanja spremenjenih podatkov o lokaciji. Poleg tega nastavimo minimalno razdaljo (v metrih), kdaj naj aplikacija prejme podatke o novi lokaciji.

removeUpdates

Metoda prekine pridobivanje podatkov o lokaciji.

5.2.2 LocationListener

Z razredom `LocationListener` lahko ob spremembi geografske lokacije pridobimo nove podatke. V Kodi 5.7 je primer, kjer se inicializira modul GPS z minimalnim časovnim intervalom eno sekundo. Če je razdalja večja od desetih metrov od stare lokacije, se kliče metoda `onLocationChanged(Location)` v `mlocListener`.

Koda 5.7: Inicializacija poslušalca za pridobitev lokacije prek GPS

```
1 LocationListener mlocListener = new MyLocationListener();
2 mlocManager.requestLocationUpdates( LocationManager.
     GPS_PROVIDER, 1000, 10, mlocListener);
```

Razred `LocationListener` sestoji iz štirih metod (Koda 5.8). `OnProviderDisabled`, `onProviderEnabled` in `onStatusChanged` sporočajo stanje ponudnika, ki pridobiva lokacijo naprave. Z metodo `onLocationChanged` pa pridobimo podatke o geografski širini (*Latitude*) in dolžini (*Longitude*) ter o natančnosti izmerjene lokacije.

Koda 5.8: Razred MyLocationListener

```
1 public class MyLocationListener implements LocationListener {
2         public void onLocationChanged(Location loc)
3         {
```

```
4                   // pridobitev lokacije
5                   loc.getLatitude();
6                   loc.getLongitude();
7                   loc.getAccuracy();
8           }
9
10          public void onProviderDisabled(String provider) {}
11
12          public void onProviderEnabled(String provider) {}
13
14          public void onStatusChanged(String provider, int
                status, Bundle extras) {}
15  }
```

5.3 HttpPost

POST je ena izmed metod, ki jo podpira protokol HTTP in spletna storitev RESTful. Ta omogoča pošiljanje podatkov na spletni strežnik. Zahtevek POST je zasnovan tako, da spletni strežnik sprejme podatke, zaprte v telesu sporočila. Uporablja se pri nalaganju datotek ali pošiljanju izpolnjenega spletnega obrazca. Podatke bi lahko na spletni strežnik pošiljali tudi preko metode GET. V tej metodi se podatki prenašajo preko naslova URL. Posledično je količina poslanih podatkov omejena. Zato smo se odločili, da za pošiljanje sporočil na strežnik izberemo metodo POST, ki omogoča prenos poljubne količine podatkov in omogoča pošiljanje datotek [3].

Vzeli bomo primer, ko želimo implementirati spletno storitev RESTful, ki bo iz aplikacije Android poslala podatek na spletni strežnik v obliki sporočila JSON. Kot odgovor spletni strežnik vrne pridobljeni podatek v obliki sporočila JSON. Spletni strežnik na izbranem naslovu URL (na primer `http://primer.com/rest/post/`) vsebuje kodo 5.9 in s pomočjo jezika PHP prejme podatek s parametrom `podatki`, v obliki JSON. Iz JSON niza pridobimo parameter `ime`, ki ga vrnemo nazaj v obliki JSON.

Koda 5.9: Koda PHP na spletnem strežniku

```
<?php
$json = $_POST['podatki'];
$json = stripslashes($json);
$obj = json_decode($json);
$ime = $obj->{'ime'};

header("Content-Type: application/json; charset=UTF-8");
echo '{"vrednost": "'.$ime.'"}';
?>
```

Android vsebuje razred `HttpPost`, ki omogoča pošiljanje podatkov preko metode POST. Da se aplikacija lahko poveže na splet, ji moramo v datoteki Manifest dovoliti povezavo z internetom, kot kaže Koda 5.10.

Koda 5.10: Potrebna dovoljenja v datoteki Manifest

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

V aplikaciji Android moramo za začetek ustvariti objekta `httpClient` in `HttpPost`, z željenim naslovom URL za izvedbo zahteve POST. S pomočjo `BasicNameValuePair` se ustvari parameter `podatki` s sporočilom JSON. Datoteko JSON zgradimo s pomočjo `JSONObject`. Ta vsebuje parameter `ime`. Nato sledi izvedba zahteve POST. Ta nam vrne objekt `HttpResponse`, s pomočjo katerega pridobimo odgovor spletnega strežnika. Ker kot dogovor pridobimo niz JSON, moramo tega še razčleniti in iz njega pridobiti odgovor. Celotna koda je prikazana v Kodi 5.11.

Koda 5.11: Pridobivanje podatka iz spletnega strežnika

```
StringBuilder response = new StringBuilder();
String rezultat = "";
try {
        JSONObject JSON = new JSONObject();
        JSON.put("ime", "moje ime");

        HttpPost post = new HttpPost();
```

```
8           URI url = new URI("http://www.primer.com/rest/post/")
                ;
9           post.setURI(url);
10          List<BasicNameValuePair> params = new ArrayList<
                BasicNameValuePair>();
11          params.add(new BasicNameValuePair("podatki", JSON.
                toString()));
12          post.addHeader("accept", "application/json");
13          post.setEntity(new UrlEncodedFormEntity(params,"UTF-8
                "));
14          DefaultHttpClient httpClient = new DefaultHttpClient
                ();
15          HttpResponse httpResponse = httpClient.execute(post);
16          if (httpResponse.getStatusLine().getStatusCode() ==
                200) {
17                  HttpEntity messageEntity = httpResponse.
                        getEntity();
18                  InputStream is = messageEntity.getContent();
19                  BufferedReader br = new BufferedReader(new
                        InputStreamReader(is));
20                  String line;
21                  while ((line = br.readLine()) != null) {
22                          response.append(line);
23                  }
24                  String rez = response.toString();
25                  JSONObject jsonRez = new JSONObject(rez);
26                  rezultat = jsonRez.getString("vrednost");
27          } else {
28                  rezutat = "error";
29          }
30  } catch (Exception e) {
31          rezultat = "error";
32  }
```

Ni priporočljivo, da se klic spletne storitve izvede sinhrono, saj se s tem zaustavi celotna aplikacija Android pred odgovorom spletnega strežnika. Zato je metodo `HttpPost` priporočljivo implementirati asinhrono z razredom `Async-`

`Task`, da se med izvajanjem spletne storitve aplikacija ne zaustavi. Sicer tudi Android od verzije 3.0 naprej ne dopušča uporabe sinhronih klicev na spletne strežnike.

5.4 Google Maps Android API

Google Maps API je dodatna knjižnica z razredom `MapView`, ki omogoča prikaz zemljevidov v aplikaciji Android. Ker ta knjižnica ni vključena v osnovni nabor knjižnic, jo je potrebno vnesti ročno v datoteko Manifest, kot prikazuje Koda 5.12. Za testiranje aplikacije v emulatorju niso dovolj standardne knjižnice Android, ampak je za to potrebno uporabljati Google API, ki vsebuje dodatno knjižnico za zemljevid. V emulatorju je zato za vsako različico operacijskega sistema v SDK Managerju poleg paketa SDK Platform potrebno prenesti paket Google APIs. Nato moramo v programu AVD Manager nastaviti virtualno napravo (operacijski sistem) z dodatnimi knjižnicami Google APIs. S tem se izognemo sesutju aplikacije Android [7].

Koda 5.12: Uvoz knjižnice Google Maps

```
1 <uses-library android:required="true"
2   android:name="com.google.android.maps">
3 </uses-library>
```

Za pravilno delovanje zemljevidov v aplikaciji potrebujemo v datoteki Manifest tri dovoljenja, ki so našteta v kodi 5.13.

Koda 5.13: Potrebna dovoljenja v datoteki Manifest

```
1 <uses-permission android:name="android.permission.INTERNET"><
    /uses-permission>
2 <uses-permission android:name="android.permission.
    ACCESS_COARSE_LOCATION"></uses-permission>
3 <uses-permission android:name="android.permission.
    ACCESS_FINE_LOCATION" ></uses-permission>
```

Razred `MapView` ponuja dostop do podatkov Google Maps. Da aplikacija lahko dostopa do teh podatkov, potrebujemo API ključ iz certifikata, ki služi

za podpis te aplikacije. S pomočjo programa Keytool pridobimo iz certifikata MD5 prstni podpis, kot prikazuje Koda 5.14. Na spletni strani `https://developers.google.com/maps/documentation/android/maps-api-signup` s pomočjo MD5 spletnega certifikata pridobimo API ključ.

Koda 5.14: Pridobivanje MD5 prstnega odtisa aplikacije za generiranje ključa

```
$ keytool -list -alias moj-alias -keystore moj-store.keystore
```

Zemljevid potrebuje grafično arhitekturo, ki jo zagotovimo z datoteko `maps.xml` v Android projektu, v mapi res/layout (Koda 5.15). Atribut `android:clickable` definira, ali se ob pritisku na zaslon iteracija na zemljevidu izvede ali ne. V primeru vrednosti atributa false iteracije ni. V atribut `android:apiKey` vpišemo API ključ, ki smo ga pridobili s pomočjo certifikata, kot je bilo opisano.

Koda 5.15: Postavitev zemljevida v datoteki maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android
        "
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="apiKey" />
```

Na zemljevid bi radi dodali točke. To storimo z razredom `ItemizedOverlay`, ki ga razširimo v svoj razred `GoogleMapsOverlay` (Koda 5.16). Tukaj je shranjen seznam objektov (točk) `OverlayItem`. Slednji je sestavljen iz geografske lokacije, imena in opisa točke. V tem razredu je nujno definirati metodi `createItem()` in `size()`, ki poskrbita, da se točke dodajo na zemljevid. Z funkcijo `boundCenterBottom()` določimo sliko (ikono) te točke.

Koda 5.16: ItemizedOverlay

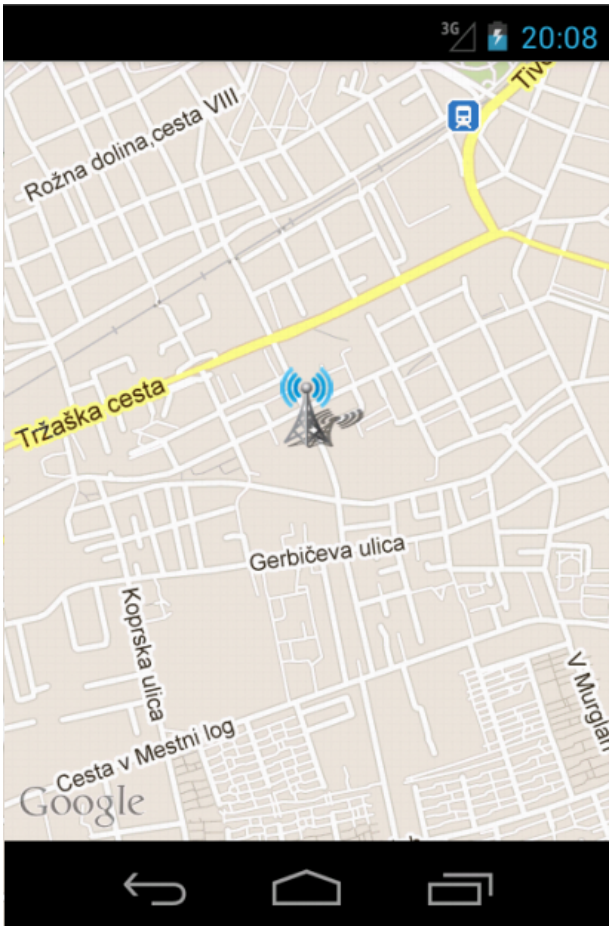
```
1  public class GoogleMapsOverlay extends ItemizedOverlay {
2
3          private ArrayList<OverlayItem> mOverlays = new
               ArrayList<OverlayItem>();
4
5          public GoogleMapsOverlay(Drawable defaultMarker) {
6                  super(boundCenterBottom(defaultMarker));
7          }
8
9          @Override
10         protected OverlayItem createItem(int i) {
11           return mOverlays.get(i);
12         }
13
14         @Override
15         public int size() {
16           return mOverlays.size();
17         }
18
19         public void addOverlay(OverlayItem overlay) {
20             mOverlays.add(overlay);
21             populate();
22         }
23 }
```

Za prikaz zemljevida ustvarimo aktivnost, ki je razširjena iz razreda `MapActivity` in nam prikaže zemljevid. V vsaki taki aktivnosti je potrebno ustvariti metodo `isRouteDisplayed()`. V standardni metodi `onCreate()` kličemo postavitev `maps.xml`. Nato z razredom `MapView` inicializiramo zemljevid in s pomočjo razreda `GoogleMapsOverlay` prikažemo točko na zemljevidu, kot prikazuje Koda 5.17.

Koda 5.17: Dodajanje točke na zemljevid

```
1  MapView mapView = (MapView) findViewById(R.id.mapview);
2
3  Drawable ikona = getResources().getDrawable(R.drawable.icon);
```

```
4 GoogleMapsOverlay itemizedoverlay = new GoogleMapsOverlay(
      ikona);
5
6 OverlayItem overlayitem = new OverlayItem(new GeoPoint(0,0),
      "ime", "opis");
7 itemizedoverlay.addOverlay(overlayitem);
8
9 mapView.getOverlays().add(itemizedoverlay);
```



Slika 5.1: Primer Google Maps na Androidu

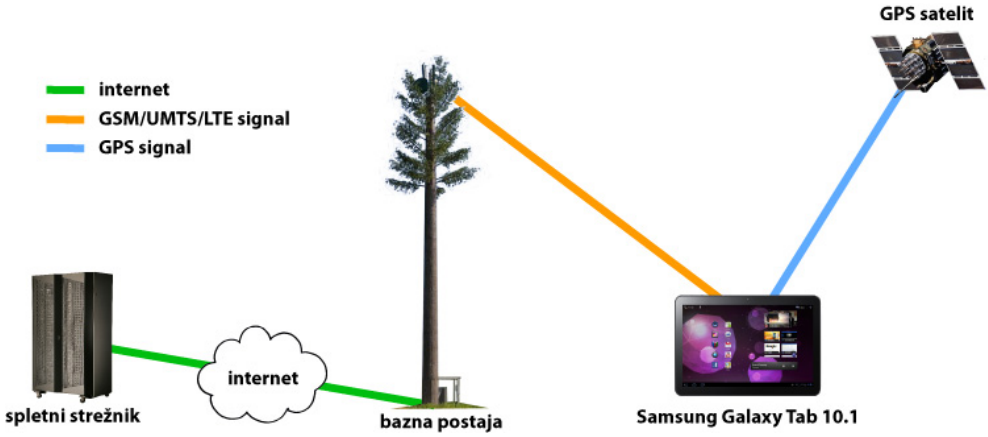
Po zgrajeni kodi se nam na zaslonu aplikacije prikaže Slika 5.1. Točko na sliki predstavlja ikona `icon.png`, ki se nahaja v datoteki `res/drawable/`. S

pomočjo razreda `MapView` je mogoče implementirati dodatne možnosti, kot so: dodatni gumbi, približevanje, oddaljevanje, izris poti, sledenje neki točki. Pomembno je tudi, da se preneseni zemljevidi shranijo na telefon. S tem prihranimo pri prenosu podatkov, ker nam zemljevidov ni potrebno prenašati še enkrat.

Poglavje 6

Razvoj aplikacije Lokator

V tem poglavju bomo opisali razvoj aplikacije Android, ki omogoča vse funkcionalnosti, ki smo jih spoznali v prejšnjih poglavjih. Aplikacija omogoča interpretiranje podatkov iz bazne postaje, na kateri je telefon prijavljen. Odločili smo se jo poimenovati Lokator. V aplikaciji želimo prikazati informacije o trenutno prijavljeni celici na bazni postaji, ne glede na to v kateri državi se nahajamo. Prav tako želimo pridobiti geografsko lokacijo s pomočjo sistema GPS ali prijavljenega omrežja. Vsaka nova zaznana celica se z njeno lokacijo pošlje v bazo Open Cell ID, s spletno storitvijo RESTful. Če se uporabnik odloči, da si želi pridobljene podatke shraniti na telefon, se ti lahko shranijo v datoteko `.log`. V Sloveniji se poleg podatkov izpiše tudi podatek o bazni postaji. Opis, lokacijo in sliko bazne postaje pridobimo s klicem na spletni strežnik preko spletne storitve RESTful. Želimo si, da bi aplikacija prikazala zemljevid, na katerem bo vidna bazna postaja ter lokacija uporabnika. Če bazne postaje še ni v sistemu, se izvedejo postopki obveščanja skrbnika aplikacije. Aplikacija omogoča pošiljanje sporočil skrbniku o napakah v podatkih o baznih postajah. Na namizju operacijskega sistema si želimo prikaz imena bazne postaje s pomočjo pripomočka (*Widget*). Aplikacijo bomo testirali na tabličnem računalniku Samsung Galaxy Tab 10.1, z verzijo Android 4.0, kot kaže Slika 6.1.



Slika 6.1: Komunikacija v aplikaciji Lokator

6.1 Spletni strežnik

Spletni strežnik nam omogoča procesiranje rezultatov, ki nam jih pošilja aplikacija Lokator na Androidu. Ima dve funkciji, prva je pošiljanje podatkov o bazni postaji, druga pa omogoča realizacijo pošiljanja sporočil skrbniku aplikacije. Obe metodi sta bili razviti v jeziku PHP s spletno storitvijo RESTful. Spletni strežnik je postavljen na operacijskem sistemu Linux. Na strežniku teče PHP verzije 5.3.13.

6.1.1 Podatki baznih postaj

Za pridobitev želenih podatkov o bazni postaji potrebujemo nekaj atributov iz aplikacije Android:

- MCC,
- MNC,
- Cell ID,
- LAC,
- jakost signala,

- generacija mobilnega omrežja (GSM, UMTS, LTE),
- geografska lokacija uporabnika (geografska širina in dolžina, ponudnik dobljene lokacije),
- verzija aplikacije,
- verzija Androida.

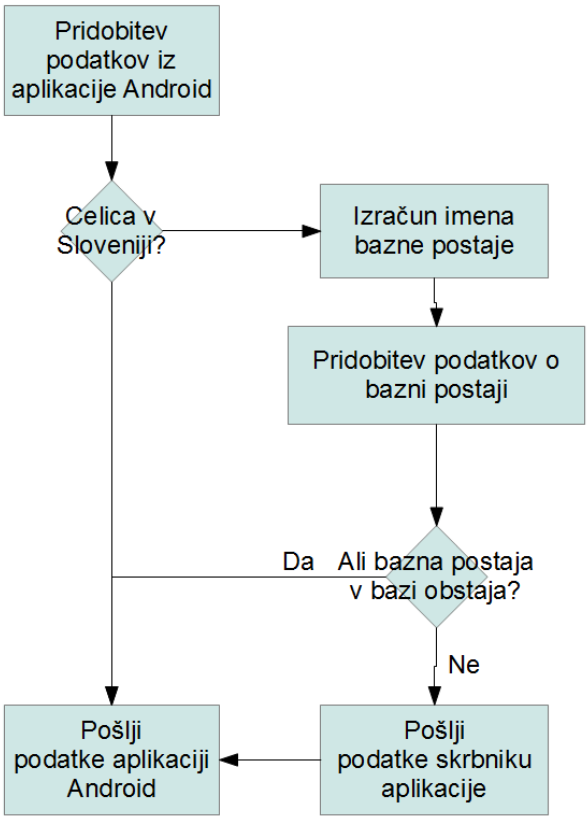
6.1.2 Storitev za obdelavo podatkov bazne postaje

Storitev kličemo z URL naslovom `http://www.svetidej.com/lokator/android/bazna/`. Klic vsebuje metodo POST z atributom `podatki`. Ta atribut vsebuje objekt JSON, kot kaže Koda 6.1.

Koda 6.1: Primer objekta JSON za pridobitev podatkov o bazni postaji

```
1 { "baznaPostajaRequest": {
2         "baznaPostaja": { "mcc": "293", "mnc": "70", "cellid"
              : "167393", "lac": "60", "nacin": "UMTS", "jakost"
              : "-83" },
3         "telefon": { "latitude": "45.881209", "longitude": "
              15.307711", "ponudnikLokacije": "GPS", "
              verzijaAplikacije": "2.09", "verzijaAndroida": "
              4.2" }
4 }}
```

S podatkoma MCC in MNC najprej preverimo, kateremu mobilnemu operaterju pripada celica. Če celica pripada slovenskemu operaterju, izračunamo ime bazne postaje, kot je opisano v Poglavju 2.3. Po pridobljenem imenu operaterja in imenu bazne postaje moramo dostopati do podatkov o seznamu baznih postaj določenega operaterja. Na strežniku za vsakega operaterja obstaja datoteka s končnico `.kml`, ki hrani seznam baznih postaj. Datoteko KML pridobimo z izvozom točk iz programa Google Earth. Seznam baznih postaj smo pridobili iz spletnih strani `http://gis.arso.gov.si/atlasokolja/`, `http://www.apek.si/frekvence` in `http://www.geoprostor.`



Slika 6.2: Diagram poteka za pridobitev podatkov o imenu bazne postaje na spletnem strežniku

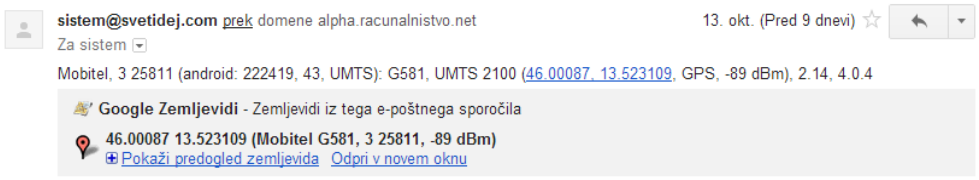
`net/PisoPortal/` ter jih nato uredili s programom Google Earth. Datoteke KML so izpeljane iz jezika XML, zato jih obdelamo kot kodo XML.

Koda 6.2: Primer vnesene bazne postaje v datoteki KML

```
1 <Placemark>
2         <name>SE632</name>
3         <description>GSM 900, UMTS 900,
4 stolp Cadraze,
5 Cadraze 7,
6 8310 Sentjernej,
7 0151738E/455201/153m,
8 31.10.2009</description>
```

```
9           <Point>
10                  <coordinates>15.29368903384951,
                        45.86685792755827,0</coordinates>
11          </Point>
12  </Placemark>
```

Koda 6.2 nam prikazuje vnos bazne postaje v datoteki KML. Označen je z imenom `Placemark`. Znotraj se nahaja `name`, ki nam pove ime bazne postaje. `Description` vsebuje opis bazne postaje. Na vrhu opisa so naštete tehnologije, ki so omogočene na izbrani bazni postaji. Nato sledi osnovni opis. Če je na koncu opisa dodan datum, pomeni aktivacijo bazne postaje okrog tistega datuma. V tagu `Point` se nahaja `coordinates`, v kateri je navedena geografska lokacija bazne postaje s pomočjo geografske širine in dolžine. Po pridobljenem opisu in lokaciji bazne postaje je treba preveriti, če obstaja na spletnem strežniku slika bazne postaje. Slike so shranjene v točno določenem direktoriju strežnika. Koda PHP preveri obstoj slike JPG z enakim imenom v izbranem direktoriju. Če ta obstaja, si zapomnimo naslov URL slike.



Slika 6.3: Primer poslanega sporočila na skrbnikov e-poštni naslov

Ker želimo biti obveščeni o novi bazni postaji ali novi tehnologiji na bazni postaji, smo izdelali, da v teh primerih skrbnik aplikacije prejme e-poštno sporočilo. To sporočilo vsebuje vse podatke, ki jih je poslala aplikacija Android in informacije, ki jih je izračunal spletni strežnik (Slika 6.3). Če obstaja geografska lokacija uporabnika, se za lažjo obdelavo podatkov v e-poštnem sporočilu prikaže tudi povezava za prikaz lokacije na spletni različici Google

Maps. Prikaz različice Androida in aplikacije Android potrebujemo za preverjanje pravilnega delovanja aplikacije. S tem ugotovimo, v kateri različici je prišlo do napake in jo lažje odpravimo.

Zadnje dejanje je pošiljanje rezultatov poizvedbe v aplikacijo Android. Kot odgovor na poizvedbo vrne objekt JSON, kot je prikazano v Kodi 6.3. Vrne nam opis, lokacijo in URL naslov slike bazne postaje. Za dodatno preverjanje točnosti podatkov dodamo še podatke o imenu bazne postaje ter kodi MCC in MNC. Če kateri podatek ne obstaja, vrne za tisti atribut vrednost `null`. Diagram poteka celotne storitve je prikazan na Sliki 6.2.

Koda 6.3: Primer rezultata poizvedbe bazne postaje

```
{ "baznaPostajaResponse": {
        "imeBazne": "SE632", "opisBazne": "GSM 900, UMTS 900,
            stolp Cadraze, Cadraze 7, 8310 Sentjernej,
           0151738E/455201/153m, 31.10.2009", "mcc": "293", "
           mnc": "70", "slika": "http://www.svetidej.com/
           slike/SE632.jpg", "latitude": "45.86685792755827",
            "longitude": "15.29368903384951", "natancnost": "
           0"
}}
```

6.1.3 Storitev za pošiljanje sporočil

Odločili smo se, da bo pošiljanje sporočila iz aplikacije Android realizirano s pomočjo spletnega strežnika. To spletno storitev kličemo na `http://www.svetidej.com/lokator/android/sporocilo/` z metodo POST, ki vsebuje atribut `podatki`. V njem je vsebovan objekt JSON, ki ima podatke o imenu in e-sporočilu pošiljatelja ter naslov in telo sporočila (Koda 6.4).

Koda 6.4: Primer objekta JSON za pošiljanje sporočila

```
{ "sporociloRequest": {
        "ime": "Mitja", "email": "mitja@svetidej.com", "
           naslov": "naslov sporocila", "besedilo": "telo
           sporocila"
}}
```

Sporočilo se pošlje na skrbnikov e-poštni naslov. Spletni strežnik aplikaciji Android vrne podatek o uspešnosti poslanega sporočila (Koda 6.5).

Koda 6.5: Primer rezultata pošiljanja sporočila

```
1 { "sporociloResponse": { "odgovor": "true" }}
```

6.2 Aplikacija na Androidu

Projekt aplikacije Lokator temelji na Androidu verzije 4.1 (Jelly Bean). Aplikacija je razvita za telefone z operacijskim sistemom verzije od 2.2 naprej. Poleg osnovnih knjižnic Androida je vključena tudi dodatna knjižnica za prikazovanje zemljevida. Aplikacija uporablja grafično predlogo `Holo Light`. Za delovanje potrebuje sedem dovoljenj, kot prikazuje Koda 6.6.

Koda 6.6: Dovoljenja aplikacije Lokator v datoteki Manifest

```
1 <uses-permission android:name="android.permission.INTERNET"><
    /uses-permission>
2 <uses-permission android:name="android.permission.
    ACCESS_COARSE_LOCATION"></uses-permission>
3 <uses-permission android:name="android.permission.
    ACCESS_COARSE_UPDATES"></uses-permission>
4 <uses-permission android:name="android.permission.
    ACCESS_FINE_LOCATION" ></uses-permission>
5 <uses-permission android:name="android.permission.
    ACCESS_NETWORK_STATE"></uses-permission>
6 <uses-permission android:name="android.permission.
    READ_PHONE_STATE"></uses-permission>
7 <uses-permission android:name="android.permission.
    WRITE_EXTERNAL_STORAGE" ></uses-permission>
```

Aplikacija je sestavljena iz sledečih aktivnosti (podstrani) in storitev:

- **LokatorActivity:** Je glavna aktivnost, ki se zažene ob zagonu aplikacije in prikaže podatke v tekstovni obliki. Poskrbi za prejem vseh zahtevanih podatkov, ki jih aplikacija potrebuje za delovanje.

- **NastavitveActivity:** V tej aktivnosti uredimo nastavitve aplikacije, kot so omogočanje internetne povezave, zapis v datoteko `.log`, vpis svojih podatkov (imena in e-poštnega naslova).
- **GoogleMapsActivity:** Aktivnost omogoča prikaz zemljevida z lokacijo uporabnika in bazne postaje.
- **SporociloActivity:** Aktivnost, ki omogoča pošiljanje sporočil skrbniku aplikacije.
- **IskanjeActivity:** Opis priporočil, kako najti bazno postajo, če ta ne obstaja v bazi podatkov.
- **VizitkaActivity:** Vizitka aplikacije.
- **prvaStran:** Omogoča prikaz logotipa ob zagonu aplikacije.
- **Widget1:** Zagotavlja prikaz pripomočka na namizju telefona. Prav tako vsebuje vse nastavitve, ki jih potrebuje pripomoček za delovanje.
- **Widget1Service:** Zagotavlja procesno logiko pripomočku Widget1.

Aplikacija je oblikovana za palčne zaslone velikosti od 3,5 do 11, torej za pametne telefone in tablične računalnike. Prav tako je na nekaterih aktivnostih v aplikaciji drugačna razporeditev objektov, če je zaslon v pokončnem ali ležečem stanju. Poleg slovenskega jezika aplikacija podpira tudi angleški jezik. Ta se samodejno nastavlja glede na nastavitve telefona. Če imamo na telefonu izbran slovenski jezik, bo aplikacija prikazana v slovenščini. V primeru, da imamo izbran drug jezik, pa bo aplikacija vedno v angleškem jeziku. Lokalizacijo aplikacije smo spoznali v Poglavju 4.3.4.

6.2.1 Prikaz in pridobitev podatkov

Vse podatke pridobivamo in beremo iz aktivnosti `LokatorActivity`. Če to aktivnost operacijski sistem uniči zaradi čiščenja pomnilnika, aplikacija več ne pridobiva podatkov. V primeru uničenja aktivnosti `LokatorActivity`, ko

uporabljamo zemljevid (aktivnost `GoogleMapsActivity`), se ta inicializira samodejno.

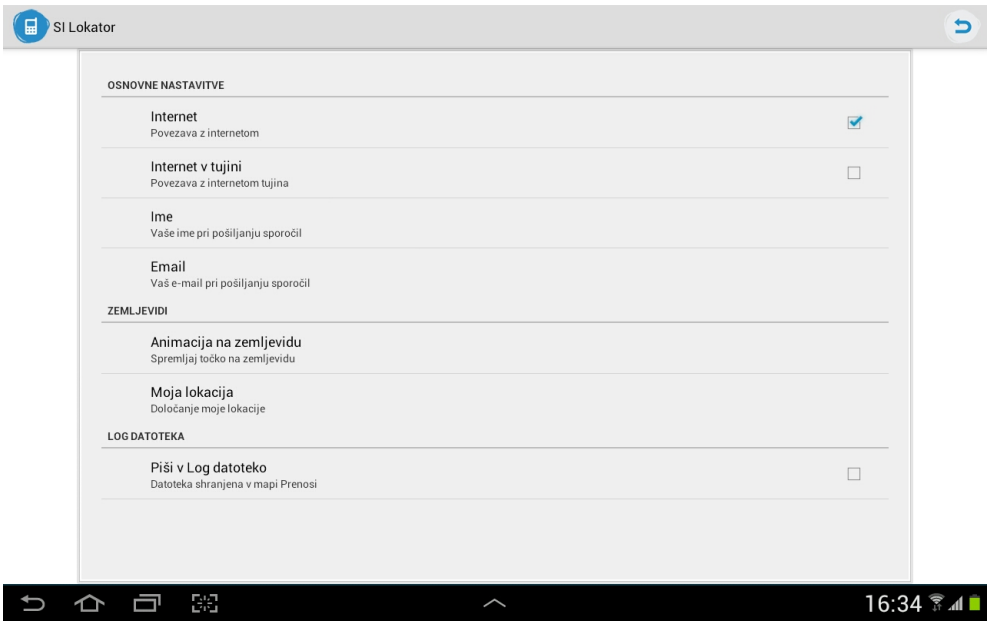
Ob inicializaciji aktivnosti se preveri, ali je aktivnost klicana prvič. Če je, se kliče aktivnost `prvaStran`, ki prikaže začetni logo aplikacije. V aktivnosti ustvarimo globalno spremenljivko `mojCellID`, v katero se shranjujejo vsi želeni podatki o sistemskih podatkih telefona. Spremenljivka je izpeljana iz razreda `PodatkiTelefon`. Nato omogočimo branje globalnih nastavitev aplikacije `SharedPreference` (Poglavje 4.3.5). Te vsebujejo tri večje razdelke (Slika 6.4):

- **Osnovne nastavitve:** V tem razdelku nastavljamo, ali želimo, da ima aplikacija dostop do interneta znotraj države ali tudi v tujini (bazna postaja ima drugačno kodo MCC, kot jo ima kartica SIM). Privzeta je internetna povezava z blokado v tujini. Da ob pošiljanju sporočil skrbniku ni potrebno vedno vtipkati imena in e-poštnega naslova, si lahko ta dva podatka shranimo v nastavitvah.

- **Zemljevidi:** Za zemljevide imamo dve nastavitvi. S prvo nastavimo, ali želimo spremljati neko točko na zemljevidu ali ne. Izbiramo med lokacijo uporabnika in bazno postajo. Druga nastavitev pa določa, kako naj aplikacija določi uporabnikovo lokacijo. To lahko izvede prek satelita GPS ali brezžičnega omrežja. Privzete nastavitve določajo iskanje moje lokacije samodejno. To pomeni, da ob neuspešni povezavi s satelitom GPS uporabi podatke podjetja Google iz brezžičnega omrežja.

- **Log datoteka:** Omogoča shranjevanje pridobljenih podatkov v datoteko s končnico `.log`.

Glavni namen aplikacije je pridobivanje informacij o lokaciji, zato omogočimo branje sistemskih podatkov o telefonu (`TelephonyManager`) ter uporabnikovi lokaciji (`LocationManager`), kot smo ju opisali v Poglavju 5. Omogočimo poslušalca `PhoneStateListener()` in mu nastavimo, naj se sproži ob spremenjenih podatkih o bazni postaji ter spremembi jakosti signala. Prav tako omogočimo poslušalca `LocationListener()`, ki ob spremembi lokacije



Slika 6.4: Nastavitve v aplikaciji Lokator (Galaxy Tab 10.1)

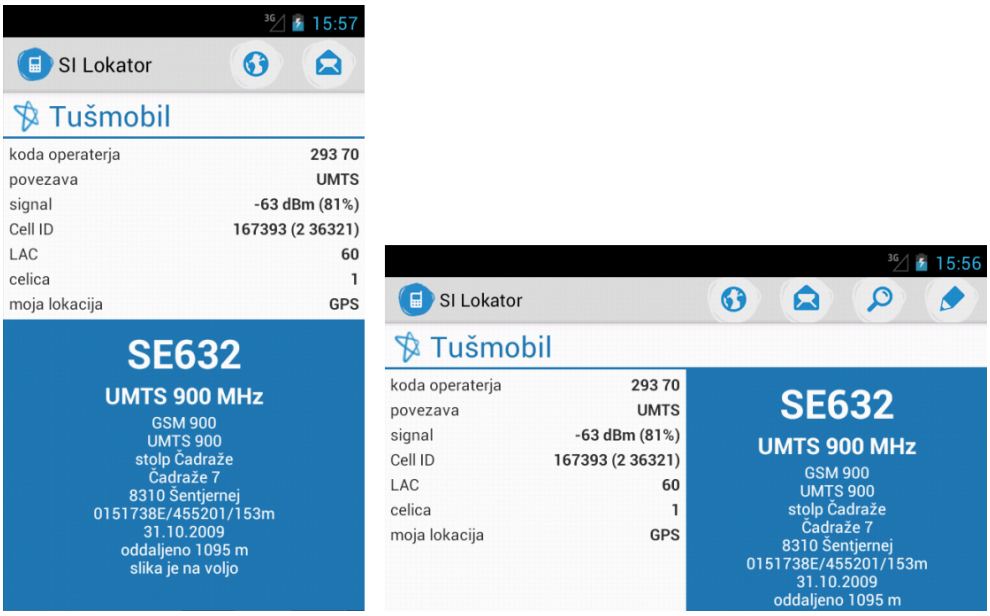
sporoči novo lokacijo uporabnika. Izbranemu modulu za določanje lokacije nastavimo, naj preverja uporabnikovo lokacijo na eno sekundo. Če je uporabnik od prejšnje shranjene lokacije oddaljen več kot deset metrov, se sproži poslušalec `LocationListener()`, ki priskrbi aplikaciji novo geografsko lokacijo uporabnika. Ko se celotna aktivnost `LokatorActivity` uniči, moramo poslušalca `PhoneStateListener()` in `LocationListener()` v metodi `onDestroy()` onemogočiti. V nasprotnem primeru bosta poslušalca delovala tudi po zaprtju aplikacije, kar vodi v večjo porabo baterije ter nezmožnost zaprtja modula GPS.

Vsaka metoda v poslušalcih `PhoneStateListener()` in `LocationListerner()` poleg tega, da shrani nove podatke, tudi kliče metodo `spremeniPodatkeZaslon()`. Ta metoda je sestavljena iz klicev metod `pridobiPodatkeCell()` ter `osveziZaslon()`. Prva metoda pridobi vse podatke o celici na bazni postaji: Cell ID, LAC, MCC in MNC iz objekta `GsmCellLocation`, ki je pridobljena iz razreda `TelephonyManager`. Prav tako pridobi podatke o kartici SIM (vstavljeni v napravo), ime operaterja, seznam sosednjih celic

ter na kateri tehnologiji (GSM, UMTS, LTE) je trenutno prijavljen telefon.

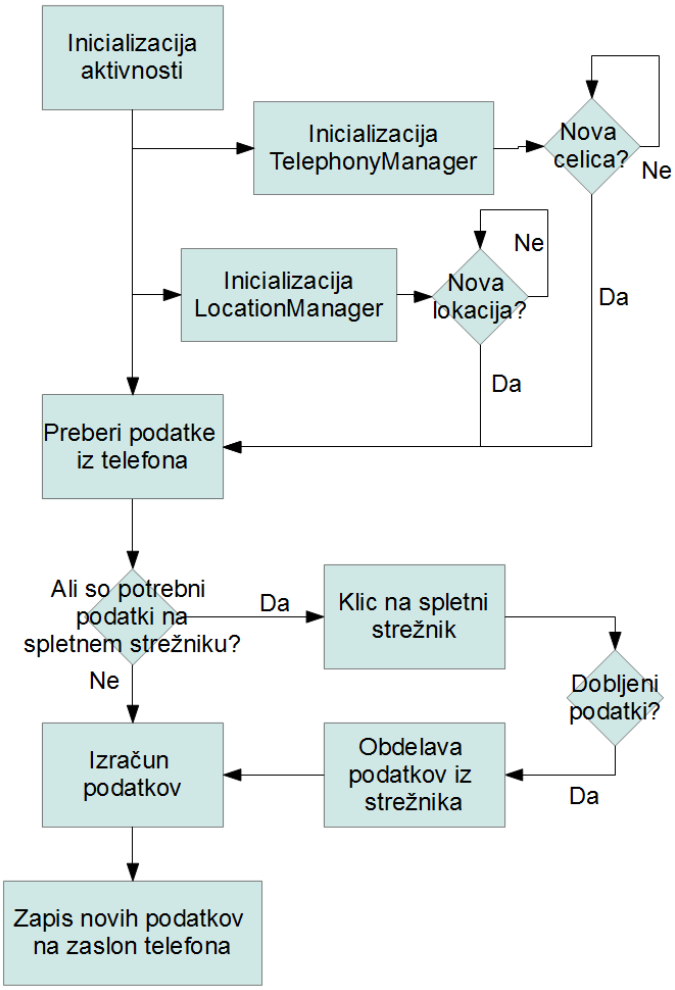
Ko pridobimo vse podatke, jih obdelamo. Kadar gre za mobilno omrežje v Sloveniji, aplikacija izračuna ime bazne postaje, kot je opisano v Poglavju 2.3. V primeru neznane bazne postaje ali postaje izven Slovenije vrne prazen niz. Ker iz telefona ne moremo pridobiti podatka, na kateri frekvenci je prijavljen telefon, ga je v nekaterih primerih možno definirati s pomočjo podatka Cell ID. Po izračunanem imenu in frekvenci bazne postaje iz podatka Cell ID preverjamo, ali ima uporabnik v nastavitvah omogočeno internetno povezavo za aplikacijo. Prav tako preverimo, če se uporabnik nahaja v tujini. Ob obeh izpolnjenih pogojih sledi obdelava podatkov, ki se nanašajo na internetno povezavo in so opisani v nadaljevanju. Če za določeno celico obstaja tudi geografska lokacija uporabnika, potem pošljemo oba podatka v bazo Open Cell ID z asinhronim klicem določenega naslova URL, kot smo opisali v Poglavju 3.4.2. Če hkrati ime bazne postaje ni prazen niz, sledi preverjanje, ali obstajajo podatki o tej bazni postaji. Najprej se preveri, če je bazna postaja shranjena v seznamu baznih postaj. Če ta obstaja, pridobimo opis, lokacijo in naslov URL slike bazne postaje. V nasprotnem primeru je potrebno z asinhronim klicem spletne storitve RESTful poslati poizvedbo na spletni strežnik, kot smo opisali v poglavju o spletnem strežniku. Po prejetju podatkov te shranimo v seznam baznih postaj ter v spremenljivko `mojCellID`, v kateri so podatki o trenutno prijavljeni bazni postaji. Podatki v seznamu baznih postaj se ob zaprtju aplikacije izbrišejo in jih ob ponovnem zagonu aplikacije ni več mogoče pridobiti. Po izvedenih dejanjih in ob omogočeni nastavitvi za zapis novih podatkov v datoteko `.log` se nam zapiše nova vrstica s trenutnimi podatki (CellID, LAC, MCC, MNC, jakost signala), s pomočjo razreda `FileWriter` (Poglavje 4.3.5).

Sledi še zadnje dejanje. Pridobljene podatke izpišemo na zaslon. Zaslon razdelimo na dva dela. V prvem delu se nahajajo osnovni podatki, ki so pridobljeni iz telefona. Če podatki za določeno postavko ne obstajajo, je naveden znak neskončno. To ne velja za sosednje Cell ID-je. Kot smo že omenili, imajo nekateri proizvajalci telefonov težave pri pridobivanju le-teh.



Slika 6.5: Pregled podatkov v aplikaciji Lokator (Emulator)

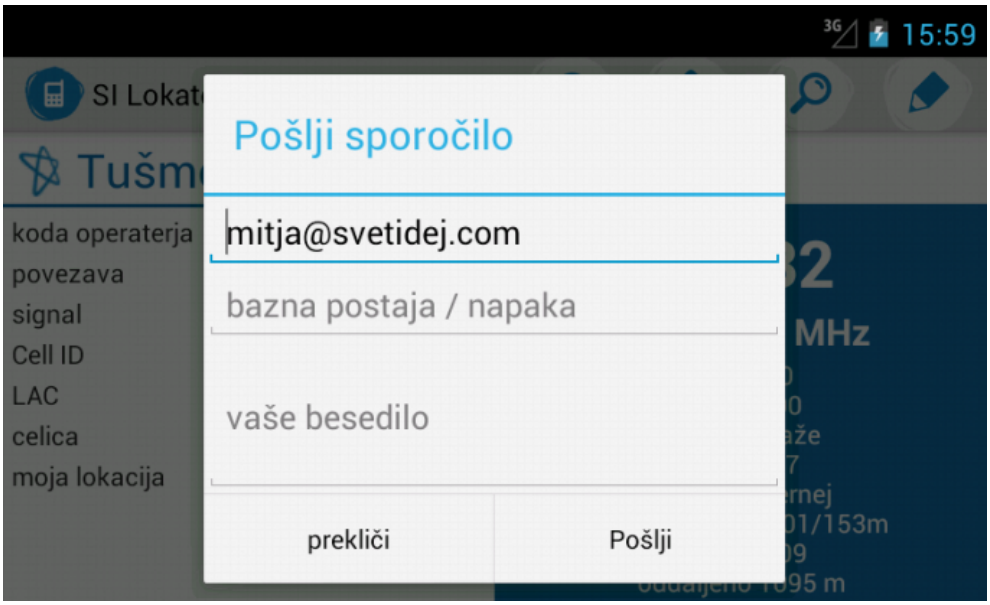
Če aplikacija ne prejme nobenega sosednjega Cell ID-ja, se ta postavka skrije. Drugi del, ki je obarvan z modro barvo, je prikazan le tedaj, ko obstajajo podatki o bazni postaji. Izpiše se ime bazne postaje, tehnologija, frekvenca ter opis bazne postaje. Če sta lokaciji bazne postaje in uporabnika znani ter je točnost lokacije uporabnika vsaj 500 metrov, se pod opisom bazne postaje izpiše tudi oddaljenost bazne postaje od uporabnika. Za bazne postaje s sliko se pod opisom izpiše, da slika obstaja. Na Sliki 6.5 si lahko ogledate izpis podatkov na zaslon aplikacije v pokončnem in ležečem položaju zaslona. Ob kliku na podatke o bazni postaji se odpre aktivnost zemljevidi. Celotni diagram poteka je predstavljen na Sliki 6.6.



Slika 6.6: Diagram poteka za pridobitev podatkov na aplikaciji Android (LokatorActivity)

Za sporočanje napak ali novih informacij razvijalcu aplikacije je to omogočeno z gumbom za pošiljanje sporočil. Prikaže se obrazec, ki vsebuje vnos treh podatkov. Obrazec je prikazan na Sliki 6.7. Prvi je e-poštni naslov uporabnika, ki se v primeru, da je shranjen v nastavitvah, izpiše samodejno. Sledita vnosa naslov in telo sporočila. Vsi podatki se pošljejo na spletni strežnik, ki poskrbi za pošiljanje sporočila skrbniku. V naslovu sporočila se ob pošiljanju le-tega dodajo še podatki o trenutno prijavljeni bazni postaji.

Tako skrbnik s pomočjo potrebnih podatkov lažje odpravi morebitno napako. Uporabnik dobi v odgovor sporočilo o uspešni dostavi poslanega sporočila (napake ali informacije).



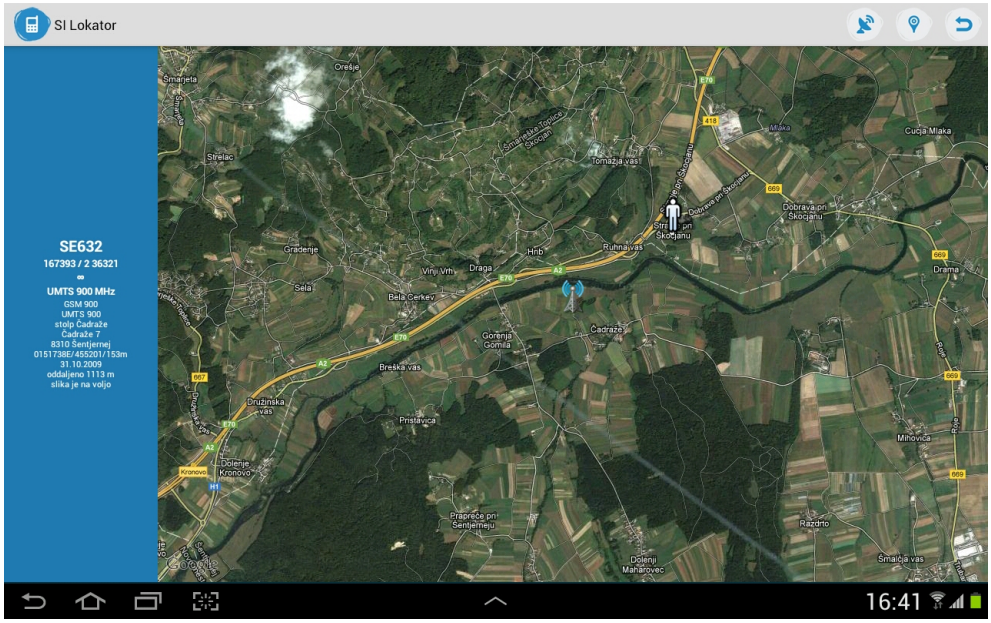
Slika 6.7: Pošiljanje sporočil v aplikaciji Lokator (Emulator)

6.2.2 Prikaz podatkov na zemljevidu

Aktivnost `GoogleMapsActivity` poskrbi za prikaz zemljevida z vsemi potrebnimi podatki. Na zaslonu želimo prikazati več aktivnosti hkrati. To nam omogočajo fragmenti (*Fragments*), ki jih je Android predstavil v različici 3.0 (Poglavje 4.3.2). Želeli smo, da naša aplikacija podpira tudi verzijo Androida 2.2 zato smo morali najti primerno rešitev tudi za to različico. V verzijah Androida, ki podpirajo fragmente, prikazujemo zemljevid in podatke o bazni postaji, v starejših različicah pa le zemljevid.

Podatke pridobimo iz spremenljivke `mojCellID` v aktivnosti `LokatorActivity`. Za inicializacijo zemljevida uporabimo Google Maps Android API (Poglavje 5.4). Namesto osnovnega prikaza zemljevida smo izbrali satelit-

ski način, ki prikazuje fotografske posnetke iz zraka (Slika 6.8). Prav tako omogočimo gumbe za spreminjanje velikosti zemljevida. Na zemljevidu sta dve vrsti ikon. Prva ikona služi za prikaz baznih postaj, druga omogoča prikaz uporabnikove lokacije. Ob spremembi lokacije se ta primerno premakne na zahtevane koordinate. Če je vrednost lokacije nedefinirana, sta geografska širina in dolžina enaki vrednosti nič. V tem primeru se ikona s takimi vrednostmi ne prikaže na zemljevidu. Glede na shranjene nastavitve v aplikaciji Lokator lahko animiramo neko točko na sredini zemljevida. Če ta ni na sredini zemljevida, se tja prestavi po določenem času. Za lažje spreminjanje te nastavitve je v meniju zemljevida na voljo gumb, ki omogoča spreminjanje animacije točk. Tako nam ni potrebno iti v aktivnosti nastavitve za spremembo teh. Poleg tega gumba imamo dodaten gumb za odpiranje lokacijskih storitev v nastavitvah operacijskega sistema Android. Tukaj določimo uporabo brezžičnega omrežja in satelitov GPS za vse aplikacije, naložene na operacijskem sistemu Android.



Slika 6.8: Zemljevid v aplikaciji Lokator (Galaxy Tab 10.1)

Verzijam Androida, ki podpirajo prikaz fragmentov, prikažemo še podatke o trenutni prijavljeni bazni postaji. Če je zaslon v pokončnem položaju, se te informacije prikažejo spodaj, v primeru, da je zaslon v ležečem položaju, pa levo od zemljevida. Prikažeta se dva tipa podatkov. V prvem se prikaže ime bazne postaje, Cell ID, jakost signala, tehnologija in frekvenca. Če kateri od podatkov ne obstaja, se ne prikaže. Drugi del je namenjen opisu bazne postaje. Če bazna postaja ni v Sloveniji, se ta razdelek skrije, v nasprotnem primeru se prikaže opis bazne postaje. Če je pod opisom bazne postaje prikazan opis: `slika je na voljo`, se ob kliku na opis bazne postaje prikaže slika bazne postaje (Slika 6.9). Prenos slike se izvrši podobno kot spletna storitev RESTful z metodo `HttpPost`, le da gre tukaj za prenos slike iz določenega naslova URL ter brez dodatnih parametrov POST. Če bazna postaja ni vnesena v sistem ali še nima točne lokacije, se prikaže besedilo o pomoči pri iskanju lokacije bazne postaje. V tem primeru se ob kliku na to besedilo odpre okno za pošiljanje sporočila razvijalcu aplikacije.

Slika 6.9: Slika bazne postaje v aplikaciji Lokator (Galaxy Tab 10.1)

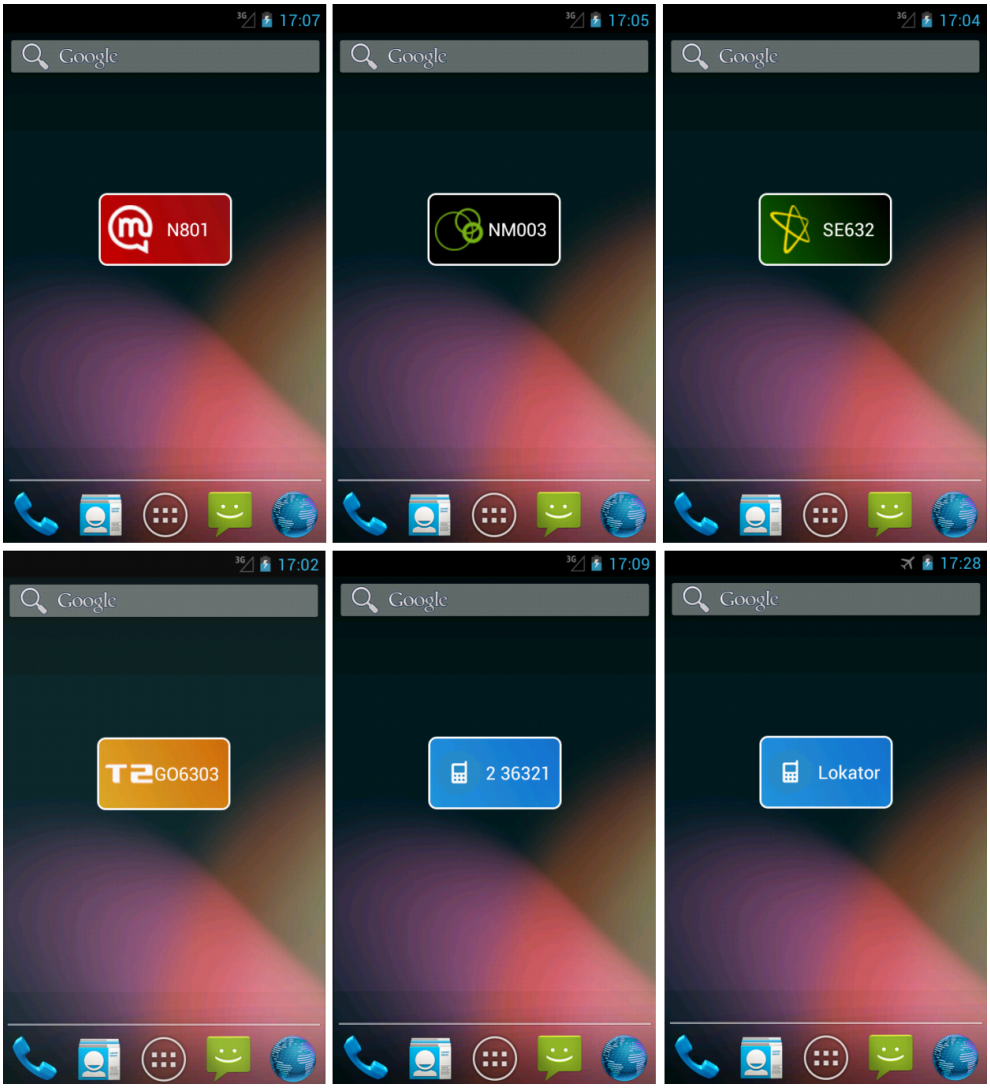
6.2.3 Pripomoček

Pripomočki Android (*Widgets*) omogočajo prikaz informacij na začetnem zaslonu Androida (v namizju). Ponavadi so implementirani v posebnem paketu, ločeno od aplikacije Android. Odločili smo se, da bomo za naš projekt Lokator implementirali pripomoček v isti paket kot aplikacijo Android. Torej z eno namestitveno datoteko namestimo na Android tako aplikacijo kot pripomoček. Pripomoček smo poimenovali Lokator Widget 2x1 in bo prikazoval operaterja in ime bazne postaje. Samostojno bo tekel brez internetne povezave tudi v primeru, če aplikacija Lokator ne bo prižgana.

Pripomoček inicializiramo z razredom `Widget1`, ki ga razširimo iz razreda `AppWidgetProvider`. Ob zagonu tega razreda se izvede razred `Widget1Service`, ki je razširjen iz razreda `Service`. V tem razredu je nastavljena logika pripomočka. Pomembno je nastaviti velikost pripomočka in čas osveževanja. Odločili smo se za višino 100dp in višino 39dp. Minimalni čas izvajanja pripomočka je iz strani Androida določen na 30 minut. Izbrali smo najkrajši možni čas osvežitve. V primeru izbire krajšega časa, se bo pripomoček osvežil šele po tridesetih minutah. Postavljeni smo bili pred izziv, kako zaobiti to omejitev. Delno smo jo rešili z uvedbo poslušalca `PhoneStateListener()`, s pomočjo razreda `TelephonyManager` v storitvi `Widget1Service`. Poslušalec je nekaj časa sprejemal podatke, nato pa je v nekem trenutku prenehal delovati. Ugotovili smo, da je storitev uničil operacijski sistem zaradi varčevanja pomnilnika. Težavo smo rešili tako, da smo metodi `onStartCommand()` v `Widget1Service` kot odgovor vrnili vrednost `START_STICKY`. S tem smo operacijskemu sistemu dali vedeti, naj storitve ne uniči, če to zares ni potrebno. S tem in s posodobitvijo pripomočka na 30 minut smo odpravili težavo delovanja pripomočka.

Ob spremembi celice na bazni postaji dobimo podatke Cell ID, LAC, MCC in MNC. Iz teh pridobimo ime operaterja in bazne postaje. Za izračun teh podatkov smo uporabili metode, ki smo jih že uporabili pri aplikaciji. Ko imamo vse potrebne podatke, sledi grafični prikaz podatkov na pripomoček. Če bazna postaja pripada enemu izmed slovenskih operaterjev, smo ozadje

pripomočka pobarvali v barvo operaterja. Levo v pripomočku smo dodali operaterjev logotip, desno pa izpis imena bazne postaje. Če Cell ID operaterja pripada državi izven Slovenije, se namesto imena bazne postaje izpiše njegov Cell ID. Če telefon ni prijavljen v nobeno mobilno omrežje, se izpiše beseda Lokator. V zadnjih dveh primerih se levo izriše logotip aplikacije ter modra barva ozadja. Vse grafične različice pripomočka so navedene na Sliki 6.10.



Slika 6.10: Pripomoček v aplikaciji Lokator (Emulator)

Kadar opazimo na pripomočku v namizju novo bazno postajo, lahko ob dotiku na pripomoček preverimo več informacij. Tako na zaslonu ne potrebujemo ikone za zagon aplikacije Lokator.

Poglavje 7

Sklepne ugotovitve

V diplomskem delu so bili pregledani koncepti in gradniki aplikacij za mobilne naprave. Spoznali smo se z zgodovino mobilne telefonije in mobilnih omrežij v Sloveniji. Predelali smo način zaznave podatkov na mobilnih platformah, geografsko lociranje in pridobivanje informacij o bazni postaji s spletno storitvijo RESTful. V delu je bil realiziran razvoj aplikacije Android za grafično prikazovanje trenutno uporabljane bazne postaje na mobilni napravi. Slednja zna razbrati in izračunati podatke iz bazne postaje. Ustvarjen spletni strežnik nam omogoča unikatno hranjenje podatkov o vseh baznih postajah različnih mobilnih operaterjev v Sloveniji. Za prikazovanje zemljevida na mobilni aplikaciji smo preučili programski paket Google Maps Android API. Za razvoj aplikacije je bilo osvojeno znanje programskih jezikov Java, XML, JSON in PHP.

Pomembnost diplomskega dela je v razvoju in izdelavi aplikacije Android. Poudarek je na povezavi aplikacije s spletnim strežnikom in prikazu geografskih podatkov, predstavljenih na zemljevidu. Prednost razvite aplikacije Lokator je v enostavnosti uporabe in namestitve. Namen aplikacije je, da uporabnikom omogoča enostaven prikaz njihove geografske lokacije in lokacije bazne postaje, na katero so prijavljeni. Sistem zna razpoznati nove ali posodobljene bazne postaje. S temi podatki skrbniki sledimo spremembam in posodabljamo celotno mrežo baznih postaj v Sloveniji.

Kljub temu da zastavljena aplikacija deluje dobro, smo tekom raziskav ugotovili, da jo je mogoče izboljšati. Trenutno nam slike baznih postaj uporabniki pošiljajo preko e-poštnega naslova. Mogoče bi bilo, da bi uporabniki pošiljali posnete fotografije baznih postaj skrbniku neposredno preko aplikacije. Aplikacijo bi lahko izboljšali tudi z administrativnimi orodji na spletnem strežniku, s pomočjo katerih bi lahko skrbnik zaznal nove bazne postaje in določil njihovo pozicijo. Ta orodja bi pridobivala podatke iz baze Open Cell ID in baze Google.

Mobilno aplikacijo Lokator smo že posredovali uporabnikom preko trgovine Google Play. Aplikacijo je možno naložiti brezplačno in ima na dan 25. oktober 2012 čez tisoč namestitev. Vsakodnevno prejemamo pritožbe in pohvale uporabnikov. Predlagane izboljšave smo realizirali. Odziv uporabnikov je zelo dober.

Literatura

[1] (2012) About the Eclipse Foundation. Dostopno na http://www.eclipse.org/org/

[2] (2012) Android Developer Tools. Dostopno na http://developer.android.com/tools/help/adt.html

[3] (2012) Android Developers - Package Index. Dostopno na http://developer.android.com/reference/packages.html

[4] (2012) Drugo četrtletje 2012: poročilo o razvoju trga elektronskih komunikacij. Dostopno na: http://www.apek.si/drugo-cetrtletje-2012:-porocilo-o-razvoju-trga-elektronskih-komunikacij

[5] (2012) Exploring the SDK. Dostopno na http://developer.android.com/sdk/exploring.html

[6] (2012) Google APIs Add-On. Dostopno na https://developers.google.com/android/add-ons/google-apis/

[7] (2012) Google Maps Android API - External Library. Dostopno na https://developers.google.com/maps/documentation/android/index

[8] (2012) How GSM Network Works. Dostopno na: http://www.techviral.com/networks/gsm-network-works/

[9] (2012) T. Korošec, Možnosti lociranja v mobilnih sistemih. Dostopno na http://www.lkn.fe.uni-lj.si/Seminarji/t_korosec.pdf

[10] (2012) Mobilna telefonija v Sloveniji. Dostopno na: http://www.mobilna-telefonija.com/

[11] (2012) OpenCellID. Dostopno na: http://www.opencellid.org/api

[12] (2012) UMTS Radio Parameter PlanningTechnical Guide. Dostopno na: http://www.scribd.com/doc/95846629/39640039-ZTE-WCDMA-Radio-Parameter-Planning-Technical-Guide

[13] (2012) Wikipedia: 1G. Dostopno na: http://en.wikipedia.org/wiki/1G

[14] (2012) Wikipedia: 2G. Dostopno na: http://en.wikipedia.org/wiki/2G

[15] (2012) Wikipedia: 3G. Dostopno na: http://en.wikipedia.org/wiki/3G

[16] (2012) Wikipedia: 4G. Dostopno na: http://en.wikipedia.org/wiki/4G

[17] (2012) Wikipedia: Android (Operationg System). Dostopno na http://en.wikipedia.org/wiki/Android_(operating_system)

[18] (2012) Wikipedia: Android Software Development. Dostopno na http://en.wikipedia.org/wiki/Android_software_development

[19] (2012) Wikipedia: Eclipse (software). Dostopno na http://en.wikipedia.org/wiki/Eclipse_(software)

[20] (2012) Wikipedia: History of mobile phones. Dostopno na: http://en.wikipedia.org/wiki/History_of_mobile_phones

[21] (2012) Wikipedia: Java (programming language). Dostopno na: http://en.wikipedia.org/wiki/Java_(programming_language)

[22] (2012) Wikipedia: JSON. Dostopno na http://en.wikipedia.org/wiki/Json

[23] (2012) Wikipedia: Mobile telephony. Dostopno na: http://en.wikipedia.org/wiki/Mobile_telephony

[24] (2012) Wikipedia: PHP. Dostopno na http://en.wikipedia.org/wiki/PHP

[25] (2012) Wikipedia: Representational state transfer. Dostopno na http://en.wikipedia.org/wiki/Representational_state_transfer

[26] (2012) Wikipedia: Satellite Navigation. Dostopno na http://en.wikipedia.org/wiki/Satellite_navigation

[27] (2012) Wikipedia: Triangulation. Dostopno na http://en.wikipedia.org/wiki/Triangulation

[28] (2012) Wikipedia: XML. Dostopno na http://en.wikipedia.org/wiki/XML