



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Никола Булатовић

**Једно решење клијентске апликације
на Андроид платформи за праћење
стања удаљених уређаја**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2012



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Никола Булатовић		
Ментор, МН:	Пап др Иштван		
Наслов рада, НР:	Једно решење клијентске апликације на Андроид платформи за праћење стања удаљених уређаја		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2012		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	<уписати статистику>		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кјуине речи, ПО:	праћење уређаја, андроид апликација		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду је описано једно решење клијентске апликације за праћење стања удаљених уређаја засноване на Андроид платформи. Апликација је реализована коришћењем Еклипсе (Eclipse) развојног окружења са Андроид СДК пакетом.		
Датум прихватања теме, ДП:			
Датум одbrane, ДО:			
Чланови комисије, КО:	Председник:	Ковачевић Др Јелена	
	Члан:	Поповић Др Мирослав	Потпис ментора
	Члан, ментор:	Пап др Иштван	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Nikola Bulatovic	
Mentor, MN:	Pap dr Istvan	
Title, TI:	One solution to the client application on the Android platform for the monitoring of remote devices	
Language of text, LT:	Serbian	
Language of abstract, LA:	Serbian	
Country of publication, CP:	Republic of Serbia	
Locality of publication, LP:	Vojvodina	
Publication year, PY:	2012	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	<upisati statistiku>	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	monitoring devices, android applications	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	This paper describes one solution to the client application for remote monitoring devices based on Android platform. The application is implemented using Eclipse development environment with the Android SDK package.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President: Member: Member, Mentor:	Kovačević Dr Jelena Popović Dr Miroslav Pap Dr Istvan
		Menthor's sign

Zahvalnost

Zahvaljujem se svom mentoru Prof. Dr Pap Ištvanu na podršci tokom izrade završnog (*bachelor*) rada.

Posebno se zahvaljujem Dejanu Stefanoviću na stručnoj pomoći i strpljenju prilikom izrade rada.

Na kraju zahvaljujem se svim svojim kolegama koji su svojim savetima pomogli u izradi rada.

SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove	2
2.1	Android aplikacija.....	3
3.	Koncept rešenja.....	6
3.1	Klasična Android aplikacija.....	6
3.2	Servis.....	7
3.3	Home screen widget	7
4.	Programsko rešenje.....	10
4.1	RemoteUnitMonitor	13
4.2	TabActivity.....	15
4.3	UnitActivity.....	17
4.3.1	MySimpleCursorAdapter.....	20
4.4	ServerNews	20
4.5	SettingsActivity.....	21
4.6	ListaActivity.....	23
4.6.1	Interactive ArrayAdapter i Model	24
4.7	UnitDetails	25
4.7.1	DataDetailsJson.....	27
4.8	UnitService	28
4.8.1	DataJson.....	29
4.9	ServerUrlData.....	29
4.10	RemoteWidgetProvider	30
4.10.1	WidgetService.....	31

4.11	RemoteWidgetConfig	32
5.	Rezultati	33
6.	Zaključak	35
7.	Literatura.....	36

SPISAK SLIKA

Slika 2.1 Glavne komponente Android operativnog sistema	2
Slika 3.1 Dijagram korišćenja	6
Slika 3.2 Redosled operacija pri pokretanju aplikacije, slučaj kada servis nije kreiran	8
Slika 3.3 Redosled operacija pri pokretanju widget-a, slučaj kada servis nije kreiran	8
Slika 3.4 Redosled operacija pri pokretanju widget-a, slučaj kada je servis kreiran	9
Slika 3.5 Redosled operacija pri pokretanju aplikacije preko widget-a	9
Slika 4.1 Razmena poruka u aplikaciji	12
Slika 4.2 RemoteUnitMonitor klasa	13
Slika 4.3 RemoteUnitMonitor aktivnost	14
Slika 4.4 Prozor za dodavanje nove URL adrese	15
Slika 4.5 TabActivity klasa	15
Slika 4.6 TabActivity kartice za promenu aktivnosti	16
Slika 4.7 UnitActivity klasa	17
Slika 4.8 UnitActivity aktivnost	19
Slika 4.9 UnitActivity filtriranje	19
Slika 4.10 MySimpleCursorAdapter klasa	20
Slika 4.11 ServerNews klasa	20
Slika 4.12 SettingsActivity klasa	21
Slika 4.13 SettingsActivity aktivnost	22
Slika 4.14 ListaActivity klasa	23
Slika 4.15 ListaActivity aktivnost	23
Slika 4.16 InteractiveArrayAdapter i Model klasa	24
Slika 4.17 UnitDetails klasa	25

Slika 4.18 UnitDetails aktivnost	26
Slika 4.19 DataDetailsJason klasa.....	27
Slika 4.20 UnitService klasa	28
Slika 4.21 DataJson klasa.....	29
Slika 4.22 ServerUrlData klasa	29
Slika 4.23 RemoteWidgetProvider klasa	30
Slika 4.24 Izgled widgeta.....	31
Slika 4.25 WidgetService klasa.....	31
Slika 4.26 RemoteWidgetConfig klasa	32
Slika 4.27 RemoteWidgetConfig aktivnost.....	32
Slika 5.1 Server klasa	33

SPISAK TABELA

Tabela 4.1 Klase projektnog zadatka	11
Tabela 4.2 Poruke koje se razmenjuju	12

SKRAĆENICE

SDK	- <i>Software Development Kit</i> , Skup računarskih alatki
REST	- <i>REpresentational State Transfer</i> , Tip arhitekture programske podrške za distribuirane sisteme
URL	- <i>Uniform Resource Locator</i> , Referenca na internet resurs
JSON	- <i>JavaScript Object Notation</i> , Tekstualni otvoren standard
API	- <i>Application Programming Interface</i> , Aplikativna programska sprega
BSD	- <i>Berkeley Software Distribution</i> , Unix operativni sistem
SGL	- <i>Scalable Graphics Library</i> , Grafički pod sistem

1. Uvod

U ovom radu opisana je realizacija jednog rešenja klijentske aplikacije za praćenje stanja udaljenih uređaja zasnovana na Android platformi. Aplikacija je realizovana korišćenjem Eclipse rezvojnog okruženja sa Android SDK paketom.

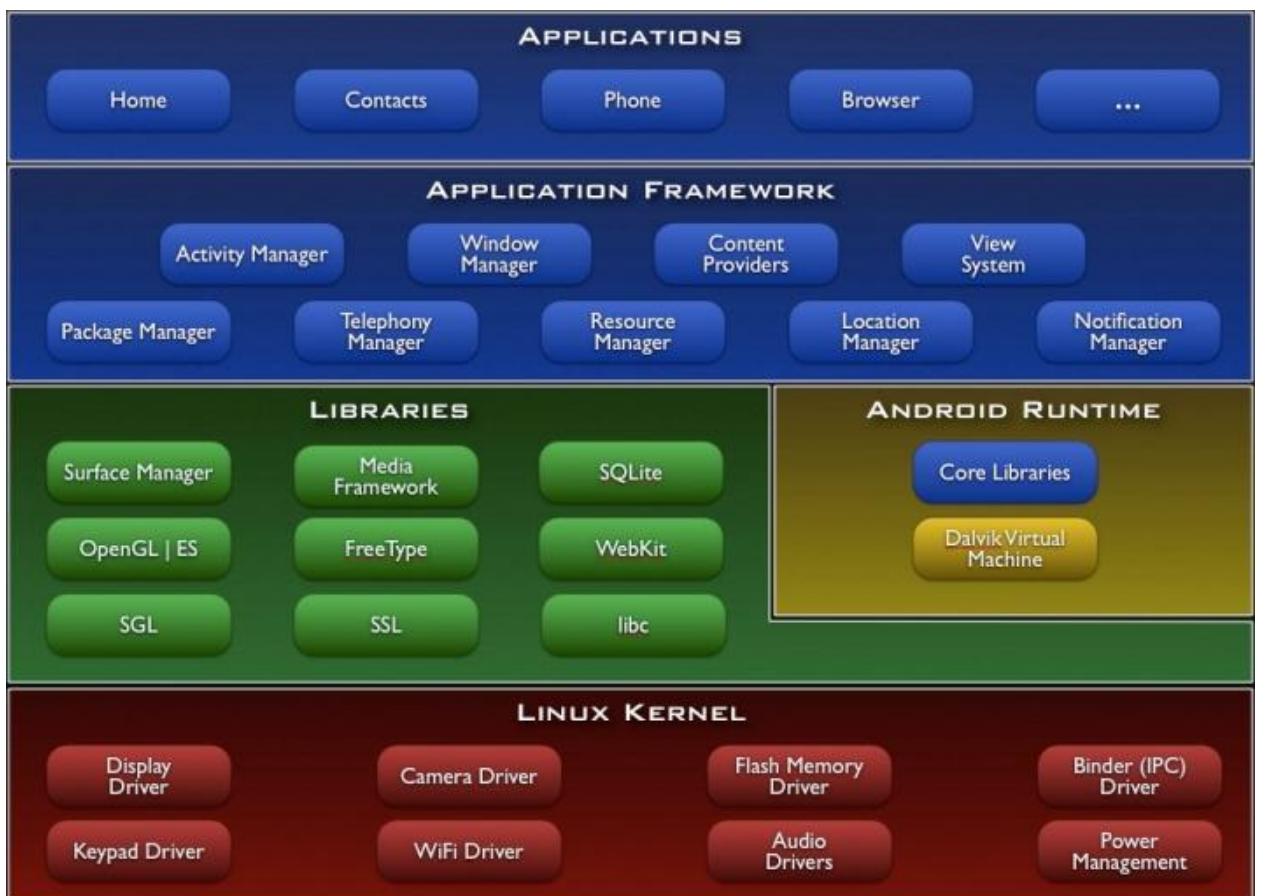
Aplikacija se sastoji iz dva dela:

1. Klasična android aplikacija, omogućava prijavljivanje korisnika na server koji obezbeđuje REST servis za praćenje stanja uređaja. Dobijeni podaci o uređajima prikazuju se u formi liste sa osnovnim podacima. Klikom na uređaj u listi prikazuju se detalji uređaja. Realizovane su osnovne opcije – dodavanje novog server URL-a, brisanje postojećih server URL-ova, podešavanje učestanosti pristupa serveru radi osvežavanja informacija o uređajima, podešavanje home screen widget-a. Aplikacija prikazuje najnovije vesti sa servera.
2. Home screen widget prikazuje stanje do pet najznačajnijih uređaja.

Komunikacija između aplikacije i servera obavlja se preko servisa koji je neprekidno aktivran na Android uređaju. Servis komunicira sa serverom preko Apache HttpClient biblioteke. Informacije koje server dostavlja aplikaciji su u JSON formatu. Za parsiranje JSON datoteke koristi se Gson biblioteka.

2. Teorijske osnove

Android je paket programske podrške za mobilne uređaje koji uključuje operativni sistem, srednji sloj (*eng.middleware*) i ključne aplikacije. Android SDK obezbeđuje alatke i API potreban za razvoj aplikacija na Android platformi korišćenjem Java programskog jezika.



Slika 2.1 Glavne komponente Android operativnog sistema

Android obuhvata skup C i C++ biblioteka koje koriste razne komponente Android sistema. Ove mogućnosti su dostupne programeru kroz Android aplikacione okvire. Neke od osnovnih biblioteka:

- Sistemska C biblioteka – BSD izvedena implementacija standardne C sistemske biblioteke, podešene za namenske Linux bazirane uređaje.
- Medija biblioteka – podrška za reprodukciju i snimanje mnogih popularnih audio i video formata, kao i statičnih slika, uključujući i MPEG4, X.264, MP3, AAC, AMP, JPG, PNG.
- Upravljač površine – upravlja pristupu podsistemu prikaza.
- Biblioteke Web pretraživača – moderan web pretraživač.
- SGL – osnovna 2D grafička podrška.
- 3D biblioteke – zasnovane na OpenGL EC 1.0 API-ju. Biblioteke koriste 3D ubrzanje fizičke arhitekture gde je dostupno ili veoma optimizovanu 3D programsku podršku.
- Slobodan tip – bitmape i vektore iscrtavanja (*eng. Free types - bitmap and vector drawing*).
- SQLite – moćan i lak alat za relacione baze podataka dostupne za sve aplikacije.

Android obuhvata skup osnovnih biblioteka koje obezbeđuju najveći deo funkcionalnosti dostupne u osnovnim bibliotekama Java programskog jezika.

Svaka Android aplikacija radi u svom procesu, sa sopstvenom instancom Dalvik virtualne mašine. Dalvik je napisan tako da uređaj može efikasno da pokrene više virtualnih mašina. Dalvik virtualna mašina izvršava Dalvik izvršne datoteke u (.dex) formatu, optimizovane za minimalnu potrošnju memorije. Virtualna mašina pokreće klase prevedene Java programskim prevodiocem u .dex formatu pomoću „dx“ alata.

Dalvik virtualna mašina oslanja se na Linux jezgro za osnovne funkcionalnosti kao što su paralelizam i pristup memoriji niskog nivoa.

Android se oslanja na Linux verziju 2.6 za osnovne usluge sistema kao što su bezbednost, upravljanje memorijom, upravljanje procesima, mrežno skladište i drajveri. Jezgro takođe spaja apstrakcije između fizičke arhitekture i ostatka programskog paketa.

2.1 Android aplikacija

Android aplikacije se pišu u Java programskom jeziku. Android SDK alati prevode kod, zajedno sa svim podacima i potrebnim datotekama u Android paket tj. u arhivsku datoteku sa

ekstenzijom .apk. Ovu datoteku Android koristi da instalira aplikaciju. Jednom instalirana svaka Android aplikacija živi u sopstvenoj sigurnoj kutiji (*eng security sandbox*).

Android operativni sistem je Linuks višekorisnički sistem u kojem svaka aplikacija predstavlja drugog korisnika. Svakoj aplikaciji sistem dodeljuje jedinstven korisnički ID (ID se koristi od strane sistema i nije dostupan aplikaciji). Sistem postavlja dozvole za sve datoteke u jednoj aplikaciji tako da samo korisnici sa odgovarajućim identifikacionim brojevima (*ID*) mogu da im pristupe.

Svaki proces ima svoju virtualnu mašinu, tako da se kod aplikacije pokreće izolovan od drugih aplikacija. Po pravilu svaka aplikacija se pokreće u nezavisnom Linux procesu. Proces se pokreće kada jedna od komponenti aplikacije treba da se izvrši. Android gasi proces kada više nije potreban ili kada je potrebno oporaviti memoriju za druge aplikacije.

Na ovaj način Android primenjuje princip najmanje privilegije. Odnosno svaka aplikacija ima pristup komponentama koje su joj potrebne za izvršenje svog zadatka. Ovo stvara vrlo bezbedno okruženje u kojem aplikacija ne može da pristupi delovima sistema za koje nema odobrenje.

Ipak postoje načini da aplikacije dele podatke sa drugim aplikacijama i da aplikacije imaju pristup uslugama sistema:

- Moguće je organizovati da dve aplikacije dele isti Linux korisnički identifikacioni broj (*ID*), pri čemu su u stanju da pristupe datotekama druge aplikacije. Takođe je moguće organizovati da aplikacije rade u istom Linux procesu i dele istu virtualnu mašinu.
- Aplikacija može da zatraži dozvolu za pristup podacima kao što su kontakti korisnika, SMS poruke, kamera, i drugo. Sve dozvole moraju biti odobrene od strane korisnika u trenutku instalacije aplikacije.

Aplikacione komponente su osnovni gradivni blokovi Android aplikacije. Svaka komponenta je drugačija tačka preko koje sistem može da pristupi aplikaciji. Nisu sve komponente stvarne polazne tačke za korisnika, a neke zavise jedna od druge, ali svaka od njih postoji kao poseban entitet i ima specifičnu ulogu. Postoje četiri različite vrste aplikacionih komponenti. Svaki tip ima jasnu svrhu i poseban životni ciklus koji definiše kako se komponenta kreira i uništava.

- Aktivnosti (*eng.Activity*) predstavljaju jedan prikaz na ekranu sa korisničkom spregom. Aplikacija može imati više aktivnosti koje obično rade zajedno da bi formirale jedinstven utisak o aplikaciji kod korisnika. Ipak svaka od aktivnosti je nezavisna i može se samostalno pokrenuti u zavisnosti od situacije i potrebe.

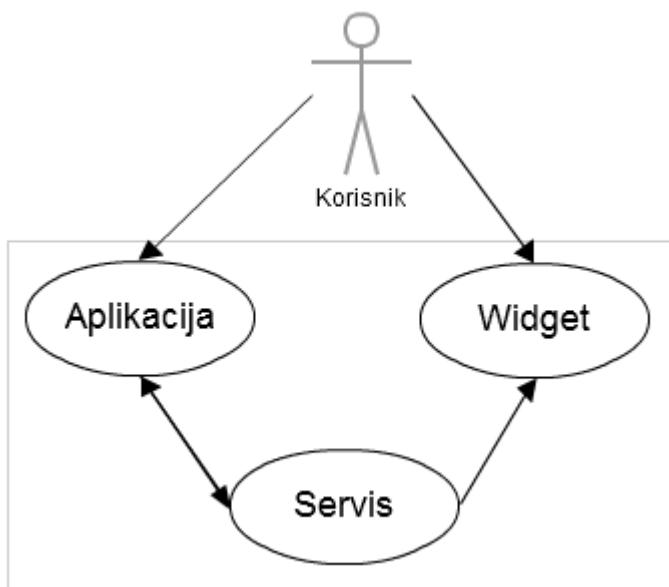
- Servis (*eng.Service*) je komponenta koja radi u pozadini i obavlja dugotrajne radnje ili radi za udaljene procese. Servis ne pruža korisničku spregu. Druge komponente mogu pokrenuti servis ili se povezati na njega kada im je potrebno da izvrše uslugu koju on obezbeđuje.
- Dobavljač sadržaja (*eng.Content provider*) omogućava deljenje podataka između aplikacija. Moguće je skladištiti podatke u sistemu datoteka, SQLite bazi podataka, na internetu ili bilo kojoj drugoj lokaciji za skladištenje kojoj aplikacija može da pristupi.
- Pošiljaoc prijemnik (*eng. Broadcast receiver*) je komponenta Android sistema koja omogućava slanje i prijem poruka kroz ceo sistem. Mnoge najave potiču iz sistema kao na primer da je ekran isključen, baterija prazna.

Pre nego što Android sistem može da pokrene neku komponentu aplikacije, sistem mora da zna da ta komponenta postoji čitajući *AndroidManifest.xml* datoteku aplikacije. Sve komponente aplikacije se moraju definisati u ovoj datoteci. Pored deklarisanja komponenti u ovoj datoteci realizuju se i sledeće opcije:

- Identifikacija korisničkih dozvola koje aplikacija zahteva (kao što su pristup internetu i sl).
- Definiše minimalni API nivo koji uređaj na koji želimo da instaliramo aplikaciju mora da poseduje da bi se aplikacija uspešno izvršavala.
- Definiše koje resurse fizičke arhitekture aplikacija koristi, kao što su kamera, bluetooth i sl.

3. Koncept rešenja

Rešenje aplikacije sastoji se iz tri nezavisna dela koja međusobno komuniciraju da bi stvorili jedinstven doživljaj aplikacije kod korisnika. Korisnik direktno komunicira sa klasičnom Android aplikacijom i home screen widget-om. Treći deo aplikacije je servis zadužen za komunikaciju sa serverom i snabdevanje druga dva dela aplikacije informacijama o uređajima koji se prate.



Slika 3.1 Dijagram korišćenja

3.1 Klasična Android aplikacija

Pokretanjem aplikacije proverava se da li je servis kreiran, u slučaju da servis postoji pokreće se aktivnost sa listom uređaja koji se prate. Ukoliko servis nije kreiran od korisnika se zahteva da odabere URL servera na koji želi da se poveže, unese korisničko ime i šifru za odabrani server. Ako prijavljivanje uspe server će aplikaciji dostaviti identifikacioni broj. Nakon

uspešne prijave otvara se aktivnost sa četiri taba. Prvi tab sadrži aktivnost sa listom uređaja koji se prate preko odabranog servera. U ovoj listi date su samo osnovne informacije o uređajima koji se prate. Klikom na uređaj u listi otvara se aktivnost sa detaljima tog uređaja. Aktivnost sadrži opciju filtriranja liste uređaja. Klikom na opciju filter otvara se pomoćni prozor u kojem se biraju parametri filtriranja. Listu je moguće filtrirati po sledećim parametrima:

- Status veze
- Grupa uređaja
- Stanje uređaja
- Ime uređaja

U drugom tabu nalazi se aktivnost sa listom vesti. Treći tab sadrži aktivnost sa osnovnim opcijama aplikacije. Realizovane su sledeće opcije:

- Dodavanje novog server URL-a
- Brisanje postojećeg server URL-a
- Promena učestanosti osvežavanja informacija o uređajima

Četvrti tab sadrži aktivnost za odabir uređaja koje korisnik želi da prati na home screen widgetu. Aktivnost je realizovana u vidu liste u kojoj su prikazani svi dostupni uređaji. Korisnik odabira pet uređaja koje želi da prati.

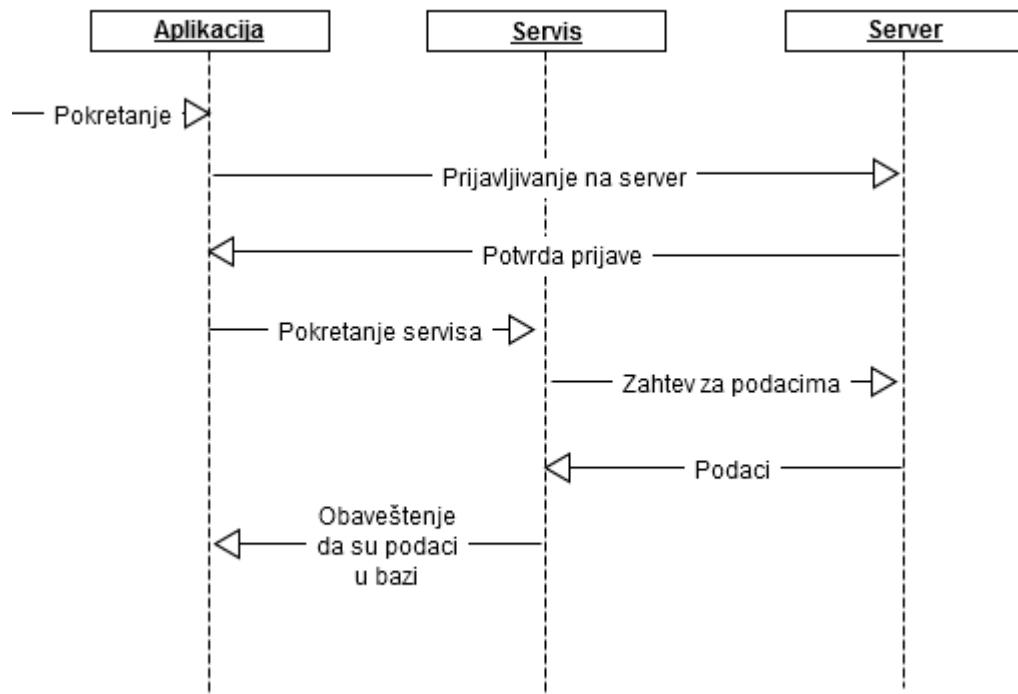
Korisnik može da raskine vezu sa serverom odabirom opcije *Log out* klikom na taster *Menu* ili da samo napusti program odabirom opcije *Exit*.

3.2 Servis

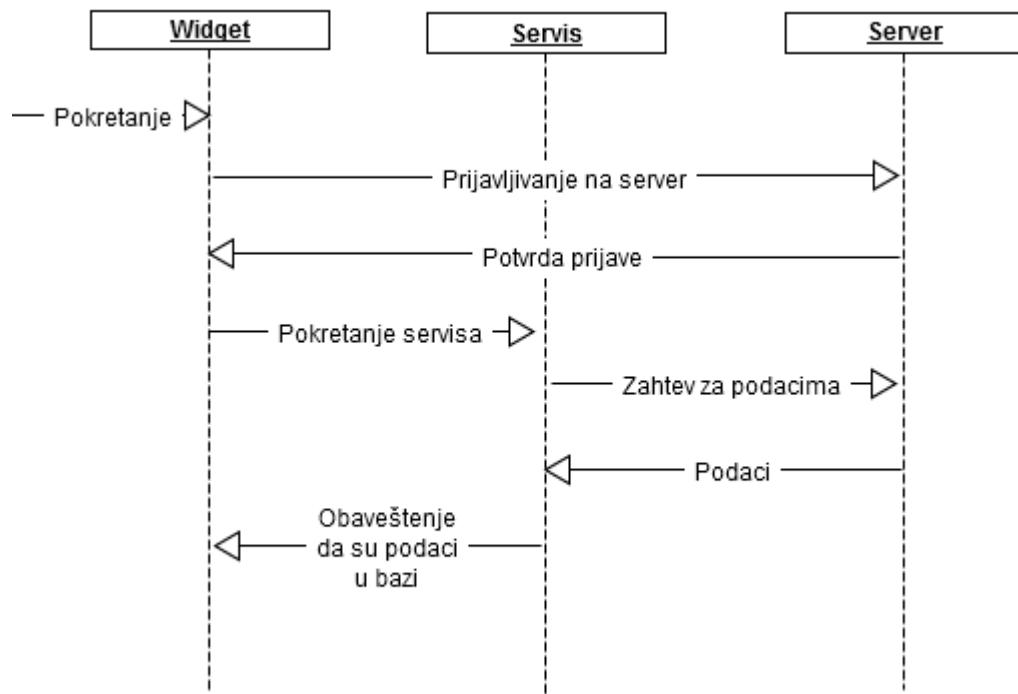
Servis je komponenta aplikacije koja nije vidljiva korisniku. Ona obavlja svoj zadatak u pozadini. Zadatak servisa u ovom slučaju je da periodično osvežava informacije o uređajima koji se prate pomoću API funkcija servera. Servis kreira aplikacija ili widget u zavisnosti ko je prvi pokrenut. Pri kreiranju servisa prosleđuje mu se identifikacioni broj koji je dobijen pri prijavljivanju na server. Pomoću ovog broja servis se autentificuje na server. Servis se može isključiti odabirom opcije *Log out* u aplikaciji.

3.3 Home screen widget

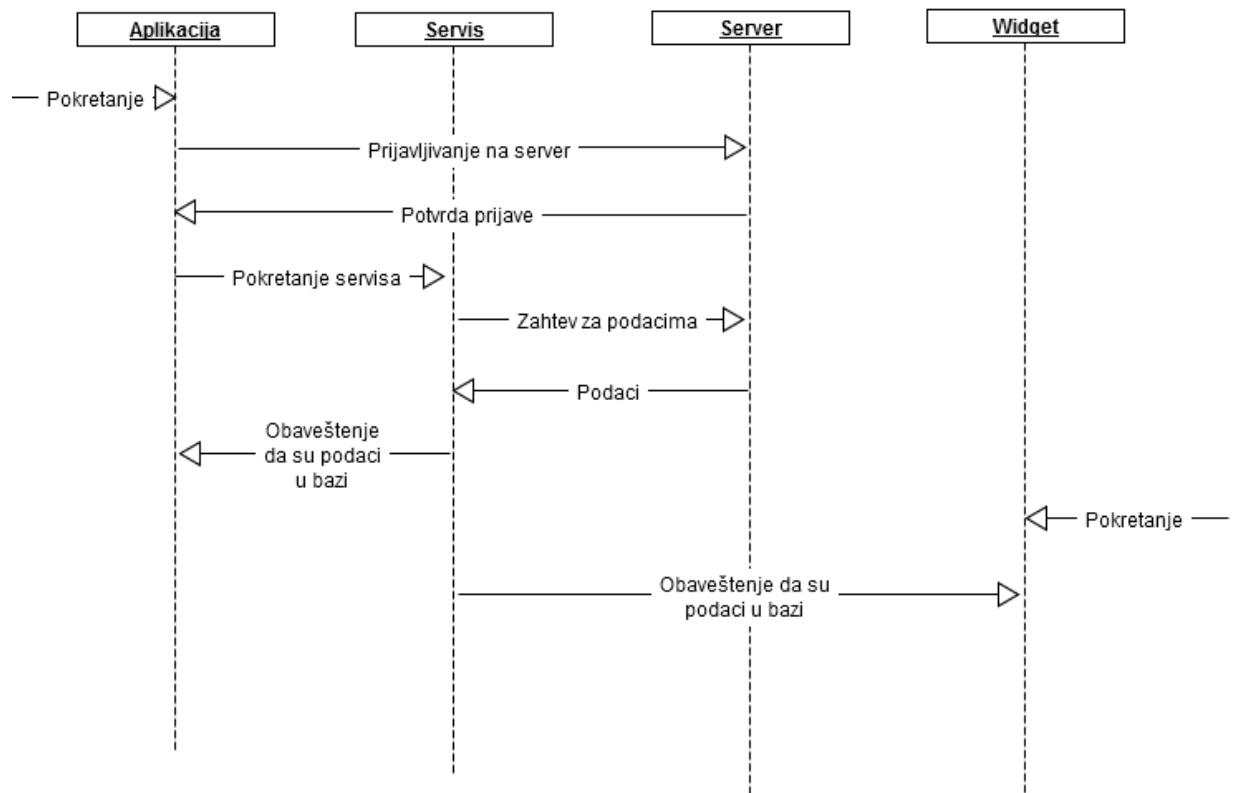
Ova komponenta aplikacije koristi se za praćenje do pet najznačajnijih uređaja na početnom ekranu Android uređaja. Ukoliko servis ne postoji u trenutku kreiranja widget-a otvara se aktivnost u kojoj se od korisnika traži da se prijavi na željeni server. Ako prijavljivanje uspe pokreće se servis kome se prosleđuje dobijeni identifikacioni broj. Widget prikazuje osnovne informacije o uređajima. Klikom na widget otvara se Android aplikacija. Iz aplikacije je moguće kontrolisati koje uređaje prikazuje widget.



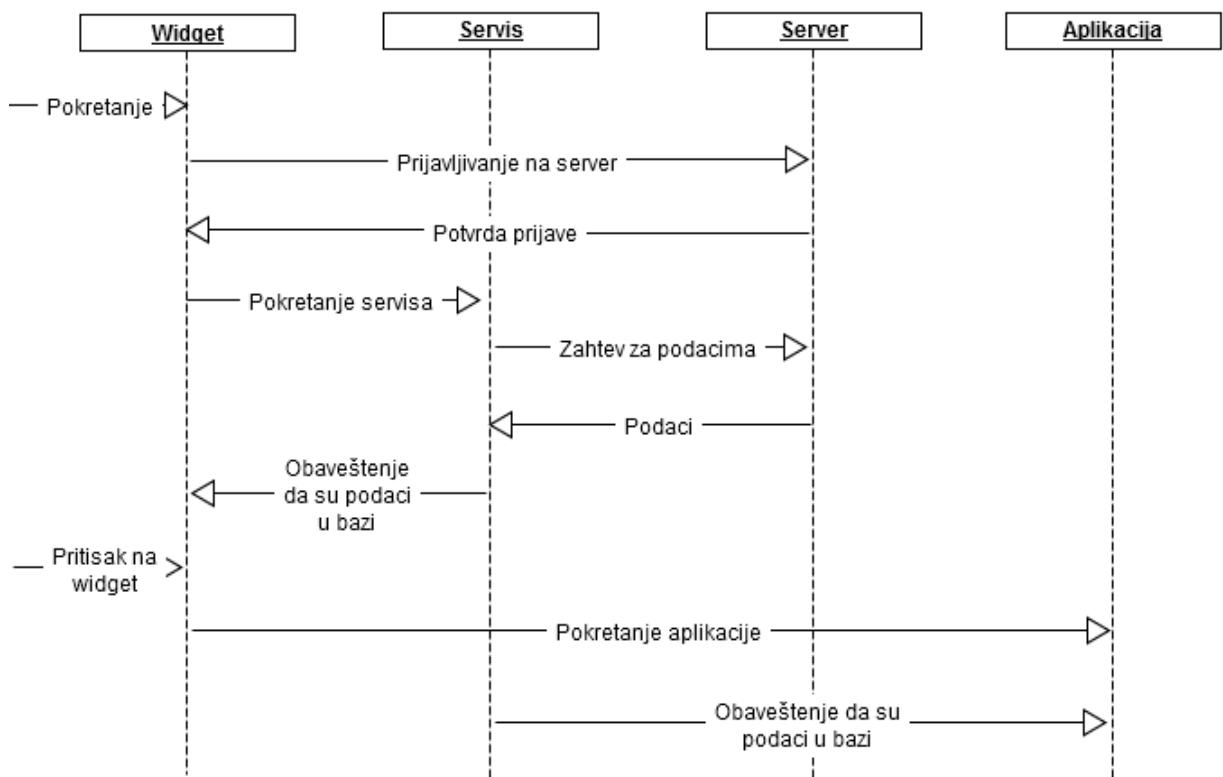
Slika 3.2 Redosled operacija pri pokretanju aplikacije, slučaj kada servis nije kreiran



Slika 3.3 Redosled operacija pri pokretanju widget-a, slučaj kada servis nije kreiran



Slika 3.4 Redosled operacija pri pokretanju widget-a, slučaj kada je servis kreiran



Slika 3.5 Redosled operacija pri pokretanju aplikacije preko widget-a

4. Programsко рење

Projektni zadatak realizovan je korišćenjem Eclipse razvojnog okruženja sa ADT Plugin-om i Android SDK Tools-om. U ovom poglavlju opisani su detalji реализације класа коришћених у изради апликације.

Klase aplikacije	Opis klase
RemoteUnitMonitor.java	Klasa sa početnom aktivnosti programa, sadrži formu za prijavljivanje korisnika na server.
TabActivity.java	Klasa koja nasledjuje TabActivity klasu. Realizuje četiri taba, svaki prikazuje različitu aktivnost.
UnitActivity.java	Aktivnost prikazuje listu uređaja koji se prate preko servera. Pokreće se unutar Tab aktivnosti.
ServerNews.java	Aktivnost sa listom vesti sa servera. Pokreće se unutar Tab aktivnosti.
SettingsActivity.java	Aktivnost u kojoj su realizovane opcije. Pokreće se unutar Tab aktivnosti.
ListaActivity.java	Aktivnost sa listom uređaja, u ovoј aktivnosti korisnik bira uređaje koje želi da prati na widgetu. Pokreće se unutar Tab aktivnosti.
UnitDetails.java	Aktivnost koja prikazuju detaljne informacije o izabranom uređaju.

UnitService.java	Klasa koja nasleđuje Servis klasu. Realizuje komunikaciju sa serverom, periodično osvežavanje podataka.
ServerUrlData.java	Klasa koja realizuje rad sa bazom podataka.
MySimpleCursorAdapter.java	Nasleđuje SimpleCursorAdapter klasu. Koristi se za prikaz podataka u listi UnitActivity aktivnosti.
InteractiveArrayAdapter.java	Nasleđuje ArrayAdapter klasu. Koristi se za prikaz podataka u listi ListaActivity aktivnosti.
Model.java	Pomoćna klasa, koristi se za smeštanje podataka o uređajima kod ListActivity klase.
DataJson.java	Klasa za deserializaciju podataka o uređajima.
DataDetailsJson.java	Klasa za deserializaciju detaljnih podataka o uređajima.
RemoteWidgetProvider.java	Klasa koja realizuje home screen widget.
RemoteWidgetConfig.java	Konfiguraciona aktivnost za widget.
WidgetService.java	Servis za popunjavanje widget-a potrebnim podacima.

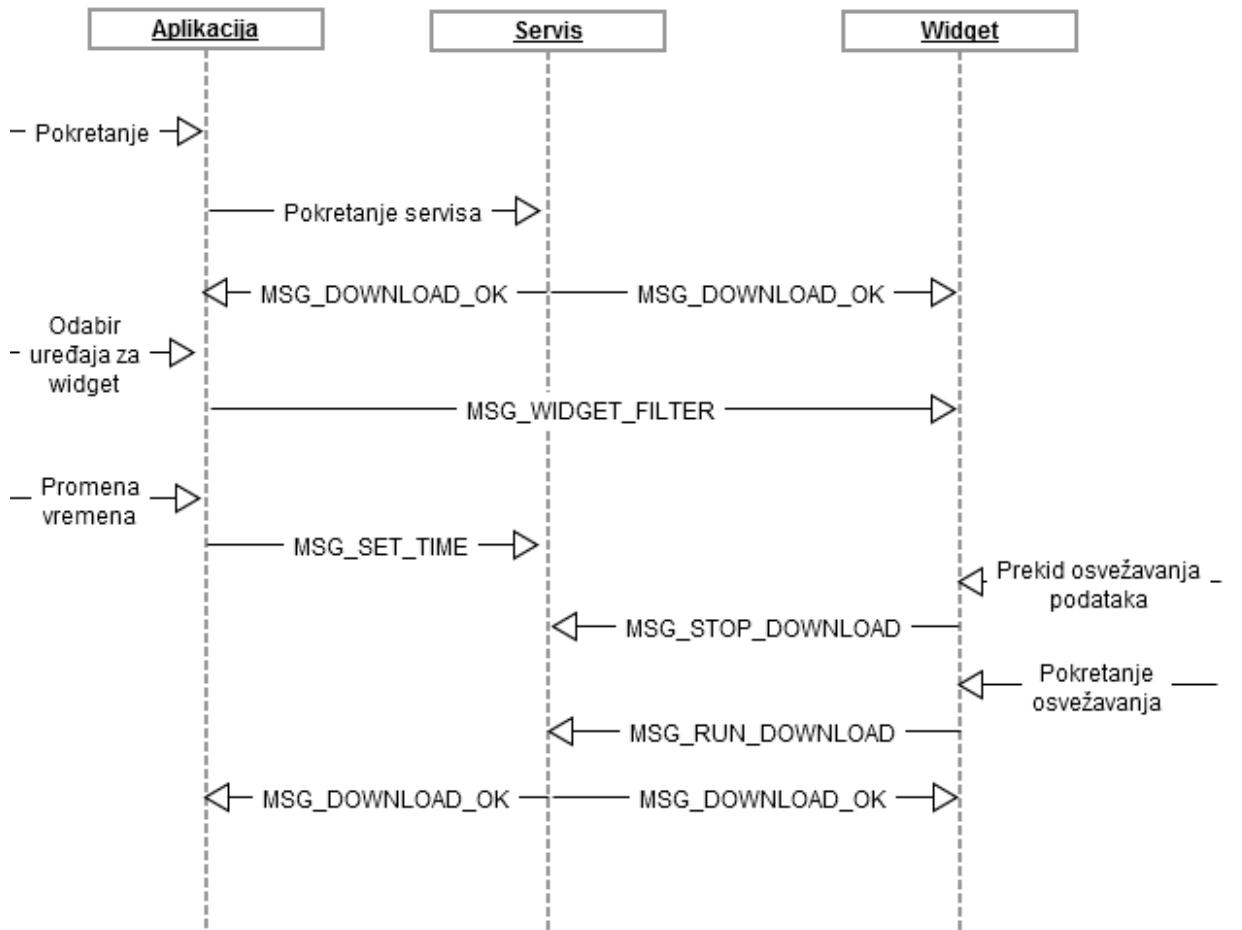
Tabela 4.1 Klase projektnog zadatka

Delovi aplikacije međusobno komuniciraju slanjem i primanjem poruka, tipovi poruka koje se razmenjuju:

Poruka	Opis
MSG_SAY_END	Signalizira servisu da treba da završi svoj rad.
MSG_DOWNLOAD_OK	Obaveštenje da su podaci osveženi novim podacima sa servera.
MSG_SET_TIME	Poruka promene vremena osvežavanja. Sadrži novo vreme osvežavanja podataka.
MSG_GET_TIME	Poruka na koju servis odgovara trenutnim vremenom osvežavanja.
MSG_GET_BACK_TIME	Odgovor na poruku zahteva za vremenom osvežavanja. Sadrži trenutno vreme osvežavanja podataka.
MSG_GET_SID	Zahtev za identifikacionim brojem.

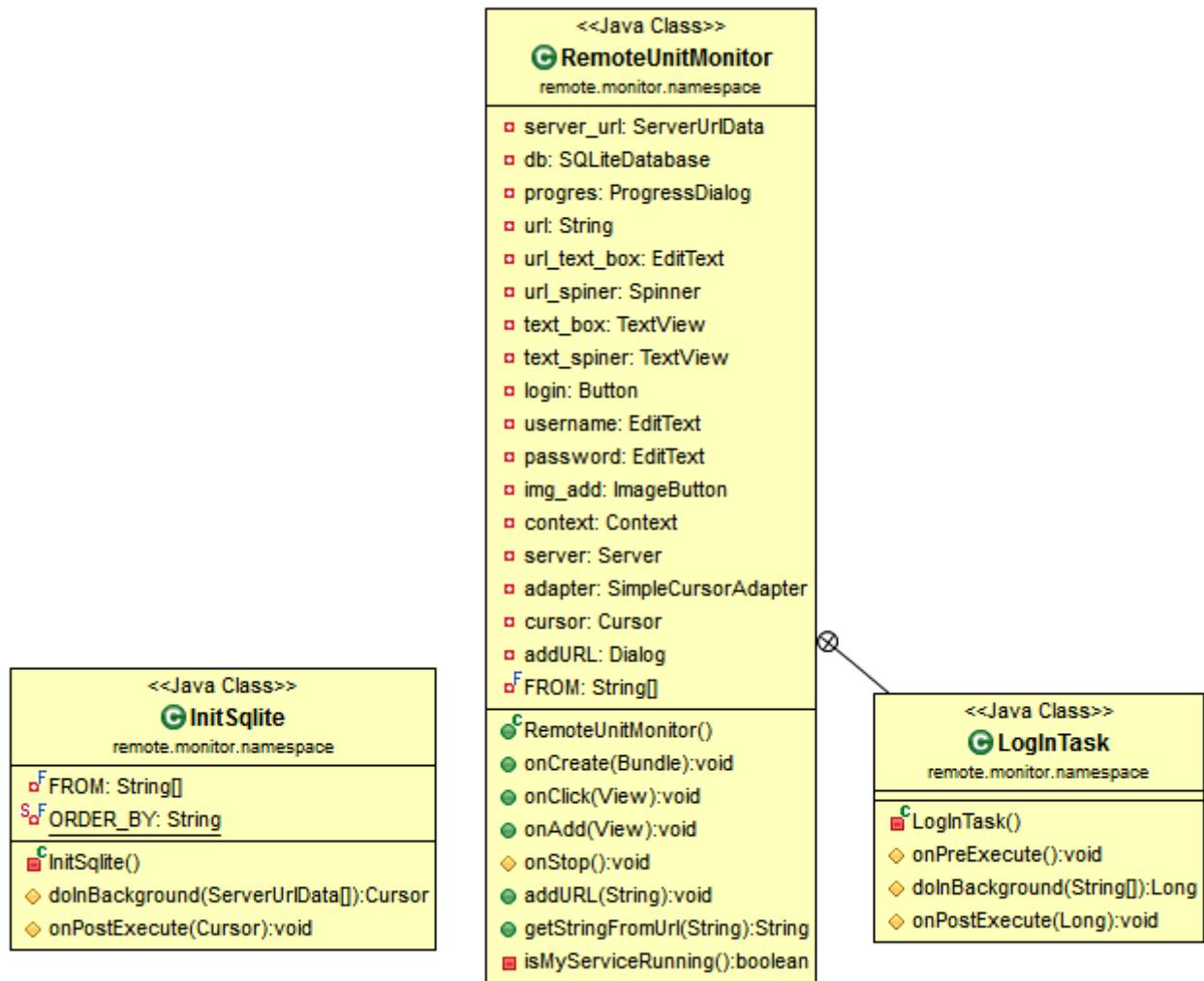
MSG_SET_SID	Odgovor na zahtev za identifikacionim brojem.
MSG_STOP_DOWNLOAD	Poruka prekida komunikacije sa serverom.
MSG_RUN_DOWNLOAD	Ponovno uspostavljanje veze sa serverom.
MSG_WIDGET_FILTER	Poruka sa izabranim uređajima koje korisnik želi da prati na widgetu.

Tabela 4.2 Poruke koje se razmenjuju



Slika 4.1 Razmena poruka u aplikaciji

4.1 RemoteUnitMonitor



Slika 4.2 RemoteUnitMonitor klasa

Klasa realizuje aktivnost aplikacije koja se prva pokreće. Na početku izvršavanja proverava se da li servis koji komunicira sa serverom postoji pomoću metode `isMyServiceRunning()`. Ako je servis već kreiran pokreće se Tab aktivnost, dok se ova aktivnost završava. Tab aktivnosti prosleđuje se URL adresa servera, dok se kao identifikacioni broj prosleđuje nula.

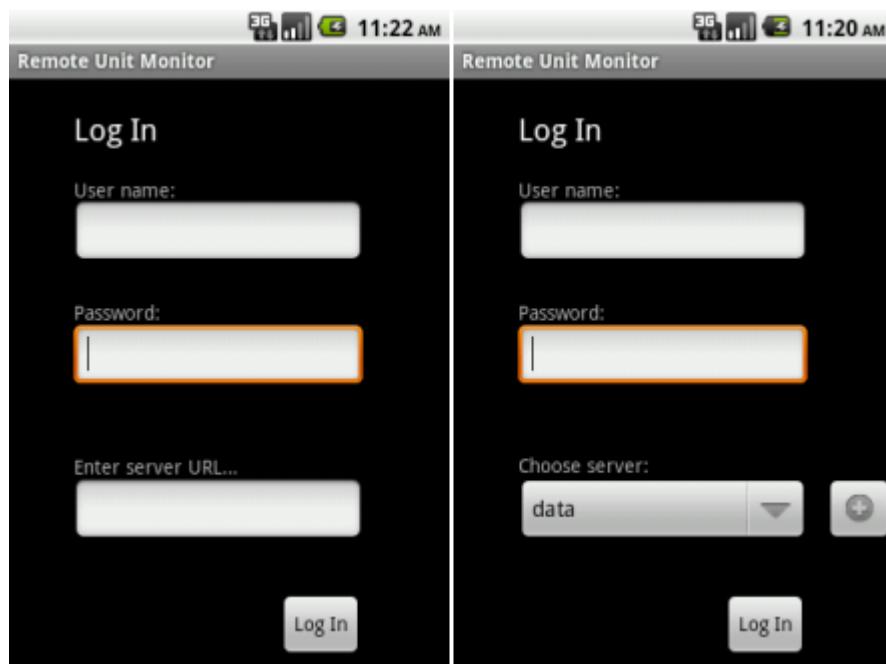
U drugom slučaju kada servis nije kreiran aktivnost isertava formu za prijavljivanje korisnika na server. Od korisnika se traži da unese korisničko ime i šifru za željeni server. Ako u bazi podataka ne postoji ni jedna URL adresa od korisnika se traži da unese novu adresu u odgovarajuće polje za unos teksta. URL adresa biće sačuvana u bazi klikom na dugme *Log In* pomoću metode `addURL(String url)`. U drugom slučaju kada u bazi podataka postoje URL adrese servera korisnik može da izabere jednu od postojećih iz padajućeg menija. Takođe realizovana je opcija dodavanja nove URL adrese ukoliko se željena adresa ne nalazi u

padajućem meniju. Pri odabiru ove opcije otvara se novi prozor u formi dialoga i od korisnika se traži da unese URL adresu servera. Klikom da dugme *OK* potvrđuje se unos nove adrese.

Kada su svi podaci ispravno uneti klikom na dugme *LogIn* poziva se *LogInTask* asinhrona aktivnost. Asinhrona aktivnost (*eng. AsyncTask*) je nasledna android klasa koja svoj zadatak obavlja u pozadini i tako ne opterećuje aplikaciju. U ovom slučaju ona ostvaruje komunikaciju sa serverom, prosleđuje mu potrebne podatke i čeka odgovor servera. Ako su podaci korisničko ime i šifra odgovarajući server vraća identifikacioni broj koji se od tog momenta koristi u komunikaciji između aplikacije i servera. U suprotnom od korisnika se zahteva da ponovo unese podatke.

Nakon uspešne prijave pokreće se Tab aktivnost, njoj se prosleđuje identifikacioni broj za komunikaciju sa serverom i URL adresa servera. *RemoteUnitMonitor* aktivnost se zatvara.

InitSqlite asinhrona aktivnost služi za inicijalizaciju baze podataka sa kojom se radi i punjenje padajućeg menija sa URL adresama iz baze ako one postoje.

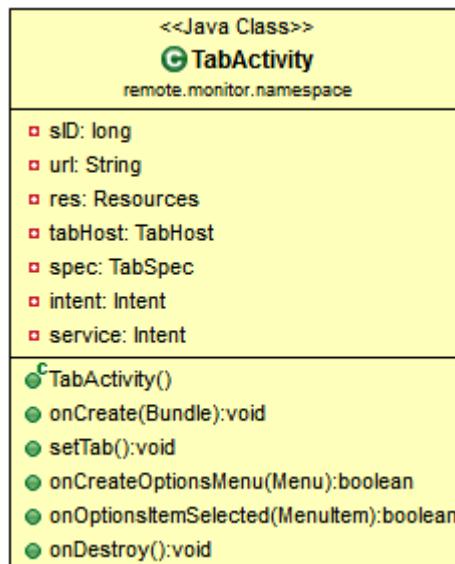


Slika 4.3 RemoteUnitMonitor aktivnost



Slika 4.4 Prozor za dodavanje nove URL adrese

4.2 TabActivity



Slika 4.5 TabActivity klasa

Tab aktivnost nasleđuje *TabActivity* android klasu koja omogućava da se u okviru ove aktivnosti pokrene više drugih aktivnosti. Aktivnostima koje se dodaju u Tab aktivnost pristupa se preko kartica, po jedna kartica za svaku instanciranu aktivnost.

Na početku izvršenja Tab aktivnosti proverava se da li je servis kreiran. Ako servis ne postoji kreira se novi kome se prosleđuje identifikacioni broj i URL adresa servera. Da li servis postoji proverava se pomoću identifikacionog broja, ako je identifikacioni broj dobijen od RemoteUnitMonitor aktivnosti nula zaključuje se da servis postoji, u suprotnom kreiramo novi.

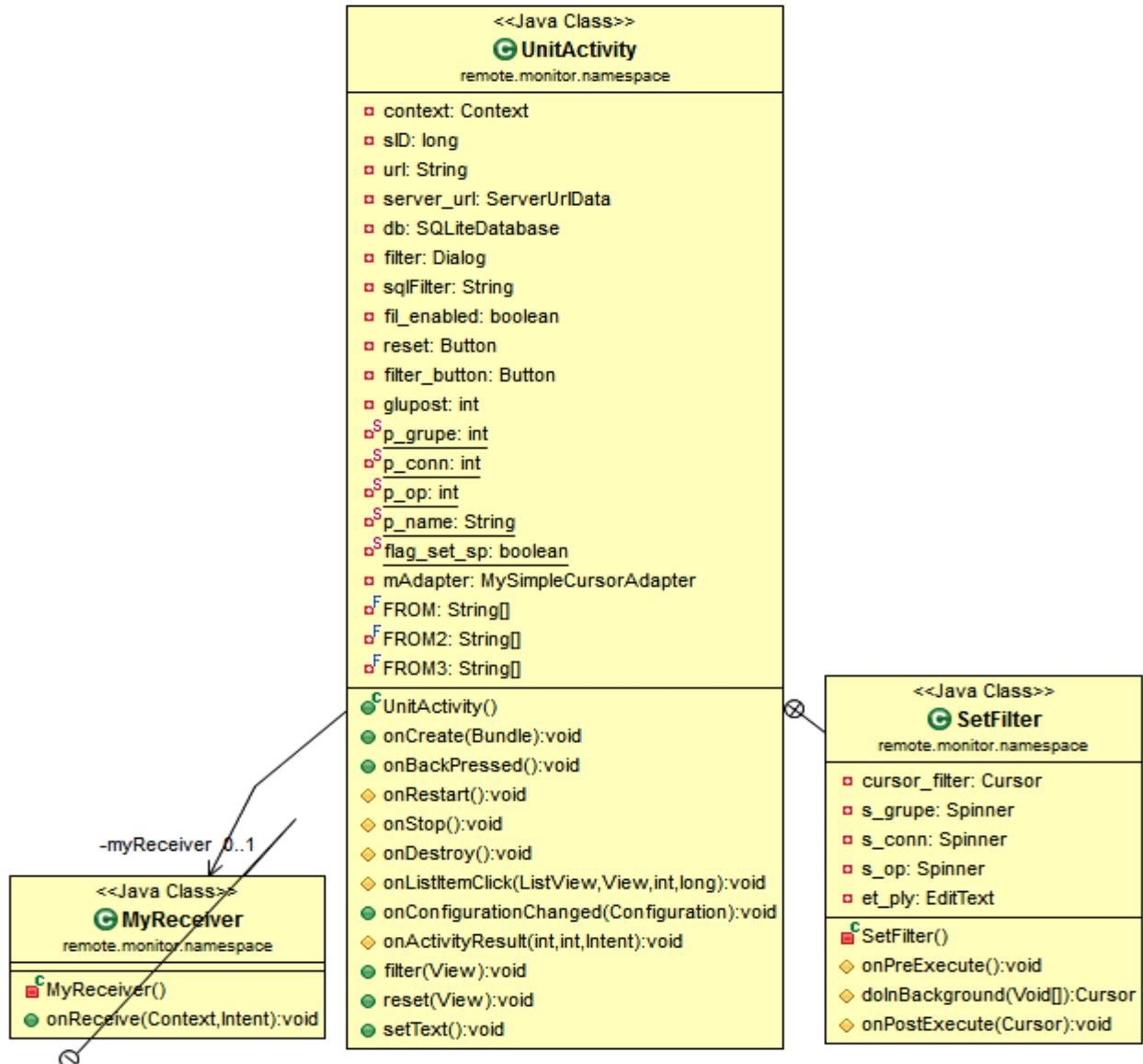
U okviru Tab aktivnosti pokreću se četiri aktivnosti: UnitActivity, ServiceNews, SettingsActivity, ListActivity. Dodavanje ovih aktivnosti u Tab aktivnost realizovano je u metodi *setTab()*.

U Tab aktivnosti realizovan je meni predefinisanjem ugrađenih metoda *onCreateOptionsMenu()* i *onOptionsItemSelected()*. Omogućene su dve opcije: *Exit* – izlaz iz aplikacije i *LogOut* – raskida vezu sa serverom, zatvara servis i pokreće ponovo početnu aktivnost programa.



Slika 4.6 TabActivity kartice za promenu aktivnosti

4.3 UnitActivity



Slika 4.7 UnitActivity klasa

UnitActivity klasa nasleđuje ListActivity Android klasu. Ova klasa omogućava da se podaci u aktivnosti prikazuju u formi liste, gde se svaki red liste popunjava željenim podacima.

UnitActivity je podešen da može da primi poruke koje šalju drugi delovi aplikacije, označenu sa *MSG_DOWNLOAD_OK* ili *MSG_SET_SID*. Prijem poruka iz drugih delova aplikacije realizuje se u klasi MyReceiver koja nasledjuje android klasu BroadcastReceiver. Kada primi poruku poziva se predefinisana metoda ove klase *onReceive()* u kojoj se poruka obradi.

Na početku izvršavanja aktivnost proverava da li je servis kreiran ranije ili je upravo kreiran pri pokretanju Tab aktivnosti. Ukoliko servis postoji od ranije poziva se metoda za punjenje liste sa podacima o uređajima *setText()* i šalje poruku servisu *MSG_GET_SID* kojom se zahteva identifikacioni broj od servisa. U suprotnom ako je servis upravo kreiran aktivnost čeka

poruku od servisa *MSG_DOWNLOAD_OK* koja signalizira aktvnosti da su podaci skinuti sa servera i smešteni u bazu podataka. Nakon ove poruke poziva se metoda *setText()* i lista se popunjava novim podacima.

Metoda *setText()* čita podatke iz baze podataka i pomoću klase MySimpleCursorAdapter listu puni podacima. Sledеći podaci se upisuju u red liste:

- Redni broj uređaja
- Naziv uređaja
- Status veze
- Grupa uređaja
- Stanje uređaja

Klikom na uređaj u listi otvara se aktivnost sa detaljnim podacima o uređaju. Ovo je realizovano predefinisanjem ugrađene metode *onListItemClick()*. Aktivnosti koja prikazuje detalje uređaja prosleđuju se identifikacioni broj, URL adresa servera, redni broj uređaja i ime uređaja.

UnitActivity realizuje opciju filtriranja liste. Klikom na dugme *Filter* otvara se prozor u formi dialoga sa opcijama za filtriranje liste. Listu je moguće filtrirati po sledećim parametrima:

- Grupa uređaja
- Status veze
- Stanje uređaja
- Ime uređaja

Kompletna realizacija funkcije filtriranja realizovana je u okviru SetFilter asinhronne aktivnosti koja se poziva klikom na dugme *Filter*.

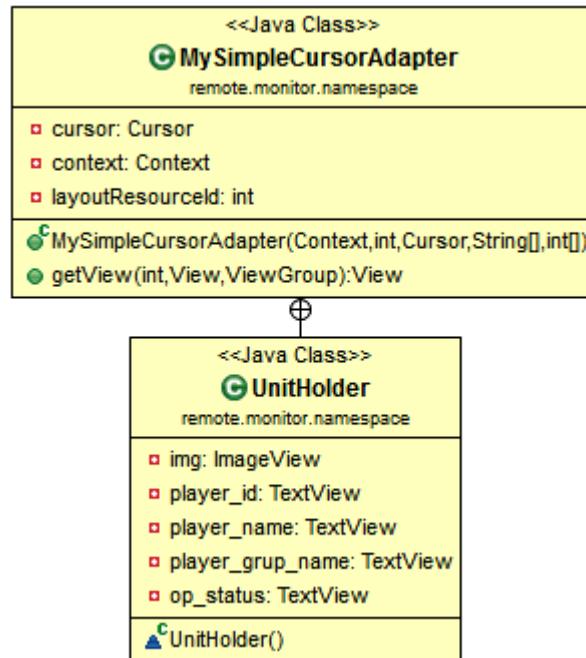


Slika 4.8 UnitActivity aktivnost



Slika 4.9 UnitActivity filtriranje

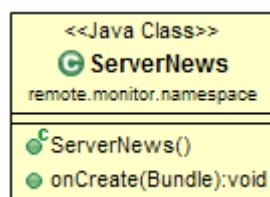
4.3.1 MySimpleCursorAdapter



Slika 4.10 MySimpleCursorAdapter klasa

Klasa nasleđuje SimpleCursorAdapter android klasu. Koristi se za unos podataka u listu UnitActivity aktivnosti. Ovo je realizovano predefinisanjem ugrađene metode SimpleCursorAdapter klase `getView()`. Podaci koji se unose u listu čitaju se iz baze podataka.

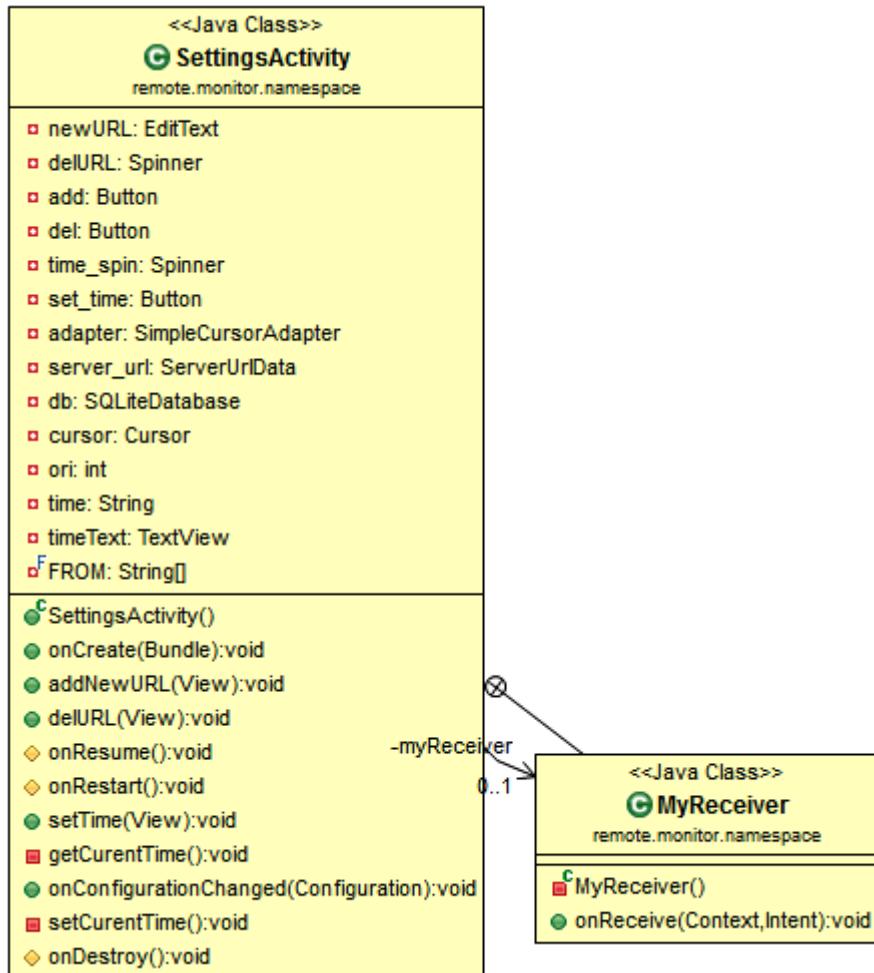
4.4 ServerNews



Slika 4.11 ServerNews klasa

U ovoj klasi realizovan je prikaz liste vesti sa servera.

4.5 SettingsActivity



Slika 4.12 SettingsActivity klasa

Klasa nasleđuje Android Activity klasu. Aktivnost realizuje osnovne operacije koje korisnik može da koristi:

- Dodavanje nove URL adrese servera
- Brisanje jedne od postojećih URL adresa servera
- Promena učestalosti osvežavanja podataka novim podacima sa servera

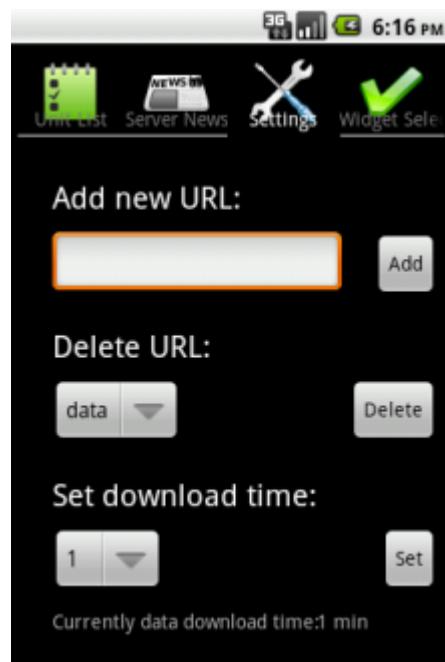
Aktivnost može da primi poruku koju šalju drugi delovi aplikacije, označenu sa `MSG_GET_BACK_TIME`. Prijem poruke realizuje se u klasi **MyReceiver** koja nasledjuje android klasu `BroadcastReceiver`. Kada primi poruku poziva se predefinisana metoda ove klase `onReceive()` koja obrađuje poruku.

Na početku izvršavanja padajući meni za brisanje URL adresa puni se adresama iz baze podataka pomoću `SimpleCursorAdapter` klase. Servisu se šalje `MSG_GET_TIME` poruka kako bi se saznalo trenutno vreme osvežavanja podataka.

Za dodavanje nove URL adrese potrebno je uneti URL adresu servera u odgovarajuće polje za unos teksta i kliknuti na dugme *Add*. Poziva se metoda *addNewURL()* koja upisuje novu URL adresu u bazu podataka.

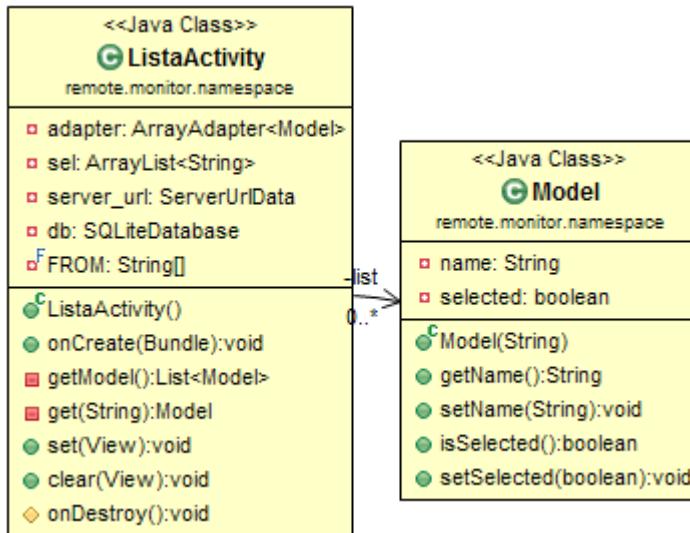
Ako korisnik želi da izbriše URL adresu iz baze podataka potrebno je da selektuje URL koji želi iz padajućeg meniju i da pritisne dugme *Delete*. Poziva se metoda *delURL()* koja briše izabrani URL iz baze podataka.

Da bi promenio učestanost osvežavanja podataka korisnik treba da izabere jednu od vrednosti iz padajućeg menija i pritisne dugme *Set*. Sve vrednosti su izražene u minutama. Pritiskom na dugme *Set* poziva se metoda *setTime()* koja šalje poruku *MSG_SET_TIME* serveru. Poruka sadrži novu vrednost perioda osvežavanja podataka.



Slika 4.13 SettingsActivity aktivnost

4.6 ListaActivity



Slika 4.14 ListaActivity klasa

`ListaActivity` nasledjuje `ListActivity` Android klasi. U ovoj klasi realizovan je odabir uređaja koje korisnik želi da prati na home screen widget. Korisnik može da selektuje do pet uređaja.

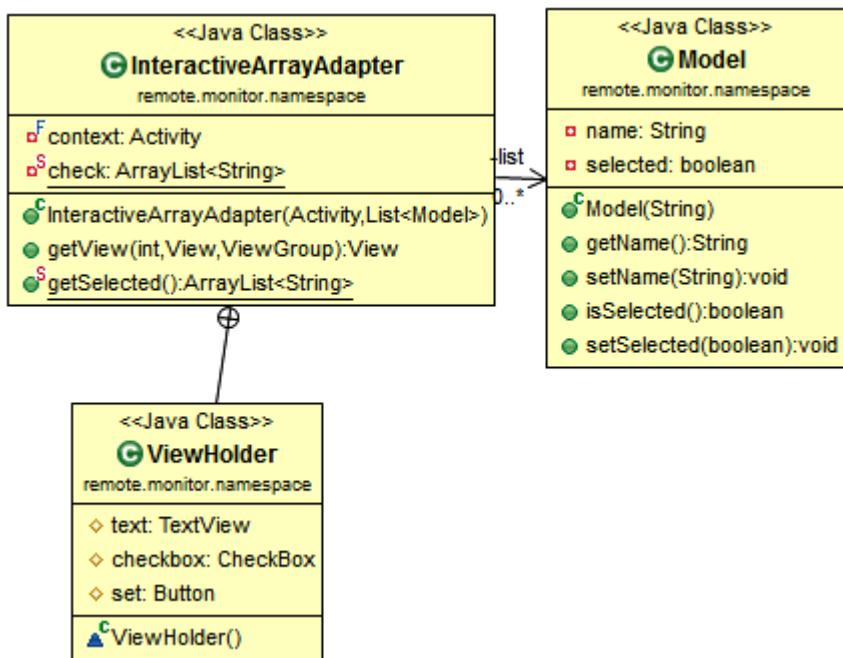
Pri kreiranju aktivnosti lista se popunjava imenima uređaja koji se prate. Popunjavanje liste radi se pomoću `InteractiveArrayAdapter` klase.

Od korisnika se traži da selektuje pet uređaja koje želi da prati. Klikom na dugme *Set* poruka sa odabranim uređajima šalje se widgetu. Poruka nosi oznaku `MSG_WIDGET_FILTER`. Aktivnost sadrži i dugme za birsanje svih selektovanih uređaja *Clear*.



Slika 4.15 ListaActivity aktivnost

4.6.1 InteractiveArrayAdapter i Model

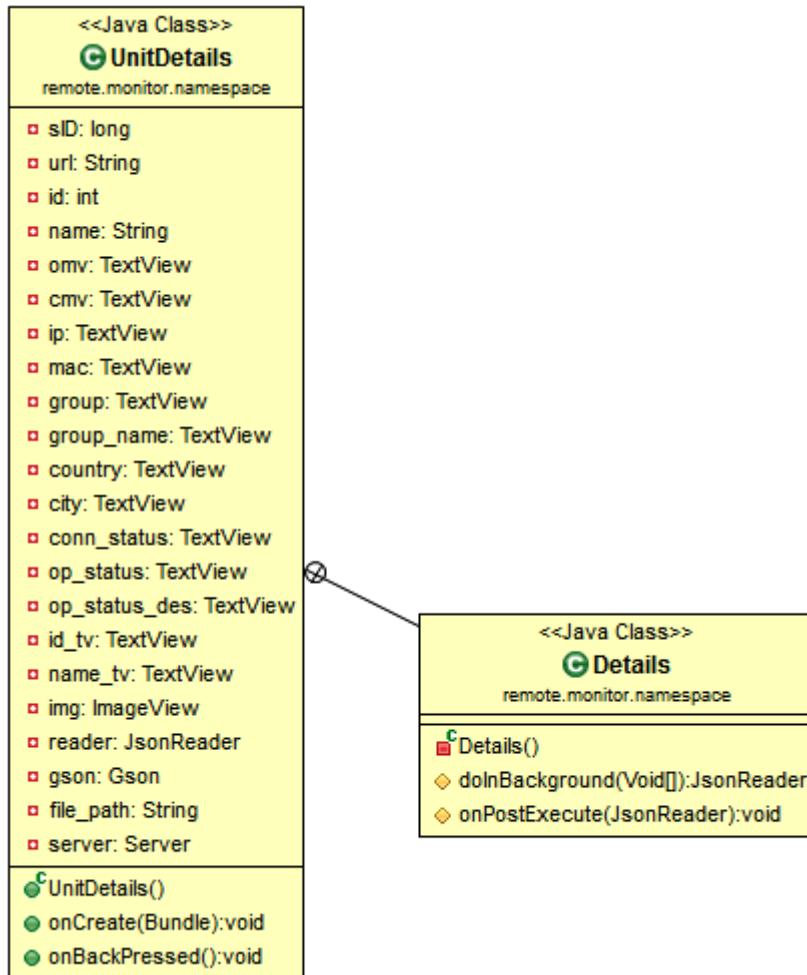


Slika 4.16 InteractiveArrayAdapter i Model klasa

Klasa InteractiveArrayAdapter nasleđuje klasu ArrayAdapter. Pomoću ove klase popunjava se lista uredaja u ListaActiviy aktivnosti. Kao element niza InteractiveArrayAdapter klasa prima objekat klase Model. Punjenje liste ListActivity aktivnosti realizovano je predefinisanjem ugrađene metode klase ArrayAdapter *getView()*.

Klasa Model je pomoćna klasa koja sadrži dva polja ime i status da li je selektovano ili ne. U polje ime unosi se ime uređaja, početno stanje je da ni jedan uređaj nije selektovan.

4.7 UnitDetails

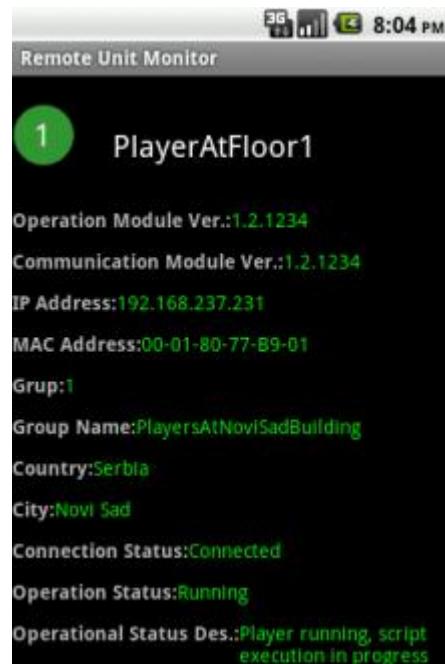


Slika 4.17 UnitDetails klasa

Klasa nasleđuje Android Activiy klasu. U ovoj aktivnosti prikazuju se detaljni podaci izabranog uređaja. Aktivnost prikazuje sledeće podatke:

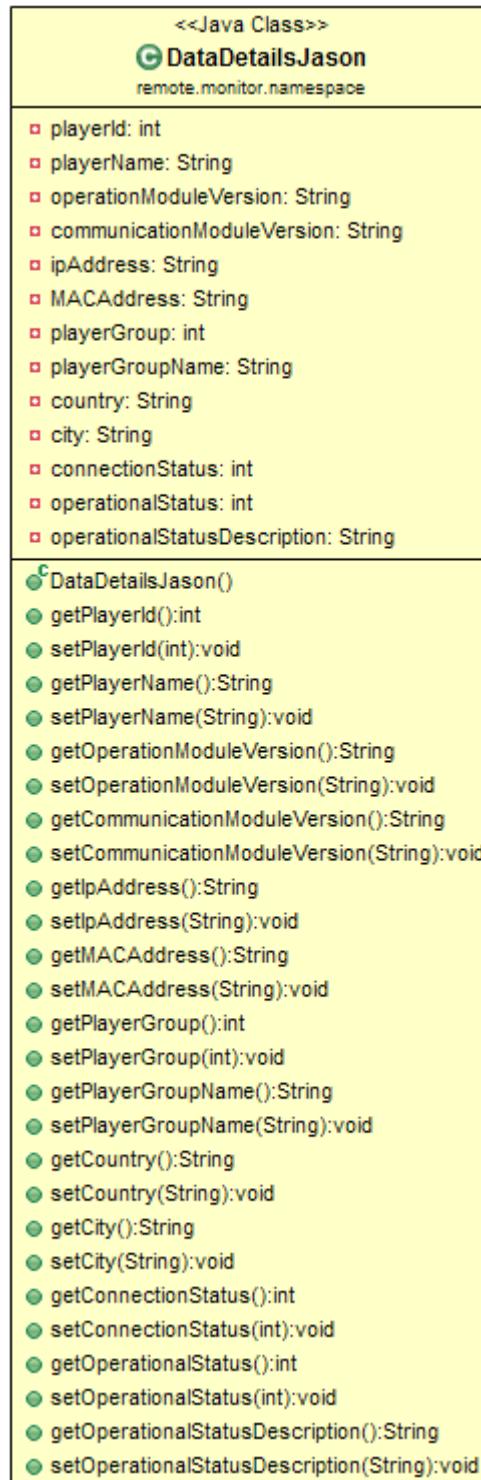
- Verzija operativnog modula
- Verzija modula veze
- IP adresa
- MAC adresa
- Grupa
- Naziv grupe
- Zemlja
- Grad
- Status veze
- Stanje uređaja
- Opis stanja uređaja

U `onCreate()` metodi aktivnosti čitaju se podaci poslati pri pokretanju ove aktivnosti: redni broj i ime željenog uređaja, identifikacioni broj i URL adresa servera. Pokreće se Details asinhrona aktivnost koja uspostavlja vezu sa serverom, traži detaljne podatke o željenom uređaju i ispisuje dobijene podatke ukoliko ih server pošalje. Pomoću DataDetailsJason klase uradi se deserializacija podataka iz JSON formata. Ako podaci nisu dostupni aktivnost se zatvara.



Slika 4.18 UnitDetails aktivnost

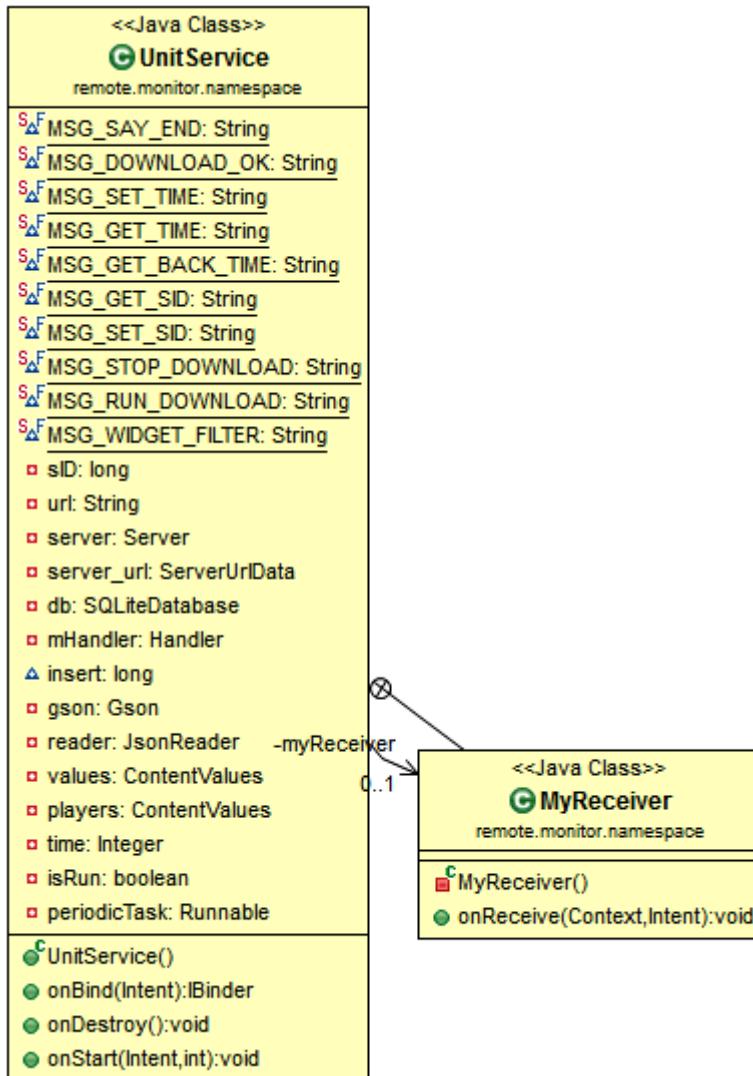
4.7.1 DataDetailsJason



Slika 4.19 DataDetailsJason klasa

Pomoćna klasa koristi se pri deserializaciji detaljnih podataka o uređajima koje server šalje u JSON formatu. Polja klase odgovaraju poljima koja se mogu pronaći u JSON dokumentu.

4.8 UnitService



Slika 4.20 UnitService klasa

Klasa nasleđuje android klasu Service, ona omogućava da se zadaci definisani u klasi koja je nasleđuje izvršavaju u pozadini ne opterećujući program.

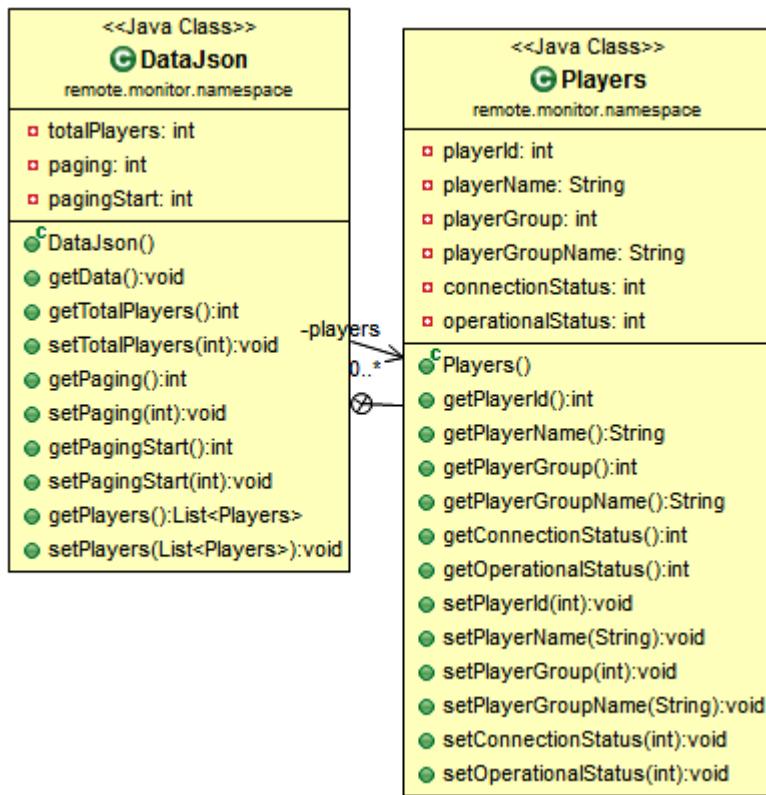
UnitService servis realizuje komunikaciju aplikacije sa serverom, periodično skida nove podatke sa servera i upisuje ih u bazu. Pokreće se pri startovanju aplikacije ili home screen widgeta ako ne postoji od ranije, prosleđuju mu se identifikacioni broj i URL adresa servera.

Servis je podešen da može da primi poruke koje šalju drugi delovi aplikacije, označenu sa `MSG_SET_TIME`, `MSG_SET_TIME`, `MSG_GET_SID`, `MSG_SAY_END`, `MSG_STOP_DOWNLOAD` ili `MSG_RUN_DOWNLOAD`. Prijem poruka iz drugih delova aplikacije realizuje se u klasi MyReceiver koja nasledjuje android klasu BroadcastReceiver. Kada primi poruku poziva se predefinisana metoda ove klase `onReceive()` koja obrađuje poruku.

Servis je neprekidno aktivan u pozadini od trenutka kreiranja. Jedini način da se servis zaustavi je odabir opcije *LogOut* u Tab aktivnosti. Kada se kreira pokrene se periodična

aktivnost u kojoj je realizovano skidanje novih podataka sa servera. Podaci koje server pošalje aplikaciji su u JSON formatu, za deserializaciju podataka koristi se Gson biblioteka. Nakon deserializacije podaci se zapisuju u bazu podataka i šalje se obaveštenje drugim delovima aplikacije da su podaci osveženi.

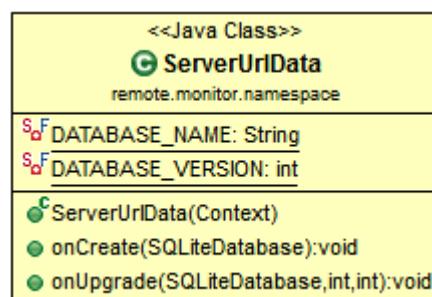
4.8.1 DataJson



Slika 4.21 DataJson klasa

Pomoćna klasa, koristi se pri deserializaciji podataka dobijenih od servera u JSON formatu. Polja klase odgovaraju poljima koja se mogu naći u JSON datoteci.

4.9 ServerUrlData

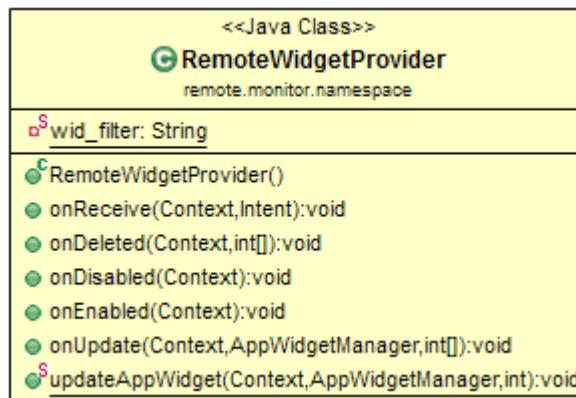


Slika 4.22 ServerUrlData klasa

Klasa nasleđuje SQLiteOpenHelper Android klasu. Realizuje rad sa bazom podataka. Baza se sastoji iz tri tabele:

- URL adresa servera
- Opštih informacija o uređajima
- Informacija o svakom uređaju pojedinačno

4.10 RemoteWidgetProvider



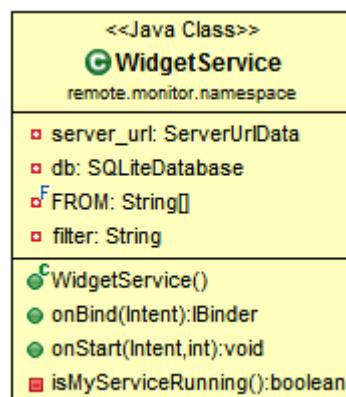
Slika 4.23 RemoteWidgetProvider klasa

Nasleđuje AppWidgetProvider Android klasu. Ona omogućava kreiranje widgeta na početnom ekranu uređaja. Klasa je podešena da može da primi poruke iz drugih delova aplikacije koje nose oznaku MSG_DOWNLOAD_OK ili MSG_WIDGET_FILTER. Obrada poruka realizovana je u metodi *onReceive()*. Za iscrtavanje željenih podataka u widgetu iz metoda klase poziva se WidgetService servis. Widget prikazuje ime uređaja, stanje veze i status uređaja u vidu slike. Takođe sadrži dva dugmeta *On* i *Off*. Prikiskom na dugme *Off* šalje se poruka MSG_STOP_DOWNLOAD koja zaustavlja osvežavanje podataka u UnitService servisu. Klikom na dugme *On* šalje se poruka MSG_RUN_DOWNLOAD koja ponovo pokreće osvežavanje podataka.



Slika 4.24 Izgled widgeta

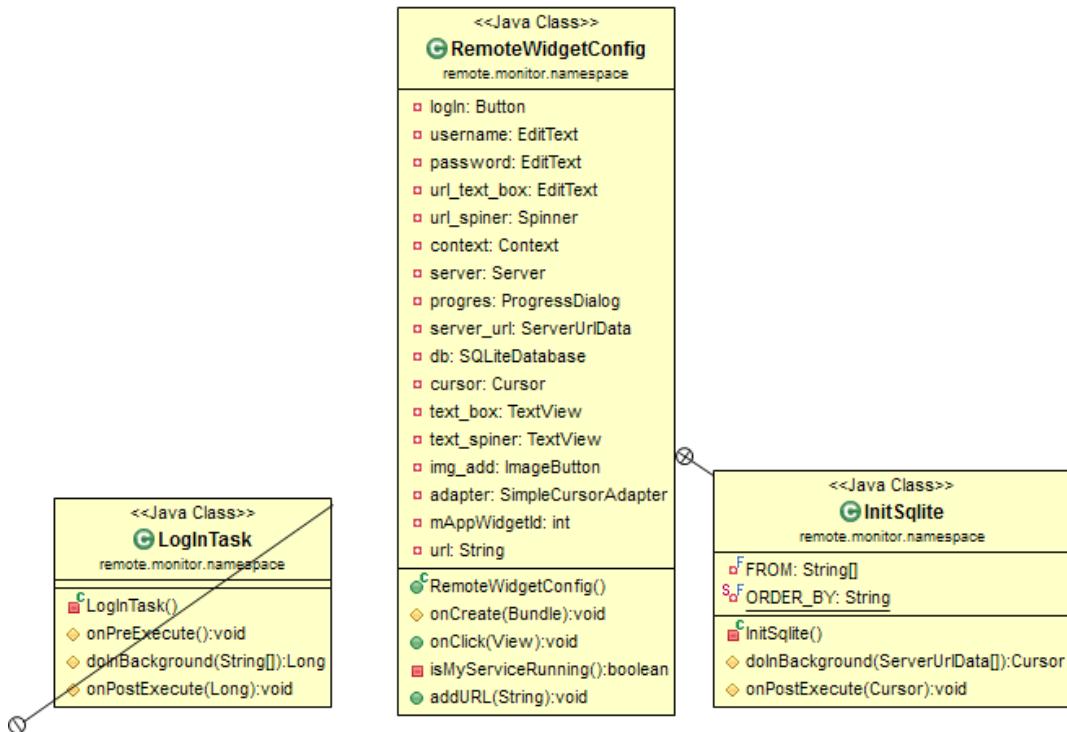
4.10.1 WidgetService



Slika 4.25 WidgetService klasa

Nasleđuje Android klasu Service. Koristi se za unos podataka u widget. Podaci se čitaju iz baze podataka. Servis sam sebe zatvori nakon što popuni widget. Kada podaci ne postoje u bazi ili servis nije kreiran widget se ne popunjava podacima već se ispisuje poruka da nema podataka.

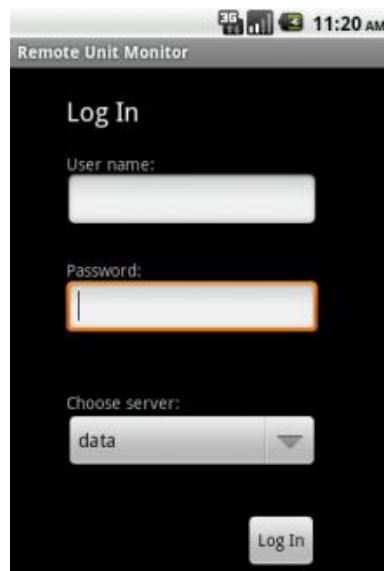
4.11 RemoteWidgetConfig



Slika 4.26 RemoteWidgetConfig klasa

Klasa nasleđuje android Activity klasu. Realizuje konfiguracionu aktivnost za widget. Pokreće se neposredno pre kreiranja widgeta.

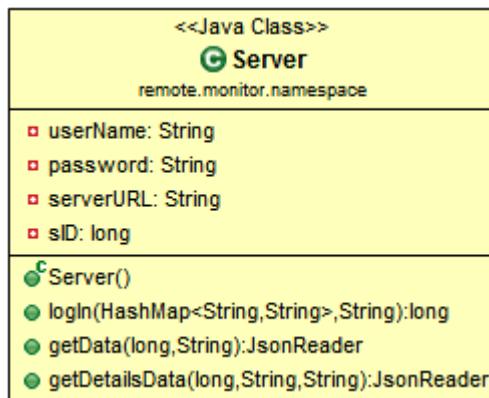
Na početku izvršenja ispituje se da li je servis pokrenut pomoću metode `isMyServiceRunning()`. Ako servis postoji aktivnost se zatvara i widget se pokreće. U drugom slučaju otvara se forma za prijavljivanje korisnika. Korisnik unosi korisničko ime, šifru i URL adresu servera na koji želi da se prijavi. Nakon uspešne prijave aktivnost se zatvara i widget se pokreće.



Slika 4.27 RemoteWidgetConfig aktivnost

5. Rezultati

Testiranje aplikacije izvršeno je korišćenjem klase Server, pomoću ove klase realizovana je simulacija realnog servera.



Slika 5.1 Server klasa

Klase sadrži tri metode:

- `logIn()` – metoda za prijavljivanje korisnika
- `getData()` – metoda za čitanje podataka o uređajima sa memorijske kartice
- `getDetailsData()` – metoda za čitanje detaljnih podataka o uređajima sa memorijske kartice

Testiranje je urađeno na Android simulatoru i realnom uređaju. Tokom testiranja korišćeno je više različitih JSON datoteka sa podacima o uređajima. Datoteke su sadržale različit broj uređaja.

Uređaji su ispravno prikazivani u listi. Detalji uređaja korektno su prikazivani ukoliko je postojala datoteka sa detaljima traženog uređaja. U suprotnom korisnik dobija obaveštenje da detalji za traženi uređaj ne postoje.

Vreme osvežavanja podataka testirano je kontrolnim ispisima i proverom vremena koje je proteklo između dva ispisa. Na isti način testirano je ponašanje aplikacije pri promeni vremena osvežavanja.

Widget omogućava zaustavljanje i pokretanje osvežavanja podataka. U zavisnosti od pritisnutog dugmeta ispisuje se odgovarajuća kontrolna poruka. Testiranje ovih opcija realizovano je kombinacijom kontrolnih poruka koje se generišu pritiskom na dugmad widgeta i poruke da su podaci osveženi.

Sadržaj tabele URL adresa servera proveravan je programom za čitanje SQL baze podataka.

Tokom testiranja aplikacija se korektno izvršavala bez grešaka.

6. Zaključak

Zadatak rada bila je realizacija aplikacije za praćenje stanja udaljenih uređaja. Trebalo je realizovati klasičnu Android aplikaciju koja će omogućiti praćenje udaljenih uređaja, davati osnovne i detaljne informacije o uređajima koji se prate, omogućeno je filtriranje uređaja radi lakšeg praćenja. Korisnik može da prati vesti sa servera. Aplikacija realizuje osnovne operacije: dodavanje i brisanje URL adresa servera, promena vremena osvežavanja podataka o uređajima. Korisniku je omogućeno da iz liste svih uređaja koji se prate izabere do pet uređaja koje želi da prati na widgetu.

Pored klasične Android aplikacije praćenje najznačajnijih uređaja omogućeno je i na widgetu. Widget prikazuje osnovne podatke za pet uređaja. U okviru aplikacije korisnik može da izabere uređaje koje želi da prati na widgetu. Pored praćenja uređaja widget omogućava pokretanje i zaustavljanje osvežavanja podataka.

Aplikacija nije testirana na realnom serveru, tako da nije provereno njen ponasanje u tim uslovima. Ipak kompletan spreg preko kojeg bi aplikacija trebala da komunicira sa serverom je realizovan.

U zadatku je realizovan samo osnovni koncept aplikacije i ostavljen je prostor za dalja unapređenja. Neka od mogućih unapređenja aplikacije:

- Omogućiti praćenje uređaja sa više različitih servera
- Realizovati prikaz uređaja na karti
- Realizovati statistiku uređaja
- Kao krajnji korak u razvoju aplikacije trebalo bi realizovati kontrolu uređaja preko aplikacije

7. Literatura

- [1] Wikipedia: The Free Encyclopedia , <http://www.wikipedia.org>
- [2] Android Developers: <http://developer.android.com>
- [3] Ed Burnette: Hello, Android: Introducing Google's Mobile Development Platform, North Carolina, 2008
- [4] Donn Felker, John Wiley & Sons: Android Application Development For Dummies, USA, 2010
- [5] Google-Gson: A Java library to convert JSON to Java objects and vice-versa ,
<http://code.google.com/p/google-gson/>