Univerzitet u Banjoj Luci
Prirodno-matematički fakultet

Diplomski rad

Tema: Dizajn i implementacija Android aplikacije za promociju fakulteta

Student:
Vladimir Jovanović

Mentor:
dr Vladimir Filipović

maj 2013.

Sadržaj

1 Uvod

Usljed velikog tehnološkog napretka posljednjih godina, rokovnike i bilješke sve više zamjenjuju elektronski uređaji, kao što su telefoni i tableti. Javlja se trend da sve informacije moraju biti dostupne odmah, bez čekanja. Vođen tom idejom javila se zamisao o izradi fakultetske Android aplikacije "miiDroid", koja bi omogućila brz i jednostavan pristup informacijama vezanim za studijski program matematika i informatika na Prirodno-matematičkom fakultetu u Banjoj Luci.

Za razvoj aplikacije odabrana je Android platforma zbog toga što je najviše korištena na mobilnim telefonima, kao i zbog toga što su telefoni sa Android operativnim sistemom na našem području najpristupačniji, a sve sa ciljem da što više potencijalnih korisnika može da je koristi. Još jedna od prednosti razvoja aplikacije za Android platformu u odnosu na ostale je to što je razvoj aplikacija za ovu platformu potpuno besplatan. Svi alati potrebni za razvoj aplikacije su besplatni, dostupni na internetu, a ukoliko programer posjeduje urećaj sa Androidom može ga koristiti za testiranje bez ikakvih dodatnih troškova.

Sama aplikacija napravljena je tako da, na uštrb vizuelnih efekata, bude minimalno zahtjevna i brza. Iz aplikacije se može pristupiti novostima sa fakultetske stranice, biografijama predavača, planovima i programima, lokaciji fakulteta na Google mapama, osnovnim informacijama o fakultetu i terminima predavanja.

Sadržaj ovog rada koncipiran je tako da u 2. glavi budu izloženi osnovni principi Android operativnog sistma kao i način razvoja aplikacija za ovaj operativni sistem. Glava 3. biće posvećena uopšteno dizajnu aplikacija za Android platformu kao i samom dizajnu aplikacije "miiDroid", dok će u glavi 4. biti predstavljeni osnovni principi implementacije aplikacije za Android platformu. Posebna pažnja u glavi 4. biće posvećena tehnologijama koje su se koristile u implementaciji aplikacije "miiDroid".

2 Operativni sistem Android

Android je operativni sistem baziran na Linux jezgru. Namijenjen je uređajima osjetljivim na dodir kao što su telefoni i tableti. To je operativni sistem otvorenog koda i Google objavljuje kôd pod Apache licencom, što znači da se softver može slobodno mijenjati od strane proizvođača hardvera, mobilnih operatera i razvojnih timova. Pored toga, Android podržava ogromna zajednica programera i dizajnera koji razvijaju aplikacije za ovaj operativni sistem.

Svaki proizvođač mobilnih telefona isporučuje na svojim telefonima operativni sistem prilagođen njegovom uređaju, kako po hardverskim osobinama, tako i po izgledu samog uređaja. Postoji i veliki broj verzija Androida razvijenih od strane nezavisnih zajednica programera, među kojima je najpoznatiji i najviše korišten Cyanogenmod.

Glavna prednost operativnog sistema Android je primjena unifikovanog pristupa razvoju aplikacija. Programeri razvijaju sve pod Android platformom, a njihova aplikacija treba da ima mogućnost izvršavanja na velikom broju uređaja, pod uslovom da ti uređaji koriste operativni sistem Android.

2.1 Istorijat operativnog sistema Android

Na početku je razvijan od strane Android, Inc. koji je finansiran i kasnije kupljen od strane kompanije Google. Kompanija Android, Inc. je osnovana u Palo Altu u Kaliforniji, oktobra 2003. godine, a osnovali su je Andy Rubin, Nick Sears, Rich Miner i Chris White. Kasnije im ponestaje novaca, pa kompaniju kupuje Google, ali oni ostaju zaposleni i nastavljaju razvoj operativnog sistema. [10]

Prvi telefon koji se prodavao sa Android OS je HTC Dream, i predstavljen je 22. oktobra 2008. Na njemu se nalazila verzija Androida 1.5 Cupcake. Od 2008, pored navedene, objavljen je veći broj verzija, a najznačajnije su: 1.6 Donut, 2.1 Eclair, 2.2 Froyo, 2.3 Gingerbread, 3.1 Honeycomb, 4.0 Ice Cream Sandwich, 4.1 Jelly Bean.

Google Nexus je linija mobilnih uređaja koji koriste operativni sistem Android. 2010. godine je predstavljen prvi telefon iz linije Nexus.[2] Nexus uređaji su obično prvi uređaji sa novom verzijom operativnog sistema Androida, tj. predstavljanjem nove verzije Androida predstavljana je i nova linija Nexus uređaja. Proizvođača Nexus uređaja bira kompanija Google.

2.2 Karakteristike operativnog sistema Android

Pošto Android ima otvoren kôd i dostupan je potpuno besplatno proizvođačima za samostalno prilagođavanje, ne postoje precizno definisane hardverske ili softverske konfiguracije. Međutim, Android podržava sljedeće:

- skladištenje – Za skladištenje podataka koristi se SQLite, jednostavna relaciona baza podataka.
- pristupanje mrežama – Podržane su GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (uključuje A2DP i AVRCP), Wi-Fi, LTE i WiMAX.
- slanje poruka – Postoji podrška za SMS i MMS.
- web čitač – Zasnovan je na WebKit otvorenom kodu, kao i na V8 JavaScript okruženju.
- multimedijalna podrška – Postoji podrška za sledeće formate medija: H.263, H.264 (u 3gp ili mp4 kontejneru), MPEG-4 SP, AMR, AMR-WB (u 3GP kontejneru), AAC, HE-AAC(u 3gp ili mp4 kontejneru), MP3, MIDI,Ogg Vorbis, WAV, JPEG, PNG, GIF i BMP.
- hardverska podrška – Akcelerometarski senzor, kamera, digitalni kompas, senzor rastojanja i

GPS

- multi-touch – Podrška za multi-touch ekrane.
- multi-tasking – Podrška za aplikacije koje izvršavaju više zadataka istovremeno.
- flash podrška – Android 2.3 podržava Flash 10.1
- povezivanje – Postoji podrška za dijeljenje internet konekcija i žičnih/bežičnih pristupnih tačaka. [3]

Android platforma je prilagođena za upotrebu na većim ekranima poput pametnih telefona koji koriste 2D grafičku biblioteku ili 3D grafičku biblioteku zasnovanu na OpenGL ES 2.0 specifikacijama. Za pohranu podataka koristi se SQLite relacioni DBMS (Database Managment System) napisan u programskom jeziku C.[4] Karakteristika ovog softvera jeste njegova biblioteka koja u svega 275kB implementira većinu SQL standarda.[6] U odnosu na druge sisteme za upravljanje bazama podataka, SQLite nije zaseban proces već je sastavni dio aplikacije koja pristupa bazi podataka.

Bluetooth podržava interfejs za upravljanje drugim uređajima, te prenos audio zapisa sa jednog uređaja na drugi. Od Android verzije 3.0 postoji podrška za spajanje uređaja za upravljanje (tastatura, miš, džojstik), dok je na prijašnjim verzijama podrška bila omogućavana od strane samih proizvođača takvih uređaja.

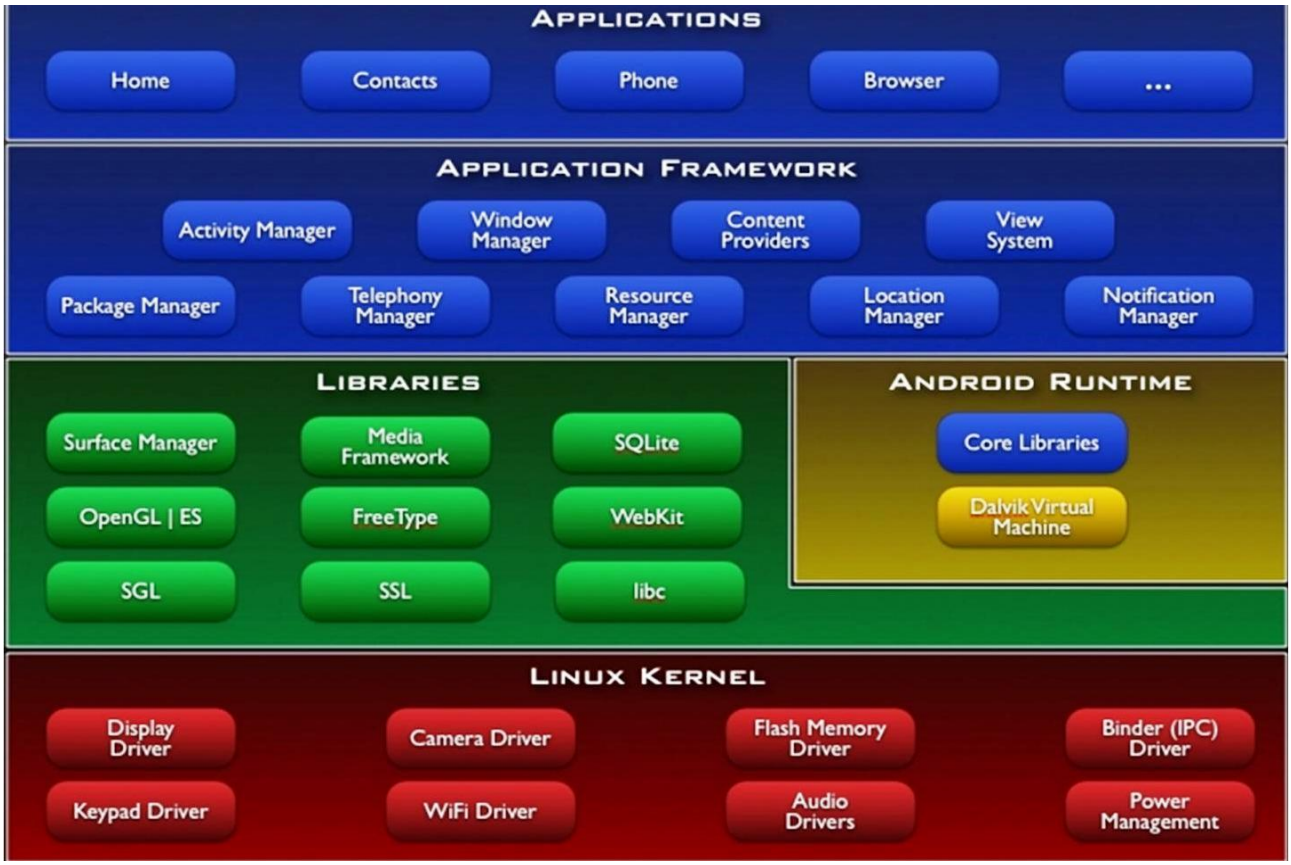
Uređaji sa instaliranom verziom Androida “Froyo” ili nekom novijom imaju mogućnost teatheringa, odnosno korištenja uređaja kao žične ili bežične pristupne tačke za pristupanje internetu. Prije verzije 2.2 ta mogućnost je postojala samo kroz korištenje neslužbenih aplikacija ili ukoliko je proizvođač uređaja omogućio tu funkciju.

Ugrađena je podrška i za RTP/RTSP12 streaming, HTML progresivni download. Adobe Flash Streaming i HTTP Dynamic Streaming podržani su kroz Flash plugin.

2.3 Arhitektura operativnog sistema Android

Način funkcionisanja operativnog sistema Android dat je na slici 1, na kojoj su prikazani različiti slojevi koji sačinjavaju operativni sistem Android.
Android je grubo podijeljen na pet sekcija i četiri osnovna sloja:

- Linux jezgro – To je jezgro na kome je Android zasnovan. Ovaj sloj sadrži sve drajvere uređaja na niskom nivou za različite hardverske komponente svakog pojedinačnog Android uređaja.
- biblioteke – Sadrže sav kod koji obezbjeđuje osnovne funkcije operativnog sistema Android.
- Android okruženje za izvršavanje – Na istom nivou kao i biblioteke, Android okruženje obezbjeđuje skup osnovnih biblioteka koje omogućavaju programerima da pišu Android aplikacije korištenjem programskog jezika Java. Android okruženje sadrži i Dalvik virtuelnu mašinu, koja omogućava svakoj Android aplikaciji da se izvršava u sopstvenom procesu, sa sopstvenom instancom Dalvik virtuelne mašine (Android aplikacija se prevodi u Dalvik izvršne datoteke). Dalvik je specijalizovana virtuelna mašina, projektovana specijalno za Android i optimizovana za mobilne uređaje koji koriste bateriju pri radu i imaju ograničene memorijske i procesorske resurse.
- radni okvir aplikacije – Omogućava se korištenje različitih mogućnosti operativnog sistema Android, tako da programeri mogu da ih koriste u svojim aplikacijama.
- aplikacije – Na ovom nivou nalaze se aplikacije koje se isporučuju sa Android uređajima (kao što su Phone, Contacs, Browser i slične), kao i aplikacije koje se preuzimaju i instaliraju korištenjem Android Marketa.



Android je zasnovan na Linux 2.6 jezgru i napisan u programskom jeziku C/C++. Obzirom na otvorenost izvornog programskog koda, aplikacije putem middlewarea imaju mogućnost komuniciranja i pokretanja drugih aplikacija na primjer za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Iako su programski jezici C i C++ korišteni za radno okruženje (framework), većina aplikacija pisana je u programskom jeziku Java uz pomoć Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u programskim jezicima C i C++, no tada se koristi Android Native Code Development Kit (NDK). Ovakvim postupkom omogućuje se bolje raspolaganje resursima i korištenje biblioteka programa iz jezgra i radnog okruženja. Ovako napisane aplikacije rade i do 10 puta brže, no pisanje samog programa je puno složenije.
Na višem nivou nalaze se biblioteke koje su pisane u programskom jeziku C/C++:

- Surface Manager – biblioteka koja nadzire iscrtavanje grafičkog interfejsa
- OpenGL | ES – biblioteka za ubrzavanje 3D prikaza (ukoliko je moguća) te za visoko optimizovanu 3D softversku rasterizaciju
- SGL – 2D biblioteka korištena za većinu aplikacija
- Media Framework – biblioteka zasnovana na OpenCORE koja podržava snimanje i reprodukciju poznatih audio/video formata
- FreeType – biblioteka namjenjena iscrtavanju fontova
- SSL (Secure Sockets Layer) - biblioteka za sigurnosnu komunikaciju putem interneta
- SQLite – biblioteka za upravljanje bazama podataka dostupna svim aplikacijama
- WebKit – engine za web pretraživače
- libc – sistemska C biblioteka prilagođena za sisteme zasnovane na operativnom sistemu Linux

Slijedi Android Runtime odnosno sloj koji služi za pokretanje aplikacija. Sastoji se od dvije važne komponente. Prva su tzv. "Core libraries" odnosno biblioteka koje sadrže većinu osnovnih funkcionalnosti programskog jezika Java. Druga komponenta je Dalvik Virtual Machine koja pokreće aplikacije kao zasebne procese odnosno kao instance virtualne mašine. DVM pretvara Java datoteke u svoj vlastiti format (.dex), kako bi bile optimizovane za minimalnu potrošnju memorije.

Nakon biblioteka dolazi aplikacioni okvir (eng. Application Framework) koji se sastoji od mehanizama koji pomažu pisanje aplikacija. Aplikacioni okvir dozvoljava upotrebu svih API-ja (Application Programming Interface) koji su korišteni za bazne aplikacije. Tako je omogućeno upravljanje programskim paketima, aktivnostima aplikacije (odnosi se na životni ciklus aplikacije), pozivima, prozorima, resursima (pohrana komponenti aplikacija koje nisu sami kôd, naprimjer slike), korištenje podataka od više različitih aplikacija, dobijanje i korištenje trenutne lokacije korisnika, prikaz obavještenja te baza pogleda i objekata koji mogu biti korišteni za dizajn aplikacije.

Na vrhu se nalaze same aplikacije. Ovaj sloj je vidljiv krajnjem korisniku i sastoji se kako od osnovnih, ugrađenih aplikacija poput e-mail klijenta, SMS programa, kalendara, web pretraživača pa sve do aplikacija koje se mogu naći na Android Marketu.

2.4 Razvoj aplikacija

Aplikacije za operativni sistem Android se obično razvijaju u programskom jeziku Java pomoću Software Development Kit (SDK) paketa. SDK je obiman skup razvojnih alata, među kojima se nalaze debuger, razne biblioteke, emulator, dokumentacija i primjeri koda. Jedno od integrisanih razvojnih okruženja (IDE) koji u potpunosti podržava SDK je Eclipse.

Android kôd, u stvari, nije pisan u programskom jeziku Java, jer kod ne podržava ustanovljene Java standarde Java SE i ME. To sprečava kompatibilnost programa pisanih za ove platforme i onih pisanih za Android. Android samo iskorištava sintaksu i semantiku Jave. [8]

Proteklih godina razvijena je velika zajednica Android programera širom svijeta. Sada je mnogo jednostavnije dobiti rješenje određenog problema, odnosno pronaći srodne programere sa kojima je moguće podijelite ideje o aplikaciji i da razmijenite iskustva. Neke od internet stranica na kojima se okupljaju ovakve zajednice programera su: www.stackoverflow.com, developer.android.com, groups.google.com/group/android-discuss.

2.5 Sigurnost i privatnost

Android aplikacija se pokreće u izolovanom prostoru operativnog sistema iz koga nema pristup ostalim sistemskim resursima, osim ako korisnik nije eksplicitno dodijelio pristup nekom sistemskom resursu prilikom instalacije. Ovakav sistem obezbjeđuje dobru sigurnost, ali programeri, zbog ograničene dokumentacije, često prave grške stavljajući nepotrebne dozvole. Par kompanija je izdalo antivirus programe za operativni sistem Android, ali ovi programi ne obezbjeđuju neku dodatnu sigurnost, jer zbog prethodno opisanog sistema instalacije aplikacija, ne mogu da skeniraju dublje sistem u potrazi za prijetnjom. [5]

Najveća prijetnja sigurnosti korisnika su aplikacije koje šalju SMS na premijum brojeve i time prave velike troškove korisniku. Pored ovih tu su aplikacije koje šalju privatne informacije korisnika E-mail ili SMS porukom, kao i one manje štetne, ali veoma dosadne koje na ekran izbacuju reklame.[1] Dosta ovakvih aplikacija se nalazi i na Google Play marketu, što je posljedica nedovoljno intenzivnog provjeravanja aplikacija prije objave na marketu.

2.6 Google Play

Google Play poznat i kao Android Market je platforma za distribuciju aplikacija za Android

osnovana od strane kompanije Google. Pored aplikacija, zavisno od države, sa marketa se mogu skinuti i drugi sadržaji, kao što su muzika, novine, knjige i TV program. Besplatne aplikacije na marketu su dostupne u cijelom svijetu, dok se aplikacije koje se plaćaju mogu skinuti samo u nekim zemljama. Ažuriranje aplikacija je riješeno veoma jednostavno. Može se podesiti da se aplikacije ažuriraju automatski ili manuelno, po želji korisnika.

Samo programeri iz veoma malog broja zemalja mogu naplaćivati svoje aplikacije na marketu. Registracija na marketu se plaća 25$. Od cijene aplikacije koja se naplaćuje, 70% ide vlasniku naloga, a 30% marketu.

3 Dizajn aplikacije “miiDroid”

Na početku dizajna aplikacije prvo se trebaju tačno odrediti mogućnosti aplikacije i ciljnu populaciju korisnika takve aplikacije. Pri izradi dizajna moraju se izdvojiti najkrupnije funkcionalne cjeline, a potom ih, imajući na umu softverske tehnologije koje će se koristiti, preslikati u entitete dizajna koji se mogu dalje implementirati. Dizajn treba prilagoditi načinima rješavanja pojedinih problema u Android programiranju i samoj objektnoj orjentisanosti programskog jezika Java.

Bitan dio razvoja aplikacije je planiranje što je moguće više njenih elemenata, da bi u procesu implementacije bilo što je moguće manje odstupanja od plana. I najmanja promjena nekog dijela plana može značajno da utiče na implementaciju ostatka aplikacije.

3.1 Kome je aplikacija namijenjena?

Na početku dizajna svake aplikacije, pa i aplikacije za Android platformu, bitno je jasno odrediti korisničku populaciju aplikacije, da bi se mogućnosti aplikacije i dizajn korisničkog interfejsa mogli prilagoditi tome. Kako je “miiDroid” fakultetska aplikacija, realno je u korisničku populaciju uvrstiti profesore i studente. Pošto bi ova aplikacija trebala biti i promotivnog karaktera u korisničku populaciju se mogu dodati još i budući studenti. Zbog toga što je Univerzitet u Banjoj Luci u procesu integracije sa univerzitetima u Evropskoj uniji, za očekivati je da bi se u budućnosti na studijski program za matematiku i informatiku mogli upisivati i strani studenti. Logičan korak je planiranje dizajna aplikacije tako da se omogući jednostavno prevođenje aplikacije na druge jezike.

3.2 Slučajevi upotrebe sistema

U nastavku će biti navedeni slučajevi upotrebe sistema koji imaju alternativni scenario.

Naziv: Učitavanje i pregled novosti

Akter: Korisnik

Učesnici: Korisnik

Osnovni scenario:

1. Korisnik inicira učitavanje novosti sa interneta
2. Aplikacija se spaja na internet
3. Aplikacija uspješno skida novosti sa interneta
4. Aplikacija prikazuje listu novosti

Alternativni scenario:

2.1 Ukoliko nije ostvareno spajanje na internet aplikacija prikazuje poruku o nemogućnosti spajanja na internet i učitava prethodno učitane novosti.

3.1 Ukoliko dođe do greške prilikom učitavanja aplikacija prikazuje poruku o grešci prilikom učitavanja novosti sa interneta i učitava prethodno učitane novosti.

Naziv: Pregled učitanih novosti

Akter: Korisnik

Učesnici: Korisnik

Osnovni scenario:

1. Korisnik inicira učitavanje novosti iz memorije
2. Aplikacija uspješno učitava novosti iz memorije
3. Aplikacija prikazuje listu novosti

Alternativni scenario:

2.1 Ukoliko ne postoje snimljene novosti u memoriji, aplikacija ispisuje poruku da nema snimljenih novosti.

2.2 Ukoliko se desi greška sa učitavanjem novosti iz memorije, aplikacija ispisuje poruku da nema snimljenih novosti.

Naziv: Skidanje materijala sa interneta

Akter: Korisnik

Učesnici: Korisnik

Osnovni scenario:

1. Korisnik inicira skidanje materijala sa interneta
2. Aplikacija se uspješno spaja na internet
3. Aplikacija skida materijal sa interneta
4. Aplikacija snima materijal u internu memoriju uređaja

Alternativni scenario:

2.1 Ukoliko se ne ostvari spajanje na internet aplikacija obavještava korisnika o nemogućnosti spajanja na internet.

3.1 Ukoliko dođe do problema sa skidanjem materijala sa interneta aplikacija obavještava korisnika o problemu prilikom skidanja materijala sa interneta.

4.1 Ukoliko dođe do problema sa snimanjem materijala u internu memoriju uređaja aplikacija obavještava korisnika o problemu prilikom snimanja materijala.

Naziv: Dodavanje novog rasporeda predavanja

Akter: Korisnik

Učesnici: Korisnik

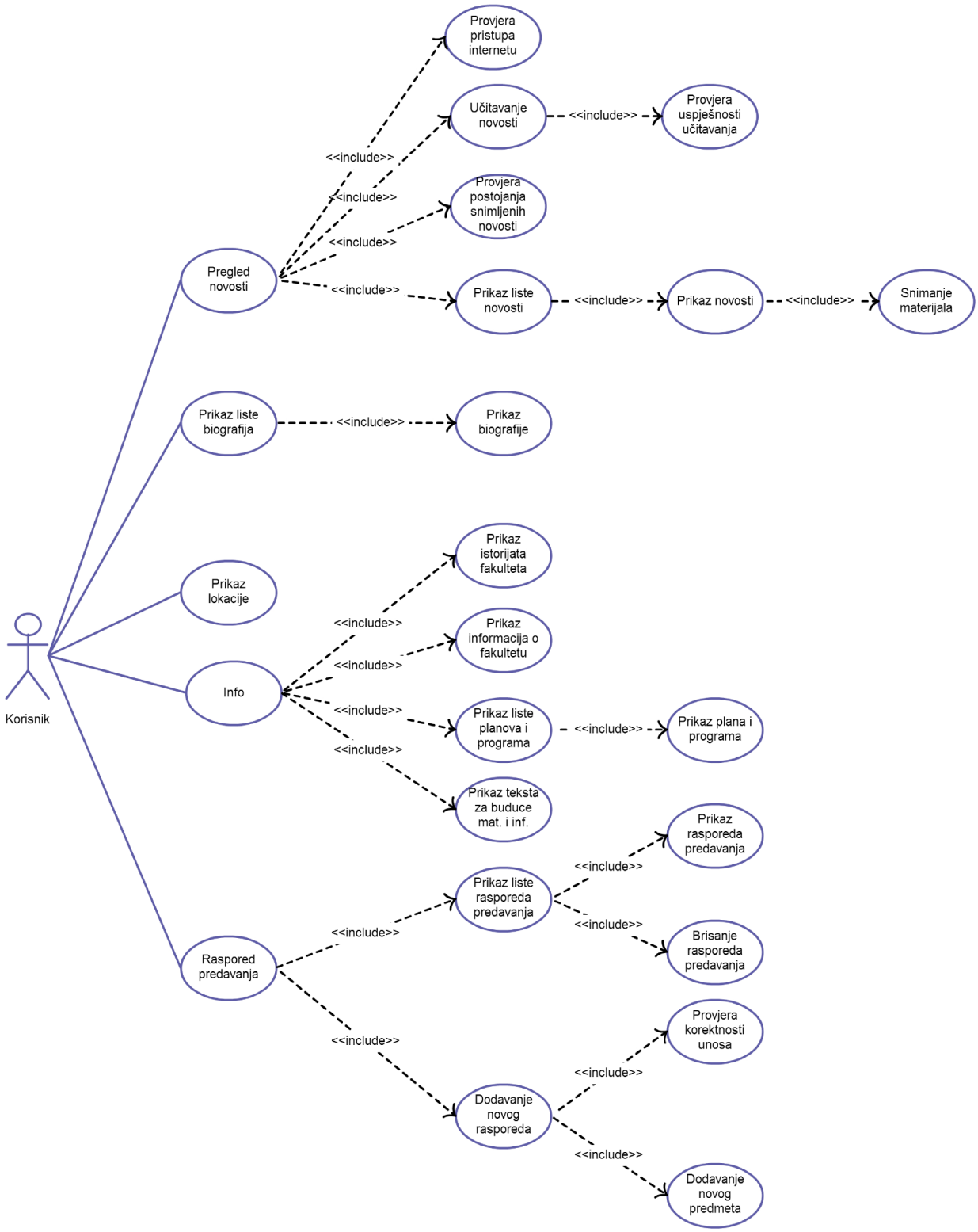
Osnovni scenario:

1. Korisnik inicira dodavanje novog rasporeda
2. Korisnik korektno unosi podatke o rasporedu
3. Korisnik unosi predavanja u raspored
4. Korisnik snima raspored

Alternativni scenario:

2.1 Ukoliko korisnik nekorektno unese podatke o rasporedu, aplikacija ispisuje poruku o nekorektnom unosu i ponudi korisniku da ispravi greške.

Slučajevi upotrebe sistema dati su na slici 2.



Slika 2

3.3 Arhitektura aplikacije “miiDroid”

Aplikacija “miiDroid” je zasnovana na monolitnoj arhitekturi sa elementima SOA[1] arhitekture. Osobina monolitne arhitekture je da se svi njeni elementi, uključujući i podatke nalaze na jednom mjestu. Iako monolitna, arhitektura aplikacije “miiDroid” se može podijeliti u cjeline.

- Prvu cjelinu čini korisnički interfejs. On je sačinjen od aktivnosti i svih elemenata koje grade korisnički interfejs, kao što su dugmad, slike, tekst itd.
- Drugu cjelinu čine elementi u koje je smještena logika koja postavlja sadržaj u aktivnosti, obrađuje interakciju korisnika sa korisničkim interfejsom i omogućuje komunikaciju između aktivnosti. Veza između ove i prve cjeline arhitekture aplikacije je dosta jaka.
- Treću cjelinu arhitekture aplikacije čini dio koji upravlja čuvanjem podataka. U ovoj cjelini se vrši smiještanje novosti i rasporeda predavanja u memoriju, njihovo čitanje i brisanje iz memorije. Ovu cjelinu čine još i strukture u kojima se čuvaju podaci.
- Četvrtu cjelinu čine elementi koji preko interneta skidaju podatke sa fakultetske stranice i pripremaju ih za čuvanje u strukturama koje se nalaze u trećoj cjelini. Ova cjelina komunicira isključivo sa trećom cjelinom.
- Peta i posljednja cjelina je zadužena za komunikaciju sa Google Maps servisom. Iako bi se mogla svrstati i kao dio četvrte cjeline odvojena je posebno, jer monolitnoj arhitekturi aplikacije “miiDroid” dodaje osobine SOA arhitekture. Ovaj dio, koristeći Google Maps API-je vrši skidanje mapa sa Google-ovih servera, njihovo slanje prvom sloju, slanje podataka potrebnih za računanje pozicije korisnika aplikacije Google-ovom serveru i primanje informacije o poziciji korisnika.

3.4 Određivanje API nivoa

Izbor API nivoa predstavlja bitan element u dizajnu Android aplikacije. Izborom nižeg nivoa dobijamo na broju potencijalnih korisnika aplikacije, to jest na broju uređaja koji će aplikaciju moći pokrenuti. Ukoliko izaberemo viši nivo dobijamo nove tehnologije u dizajnu i funkcionalnosti aplikacije, ali gubimo u broju podržanih uređaja. Trenutan odnos broja korisnika različitih verzija Android operativnog sistema dat je u sledećoj tabeli.

Verzija	Kodno ime	Datum izlaska	API nivo	Zaključko sa 14. martom 2013.
4.2.x	*Jelly Bean*	13. novembar 2012.	17	1.6%
4.1.x	*Jelly Bean*	9. jul 2012.	16	14.9%
4.0.x	*Ice Cream Sandwich*	16. decembar 2011.	15	28.6%
3.2	*Honeycomb*	15. jul 2011.	13	0.9%
3.1	*Honeycomb*	10. maj 2011.	12	0.3%
2.3.3–2.3.7	*Gingerbread*	9. februar 2011.	10	44%

1 SOA(Service Oriented Aplication) je aplikacijona arhitektura u kojoj su sve funkcije i usluge (servisi) definisani pomoću standardizovanog jezika te posjeduju pristupan interfejs pomoću kojih se pozivaju s ciljem potpore određenim segmentima poslovnih procesa. Svaki pristup i interakcija između servisa je nezavisna od ostalih interakcija i komunikacijonih protokola. Budući da su interfejsi nezavisni od platforme, klijent može pristupati i koristiti servise s bilo kojeg uređaja, bilo kojeg operativnog sistema i programskog jezika.

Verzija	Kodno ime	Datum izlaska	API nivo	Zaključko sa 14. martom 2013.
2.3–2.3.2	*Gingerbread*	6. decembar 2010.	9	0.2%
2.2	*Froyo*	20. maj 2010.	8	7.6%
2.0–2.1	*Eclair*	26. oktobar 2009.	7	1.9%
1.6	*Donut*	15. septembar 2009.	4	0.2%

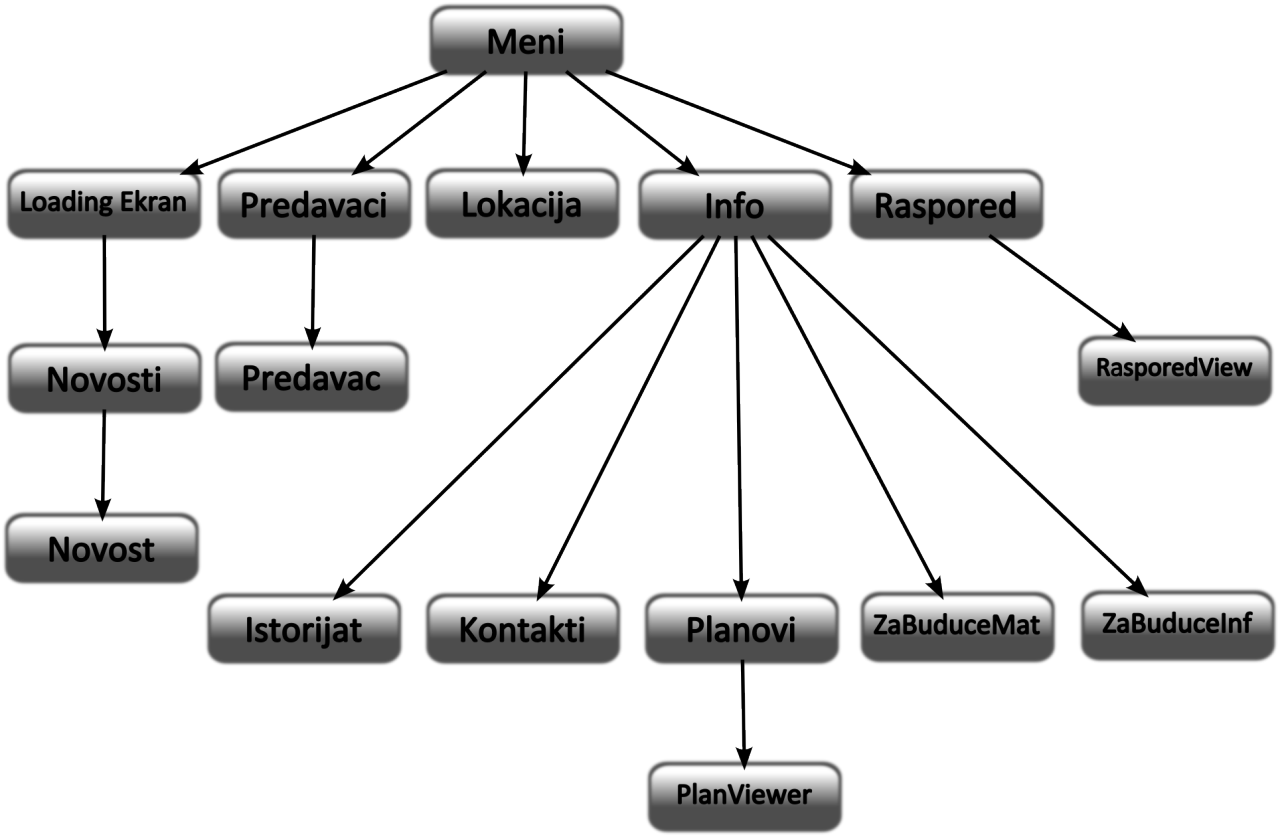
Za izradu aplikacije "miiDroid" korišten je API nivo 10 to jest minimalna verzija Android operativnog sistema neophodna za korištenje ove aplikacije je 2.3.3. To je trenutno najviše korištena verzija Androida i daje optimalan odnos dostupnih tehnologija za izradu aplikacije i broja uređaja koji će aplikaciju moći pokrenuti. Trenutno oko 9,7% korisnika mobilnih telefona sa Androidom neće moći koristiti ovu aplikaciju, a taj broj će u budućnosti sve više opadati.

3.5 Dizajn grafičkog interfejsa

Dizajnu grafičkog interfejsa potrebno je posvetiti posebnu pažnju. To je element aplikacije koji prosječan korisnik prvo zapazi. Pored samog izgleda potrebno je obratiti i dodatnu pažnju na funkcionalnost grafičkog interfejsa. Dobar grafički interfejs mora da posjeduje sledeće osobine:

- preglednost – Moraju da se raspoznaju cjeline grafičkog interfejsa. Elementi moraju biti jasno odvojeni i pregledno raspoređeni;
- povratni osjećaj – Prilikom svake interakcije korisnika sa ekranom mora mu biti jasno da li je kliknuo na neki element grafičkog interfejsa ili ne;
- odziv – Aplikacija mora da bude brza, a da pri bilo kakvim obimnijim procesima, gdje korisnik mora da sačeka da se neka radnja završi, obavijesti korisnika o izvršavanju te radnje.
- ponovno iskorištavanje grafičkih elemenata – Kako se Android aplikacija obično izvršava na uređajima sa ograničenom memoriom, poželjno je grafičke elemente koristiti što je moguće više puta bez pravljenje novih, a da se pri tome očuvaju gornje osobine;
- jedinstvenost – Treba se truditi da aplikacija, što je moguće više, bude drugačija od aplikacija slične namjene;
- testiranje – posljednja, ali ne manje bitna stvar je testiranje aplikacije među korisničkom populaciom. Koliko se god dizajneru ili programeru činilo da je sasvim jasno čemu služi koji element grafičkog interfejsa to ne mora biti tako. Zato je aplikaciju potrebno dati stvarnim korisnicima na testiranje i vidjet nejasnoće koje će se pojaviti u toku korištenja aplikacije.

Prije bilo kakve implementacije korisničkog interfejsa potrebno je pomno isplanirati izgled svake aktivnosti (eng. Activity). Treba još jasno isplanirati iz koje aktivnosti će se moći pozvati ostale aktivnosti, tj. treba tačno odrediti strukturu prikaza u aplikaciji. Struktura prikaza aplikacije "miiDroid" data je na slici 3.


Meni
Loading Ekran
Predavaci
Lokacija
Info
Raspored
Novosti
Predavac
RasporedView
Novost
Istorijat
Kontakti
Planovi
ZaBuduceMat
ZaBuduceInf
PlanViewer

Slika 3

4 Implementacija aplikacije “miiDroid”

Najvažniji i vremenski najzahtjevniji dio procesa izrade aplikacije je njena implementacija. U ovom radu će, iz toga razloga, upravo ovom dijelu biti posvećeno najviše pažnje. Implementacija aplikacije se umnogome može olakšati i pojednostaviti, ako se dizajnu aplikacije posveti dovoljna pažnja. U ovome poglavlju ćemo razmotriti sve bitne elemente implementacije aplikacije “miiDroid” i razmotriti i opisati tehnologije koje su se u tom procesu koristile.

4.1 Anatomija Android aplikacije

Prije bilo kakvog teksta o implementaciji same aplikacije, razmotrićemo elemente jednog Android projekta i kako pojedini elementi omogućavaju funkcionisanje aplikacije. Android projekat sačinjavaju sledeći folderi:

- src – Sadrži .java izvorne datoteke u projektu.
- gen – Sadrži R.java datoteku koju je generisao prevodilac, a koja referencira sve resurse u projektu. Nju ne treba modifikovati. Svi resursi u projektu se automatski prevode u ovu klasu, tako da mogu da budu referencirani njenim korištenjem.
- Android 2.3.3 library – Ovaj folder sadrži samo jednu datoteku – android.jar, koja sadrži sve biblioteke klasa neophodne za jednu Android aplikaciju.
- assets – Ovaj folder sadrži sva sredstva koja koristi aplikacija, kao što su HTML, tekstualne datoteke, baze podataka itd.
- bin – U njemu su sadržane datoteke koje je kreirao ADT u procesu prevođenja. Konkretno, generiše se .apk datoteka. Datoteka sa ekstenziom .apk je binarni kod Android aplikacije. Ona sadrži sve neophodno za izvršavanje te aplikacije.
- res – Ovaj folder sadrži sve resurse koji se koriste u aplikaciji koji su raspoređeni u podfolderima drawable, layout i values. [7]

Pored foldera u osnovnom folderu Android projekta nalazi se i datoteka AndroidManifest.xml. U njemu se definišu privilegije koje su neophodne za aplikaciju kao i ostale funkcije. AndroidManifest.xml datoteka sadrži detaljne informacije o aplikaciji:

- Definiše naziv paketa aplikacije.
- Sadrži vrijednost android:versionCode koja se koristi za identifikovanje verzije aplikacije. Ova vrijednost se koristi za programsko utvrđivanje da li je neophodno ažuriranje aplikacije.
- Naziv verzije aplikacije (dat u atributu android:VersionName) koja predstavlja broj oblika 1.2.1 koji ne mora biti isti kao android:versionCode i biće dostupan korisniku.
- Atribut android:minSdkVersion određuje minimalnu verziju operativnog sistema na kojem će se aplikacija izvršavati.
- Naziv aplikacije je određen stringom pod nazivom app_name.
- Na kraju su navedene sve aktivnosti koje se koriste u aplikaciji, njihove osobine i jedna glavna aktivnost.

4.2 Korisnički interfejs

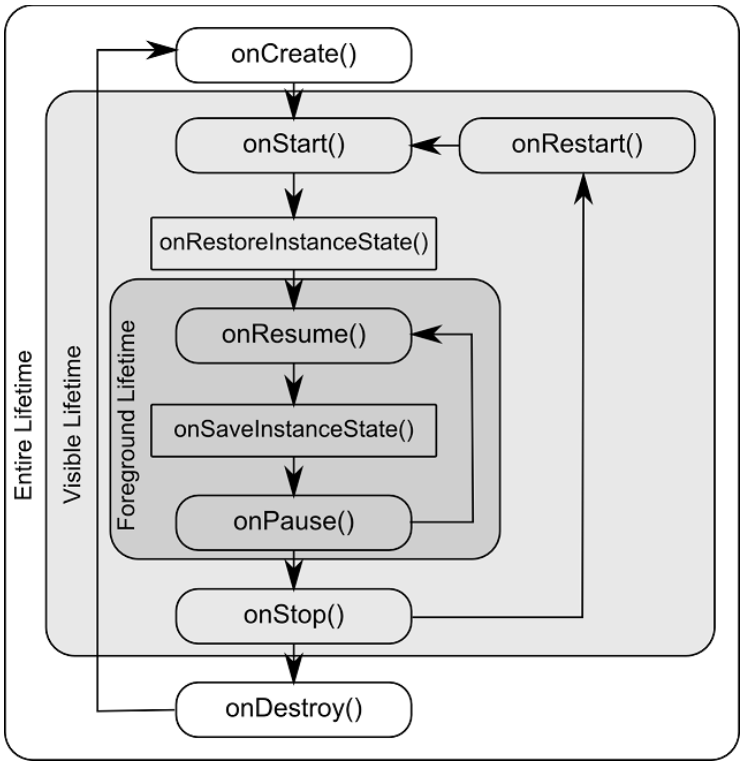
Implementacija korisničkog interfejsa u Androidu je jedna od osnovnih i stvari koje se prve nauče kada se uči Android programiranje. Android API-ji pružaju dosta naprednu i laganu kontrolu sadržaja koji će se prikazivati na ekranu.

4.2.1 Aktivnosti

Aktivnost (eng. Activity) je komponenta aplikacije koja obezbjeđuje korisniku pristup ekranu, tako da korisnik preko ekrana može vršiti interakciju sa aplikacijom. Svaka aktivnost daje prozor u kome se iscrtava grafički interfejs. Android aplikacija obično sadrži veći broj aktivnosti. Postoji jedna glavna aktivnost koji se poziva prilikom pokretanja aplikacije. Svaka aktivnost može pokrenuti neku drugu aktivnost i prilikom toga ona gubi fokus, to jest prestaje biti vidljiva korisniku i smiješta se na stek. Način pozivanja nove aktivnosti prikazan u sledećem primjeru.

```
Intent i = new Intent();
i.setClassName("putDoAktivitija", "putDoAktivitija.ImeAktivitija");
startActivity(i);
```

Kada korisnik završi sa korištenjem trenutne aktivnosti i pritisne dugme Back, aktivnost koja ju je pozvala se vraća sa steka i prikazuje korisniku, ukoliko u aplikaciji nije podešeno drugačije. Životni vijek svake aktivnosti je prikazan na slici 4. Metode koje se pozivaju prilikom ulaska aktivnosti u neki životni ciklus i njihova funkcija su sledeće:



Slika 4

- onCreate() - Poziva se prilikom prvog definisanja aktivnosti. Koristi se za inicijalizaciju promjenjivih koje se u toj aktivnosti koriste, kao i za prvo iscrtavanje grafičkog interfejsa.
- onStart() - Poziva se kada aktivnost postane vidljiva korisniku. Koristi se kada se nešto želi prikazati prilikom svakog ponovnog prikazivanja aktivnosti.
- onResume() - Poziva se kada korisnik započne interakciju. Koristi se kada se želi napraviti neka promjena prilikom interakcije korisnika sa aplikacijom (npr. pojačati svjetlina ekrana).
- onPause() - poziva se kada se privremeno zaustavi trenutna aktivnost, a nastavi prethodna aktivnost. Koristi se kada korisnik ne vrši interakciju sa datom aktivnošću, pa se želi napraviti neka promjena u prikazu.
- onStop() - Poziva se kada aktivnost više nije vidljiva korisniku. Koristi se da se zaustave procesi koji bi nepotrebno trošili memoriju ili procesorsku snagu na prikazivanje nekog sadržaja na ekranu. (npr. animacija)
- onDestroy() - Poziva se prije nego što se ukloni određena aktivnost iz sistema. Koristi se da bi se ručno uklonio neki sadržaj iz memorije.
- onRestart() - Poziva se kada se aktivnost zaustavi, a zatim ponovo startuje. Koristi se da bi se nastavili neki procesi koji su zaustavljeni kada je aktivnost prestala biti vidljiva korisniku.

U prethodnom tekstu su navedene samo neke od najčešćih primijena navedenih metoda.

4.2.2 Pogledi

Grafički interfejs jedne Android aplikacije se ne može zamisliti bez pogleda-objekata klase View i njenih podklasa. Osnovna hijararhija strukture klase View je prikazana na slici 4. U nju spadaju svi elementi pomoću kojih se kreira grafički interfejs aplikacije, kao što su dugmad, radio-dugmad, pregled slike, pregled teksta i tako dalje.

Podklasa ViewGroup je podklasa u kojoj su smještena poravnanja. Oni predstavljaju neku vrstu kontejnera u koji se dodaju osnovni elementi. Svaki element prikaza se može dodati ili iz koda ili pomoću XML datoteke za dato poravnanje. Svaki objekat klase View se može posebno podesiti, može mu se dodati osluškivač tj. objekat klase Listener, jedinstveni ID, animacija itd. U sledećem primjeru dat je dio kôda u kome je implementiran ekran prikazan na slici 5, dok je XML vezan za isti ekran dat u sledećoj glavi(4.1.3).

```
public void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
		requestWindowFeature(Window.FEATURE_NO_TITLE);
		getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
			WindowManager.LayoutParams.FLAG_FULLSCREEN);
		super.onCreate(savedInstanceState);
		setContentView(R.layout.predavac);
		Bundle extras = getIntent().getExtras();
		int profesor = extras.getInt("profesor");
		final Biografija biografija = Predavaci.biografija[profesor];
		TextView tv = (TextView)findViewById(R.id.imePredavac);
		tv.setText(biografija.getIme());
		tv.setBackgroundResource(R.drawable.textbar);
		tv.setTextSize(20);
		tv.setGravity(Gravity.CENTER);
		tv.setTextColor(Color.WHITE);
		ImageView iw = (ImageView)findViewById(R.id.slikaPredavac);
		iw.setBackgroundResource(biografija.getSlika());
		TextView tvMail = (TextView)findViewById(R.id.mailPredavac);
		tvMail.setText(biografija.getMail());
		tvMail.setTextColor(Color.BLUE);
		tvMail.setTypeface(null, Typeface.BOLD_ITALIC);
		tvMail.setOnClickListener(new View.OnClickListener() {
			public void onClick(View v) {
				Intent i = new Intent(Intent.ACTION_SEND);
				i.setType("message/rfc822");
				i.putExtra(android.content.Intent.EXTRA_EMAIL, new
					String[]{ getString(biografija.getMail()) });
				startActivity(Intent.createChooser(i,
					getString(R.string.posaljimail)));
			}
		});
		TextView tvBiografija =
			(TextView)findViewById(R.id.biografijaPredavac);
		tvBiografija.setText(biografija.getBiografija());
		tvBiografija.setTextColor(Color.BLACK);

}
```

Prilikom definisanja veličine elemenata u Android korisničkom interfejsu treba voditi računa da se koriste sledeće jedinice mjere:

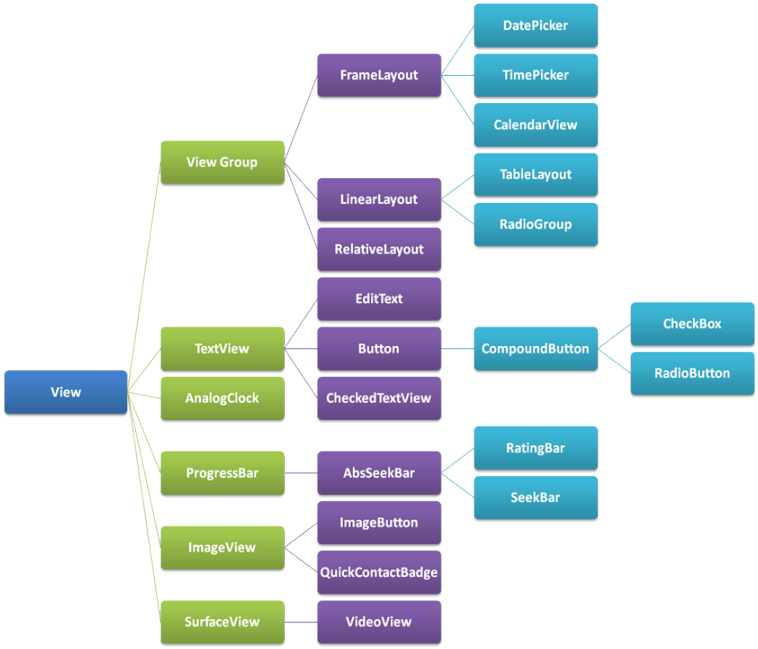
- dp – piksel nezavisan od gustine (density-independent pixel). Jedan dp je ekvivalentan

jednom pikselu na ekranu rezolucije 160dpi. Ovo je preporučena jedinica mjere prilikom specifikacije dimenzija pogleda za odgovarajući raspored elemenata.

- sp – piksel nezavisan od rezolucije (scale-independent pixel). Preporučuje se za definisanje veličine fonta.
- pt – tačka (point). Definisana je kao 1/72 inča, na osnovu fizičke veličine ekrana.
- px – piksel (pixel). Odgovara aktuelnoj veličini piksela na ekranu. Korištenje ove jedinice nije preporučljivo, jer korisnički interfejs neće biti ispravno prikazan na uređajima koji imaju različitu rezoluciju ekrana.

4.2.3 XML

Dizajn grafičkog interfejsa se umnogome oslanja na XML. Dizajn se vrši ili direktno kucanjem XML datoteke ili pisanjem Java koda koji će u trenutku izvršenja generisati potreban XML kod.
XML (Extensible Markup Language) je standardan skup pravila za definisanje formata podataka u elektronskoj formi. Propisan je od strane W3C.[9] Ideja je bila da se stvori jezik koji će i ljudi i računarski programi moći jednostavno da čitaju. On definiše opštu sintaksu za označavanje podataka pomoću odgovarajućih etiketa koje imaju poznato i lako razumljivo značenje.
Jedna od XML datoteka iz aplikacije izgleda ovako:



Slika 5

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
	android:layout_width="fill_parent"
	android:layout_height="fill_parent"
	android:orientation="vertical"
	android:background="@drawable/pmf"
	android:id="@+id/predavacLayout">
	<TextView
		android:id="@+id/imePredavac"
		android:layout_width="fill_parent"
		android:layout_height="wrap_content" >
	</TextView>
	<ScrollView
		android:layout_width="fill_parent"
		android:layout_height="fill_parent"
		android:orientation="vertical"
		android:background="@drawable/okvir" >
		<LinearLayout
			android:layout_width="fill_parent"
			android:layout_height="wrap_content"
			android:orientation="vertical"
			android:id="@+id/predavacLinearLayout">
			<ImageView
				android:id="@+id/slikaPredavac"
```

```
                        android:layout_height="wrap_content"
                        android:layout_width="wrap_content"
                        android:layout_gravity="center_horizontal"
                        android:scaleType="centerInside">
                    </ImageView>
                    <TextView
                        android:id="@+id/mailPredavac"
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:textStyle="bold"
                        android:layout_gravity="center_horizontal">
                    </TextView>
                    <TextView
                        android:id="@+id/biografijaPredavac"
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content" >
                    </TextView>
                </LinearLayout>
            </ScrollView>
</LinearLayout>
```

Slika 6

a izgled ekrana dat ovim kodom dat je na slici 6 gdje je sadržaj ubačen iz kôda, koji je dat u prethodnoj glavi (4.1.2).

4.2.4 Poruke obavještenja

Bitan dio implementacije korisničkog interfejsa predstavljaju obavještenja o uspješno obavljenom poslu ili o problemima prilikom izvršavanja neke naredbe. To su objekti klase Toast Kreiraju se na sljedeći način:

```
Context contekst = getApplicationContext();
CharSequence tekst = "Zdravo!";
int trajanje = Toast.LENGTH_SHORT;
```

```
Toast obavjestenje = Toast.makeText(contekst, tekst, trajanje);
obavjestenje.show();
```

Postoje još naredbe za njihovo precizno pozicioniranje na ekranu. Svakom obavještenju može se dodijeliti poravnanje u kome se izgled obavještenja može podesiti do najsitnijih detalja.

4.2.5 Grafički elementi

Nijedna moderna aplikacija se ne može zamisliti bez grafičkih elemenata korisničkog interfejsa. Za izradu grafičkih elemenata ove aplikacije korišten je alat za vektorsku grafiku "Inkscape". Ova aplikacija je potpuno besplatna i postoje verzije i za Windows i za Linux. Korišten je vektorski alat zato što se grafički element može veoma jednostavno izvesti u veliki broj različitih rezolucija. Na slici 7 se mogu vidjeti svi grafički elementi korišteni u aplikaciji "miiDroid". Poželjno je u aplikaciji koristiti što manje različitih grafičkih elemenata, jer se time štedi na memoriji koja je kod mobilnih telefona itekako dragocijena.

Slika 7

Pri izradi Android aplikacija se uglavnom koriste dva formata za skladištenje slika, a to su .png i .jpeg. Prvi format (.png) se obično koristi za ikonice, jednostavnije dizajne sa manjim brojem boja, geometrijske figure i slike sa dosta teksta; dok se drugi format (.jpeg) koristi za fotografije, realistične slike, slike sa mnogo boja.
Kod izrade grafike za aplikaciju treba obratiti pažnju na par stvari:

- Veoma tamne, u najgorem slučaju crne, pozadine prave probleme sa refleksijom, pa je sadržaj veoma teško vidjeti na direktnom svjetlu.
- Veoma svijetle, u najgorem slučaju bijele, pozadine koriste značajno više energije za prikazivanje na aktivnim ekranima kao što je AMOLED.
- Čitanje bijelog teksta na tamnoj pozadini veoma brzo zamara oči.
- Elementi jarkih boja su veoma zamorni za oči.
- Plava pozadina se veoma lako stopi sa refleksijom neba prilikom korištenja aplikacije napolju.

4.3 Prilagođenost različitim rezolucijama ekrana

Najveća pogodnost operativnog sistema Android, mogućnost rada na velikom broju uređaja, je ujedno i stvar koja Android programerima zadaje i najveće glavobolje. Dok kod naprimjer iOS uređaja postoje svega par rezolucija ekrana i samo dva formata ekrana, kod uređaja sa Androidom taj broj je ogroman i svakim danom se sve više povećava. Postoji nekoliko metoda koje umnogome olakšavaju rješavanje ovog problema.
U samoj strukturi Android projekta nalaze se sledeći folderi:

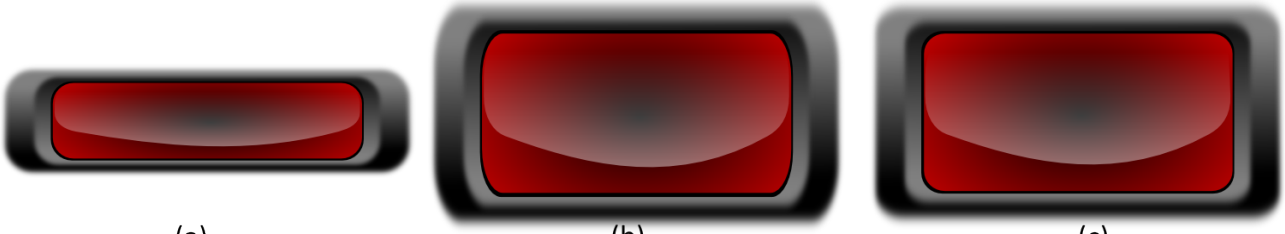
- ldpi – služi za smiještanje grafičkih elemenata namijenjenih prikazu na ekranima male

gustine (približno 120dpi[2])

- mdpi – služi za smiještanje grafičkih elemenata namijenjenih prikazu na ekranima srednje gustine (približno 160dpi)
- hdpi – služi za smiještanje grafičkih elemenata namijenjenih prikazu na ekranima velike gustine (približno 240dpi)
- xhdpi – služi za smiještanje grafičkih elemenata namijenjenih prikazu na ekranima veoma velike gustine (približno 320dpi)

Aplikacija automacki određuje iz kojeg foldera će da uzima grafičke resurse zavisno od karakteristika uređaja na kojem se aplikacija izvršava.

Još jedan od načina rješavanja ovog problema je skaliranje pojedinih elemenata interfejsa na određenu veličinu, zavisno od velicine i rezolucije ekrana. Kod skaliranja elemenata javlja se problem nepravilnih ćoškova kao što je prikazano na slici 8 pod (b). Ovaj problem se može riješiti korištenjem aplikacije draw9patch koja je dio standardnog paketa ADT.



(a) (b) (c)

Slika 8 (a) originalna slika; (b) slika razvučena po visini bez korištenja draw9patch aplikacije; (c)slika razvučena uz korištenje draw9patch aplikacije

Alat draw9patch editovanu sliku prosiruje sa svake strane za po jedan red piksela u koje se zapisuje način širenja slike, kao i prostor u koji će biti moguče unijeti neki sadržaj, npr. u dato dugme upisati tekst. Takva slika se snima u .png format, ali u obliku imeSlike.9.png.

4.4 Lokalizacija

Bitan element svake promotivne aplikacije, u ovom slučaju aplikacije za promociju fakulteta, je mogućnost aplikacije da sadržaj prikaže na što više različitih jezika. Prilikom implementacije ovakve aplikacije potrebno je na početku planirati višejezičnost aplikacije. Ovime se implementacija ove osobine može veoma pojednostaviti. Za svaki jezik se formira string.xml datoteka u koju se upisuje rječnik pojmova u sledećem obliku:

```
<string name="ime_stringa">Sadržaj stringa</string>
```

U svaku string.xml datoteku upisuju se stringovi sa istim imenom, ali sa tekstom na različitom jeziku. Iz kôda se podešava koji jezik se koristi, pomoću lokalizacionih kodova.

```
Resources res = getResources();
android.content.res.Configuration conf = res.getConfiguration();
DisplayMetrics dm = res.getDisplayMetrics();
String language_code = "sr";
conf.locale = new Locale(language_code);
res.updateConfiguration(conf, dm);
```

Ukoliko se ovakva podešavanja ne obave, aplikacija će automacki prepoznati jezičko podešavanje uređaja i odabrati njegov jezik, ukoliko u aplikaciji postoje postavke za taj jezik.

2 dpi predstavlja broj piksela na jednom kvadratnom inču ekrana

4.5 Čitanje novosti sa fakultetske stranice

U današnje vrijeme veoma je bitno da iz aplikacije bude omogućen pristup aktuelnim novostima. Aplikacija "miiDroid" ima mogućnost čitanja novosti sa fakultetske stranice. Problem čitanja novosti je riješen parsovanjem HTML koda, što nije najljepši ili najbrži način, ali ukoliko nije moguće dobiti pristup bazi podataka stranice, nameće se kao jedino rješenje. Za parsovanje HTML-a mogu se koristiti gotovi parseri koji koriste regularne izraze, ili se parsovanje može uraditi ručno. U izradi aplikacije "miiDroid" se pribjeglo drugom rješenju, jer je HTML prilično jednostavan, pa nema potrebe za korištenjem glomaznih parsera.
Sam HTML kod je učitan korištenjem AsyncTask klase koja omogućava da se datoteke učitavaju u pozadini i da se pri tome ne blokira prikaz korisničkog interfejsa. Klasa za čitanje novosti je implementirana na sledeći način.

```
private class DownloadWebPageText  extends AsyncTask<String,String,String> {
      protected void onPreExecute(String init){
            //inicijalizacija varijabli
            //postavljanje loading menija na početnu poziciju
      }
      protected String doInBackground(String... urls) {
             //Učitavanje vijesti sa stranice
      }
      protected void onPostExecute(String result) {
            //parsovanje učitanog HTML-a i smiještanje u varijable
            //pokretanje aktivitija za prikaz novosti
      }
}
```

Funkcija koja se poziva u doInBackground() je data u sledećem kodu.

```
private String downloadUrl(String myurl) throws IOException {
      InputStream is = null;
      try {
            URL url = new URL(myurl);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setReadTimeout(5000);
            conn.setConnectTimeout(10000);
            conn.setRequestMethod("GET");
            conn.setDoInput(true);
            conn.connect();
            is = conn.getInputStream();
            String contentAsString = convertStreamToString(is);
            return contentAsString;
          } finally {
            if (is != null) {
                  is.close();
            }
      }
}
```

U aplikaciji je omogućeno i prikazivanje učitanih novosti kada uređaj nema pristup internetu. Provjera dostupnosti interneta obavlja se sledećom metodom.

```
private boolean isNetworkConnected() {
      ConnectivityManager cm = (ConnectivityManager)
      getSystemService(Context.CONNECTIVITY_SERVICE);
```

```
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null && netInfo.isConnected()) {
                return true;
        }
        return false;
}
```

Čuvanje novosti vrši se pomoću objekta koji je primjerak klase SharedPreferencies. SharedPreferencies predstavljaju okvir za čuvanje i učitavanje snimljenih primitivnih podataka. Pomoću njih se može sačuvati bilo koji primitivan tip podataka: int, float, boolean, long i string. Način na koji je ovo rješeno prikazan je u sledećim okvirima. U prvom način na koji se novosti čuvaju, a u drugom način na koji se novosti učitavaju.

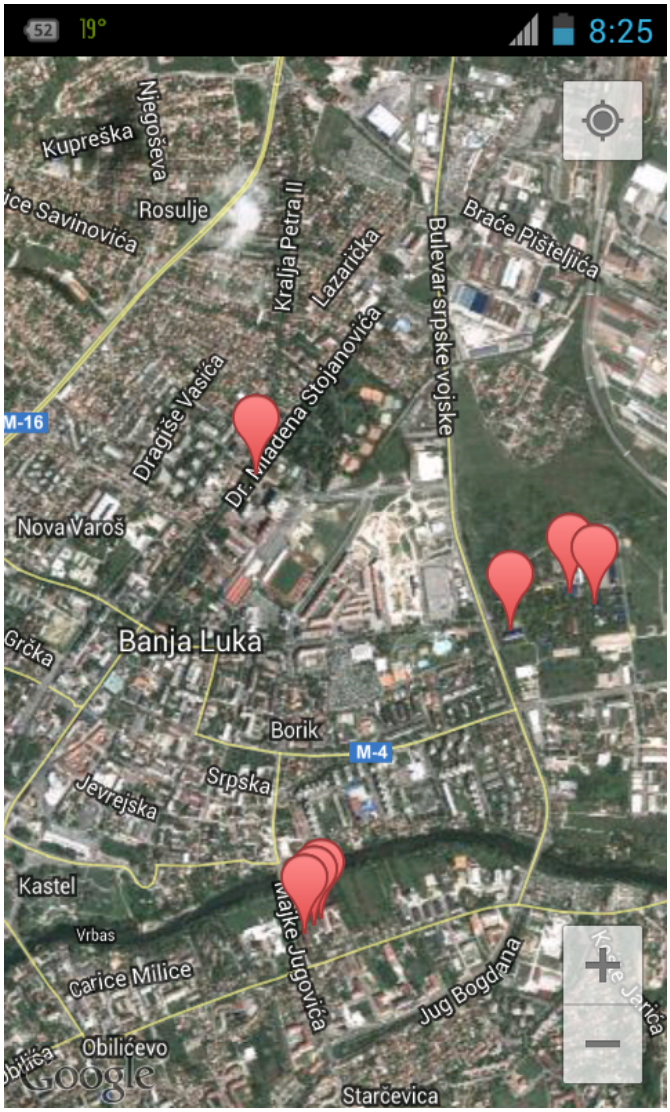
```
SharedPreferences.Editor ed = mPrefs.edit();
ByteArrayOutputStream arrayOutputStream = new ByteArrayOutputStream();
ObjectOutputStream out;
try {
        out = new ObjectOutputStream(arrayOutputStream);
        out.writeObject(novosti);
        out.close();
        arrayOutputStream.close();
} catch (IOException e) {
        e.printStackTrace();
}
ed.putString("novosti", arrayOutputStream.toString());
ed.commit();
```

```
ByteArrayInputStream byteArray = new
ByteArrayInputStream(mPrefs.getString("novosti","null").getBytes());
try {
        in = new ObjectInputStream(byteArray);
} catch (Exception e) {
        e.printStackTrace();
}
try {
        novosti = (NovostiHolder) in.readObject();
}catch (Exception e) {
        e.printStackTrace();
}
```

Kao što se može primijetiti, aplikacija u SharedPreferencies snima objekat klase NovostiHolder. Ova klasa je namijenjena za prenos novosti. Iz prethodnog imamo da se u SharedPreferencies mogu snimiti samo primitivni tipovi podataka, pa se snimanje objekta klase NovostiHolder obavlja pretvaranjem ovog objekta u string. Da bi ovo bilo moguće clasa NovostiHolder mora omogućavati serijalizovanos.

4.6 Google mape

Google mape je Google-ova tehnologija besplatnih digitalnih karata, koje čine osnovu mnogih servisa i usluga, kao što su prikazivanje satelitskih snimaka, planiranje trase putovanja, lociranje traženih mijesta. Dopušta jednostavnu implementaciju na različite web stranice, kombinovanje sa drugim aplikacijama, razvoj dodataka i prilagođavanje specifičnim potrebama. Dio razvoja Google Mapa uključuje i Google Maps API, interfejs za programiranje primarno namijenjen integraciji u drugim aplikacijama.



Slika 9

Veliki problem sa Google mapama predstavlja loš kvalitet ovih mapa u Bosni i Hercegovini. Zbog toga je u aplikaciji "miiDroid" korišten hibridni pogled. Ovaj pogled predstavlja kombinaciju satelitskog snimka i Google mapa, te omogućava da se lakše primijete greške u mapama kao i da se vide ulice koje u Google mape nisu još unesene. Prikaz Google mapa u aplikaciji "miiDroid" dat je na slici 9.

U aplikaciji "miiDroid" su preko prikaza mapa prikazuju lokacija Prirodno-matematičkog fakulteta, kao i lokacije značajnih zgrada Univerziteta u Banjoj Luci. Na mapi je omogućeno zumiranje pritiskom na dugmad na ekranu ili korištenjem multi-touch komandi (eng. Gestures). Pored toga, omogućeno je i prikazivanje trenutne lokacije korisnika aplikacije.

Od kada se pojavilo Google Maps API v 1.0 izdanje, neophodno je registrovati se da bi se dobila besplatna Google Maps API šifra, koja omogućava integraciju Google Maps servisa sa aplikaciom. Pri izradi aplikacije "miiDroid" korišteni su Google Maps API v 2.0 koji omogućavaju dosta lakšu implementaciju i naprednije korištenje Google Maps servisa. O samom implementiranju Google mapa u aplikaciju mogao bi se napisati poseban rad, pa bi uključivanje u tekst rada detaljnog opisa načina GoogleMap implementacije dovelo do toga da rad bude preobiman i neodgovarajuće struktuiran.

5 Zaključak

Svakim danom Android platforma uzima sve veći dio mobilnog tržišta, a polako počinje da prelazi i na tržište satova, pametnih televizora i sličnih uređaja. Zbog toga programiranje za Android platformu sve više dobija na značaju. U ovom radu je pokušano da se iznese osnovna problematika dizajna i implementacije Android aplikacije za promociju fakulteta „miiDroid“, kao i da se objasne tehnologije koje su se prilikom njene implementacije koristile.

Sama aplikacija je napravljena tako da bude jednostavna za kasnije dorade i implementaciju nekih novih elemenata. Prva stvar koja će biti poboljšana je iscrtavanje putanje do od trenutne pozicije korisnika do neke od označenih lokacija u Banjoj Luci. Pored toga, u budućnosti bi trebalo omogućiti da se kreiraju rasporedi predavanja za profesore. Ako bude potrebe aplikacija se veoma jednostavno može prevesti na još jezika, te time povećati potencijalnu korisničku populaciju aplikacije. U budućnosti će, također, biti pokušano pronalaženje profesionalnog dizajnera za ponovno crtanje grafike i dizaj cijelog korisničkog interfejsa.

Ovaj rad bi trebao da posluži kao fina podloga studentima matematike i informatike koji žele da počnu sa učenjem programiranja za Android platformu, kao i onima koji žele da se upute u programiranje za mobilne uređaje uopšte.

6. Listing izabranih dijelova koda aplikacije “miiDroid”

-Dio klase kojom je riješena upotreba Google Maps API-ja:

```
public class Lokacija extends FragmentActivity implements OnMarkerClickListener{
private static GoogleMap myMap;
ArrayList<LatLng> markerPoints;
@Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maplokacija_layout);
    FragmentManager myFragmentManager = getSupportFragmentManager();
    SupportMapFragment mySupportMapFragment
     = (SupportMapFragment)myFragmentManager.findFragmentById(R.id.map);
    myMap = mySupportMapFragment.getMap();
    myMap.setMyLocationEnabled(true);
    myMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
    markerPoints = new ArrayList<LatLng>();
    myMap.getUiSettings().setZoomControlsEnabled(true);
    myMap.getUiSettings().setCompassEnabled(true);
    myMap.getUiSettings().setMyLocationButtonEnabled(true);
    myMap.getUiSettings().setRotateGesturesEnabled(true);
    myMap.getUiSettings().setScrollGesturesEnabled(true);
    myMap.getUiSettings().setZoomGesturesEnabled(true);
    CameraUpdate center =
        CameraUpdateFactory.newLatLng(new LatLng(44.774402,17.194548));
    CameraUpdate zoom = CameraUpdateFactory.zoomTo(14);

     myMap.moveCamera(center);
     myMap.animateCamera(zoom);

     myMap.setOnMarkerClickListener(this);

     myMap.addMarker(new MarkerOptions()
     .position(new LatLng(44.778975,17.198238)) //marker za pmf
     .title(getString(R.string.pmf)));
  }
}
```

-Klasa namijenjena za čuvanje rasporeda:

```
public class RasporediHolder implements Serializable {
      private static final long serialVersionUID = -415258755476089790L;
      public  ArrayList<RasporedHolder>  rasporedi;
      public  ArrayList<Integer> rasporediId;
      public void init(){
            rasporedi= new ArrayList<RasporedHolder>();
            rasporediId = new ArrayList<Integer>();
      }
      public int size(){
            return rasporedi.size();
      }
      public int getId(int i){
            return rasporediId.get(i).intValue();
      }
      public RasporedHolder getRaspored(int i){
```

```
            return rasporedi.get(i);
      }
      public void add(RasporedHolder raspored){
            rasporedi.add(raspored);
            rasporediId.add(Integer.valueOf(generateId()));
      }
      public void remove(int i){
            rasporedi.remove(i);
            rasporediId.remove(i);
      }
      public void removeById(int id){
            int placeOfId = -1;
            for(int i = 0; i<rasporedi.size();i++){
                  if(rasporediId.get(i).intValue() == id){
                        placeOfId = i;
                  }
            }
            rasporedi.remove(placeOfId);
            rasporediId.remove(placeOfId);
      }
      public RasporedHolder getById(int id){
            int placeOfId = -1;
            for(int i = 0; i<rasporedi.size();i++){
                  if(rasporediId.get(i).intValue() == id){
                        placeOfId = i;
                  }
            }
            return rasporedi.get(placeOfId);
      }
      private int generateId(){
            int id = 0;
            for(int i = 0; i<rasporediId.size();i++){
                  if(rasporediId.get(i).intValue()>id){
                        id = rasporediId.get(i).intValue();
                  }
            }
            return (id+1);
      }
}
```

-Klasa koja prikazuje listu novosti:

```
public class Novosti extends Activity{
ObjectInputStream in;
private Vibrator vibe;
public static NovostiHolder novosti;
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                             WindowManager.LayoutParams.FLAG_FULLSCREEN);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.novosti_layout);
        Display display = ((WindowManager)
            getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay();
      int height = display.getHeight();
      TextView textView = (TextView)findViewById(R.id.textViewPredavaci);
      textView.setMaxHeight(height/8);
      vibe = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE) ;
```

```
LinearLayout linearLayout =
      (LinearLayout)findViewById(R.id.novostiLinearLayout);
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.toast_layout,
      (ViewGroup) findViewById(R.id.toast_layout_root));
TextView text1 = (TextView) layout.findViewById(R.id.text);
text1.setBackgroundResource(R.drawable.textbar);
text1.setTextColor(Color.WHITE);
Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.TOP, 0, height/2);
toast.setDuration(Toast.LENGTH_SHORT);
Bundle extras = getIntent().getExtras();
 boolean ucitao = extras.getBoolean("ucitao");
 if(ucitao){
      try {
            AndroidUtils.saveNovosti(novosti);
      } catch (Exception e) {
            e.printStackTrace();
      }
 }
 else{
      try {
            novosti = AndroidUtils.loadNovosti();
      } catch (Exception e) {
            e.printStackTrace();
      }
 }
 if (novosti == null){
      text1.setText(getText(R.string.nemanovosti));
      toast.setView(layout);
      toast.show();
 }
 else{
      for(int i = 0; i<novosti.getBrojNovosti(); i++){
            TextView tv=new TextView(getApplicationContext());
            tv.setText(novosti.getNovost(i).getNaslov());
            tv.setClickable(true);
            tv.setBackgroundResource(R.drawable.textbar);
            tv.setTextSize(20);
            tv.setGravity(Gravity.CENTER);
            tv.setId(i+600);
            tv.setTextColor(Color.WHITE);
            Animation anim = new TranslateAnimation(0, 0, height, 0);
            anim.setDuration(800);
            anim.setStartOffset(100*i);
            tv.setAnimation(anim);
            tv.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                        vibe.vibrate(50);
                        Intent i = new Intent();
                        i.setClassName("com.vlad1m1r.miidroid",
                            "com.vlad1m1r.miidroid.Novost");
                        i.putExtra("novost", v.getId()-600);
                        startActivity(i);
                  }
            });
            linearLayout.addView(tv);
            anim.start();
      }
```

```
        }
}
@Override
    public void onResume() {
        super.onResume();
        overridePendingTransition(R.anim.fadein,R.anim.fadeout);
    }
}
```

Korištena literatura

[1] Emil Protalinski, http://www.zdnet.com/android-malware-numbers-explode-to-25000-in-june-2012-7000001046/ , 23.12.2012.

[2] Guardian, http://www.guardian.co.uk/technology/2010/mar/14/google-mobile-phone-launch-delay , 25.03.2013.

[3] Lee, Wei-Meng (2012) Android 4, Razvoj aplikacija, Kompjuter biblioteka, Čačak, 3-5.

[4] Meier, Reto (2008) Profesional Android Application Development, Wiley Publishing, Inc., 7.

[5] Ryan Whitwam, http://www.extremetech.com/computing/104827-android-antivirus-apps-are-useless-heres-what-to-do-instead, 23.03.2013.

[6] SQLite, http://www.sqlite.org/different.html , 06.11.2012.

[7] Steele, James (2010) The Androids Developer's Cookbook, Addison-Wesley, 26.

[8] Stephen Shankland, http://news.cnet.com/8301-13580_3-9815495-39.html , 03.04.2013.

[9] W3C, http://www.w3.org/TR/REC-xml/ , 18.12.2012.

[10] BusinesWeek, http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal, 26.10.2012.