SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI

Filip Jakovljević

DIPLOMSKI RAD

RAZVOJ MOBILNE APLIKACIJE ZA PREGLED I KOMENTIRANJE WEB-SADRŽAJA

Mentor : dr. sc. Vedran Juričić

Zagreb, 2014.

Sažetak

U ovome radu opisan je razvoj mobilne aplikacije za pregled i komentiranje web-sadržaja, točnije web-portala s filmskim vijestima, te način rada i korištenja iste. Aplikacija je razvijana za operativni sustav Android, koji je uz iPhone OS trenutno najpopularniji operativni sustav za mobilne uređaje, a programski kôd je napisan u programskom jeziku Java.

Aplikacija ima nekoliko osnovnih funkcionalnih dijelova: pozadinski server koji se spaja na web-portal i dohvaća podatke, XML parser koji „prevodi“ te podatke, adapter koji podatke strukturira i fragment u kojem su podaci prikazani u listi. U aplikaciji je također implementirana i funkcionalnost za preslušavanje audio-sadržaja, jer web-portal između ostaloga objavljuje i epizode audio-serijala (*podcast*).

Aplikacija nije rađena isključivo u svrhu diplomskog rada, već postoji realna potreba za ovom aplikacijom i želja da se ona aktivno koristi među čitateljima web-portala.

Sadržaj

Popis slika

Popis tabela

1. Uvod

U ovome radu je detaljno predstavljena i opisana izrada mobilne aplikacije za prikaz i komentiranje postojećeg web-sadržaja. Prvi dio rada bavi se tehnologijama korištenim u izradi aplikacije, poput platforme Android za koju je aplikacija napravljena, mobilnog uređaja i verzija Androida na kojima je obavljeno testiranje aplikacije, softvera Eclipse koji je korišten za pisanje programskog kôda itd. U tom je dijelu predstavljena i teorija, kratka povijest razvoja i druge informacije o mobilnim aplikacijama. Cilj je izložiti potrebne informacije o pozadini rada na aplikaciji kako bi drugi dio rada bio jasniji i kako bi bilo što manje potrebe za objašnjavanjem pojedinih pojmova, procesa i korištenja tehnologija u drugom dijelu rada.

Mobilna aplikacija objašnjena u radu zapravo je jasniji i pristupačniji pregled web-stranice, odnosno određenog portala s vijestima. Cilj aplikacije jest omogućiti posjetiteljima portala lakše čitanje vijesti i drugih tipova objavljenih članaka, kako stranicu ne bi morali pregledavati u web-pregledniku na mobilnom uređaju, odnosno kako bi mogli lakše i brže navigirati po informacijama koje stranica nudi i doći do potrebnih informacija prikazanih u jednostavnom, ali cjelovitom prikazu.

Drugi dio rada objašnjava proces planiranja rada na aplikaciji. Najprije je detaljnije objašnjeno problemsko područje, odnosno ideja i potreba za ovim tipom aplikacije. Nakon toga definirani su zahtjevi vlasnika web-stranica. Definirano je što sve stranica mora sadržavati, kako bi to trebalo biti prikazano i posloženo u aplikaciji te na koje se aspekte aplikacije i informacija treba staviti naglasak.

Treći dio rada bavi se implementacijom, odnosno samim radom na programiranju aplikacije. Opisan je funkcionalni dizajn aplikacije, što i kako radi, uz objašnjenja dijelova korištenog kôda. Također je predstavljen i grafički dizajn aplikacije.

Cilj rada jest uvesti programere s manjim iskustvom u svijet izrade mobilnih aplikacija kroz teoriju i konkretan primjer izrade jedne naprednije, ali ne prekomplicirane aplikacije.

2. Mobilne aplikacije

Mobilne aplikacije specijalna su vrsta aplikacijskog softvera programiranog i dizajniranog za pokretanje i korištenje na tzv. pametnim telefonima (eng. *smartphone*), tabletima i drugim mobilnim uređajima. Distribucija se odvija preko specijaliziranih platformi koje nude originalne verzije, kao i sva naredna ažuriranja, nadogradnje i zakrpe aplikacija. Najpopularnije platforme toga tipa su App Store, Google Play, Windows Phone Store i BlackBerry App World. Mobilne aplikacije mogu biti besplatne (u cijelosti ili kao demo, odnosno *trial* verzija) ili se mogu naplaćivati, pri čemu 20 – 30% zarade od aplikacije prima platforma s koje je skinuta, a ostatak programer ili tvrtka koja ju je razvila. Uz korištenje na mobilnim uređajima neke se mobilne aplikacije mogu instalirati i na stolna ili prijenosna računala.

U svojim počecima mobilne su aplikacije bile zamišljene kao softver za rad s općim informacijama kao što su vijesti, elektronička pošta, kalendar (obično s dodatnim funkcijama), vremenska prognoza i slično, ali su zbog velike popularnosti i jednostavnosti korištenja postale osnovni oblik softvera za mobilne uređaje.[1] Uz stalni rast popularnosti pametnih mobitela, ali i tableta, ovo je trenutno najunosnija grana računalnog programiranja (prema istraživanjima tvrtke Gartner[2] samo u 2013. godini preuzete su preko 102 milijarde aplikacija, od čega je oko 91% bilo besplatno, ali je svejedno ostvaren profit od 26 milijardi američkih dolara, što je rast od 44,4% u odnosu na 2012. godinu).

[1] Wikipedia - Mobile App: http://en.wikipedia.org/wiki/Mobile_app (2014.)
[2] Lunden, Ingrid. Gartner: 102B App Store Downloads Globally In 2013, $26B In Sales, 17% From In-App Purchases, 19. rujna 2013., http://techcrunch.com/2013/09/19/gartner-102b-app-store-downloads-globally-in-2013-26b-in-sales-17-from-in-app-purchases/ (2014.)

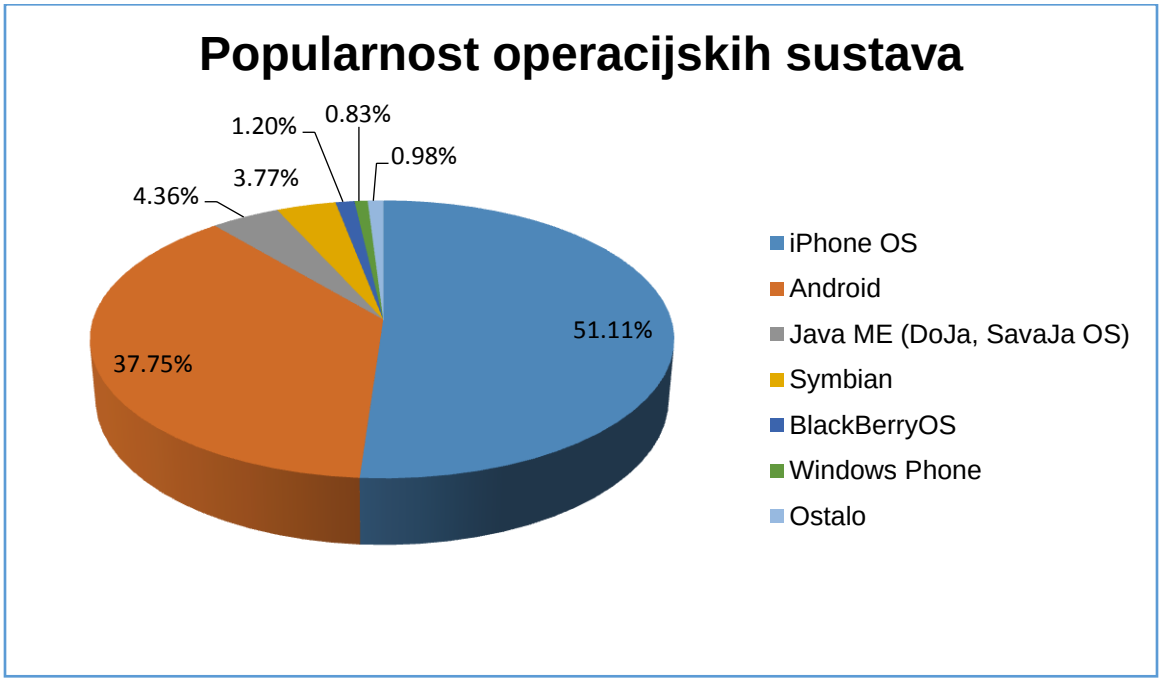
2.1. Razvoj mobilnih aplikacija

Različiti mobilni uređaji koriste različite operativne sustave (platforme) te se mobilne aplikacije moraju specijalno programirati za rad na svakom od njih. Najpopularnije platforme trenutno su Android (tvrtke Google), iOS (Apple Inc.) i Windows Phone (Microsoft). Budući da je aplikacija opisana u ovom radu rađena samo za platformu Android, taj je operativni sustav detaljnije opisan kasnije u radu. Razvijanje aplikacija odvija se u specijalnim alatima, odnosno integracijskim razvojnim okruženjima (eng. IDE – *Integrated development environment*), poput Eclipsea (korištenog u ovom slučaju), NetBeansa, Xcodea, Visual Studia i drugih. Testiranje mobilnih aplikacija može se odvijati na emulatorima na računalu i/ili na samim mobilnim uređajima, ukoliko su uključene i podešene postavke za programere. Primjeri najčešće korištenih emulatora su Google Android Emulator, Android SDK Emulator, MobiOne, TestiPhone, iPhoney, BlackBerry Simulator, Windows UI Automation i drugi.[3][4] Sljedeća tablica prikazuje kratku specifikaciju najpopularnijih operacijskih sustava za mobilne uređaj

[3] Wikipedia - Mobile application development: http://en.wikipedia.org/wiki/Mobile_application_development (2014.)
[4] Feigin, Ben. Mobile Application Development, The Search for Common Ground in a Divided Market, http://www.cs.cmu.edu/~bam/uicourse/830spring09/BFeiginMobileApplicationDevelopment.pdf (2014.)

Tabela 1 - Popularnost operacijskih sustava za mobilne uređaje s detaljima[5][6]

Proizvođač	Operacijski sustav	Programski jezik	Korištenost (u odnosu na sve korištene operacijske sustave, travanj, 2014.)
Apple	iPhone OS	Objective-C	51.11%
Open Handset Alliance	Android	Java	37.75%
Sun Microsystems	Java ME (DoJa, SavaJa OS)	Java	4.36%
Symbian Foundation	Symbian	C++	3.77%
RIM	BlackBerry OS	Java	1.20%
Microsoft	Windows Phone	Visual C#/C++	0.83%
Ostalo			0,98



Slika 1 - Dijagram korištenosti operacijskih sustava za mobilne uređaje

[5]Mobile Applications, ITU-T TechWatch Alert, srpanj 2009., http://www.itu.int/dms_pub/itu-t/oth/23/01/T230100000C0004PDFE.pdf (2014.)
[6] AppBrain Stats: http://www.appbrain.com/stats/number-of-android-apps(2014.)

3. Korištene tehnologije

Kao što je već ukratko navedeno, proces razvijanja mobilnih aplikacija zahtijeva nekoliko različitih vrsta tehnologija. Za početak predstavljen je operativni sustav Android, za koji je aplikacija razvijana, a zatim i ostale tehnologije koje su korištene u radu.

3.1. Android

Android je trenutno najpopularniji operativni sustav za mobilne uređaje u svijetu. Dizajnirala ga je tvrtka Android Inc., primarno za pametne telefone i tablete s ekranima osjetljivima na dodir, a kasnije ga je otkupio Google. Ovaj je operativni sustav baziran na Linux kernelu, operativnom sustavu otvorenog koda (eng. *open source*), što znači da omogućava vrlo opširnu personalizaciju i prilagodbu potrebama programera, odnosno zahtjevima klijenta. Uz aplikacije, Android nudi veliki broj raznih *widgeta*. Razlika između aplikacija i *widgeta* je u tome što je za korištenje aplikacije potrebno ući u aplikaciju, dok su *widgeti* dizajnirani kao *hands-on* alati, što znači da izravno prikazuju određene informacije ili funkcionalnosti dok su smješteni na radnoj površini uređaja.

Korisničko sučelje Androida vrlo je lako prepoznatljivo. Početni zaslon na dnu nudi traku za najčešće korištene aplikacije i funkcije (poput imenika, SMS-a, poziva, internetskog pretraživača, gumba za prikaz popisa aplikacija), dok je ostatak ekrana predviđen za personalizaciju uređaja prema potrebama korisnika (za postavljanje *widgeta* i drugih često korištenih aplikacija). Android također nudi mogućnost dodavanja dodatnih zaslona na koje se može navigirati jednostavnim povlačenjem početnog zaslona, ukoliko korisniku ponestane mjesta za dodavanje novih *widgeta* i aplikacija. Sučelje se prema tome bazira na direktnoj manipulaciji objekata na zaslonu, pri čemu se koriste aktivnosti koje Android čine vrlo jednostavnim i interaktivnim, poput povlačenja ikona za pokretanje aplikacija na mjesto gdje će nam biti nadohvat ruke,

dodira ikone kako bi se aplikacija pokrenula ili produženog dodira na objekt kako bi se otvorile postavke ili popis mogućih radnji povezanih s objektom.

Slika 2 - Primjer glavnog zaslona mobilnog uređaja s operacijskim sustavom Android

Broj aplikacija za Android u nevjerojatnom je porastu. Prema podacima s web-stranice AppBrain Stats od 11. svibnja 2014. godine, trenutno je na Google Playu dostupno 1 208 060 aplikacija, od čega je 994 722 besplatno. Najveći postotak aplikacija spada u kategoriju zabave, a slijede *lifestyle*, personalizacija, obrazovanje, alati, knjige, posao i putovanja.

Android nudi cijeli niz korisnih funkcionalnosti od kojih je, s obzirom na to da je Android primarno operacijski sustav za mobilne uređaje, naravno najvažnija mogućnost slanja i primanja SMS i MMS poruka i mogućnost uspostave telefonskih poziva. Android, kao i svi noviji operacijski sustavi, koristi prepoznatljivi prikaz za čitanje i pisanje SMS poruka koji podsjeća na *chat*. Jedna od vrlo korištenih funkcionalnosti jest i web-preglednik. Android nudi mogućnost instalacije različitih popularnih web-preglednika podešenih za rad na mobilnim uređajima (Google Chrome, Opera, Mozilla Firefox...). Još jedna vrlo zanimljiva funkcionalnost jest korištenje glasovnih naredbi (od

verzije 2.2 nadalje) za pokretanje aplikacije, pisanje i slanje poruka te manipuliranje drugim funkcionalnostima. Ostale mogućnosti uključuju odrađivanje više zadataka istovremeno (eng. *multitasking*), slikanje zaslona (eng. *screenshot*), videokonferneciranje, programe za pretvaranje teksta u govor za slabovidne osobe, dostupnost na velikom broju jezika itd. Interaktivnost Androida temelji se na korištenju takozvanih *push* notifikacija, koje mogu služiti za informiranje korisnika o primljenim porukama, potrebnim intervencijama ili ažuriranjima u aplikacijama, greškama u radu uređaja i slično.

Načini povezivanja uključuju Bluetooth, GSM/EDGE, Wi-Fi, LTE, CDMA, EV-DO, UMTS, NFC, IDEN i WiMAX. Android nudi i veliki opseg podržanih medijskih formata poput *streaminga* pomoću RealPlayera za Android, cijelog niza slikovnih formata (JPG, PNG, GIF, BMP...), audioformata (WAV, MP3, FLAC, MIDI...), videoformata, tekstualnih dokumenata itd. Podržan je i veliki broj hardverskih dodataka, uključujući GPS, videokamere, barometre, žiroskope, termometre i druge dodatke.

Android ima specifičan način upravljanja memorijom. Kada se neka aplikacija u Android OS-u prestane koristiti, sustav je automatski suspendira iz memorije, a ona, iako je zapravo još uvijek otvorena, ne troši nikakve resurse (poput baterije ili radne memorije), čime se smanjuje potrošnja baterije i povećava brzina rada mobilnog uređaja. Aplikacija tako „čeka“ u pozadini dok se opet ne počne koristiti. Ukoliko je nekoliko aplikacija otvoreno, što rezultira s premalo dostupne radne memorije, sustav automatski počinje gasiti aplikacije i procese koji se nisu koristili duže vrijeme, kako bi oslobodio potrebnu memoriju.

Logički i funkcionalni ustroj Android aplikacija se temelji na samostalnim funkcionalnim dijelovima koji se nazivaju *activityjima.* Postoje tri vrste *activityja*:

- *Activity* u prvom planu (eng. *foreground activity*) – tip *activity*ja koji sadrži funkcionalnosti koje direktno ovise o odlukama korisnika, odnosno reagiraju na korisnikove aktivnosti unutar aplikacije (pritisak gumba, povlačenje zaslona, unos teksta...). Ovaj tip *activityja* uvijek ima svoj grafički prikaz na zaslonu.

- Pozadinski servis – pozadinski servisi svoje funkcionalnosti odrađuju bez korisnikovog znanja i mogu, ali ne moraju, biti aktivirani korisnikovim aktivnostima u aplikaciji. Ovaj tip *activityja* nema grafički prikaz jer nikada ne dolazi u prvi plan, već se odrađuje u pozadini.
- Intermitentni *activity* – kombinacija prva dva tipa activityja. Mogu imati grafički prikaz i reagirati na korisnikove aktivnosti, ali isto tako mogu koristiti iste informacije za pozadinske funkcionalnosti poput slanja notifikacija korisniku.[7]

Još jedan vrlo važan aspekt Androidovih aplikacija jesu *intenti* (namjere, intencije). *Intent* je naredba koju namjeravamo izvršiti, ali ne unutar *activityja* u kojem se trenutno nalazimo. Sastoji se od dva dijela: aktivnosti i podataka. Aktivnost je sama naredba koja se treba izvršiti (poput otvaranja nekog drugog *activityja* ili pokretanja pozadinskog servisa), a podaci su informacije koje imamo u trenutnome *activityju*, a potrebne su nam za uspješan rad *activityja* koji se treba otvoriti. Postoje dva tipa:

- Eksplicitni *intent* – ima definiranu komponentu (primjerice klasu, odnosno *activity*) koju pokreće. Može i ne mora sadržavati dodatne podatke koje šalje toj komponenti.
- Implicitni *intent* – nema definiranu komponentu, već mora sadržavati dovoljno podataka kako bi sustav mogao odlučiti koju komponentu je najbolje pokrenuti ovisno o tim podacima.[8]

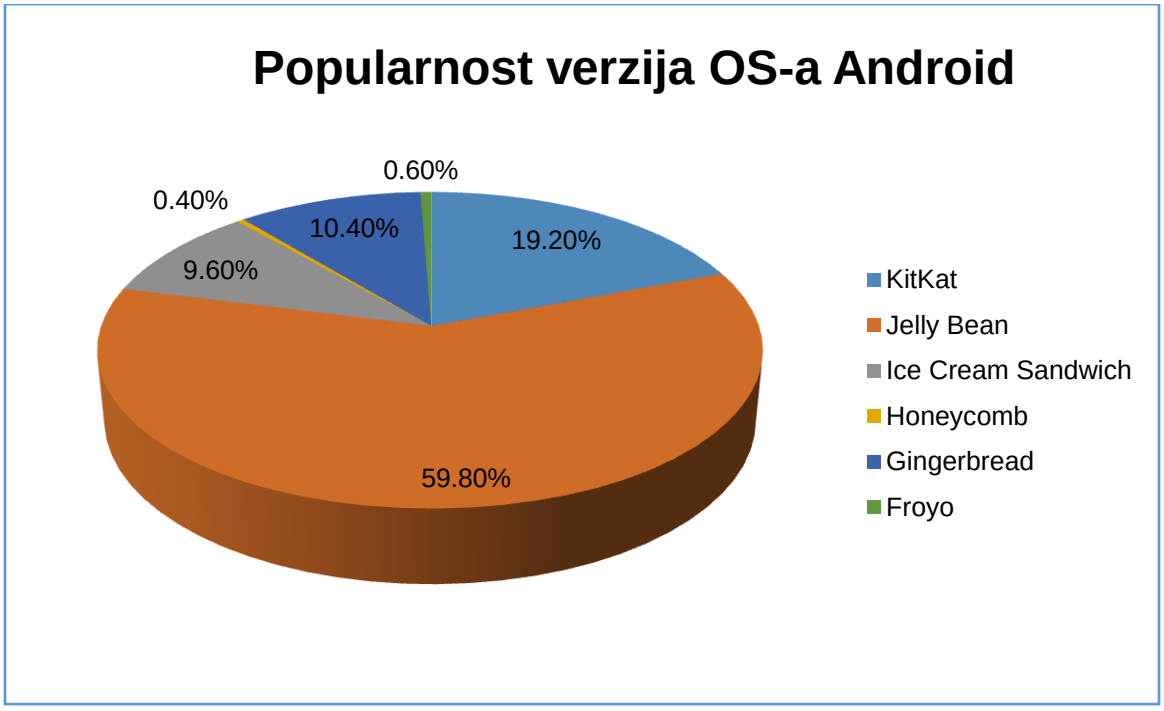
Najnovija verzija Android platforme trenutno je Android 4.4 (KitKat), ali trebat će proći još neko vrijeme dok ne postane i najpopularnija verzija Androida. Sljedeća tablica prikazuje popis trenutno najkorištenijih verzija Androida s datumima njihovih izlazaka na tržište i njihovom popularnosti, odnosno korištenosti (podaci od 1. svibnja 2014. temeljeni na verzijama Androida na mobilnim uređajima koji su pristupali Google Playu):

[7] Meier, Reto. Professional Android Application Development. Wiley Publishing, Inc.2009. (2014.) str. 29-30

[8] Android Developers – Intent: http://developer.android.com/reference/android/content/Intent.html (2014.)

Tabela 2 - Verzije operacijskog sustava Android s detaljima

Verzija	Ime	Datum izlaska	API level	Korištenost
4.4	KitKat	31. listopada 2013.	19	19.2%
4.3.x	Jelly Bean	24. srpnja 2013.	18	59.8%
4.2.x		13. studenog 2013.	17	
4.1.x		9. srpnja 2012.	16	
4.0.3 – 4.0.4	Ice Cream Sandwich	16. prosinca 2011.	15	9.6%
3.2	Honeycomb	15. srpnja 2011.	13	0.4%
2.3.3 – 2.3.7	Gingerbread	9. veljače 2011.	10	10.4%
2.2	Froyo	20. svibnja 2010.	8	0.6%[9]



Slika 3 - Dijagram korištensoti pojedine verzije Android OS-a

Androidova laka prilagodljivost omogućava i njegovo korištenje u drugim uređajima, osim mobilnih telefona i tableta, poput laptopa, *netbookova*, *smartbookova*,

[9] AppBrain Stats: Most common SDK versions: http://www.appbrain.com/stats/top-android-sdk-versions (26. Kolovoza 2014.)

„pametnih TV-a“ (eng. *smart TV*), videokamera itd. Novije implementacije Androida javljaju se i u inovacijama poput „pametnih naočala“ (eng. *smart glasses*), poput Google Glassa, „pametnih satova“ (eng. *smartwatch*), koji će biti uključeni u projekt Android Wear, električnih zrcala, te manjih uređaja poput slušalica ili CD i DVD čitača za automobile. Kampanja za financiranje Ouye, konzole za videoigre koja radi na Android operacijskom sustavu, jedna je od najuspješnijih na Kickstarteru, web-stranici za prikupljanje financijskih sredstava za različite projekte, s prikupljenih 8.5 milijuna američkih dolara. Razvoj Ouye je kasnije popraćen i razvojem drugih konzola temeljenih na Androidu poput Project Shield tvrtke Nvidia.

Godine 2011. Google je predstavio Android@Home, sustav temeljen na Androidu, koji se koristi za manipuliranje raznim uređajima u kućanstvu poput prekidača za svjetlo, termostata, električnih zavjesa, televizora, kuhinjskih uređaja itd. Najavljeni su i prototipovi žarulja koje bi mogle biti kontrolirane preko mobilnih uređaja.

Stereosustavi za automobile doživjeli su veliki napredak zahvaljujući korištenju Androida. Tvrtka Parrot 2011. je godine pustila u proizvodnju sustav nazvan Asteroid, a godinu dana kasnije naslijedio ga je Asteroid Smart koji je koristio zaslon osjetljiv na dodir. U siječnju 2014. Google je najavio suradnju s nekoliko velikih proizvođača automobila (Audi, General motors, Hyundai i Honda), čiji je cilj izrada različitih sustava za zabavu isključivo za automobile tih proizvođača.[10][11]

3.2. Eclipse

Softver korišten za izradu ove mobilne aplikacije bio je Eclipse. To je integrirano razvojno okruženje u kojem je primarni programski jezik za razvoj aplikacija Java, ali podržava i razvijanje aplikacija u drugim programskim jezicima (Ada, ABAP, C, C++,

[10] Wikipedia – Android (operating system): http://en.wikipedia.org/wiki/Android_(operating_system) (2014.)
[11] Android, the world's most popular mobile platform: http://developer.android.com/about/index.html (2014.)

COBOL, Fortran, JavaScript, Lasso, Natural, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy, Scheme, Erlang). Eclipse se sastoji od osnovnog razvojnog okruženja i opsežnog sustava priključaka (eng. *plug-in*) za prilagođavanje okruženja potrebama programera. Osnovni paket alata za razvijanje u Javi za Eclipse je Eclipse SDK (*software development kit*), a programeri mogu proširiti njegove funkcionalnosti instalacijom priključaka razvijenih za Eclipse platformu, poput paketa razvojnih alata za druge programske jezike. Eclipse SDK je besplatan softver otvorenog koda. Ime Eclipse dolazi od činjenice da je u početku glavna konkurencija Eclipseovim proizvodima bio već dokazani i široko korišteni Microsoft Visual Studio kojeg je novi softver trebao zasjeniti (eng. *eclipse*, *overshadow*).

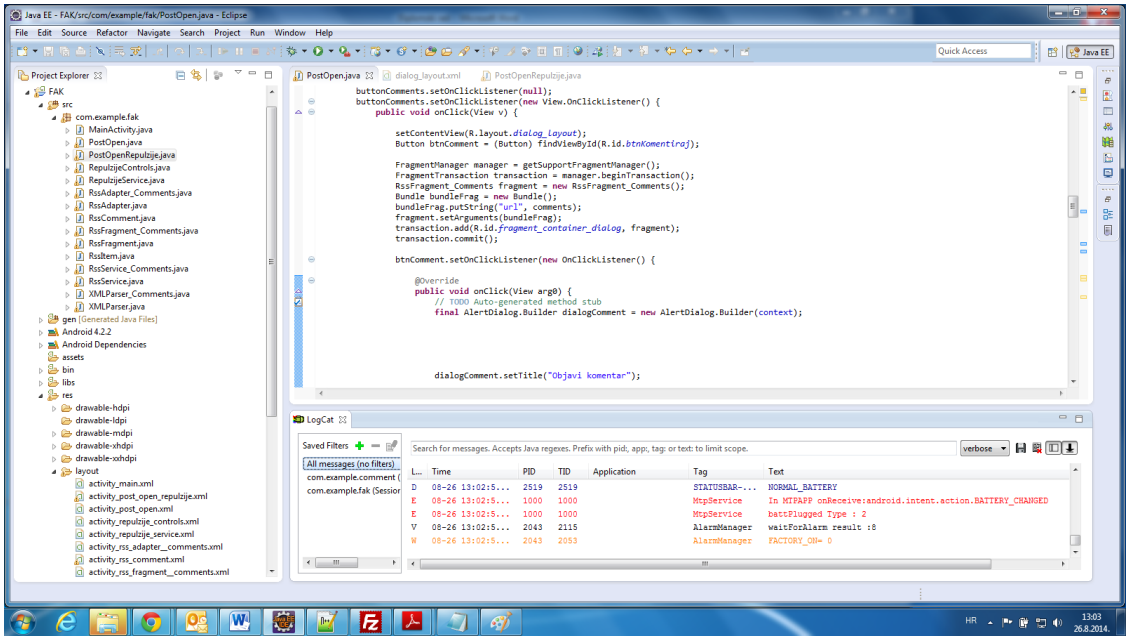
Eclipse pruža programeru mogućnost razvijanja aplikacija temeljenih na tzv. Rich Client Platform alatu koji omogućava integraciju softverskih komponenti koje su neovisne jedna o drugoj i gdje se većina funkcija i obrade podataka u aplikaciji odvija na klijentskoj strani. Rich Client Platform u Eclipseu se sastoji od sljedećih komponenti:

- Equinox OSGi – modula koji omogućava implementaciju aplikaciju u obliku više nakupina (eng. *bundle*) podataka
- osnovne platforme s priključcima
- paketa alata za *widgete* – Standard Widget Toolkit
- JFace - skupa klasa za prikazivanje podataka u aplikaciji
- Eclipse Workbench – integriranog *frameworka* za olakšani razvoj aplikacija (uključuje editore, poglede, čarobnjake za lakše kreiranje i modifikaciju komponenti)

Osnovna arhitektura aplikacije u Eclipseu temelji se na razdjeljivanju projekta na funkcionalne dijelove (*activity*). Svaki kreirani *activity* sa sobom automatski kreira i svoj grafički prikaz, odnosno *layout* (koji se koristi za uređivanje korisničkog sučelja za taj *activity*, ukoliko *activity* sadrži prikaz podataka). *Activity* može raditi i u pozadini, bez otvaranja u aplikaciji kao produžetak ili podrška *activityju* koji je trenutno prikazan u aplikaciji, a sastoji se od jedne ili više klasa koje sadrže definicije podataka i funkcionalnosti *activityja*, pri čemu je jedna klasa glavna, a ostale su sporedne. Format datoteke u kojoj se pohranjuje *activity* jest .java, dok se *layout* sprema u datoteke

formata .xml. Ostali formati datoteka koji se često koriste u razvoju su .png (za unos slika u aplikaciju), .jar (Java Archive, format datoteke knjižnica za Javu), .properties, .dex, .txt, itd., no najvažniji je „krajnji proizvod“, odnosno datoteka formata .apk. Ta je datoteka zapravo Androidov paket koji sadrži sve podatke i resurse za dijeljenje i instalaciju aplikacije.[12][13]

Radna površina Eclipsea vrlo je slična ostalim programima za programiranje. Sastoji se od niza prozora koji se mogu razmještati po sučelju ovisno o potrebama programera. Slika prikazuje najkorištenije dijelove Eclipsea. U sredini se nalazi prostor za uređivanje koda (eng. *editor*) koji zauzima najveći dio prikaza. S lijeve je strane prozor za upravljanje objektima (eng. *object manager*) u kojem se nalazi cijela hijerarhija objekata u svim projektima trenutno otvorenima u programu. Ispod prostora za uređivanje koda nalazi se prozor za praćenje i zapisivanje procesa koji se odvijaju u Eclipseu, kao i prozor za čitanje mogućih grešaka (eng. *log*), a iznad alatna traka. Svaka od kartica na vrhu editora sadrži kôd jedne klase, *activityja* ili nekog drugog dijela razvijane aplikacije.



Slika 4 - Primjer radne površine u programu Eclipse

[12] Wikipedia – Eclipse (software): http://en.wikipedia.org/wiki/Eclipse_(software) (2014.)
[13] About the Eclipse Foundation: http://www.eclipse.org/org/ (2014.)

3.3. Objektno orijentirano programiranje

Objektno orijentirano programiranje (OOP) oblik je računalnog programiranja u kojem je sve grupirano u objekte. Objekti su svi elementi koji mogu odrađivati neki skup funkcija. Skup funkcija koje objekt može odraditi određuje kojeg je tipa taj objekt. Jedan objekt može pozvati drugi objekt da odradi neku funkcionalnost,a isto tako može biti pozvan od strane drugog objekta. Pozivanje se radi preko poruke koja sadrži informaciju koja se funkcionalnost objekta treba odraditi i po potrebi parametre ili neke druge informacije potrebne za pravilno izvršavanje funkcije. Funkcije unutar objekta definirane su pomoću metoda. Svaka metoda može, ali ne mora, primati ulazne parametre koji ukoliko su definirani moraju biti poslani metodi kako bi se mogla izvršiti.

Objekti najčešće predstavljaju elemente iz stvarnog svijeta, a definirani su pomoću klasa. Klase sadrže detalje o objektu i metode koje se mogu pozvati nad tim objektom. Klase podilaze zakonima nasljednosti u programiranju. To znači da postoji glavna klasa koja može imati podklase čije metode može pozivati. Primjerice, u stvarnom svijetu imamo neko motorno vozilo i to je objekt u programiranju. Taj je objekt definiran klasom koja sadrži podatke o tipu vozila, brzini, boji i cijeni, te metodu za promjenu boje vozila.

Koncepti na kojima se OOP temelji:

- klasa – nacrt ili predložak objekta koji sadrži detalje o objektu i metode koje se nad objektom mogu izvršavati
- objekt – instanca klase, temeljen na elementu iz stvarnog svijeta
- metoda – funkcija koja se izvršava nad objektom
- ovijanje – grupiranje podataka i funkcija u pojedine komponente
- nasljeđivanje – svojstvo objekata i klasa koje im omogućava da budu temeljeni na nekom drugom objektu ili klasi. Omogućava korištenje funkcionalnosti jednog objekta u drugome.
- otvorena rekurzija – svojstvo objekta da može pozivati metode objekata koji su u kôdu definirani nakon njega

- apstrakcija – način smanjivanja i razdvajanja detalja kako bi se mogli usmjeriti na manje koncepata. Omogućava da se svojstva više pojedinih, ali sličnih objekata definiraju unutar iste klase.[14][15]

3.4. Java

Java je programski jezik koji od 1995. godine razvija firma Sun Microsystems (koji je kasnije postao dio firme Oracle Corporation). Jedan je od najkorištenijih programskih jezika današnjice i većina naprednijih web-stranica ne može normalno raditi bez nje. Java Runtime Environment je okruženje koje je potrebno instalirati na klijentsko računalo kako bi takve web-stranice mogle normalno raditi, odnosno kako bi se na njima mogle izvršavati funkcionalnosti razvijene u Javi.

Java se temelji na klasama i objektno orijentiranom programiranju te omogućava izvršavanje više funkcionalnosti u aplikaciji istovremeno. Ovaj programski jezik teži WORA (*write once, run anywhere*) principu koji bi trebao programerima omogućiti da kôd koji su napisali za jednu platformu ne mora biti promijenjen kako bi radio na drugoj.

Kreator Jave, James Gosling, temeljio je Javu na 5 osnovnih pravila:

- trebala bi biti jednostavna, objektno-orijentirana i prepoznatljiva
- trebala bi biti robusna i sigurna
- trebala bi biti neovisna o arhitekturi platforme i lako prijenosna
- trebala bi funkcionirati na visokom nivou
- trebala bi biti interpretirana, temeljena na nitima (eng. *thread*) procesa i dinamična.

[14] Introduction: What is Object-Oriented Programming?: http://www.inf.ufsc.br/poo/smalltalk/ibm/tutorial/oop.html (2014.)
[15] Wikipedia - Object oriented programming: http://en.wikipedia.org/wiki/Object-oriented_programming (2014.)

Gosling se tih pravila čvrsto držao i rezultat je Java koja je prijenosna (programi napisani u Javi mogu raditi slično na bilo kojem hardveru ili operacijskom sustavu). To se postiže prevođenjem (eng. *compile*) Jave u Java *bytecode* umjesto da se izravno prevodi u kôd sustava na kojem bi trebala raditi. Java *bytecode* tada šalje naredbe sustavu koji mora imati implementiran neki oblik virtualne mašine koja ga može interpretirati i lokalizirati (najrašireniје takvo okruženje je Java Runtime Environment).

Programi napisani u Javi imaju reputaciju da su sporiji od onih napisanih u C++, te da koriste više radne memorije za vrijeme rada. Njihova se brzina znatno popravila uvođenjem JIT prevoditelja (*Just-in-time compiler*) 1997. godine za verziju Jave 1.1. Ovaj tip prevoditelja omogućuje prevođenje Java *bytecodea* u kôd sustava za vrijeme rada programa (eng. *at runtime*).

Java koristi tzv. „automatski sakupljač smeća“ odnosno automatski proces manipulacije radnom memorijom koji omogućava povrat neiskorištene radne memorije kada se neki objekt prestane koristiti. Kada nestanu sve reference na određeni objekt, Java radnu memoriju koja je bila iskorištena za rad s tim objektom vraća i ponovno je čini dostupnom za druge aktivnosti i objekte, čime poboljšava brzinu rada aplikacije i smanjuje mogućnost gubitka, odnosno curenja memorije (eng. *memory leak*). Do gubitka memorije ipak može doći u određenim slučajevima. Jedan od najčešćih primjera jest slučaj kada se neki objekt više ne koristi ali i dalje postoji referenca na primjerice klasu u kojoj se taj objekt nalazi što rezultira daljnjim čuvanjem radne memorije za taj objekt i ima utjecaj na performanse programa. Unatoč tome ova funkcionalnost Jave pošteđuje programere od mukotrpnog posla pisanja kôda za ručno manipuliranje memorijom.

Velik dio sintakse Jave proizlazi iz programskog jezika C++. Međutim, velika je razlika u tome što C++ kombinira sintaksu za strukturno, generičko i objektno orijentirano programiranje, dok je Java gotovo u potpunosti objektno orijentirana. Sav je kôd napisan unutar klasa i svaka komponenta u kôdu je objekt (osim primitivnih tipova podataka poput *integera*, floata, booleanskih vrijednosti i znakova). Java za razliku C++-a ne podržava ni višestruko nasljeđivanje kod klasa, što je čini jednostavnijom i smanjuje mogućnost pogrešaka. U svrhu testiranja različitih verzija kôda u aplikaciji

često je korišteno komentiranje dijelova postojećeg kôda, zbog čega se treba kratko posvetiti i tom aspektu Jave. Metode komentiranja su vrlo slične kod oba programska jezika, a postoje tri tipa:

- komentiranje pojedinog retka kôda (//)
- komentiranje dijela kôda od više redaka (otvaranje komentara sa /*, te zatvaranje sa */)
- Javadoc komentiranje (otvaranje komentara sa /**, te zatvaranje sa */) – omogućava generiranje API dokumentacija u HTML-u iz kôda napisanog u Javi.

Sun Microsystems definirao je i podržava četiri tipa (edicije) Jave od kojih svaka cilja na različita aplikacijska okruženja. To su:

- Java Card – za Smart kartice
- Java ME (Java Platform, Micro Edition) – za platforme s limitiranim resursima (mobilni uređaji)
- Java SE (Java Platform, Standard Edition) – za lokalne i serverske radne platforme (stolna i prijenosna računala)
- Java EE (Java Platform, Enterprise Edition) – za veće radne platforme (organizacijske i internetske platforme).[16][17]

3.5. RSS

RSS (*Rich Site Summary*, često nazivan i *Really Simple Syndication*) koristi standardne web-formate za prikazivanje informacija koje se često osvježavaju poput zapisa na blogu, vijesti, sportskih rezultata itd. RSS se prikazuje u obliku *feeda*, odnosno liste zapisa koji se pojavljuju na izvoru, a može uključivati samo naslove, neke

[16] Learn About Java Technology: http://www.java.com/en/about/ (2014.)
[17] Wikipedia – Java (programming language): http://en.wikipedia.org/wiki/Java_(programming_language) (2014.)

opise zapisa ili cijele zapise, kao i podatke o zapisu poput podataka o kreatoru zapisa ili vremena kada je zapis kreiran.

Standardni tip dokumenta u koji RSS koristi za izvlačenje informacija je XML dokument generiran najčešće na izvoru, koji RSS zatim parsira i ispisuje u obliku lakše čitljivom za korisnika. Korištenjem XML dokumenata omogućava se kompatibilnost RSS-a s velikim brojem različitih sustava, programa i uređaja.

RSS omogućava korisniku pregled informacija, odnosno različitih zapisa (tekst, audio, video, slika) s izvora (najčešće web-stranica) bez da pristupa samom izvoru. Tako, naprimjer, korisnik može pročitati naslove najnovijih vijesti na nekom portalu bez da uopće otvara web-preglednik te zatim, ukoliko nađe nešto zanimljivo , otići na tu web-stranicu i pročitati vijesti koje ga zanimaju, odnosno koje je našao u RSS *feedu*. Naravno, ukoliko je RSS tako podešen, korisnik preko njega može pročitati i cijele vijesti bez potrebe za otvaranjem web-preglednika.

Još jedna od vrlo korisnih mogućnosti RSS-a jest filtriranje prikazanih informacija po kategorijama, tematici ili nekim drugim parametrima što omogućava laku personalizaciju *feeda* prema potrebama i željama korisnika.

Softver za prikaz RSS *feeda*, nazvan RSS čitač (eng. *RSS reader*), može biti napravljen kao samostalna web, desktop ili mobilna aplikacija ili implementiran kao dio jedne od njih. RSS čitač samostalno automatski osvježava *feed* u određenom vremenskom periodu. Ovaj softver iz xml dokumenta izvlači informacije tako što traži definirane oznake (eng. *tag*) te prema njima razumije strukturu svake stavke (eng. *item*). Jedna stavka obično predstavlja i jedan zapis s izvora, a unutar nje onda su definirane oznake koje specificiraju koji podaci iz zapisa trebaju biti prikazani u *feedu*.

Od 1999. godine izašlo je nekoliko verzija RSS-a:

- RSS 1 – uključuje RSS 0.90, RSS 1.0 i RSS 1.1
    - RSS 0.90 – originalni Netscapeov proizvod iz 1999. godine, poznat i kao RDF Site Summary. Temeljio se na ranoj logici RDF standarda.
    - RSS 1.0 – proizvod tvrtke RSS-DEV Working Group, također poznat kao RDF Site Summary. Ova je verzija, kao i prethodni RSS 0.90, bila

temeljena na RDF standardu, ali nije u potpunosti kompatibilna s prethodnikom.

  - RSS 1.1 – nezavisni proizvod otvorenog formata, nepodržan od strane RSS-DEV Working Group ili bilo koje druge organizacije, a napravljen s ciljem da u potpunosti zamijeni RSS 1.0, a ne kao napredna verzija istog.

- RSS 2 – uključuje RSS 0.91, RSS 0.92 i RSS 2.0.1
  - RSS 0.91 – pojednostavljena verzija RSS-a razvijane od strane Netscapea. Ova je verzija preuzela ime *Rich Site Summary*, s obzirom na to da više nije bila temeljena na RDF standardu.
  - RSS 0.92 (do RSS 0.94) – ekspanzije verzije RSS 0.91 koje su većinom bile kompatibilne međusobno.
  - RSS 2.0.1 – od ove se verzije puni naziva RSS-a mijenja u Really Simple Syndication. Glavna je karakteristika ove verzije uključivanje logike za prepoznavanje xml oznaka.[18]

3.6. XML

Extensible Markup Language (XML) je jezik za označavanje koji služi za definiranje skupa pravila za kodiranje dokumenata u formatu koji lako mogu čitati i informacijski sustavi i čovjek. U dizajnu XML-a naglasak je na jednostavnosti, općenitosti i lakoći korištenja na internetu. To je tekstualni podatkovni format s jakom podrškom za različite ljudske jezike kroz Unicode standard za kodiranje. Iako se dizajn XML-a fokusira na dokumente, najčešće se koristi za prezentiranje podataka u hijerarhijskoj i logičkoj strukturi.

Od 2009. godine razvijene su stotine formata dokumenata koji koriste XML sintaksu, poput RSS-a, Atoma, SOAP-a i XHTML-a, a koristi se i u programima

[18] Wikipedia – RSS: http://en.wikipedia.org/wiki/RSS (2014.)

Microsoft Officea (Office Open XML), kao i Appleovom iWork-u. XML je također implementiran kao temeljni jezik za neke komunikacijske protokole poput XMPP-a, a mnogo aplikacija razvijenih u Microsoft .NET-u koristi XML u definiranju konfiguracijskih dokumenata.

Ključni pojmovi:

- (Unicode) znak – po definiciji XML dokument je niz znakova, a gotovo svaki Unicode znak može se pojaviti u XML dokumentu
- procesor (XML parser) – analizira oznake u XML dokumentu i šalje strukturirane podatke aplikaciji
- oznake i sadržaj – nizovi znakova u XML dokumentu dijele se na oznake i sadržaj, koji se razlikuju implementacijom jednostavnih sintaktičkih pravila. Nizovi znakova koji predstavljaju oznake su obično ograđeni izlomljenim zagradama, odnosno njihov se početak označava znakom „<“, a završetak znakom „>“. Nizovi znakova koji ne predstavljaju oznake nazivamo sadržajem. Sadržaj se obično nalazi između otvarajuće i zatvarajuće oznake (zatvarajuća se oznaka označava kosom crtom (/) i istim imenom kao i otvarajuća oznaka). Oznake mogu predstavljati i prazan sadržaj (čija je vrijednost *null*, odnosno nemaju vrijednost), a u tom se slučaju ne koriste otvarajuće i zatvarajuće oznake već samo jedna oznaka koja nakon imena ima razmak i kosu crtu. Ukoliko napišemo otvarajuću i zatvarajuću oznaku bez sadržaja između njih, parser taj sadržaj prepoznaje kao prazno mjesto, ali mu i to predstavlja vrijednost. Primjeri:
  - otvarajuća oznaka - <ime>
  - zatvarajuća oznaka - </ime>
  - oznaka bez sadržaja - <ime />
  - sadržaj između oznaka - <ime>**Marko**</ime>
- element – skup koji se sastoji od otvarajuće oznake, sadržaja i zatvarajuće oznake (kao u ranije navedenom primjeru)
- atribut – služi za opisivanje sadržaja ili definiranje vrijednosti koje nisu tekstualni sadržaj poput slika. Atributi se sastoje od imena i vrijednosti, a zapisuju se unutar otvarajuće oznake ili oznake bez sadržaja, iza njihovog imena. Primjer:

    - atribut za definiranje sadržaja koji je slika: - ova oznaka ima dva atributa: src koji definira dokument sa slikom i alt koji služi kao opis slike u dokumentu
- XML deklaracija – XML dokumenti mogu započeti deklaracijom koja sadrži neke podatke o dokumentu. Deklaracija izgleda slično kao oznaka, ali je definirana znakovima ? na početku i kraju. Primjer:
    - <?xml version="1.0" encoding="UTF-8"?>

XML dokumenti dopuštaju korištenje isključivo znakova iz Unicode standarda. Gotovo svi znakovi koji su definirani Unicode standardom mogu se koristiti u sadržaju, izuzev nekolicine kontrolnih znakova. Set znakova iz Unicode standarda može se podijeliti u bajtove za lakoću spremanja. Sam standard definira kodiranja koja uključuju sve moguće znakove Unicode standarda, od kojih su najpoznatija UTF-8 i UTF-16. Postoje i stariji standardi kodiranja poput ASCII-a i ISO/IEC 8859, ali njihovi su setovi znakova gotovo uvijek uključeni kao podskupovi Unicodeova seta. XML u sebi sadrži i mehanizam koji omogućava XML parseru lako prepoznavanje o kojem se kodiranju radi.

XML koristi neke znakove koji su dio Unicode seta, ali se ne mogu koristiti u sadržaju XML dokumenta. Znakovi poput „<" i „&" sintaksni su znakovi i nikada se ne pojavljuju u sadržaju. Jedan od problema Unicodea i XML-a jest globalno korištenje, što znači da se koristi velik broj znakova koji nisu dostupni na svim jezicima i računalima. Primjerice, u hrvatskom se jeziku koristi slovo „č" koje nije dostupno ukoliko su jezične postavke konfigurirane na engleski jezik. Zbog toga Unicode omogućava korištenje takozvanih *glyphova*, odnosno nizova znakova latinskog pisma koji se koriste u svim jezicima. Svaki takav niz znakova označava jedan „specijalni" znak (č, ć, ÿ, ė...). Neki od setova znakova mogu predstavljati i dijelove sintakse dokumenta:

- < predstavlja „<"
- > predstavlja „>"
- & predstavlja „&"

- ' predstavlja „'“
- " predstavlja „““

Svi se znakovi mogu prikazati i brojevnom referencom. Primjerice, kineski znak „中“ može se prikazati kao 中 ili 中

Komentari se u XML dokumentu mogu pojaviti bilo gdje osim ispred deklaracije dokumenta. Otvaraju se nizom znakova „<!--“, a zatvaraju nizom znakova „-->“.

Kako bi se održala struktura i jednostavnost parsiranja XML dokumenti se temelje na velikom broju pravila, od kojih su neka od najvažnijih:

- dokument može sadržavati samo znakove definirane Unicode standardom
- znakovi poput „<“ i „&“ ne mogu se pojaviti ni u jednom obliku ili funkciji osim kod označavanja funkcionalnosti drugog znaka ili niza znakova
- oznake za otvaranje, zatvaranje i oznake bez sadržaja koje definiraju elemente moraju biti ispravno ugniježđene, nijedna ne smije nedostajati i dvije se ne mogu ni u jednom segmentu preklapati
- imena oznaka osjetljiva su na korištenje velikih i malih slova,a pâr oznaka za otvaranje i zatvaranje mora imati identično ime
- imena oznaka ne mogu sadržavati nijedan od sljedećih znakova: !"#$%&'()*+,/;<=>?@[\]^`{|},~
- imena oznaka ne mogu započeti brojkom ili znakovima „-“ i „.“
- postoji samo jedan korijenski (eng. *root*) element, svi ostali su mu podređeni.

XML dokumenti mogu biti validni, što znači da u njima postoji referenca na drugi dokument tipa DTD (*Document Type Definition*), te su njegovi elementi i atributi deklarirani u tom dokumentu. DTD dokument služi za definiranje sheme, odnosno gramatike XML dokumenta. XML shema definira, odnosno ograničava skup elemenata koji se mogu koristiti u dokumentu, koji se im se atributi mogu pridružiti, redoslijed kojim su poslagani i odnose između elemenata o podređivanju (roditelj/dijete). Nakon DTD-a pojavio se i XSD (XML *Schema Definition*), koji je sve popularniji zbog detaljnijeg

opisivanja XML shema. Ostali standardi koji se koriste za opisivanje XML shema su RELAX NG, Schematron, ISO DSDL, itd.

XML se može povezivati i s drugim tipovima dokumenata, od kojih svaki služi za specifikaciju nekog aspekta XML-a koji nije definiran u samom dokumentu, poput XML Namespaces, XML Base, XML Information Set, XPath, XSLT, XSL Formatting Objects, XQuery, XML Signature, XML Encryption, XML Core, XInclude, XLink, XPointer.[19][20]

Kao što je ranije objašnjeno, jedan je od osnovnih ciljeva XML-a olakšavanje pisanja koda za programe koji će koristiti XML dokumente. Unatoč tome XML specifikacija ne sadrži nikakve informacije o tome kako tu karakteristiku XML-a iskoristiti pri programiranju.

3.7. WordPress

Iako nije direktno povezan s razvojem i radom aplikacije opisane u ovom radu, potrebno je ukratko predstaviti i Wordpress alat. Wordpress se koristi za manipuliranje i uređivanje podataka na web-stranici koja funkcionira kao blog. To je alat pomoću kojeg se članci na web-stranici uređuju, objavljuju i brišu, a pokriva i cijeli niz drugih funkcionalnosti stranice. Temelji se na arhitekturi priključaka (eng. *plug-in*) koji proširuju njegove osnovne funkcionalnosti ili dodaju nove. Wordpress je trenutno najpopularniji alat za pisanje blogova.

U radu aplikacije sudjeluje kôd generiranja RSS *feeda*. Korištenjem priključka definirano je da RSS *feed* prikazuje 20 najsvježijih članaka i filtrirane su kategorije članaka koje trebaju biti sadržane u RSS *feedu*. Wordpress također nudi i automatsko generiranje RSS *feeda* s komentarima za pojedine članke, što je još jedna funkcionalnost koja se koristi u prikazu podataka u ovoj aplikaciji.

---

[19] Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M.; Maler, Eve; Yergeau, Francois: Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml/, 26. studenoga 2008. (2014.)

[20] Wikipedia – XML: http://en.wikipedia.org/wiki/Xml (2014.)

Wordpress kao alat, odnosno aplikacija, postoji i za mobilne uređaje. Iako mu je funkcionalnost u ovome obliku limitirana, omogućava dodavanje novih članaka i komentara, moderiranje komentara, uređivanje objavljenih članaka, pregled statistika i slično.[21][22]

3.8. Podcast

Podcast je digitalni medij koji se sastoji od audiozapisa ili audio-videozapisa čiji je sadržaj tematski povezan, a objavljuje se periodično u epizodama. Ime je nastalo spajanjem engleskih riječi *broadcast* (prijenos, emitiranje) i *POD*, što je skraćenica od *Playable on Demand* (reproducirano na zahtjev). *Podcast* se najčešće objavljuje na jednom mjestu na serveru autora u obliku web-*feeda* i zatim ga korisnici mogu povući i reproducirati na svom web-pregledniku ili direktno na uređaju pomoću softvera koji se nazivaju *podcatchers* (hvatači *podcasta*). *Podcatcher* pronalazi *podcast* na serveru i skida u željenome formatu na mobilni uređaj ili osobno računalo, odnosno prilagođava ga za reprodukciju u web-pregledniku. Još jedan način preslušavanja/pregledavanja *podcasta* jest strujanjem medija (eng. *media streaming*).[23]

3.9. Strujanje medija

Strujanje medija jest tehnologija koja omogućava prijem i istovremeno reproduciranje audiopodataka i videopodataka putem računalne mreže. Princip je sličan klasičnom prijenosu radijskog i televizijskog programa. Uređaj ili softver koji preuzima podatke s nekog mjesta na mreži može krenuti s reproduciranjem zapisa i prije nego je cijeli zahtjev dohvaćen. Streaming također omogućava reprodukciju zapisa bez skidanja

[21] WordPress: http://wordpress.org/about/
[22] Wikipedia - WordPress: http://en.wikipedia.org/wiki/WordPress (2014.)
[23] Wikipedia – Podcast: http://en.wikipedia.org/wiki/Podcast (2014.)

samog zapisa na tvrdi disk ili neku drugu lokaciju na mreži. Zapis se reproducira, ali se ne pohranjuje nigdje. Streaming zapisa može biti uživo (primjerice prijenos neke nogometne utakmice ili drugog događaja uživo) ili može biti asinkron ukoliko se radi o ranije napravljenim zapisima.[24]

[24] Wikipedia – Streaming Media: http://en.wikipedia.org/wiki/Media_streaming (2014.)

4. Planiranje rada na mobilnoj aplikaciji

4.1. Lazy User Model

Planiranje svakog projekta kreće od nekog modela, odnosno načina donošenja odluke o tome koji je najbolji pristup, odnosno plan rada. U području informacijskih tehnologija jedan od najpopularnijih je *Lazy User Model* (model lijenog korisnika).

Lazy User Model jest model planiranja informacijskih sustava koji su predstavili Mikael Collan i Franck Tetard. Oni su tim modelom pokušali objasniti kako jedna osoba odabire najbolje rješenje iz skupa mogućih rješenja. LUM se temelji na teoriji da se rješenje odabire ovisno o količini napora koje osoba (u ovom slučaju programer) mora uložiti – programer odabire rješenje koje uključuje ispunjenje svih zahtjeva, ali s najmanjom količinom napora. Model se može primijeniti i u drugim tipovima situacija, odnosno i u drugim područjima osim informatičkih tehnologija.

Model se zasniva na istraživanjima iz područja psihologije preslikanim u aspekt informacijskih tehnologija, a kreće od osnovice da postoji potreba za izradom nečega, odnosno da postoji jasno definirana volja za radom koja na kraju treba biti zadovoljena. Programer pred sobom ima problem koji mora riješiti, a rezultat rješavanja problema bit će zadovoljstvo (koje nije nužno osobno, nego može biti manifestirano u obliku izrađenog produkta ili u monetarnom obliku). Potreba za rješavanjem problema definira skup mogućih rješenja koja omogućavaju to zadovoljstvo (u ovom slučaju izrada aplikacije koja zadovoljava sve zahtjeve korisnika). Osnovni model zbog jednostavnosti pretpostavlja da postoje potrebe koje se mogu u potpunosti zadovoljiti i rješenja koja u potpunosti zadovoljavaju potrebe. To znači da su važna samo rješenja koja mogu riješiti problem, a najčešće je iskorišteno ono koje zahtjeva najmanje napora. Napor se gleda

kao kombinacija monetarnih izdataka, utrošenog vremena i fizičkih/mentalnih napora koji su potrebni.[25][26]

4.2. Cilj i problemsko područje

Prvi korak u izradi aplikacije je planiranje. Potrebno je definirati zahtjeve klijenta, odnosno potrebe korisnika, predstaviti okvirni funkcionalni i grafički dizajn i predvidjeti vremenski period rada na aplikaciji sukladno svojem iskustvu u programiranju kao i iskustvu u radu s tehnologijama korištenima u izradi i radu aplikacije.

Nakon dužeg razmišljanja i analiziranja različitih ideja za ovaj je rad izabrana mobilna aplikacija za prikaz i komentiranje web-sadržaja. Točnije, odlučeno je da će se izraditi aplikacija za prikaz vijesti s jednog web-portala s filmskim vijestima, recenzijama i drugim sadržajima vezanim uz kritiku filma. Navedenu web-stranicu vodi nezavisna grupa filmskih kritičara i zbog velike popularnosti javila se potreba za kvalitetnijim prikazom sadržaja na mobilnim uređajima. Web-stranica je u pregledniku na mobilnom uređaju bila presitna i bilo je potrebno previše navigiranja po stranici kako bi se prošao sav sadržaj, kao i često povećavanje i smanjivanje prikaza na zaslonu kako bi se mogao čitati tekst ili gledati slike. Problemi su se iz sličnih razloga javljali i kod gledanja videa na stranici, kao i slušanja zvučnih kanala (*podcasta*).

Ideja je bila da redoviti posjetitelji stranice radije i češće pregledavaju sadržaje ukoliko su im dostupni i kvalitetno prikazani na mobilnom uređaju, a ukoliko se aplikacija pokaže prihvatljivom mogla bi privući i nove posjetitelje.

[25] Hayes, James D. Lazy User Theory and Interpersonal Communication Networks, prosinac 2009., https://etd.ohiolink.edu/!etd.send_file?accession=csu1336150484&disposition=inline (2014.), str. 12-16

[26] Wikipedia – Lazy User Model: http://en.wikipedia.org/wiki/Lazy_User_Model (2014.)

4.3. Specifikacija zahtjeva

Nekoliko razgovora s klijentima bilo je potrebno da bi se pripremila specifikacija zahtjeva. Aplikacija je kao osnovno korisnicima (odnosno posjetiteljima portala) morala ponuditi jednostavniji prikaz na zaslonu mobilnog uređaja od onog koji nudi prikaz u web-pregledniku. Najbolji pristup prema ostvarivanju tog cilja bilo je pojednostaviti navigaciju i pregled informacija. Dogovoreno je kako u tu svrhu neke kategorije članaka koje su ponuđene na web-stranici neće biti dostupne u aplikaciji. Te kategorije bile su ili manje popularne ili bi njihov prikaz i u aplikaciji bio prekompliciran i nerazumljiv, što bi u konačnici pregazilo glavni cilj aplikacije. Prvi susret s aplikacijom trebao je ponuditi logo udruge koja drži web-stranicu, ali na samo nekoliko sekundi. Odlučeno je da će biti isključena potreba za prijavom korisnika u aplikaciju pomoću korisničkog imena i lozinke jer za tim u većem dijelu aplikacije nema potrebe. Jedina situacija koja bi zahtijevala prijavu korisnika jest kod postavljanja komentara na članke, ali budući da se na web-stranici koristi model komentiranja gdje bilo tko može bez prijave komentirati sadržaj na način da upiše bilo koji *alias* pod kojim želi komentirati i komentar te jednostavno klikne gumb za postavljanje komentara, odlučeno je da će se isti model koristiti u aplikaciji kako bi funkcionalnost komentiranja bila dostupna bilo kome tko skine aplikaciju na svoj mobilni uređaj.

Nakon prve slike koja služi kao uvod u aplikaciju (logo web-stranice) pojavljuje se prvi zaslon koji odmah nudi posljednjih 20 objavljenih članaka iz svih kategorija. Tu do izražaja dolazi jednostavnost pregleda i lakoća navigacije. Svaka stavka na prikazanoj listi prikazuje naslov članka i vrijeme objave. Pritiskom na bilo koju stavku otvara se članak u potpunosti (s cijelim tekstom i svim slikama) prikazan tako da nema potrebe za povećavanjem ili smanjivanjem prikaza na zaslonu kako bi se povećao tekst ili slike ili prikazao sav sadržaj članka. Na dnu prikazanog članka nalazi se gumb za navigiranje na objavljene komentare za taj članak. Odlučeno je da će komentari biti prikazani na zasebnom zaslonu od članka zbog boljeg pregleda odnosno kako korisnik ne bi morao puno povlačiti zaslon da bi pročitao članak i sve komentare. Na dnu zaslona s komentarima nalazi se gumb za postavljanje komentara. Pritiskom na gumb otvara se

manji prozor u koji se upisuje alias i komentar, te gumbi za postavljanje komentara, kao i za odustajanje i vraćanje na zaslon s objavljenim komentarima.

U suštini to je temelj aplikacije i njene osnovne funkcionalnosti. Zahtijevana je još i opcija filtriranja prikazanih članaka po kategorijama. Odabirom jedne kategorije bilo bi prikazano najnovijih 20 članaka, ali samo iz te kategorije. Specijalna kategorija je i *podcast*. Dok sve ostale kategorije članaka sadrže samo slike i tekst, ova je kategorija trebala sadržavati samo *audioplayer* s pojedinom epizodom *podcasta*. U tekstu članaka na web-stranici postoje pojmovi koji su postavljeni kao poveznice na druge web-stranice. Ta je funkcionalnost morala biti implementirana i u mobilnu aplikaciju.

4.4. Priprema razvojnog okruženja

Nakon što su definirani zahtjevi i dogovoren okvirni funkcionalni i grafički dizajn, potrebno je osigurati kvalitetno razvojno okruženje. Kao softver za pisanje programskog kôda odabran je Eclipse, a već je ranije dogovoreno da će ciljana platforma biti Android. Uz instalaciju Eclipsea na računalo na kojem će se razvijati aplikacija, bilo je potrebno i instalirati Android SDK alate te ih implementirati u Eclipse, te ih ažurirati kako bi se radilo s najsvježijim funkcionalnostima koje Android nudi, a koje se sve brže mijenjanju, odnosno razvijaju. Potrebno je osigurati i testno okruženje. U ovom je slučaju korišten emulator za Android i pretežito mobilni uređaj Samsung Galaxy S3 mini s verzijom Androida 4.1.2. Za testiranje na mobilnom uređaju bilo je potrebno osigurati i kabel za spajanje s računalom preko USB ulaza. Na mobilnom je uređaju trebalo omogućiti testiranje uspostavljanjem internetske veze i ispravnim postavljanjem postavki za programere na samom uređaju.

5. Implementacija

5.1. Dijelovi aplikacije

Aplikacija se sastoji od četiri osnovna dijela:

- Servis za povezivanje i dohvaćanje podataka
- XML Parser
- Adapter
- Fragment

Prikaz sadržaja

Servis služi za povezivanje s web-stranicom i dohvaćanje podataka. To je pozadinski servis čiji se *activity* ne otvara u prozoru već odrađuje svoju funkcionalnost bez korisnikovog znanja, ali zbog reagiranja na korisnikove odluke u aplikaciji (poput odabira filtriranja članaka po kategoriji) postaje intermitentni *activity*. U kodu je definirana web-stranica na kojoj se nalazi *feed* s vijestima s web-portala. Nakon uspješnog povezivanja s web-stranicom povlači se sadržaj izvornog koda cijele stranice koji je u XML formatu, a taj se sadržaj provlači kroz XML parser kako bi bio spreman za korištenje u aplikaciji. Rezultat je lista RSS stavki. Nakon što cijeli proces završi servis šalje generiranu listu i dopuštenje sljedećem *activityju* da su dohvaćeni svi podaci i da ih može pripremiti za prikaz na zaslonu. Taj dio obavlja RSS Fragment i adapter, čija je funkcionalnost objašnjena kasnije u radu. Da bi servis funkcionirao potrebna je naravno aktivna veza prema internetu. U manifestu aplikacije potrebno je stoga dopustiti aplikaciji spajanje na internet.

Prije nego podaci koje servis dohvatio mogu biti korišteni u aplikaciji, potrebno ih je „prevesti“, odnosno parsirati. Tomu služi XML parser, odnosno XML procesor. Budući da podaci s web-stranice kroz servis dolaze u XML formatu potrebno ih je interpretirati i sistematizirati kako bi ih aplikacija lakše implementirala. XML parser prolazi kroz cijeli

XML dokument i traži imena oznaka koje su definirane u kodu. Svaku od RSS stavki koju prepozna pomoću oznake imena *item* parser sprema u jednu cjelinu. Svaka od tih cjelina zatim se sprema u listu. Međutim, prije nego bude pohranjena u listi svaka je cjelina parsirana tako da se sav sadržaj između oznaka pohranjuje u varijable u tekstualnom obliku tipa string. Set tih varijabli koji dolaze iz XML parsera zapravo čini jednu stavku. Uzmimo sljedeći dio XML dokumenta kao primjer jedne stavke RSS *feeda*:

```
<item>

<title>Prvi pogled: Novi film Caryja Fukunage s Idrisom Elbom u glavnoj
ulozi</title>

<link>http://www.fak.hr/vijesti/prvi-pogled-novi-film-caryja-fukunage-s-
idrisom-elbom-u-glavnoj-ulozi/</link>

<comments>http://www.fak.hr/vijesti/prvi-pogled-novi-film-caryja-fukunage-s-
idrisom-elbom-u-glavnoj-ulozi/#comments</comments>

<pubDate>Sun, 29 Jun 2014 22:05:34 +0000</pubDate>

<dc:creator><![CDATA[Sven Mikulec]]></dc:creator>

<category><![CDATA[Vijesti]]></category>

<category><![CDATA[Cary Fukunaga]]></category>

<guid isPermaLink="false">http://www.fak.hr/?p=48043</guid>

<description><![CDATA[Što nam spremaju redatelj 'True Detectivea' i zvijezda
'Žice' i 'Luthera'?]]></description>

<content:encoded><![CDATA[<div align="center"><a href="http://www.fak.hr/wp-
content/uploads/2014/06/179.jpg"><img class="aligncenter size-full wp-image-
48044" src="http://www.fak.hr/wp-content/uploads/2014/06/179.jpg" alt="1"
width="500" height="334" /></a></div>

<p>Kadsnimišneštošto valja, oči svijeta pratit će tvoj sljedećikorak I
eventualnikikskspremnodočekatinanož. Rijetko tko mora se nositi s takvim
pritiskom kao što se nosi Cary Fukunaga, koji se svijetu predstavio jednom od
najboljih serija u posljednje vrijeme, <i>True Detective</i>. Filmašev
sljedeći projekt bit će film <i>Beasts Of No Nation</i>, koji priča priču o
dječaku prisiljenome postati vojnikom. Mladome Aguu pušku u ruke gurnut će
Idris Elba, možda i na vrijeme da prasak oružja dovoljno glasno odjekne i na
sljedećim Oscarima.</p>
<p>Izašla je prva fotka filma koji, uz imena Fukunage i Elbe, ni ne treba
kvalitetne fotke, teasere, trailere i slično da se proda. Drž&#8217;mo fige
da ekipa stigne <i>BONN</i> završiti što prije. S ovim dvojcem u glavnim
ulogama, teško da će promašiti metu.</p>

<div align="center"><a href="http://www.fak.hr/wp-
content/uploads/2014/06/1ryan33.jpg"><img src="http://www.fak.hr/wp-
content/uploads/2014/06/1ryan33.jpg" alt="1ryan" width="640" height="450"
class="aligncenter size-full wp-image-48045" /></a></div>
]]></content:encoded>
```

```
<wfw:commentRss>http://www.fak.hr/vijesti/prvi-pogled-novi-film-caryja-
fukunage-s-idrisom-elbom-u-glavnoj-ulozi/feed/</wfw:commentRss>

<slash:comments>0</slash:comments>

</item>
```

Recimo da je XML parser podešen tako da prepoznaje oznake imena *title, link, pubDate, wfw:commentRSS, content:encoded* i *category*. Parser prolazi kroz dokument i dijeli ga na stavke, te zatim prepoznaje svaku od ovih oznaka i povlači sadržaj koji se nalazi unutar njih. Ti se dijelovi sadržaja spremaju u varijable imena **naslov** (za sadržaj između oznaka imena title), **link** (za sadržaj između oznaka imena link), **datumObjave** (za sadržaj između oznaka imena pubDate), **linkKomentari** (za sadržaj između oznaka imena wfw:commentRSS), **sadržaj** (za sadržaj između oznaka imena content:encoded) i **kategorija** (za sadržaj između oznaka imena category). Rezultat parsiranja jest lista stavki od kojih stavka iz ovog primjera konkretno izgleda kao niz varijabli sa sljedećim tekstualnim vrijednostima:

- naslov = Prvi pogled: Novi film Caryja Fukunage s Idrisom Elbom u glavnoj ulozi
- link = http://www.fak.hr/vijesti/prvi-pogled-novi-film-caryja-fukunage-s-idrisom-elbom-u-glavnoj-ulozi/
- datumObjave = Sun, 29 Jun 2014 22:05:34 +0000
- linkKomentari = http://www.fak.hr/vijesti/prvi-pogled-novi-film-caryja-fukunage-s-idrisom-elbom-u-glavnoj-ulozi/feed/
- sadržaj = <![CDATA[Kad snimiš nešto što valja, oči svijeta pratit će tvoj sljedeći korak i eventualni kiks spremno dočekati na nož. Rijetko tko mora se nositi s takvim pritiskom kao što se nosi Cary Fukunaga, koji se svijetu predstavio jednom od najboljih serija u posljednje vrijeme, True Detective. Filmašev sljedeći projekt bit će film Beasts Of No Nation, koji priča priču o dječaku prisiljenome postati vojnikom. Mladome Aguu pušku u ruke gurnut će Idris Elba, možda i na vrijeme da prasak oružja dovoljno glasno odjekne i na sljedećim Oscarima. Izašla je prva fotka filma koji, uz imena Fukunage i

Elbe, ni ne treba kvalitetne fotke, teasere, trailere i slično da se proda. Drž’mo fige da ekipa stigne BONN završiti što prije. S ovim dvojcem u glavnim ulogama, teško da će promašiti metu.</p>]]>

- kategorija = Vijesti [27]

Primljeni sadržaj kasnije je potrebno dodatno urediti kroz pravila, odnosno automatske modifikacije u adapteru. Tekstualni podaci tipa string lako se uređuju i po potrebi pretvaraju u većinu drugih tipova koje aplikacija može koristiti.

Sljedeći dio koda prikazuje objašnjeni proces gdje je XML parseru prvo inicijalizirana Lista RSS stavki, zatim su definirane varijable tipa string i imena oznaka u XML dokumentu, te mapiranje definiranih varijabli na sadržaj unutar tih oznaka kako bi parser znao koji dio sadržaja treba pohraniti u koju varijablu. Na kraju operacijom *add* parser svaku od tih varijabli pohranjuje u stavku.

[27] Iako se unutar stavke nalazi više parova oznaka s imenom *category*, parser izvlači samo vrijednost prvog takvog pojavljivanja i nakon toga prestaje tražiti ostale parove oznaka s istim imenom. Za potrebe aplikacije opisane u ovome radu to je dovoljno. Ukoliko bi bilo potrebno izvući sadržaj iz svih parova oznaka istog imena unutar jedne stavke, bilo bi potrebno proći parserom kroz stavku više puta i podesiti njegovu funkcionalnost tako da pri svakom prolasku dohvaća vrijednost sljedećeg para oznaka istog imena.

```
private List<RssItem>readFeed ( XmlPullParser parser)throws
XmlPullParserException, IOException
{
    parser.require(XmlPullParser.START_TAG,null,"rss");
    String title =null;
    String link =null;
    String published =null;
    String comments =null;
    String content =null;
    String category =null;
    List<RssItem> items =new ArrayList<RssItem>();
      while(parser.next()!= XmlPullParser.END_DOCUMENT)
      {
            if(parser.getEventType()!= XmlPullParser.START_TAG){
                  continue;}
      String name = parser.getName();
            if(name.equals("title")){
            title = readTitle(parser);}
            elseif(name.equals("link")){
      link = readLink(parser);}
            elseif(name.equals("pubDate")){
      published = readPubDate(parser);}
            elseif(name.equals("wfw:commentRss")){
      comments = readCommentsUrl(parser);}
            elseif(name.equals("content:encoded")){
      content = readContent(parser);}
            elseif(name.equals("category")){
      category = readCategory(parser);}

            if(title !=null&& link !=null&& published !=null&&content
      !=null){
                  RssItem item =new RssItem(title, link, published,
                  comments,content,category);
      items.add(item);
            title =null;
      link =null;
      published =null;
      comments =null;
      content =null;
      category =null;
            }
}
      return items;
}
```

Nakon parsiranja podatke preuzima adapter. Podaci, kao što je ranije objašnjeno, u adapter ulaze u obliku stavki, u kojima je svaki dio teksta posebno naznačen (naslov, sadržaj, datum...). Posao je adaptera da te stavke posloži u privremeni spremnik iz kojeg fragment kasnije povlači strukturirane stavke i prikazuje ih u obliku liste pomoću ListView kontrole. ListView kontrola funkcionira na način da se

definira struktura, odnosno izgled jedne stavke, te je isti onda korišten za cijelu listu. Adapter prvo postavlja parametar za veličinu liste, odnosno broj stavki u listi. U slučaju ove aplikacije, prvi zaslon prikazuje 20 najsvježijih objavljenih članaka, što znači da će lista imati 20 stavki. Nakon toga dohvaća jednu po jednu stavku i pohranjuje ih u spremnik.

Kako bi tekst u listi bio čitljiv i u formatu prihvatljivom za korisnike treba biti očišćen, odnosno formatiran. Prvo takvo formatiranje odvija se na datumu objave. Datum dolazi u formatu na engleskom jeziku koji izgleda primjerice ovako:

Mon, 14 Jul 2014 12:45:39 +0000 ,

Dok je željeni format u aplikaciji ovakav:

Pon, 14.07.2014. 12:45:39

Kako bi se to ostvarilo adapter za početak iz stavke očitava vrijednost varijable koja sadrži datum objave. Taj podatak zatim rastavlja na dijelove (eng. *substrings*), prema definiranim rasponima indeksa. Svaki od dijelova sprema se u zasebnu varijablu, od kojih svaka sadržava dio datuma (dan, mjesec, godina...):

```
String danUMjesecu = published.substring(5,7);
String godina = published.substring(12,16);
String sati = published.substring(17,25);
```

Prvi dio datuma, ime dana i sl. mora se prevesti na hrvatski jezik i to tako što adapter čita prva tri znaka u engleskom formatu, odnosno niz znakova na rasponu indeksa 0 do 3. Ovisno o očitanoj vrijednosti u novu varijablu zapisuje vrijednost na hrvatskom jeziku. Ne koristi se nikakav automatski prevoditelj, već je sve zapisano u kodu i ne mijenja se. Primjerice, ukoliko se radi o ponedjeljku, niz od prva tri znaka u originalnom je formatu „Mon“. Adapter to prepoznaje i u varijablu pohranjuje vrijednost koja mu se slaže s tim nizom, a to je „Pon“:

```
if(published.substring(0,3).equals("Mon")){
            dan ="Pon";
}
```

Isti je proces i s prepoznavanjem i formatiranjem mjeseca, samo što se u tom slučaju radi o drugom rasponu indeksa i rezultat je broj mjeseca u godini:

```
if(published.substring(8,11).equals("Jan")){
            mjesec ="01";
}
```

Niz znakova s kraja originalnog formata („+0000“) ne koristi se pa se jednostavno ne sprema ni u jednu varijablu. Kada su sve vrijednosti očitane, po potrebi formatirane i pohranjene u svoje varijable, adapter ih jednostavno spaja u novi niz znakova koji odgovaraju željenom formatu datuma. Taj niz znakova pohranjuje se u zasebnu varijablu koja je korištena kod strukturiranja stavke, odnosno prikaza podataka u istoj:

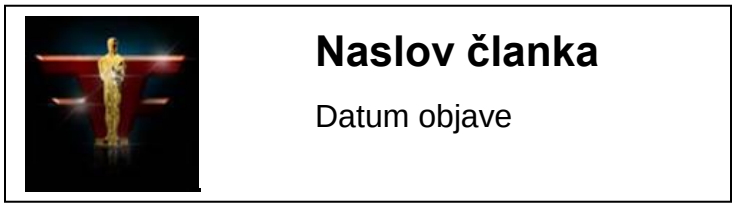
```
String publishedFinal = dan+", "+danUMjesecu+"."+mjesec+"."+godina+".
"+sati;
```

Nakon toga adapter definira u koju kategoriju članaka ulazi trenutna stavka. To radi na način da prepoznaje dio web-adrese koji je uvijek identičan za pojedinu kategoriju. Ukoliko web-adresa sadrži niz znakova „http://www.fak.hr/fak-top-13“ adapter prepoznaje da se radi o kategoriji „FAK Top 13“ i tu vrijednost sprema u novu varijablu.

Ostali dijelovi stavke, osim sadržaja, ulaze u adapter u čitljivom formatu te se jednostavno pohranjuju u nove varijable koje će adapter koristiti. Sadržaj članka ovdje se ne prikazuje pa ga nije potrebno procesirati.

Podatke, koji su sada formatirani i pohranjeni u spremnik adaptera, preuzima fragment. Fragment se može opisati kao glavni funkcionalni dio koji služi za pokretanje osnovnih naredbi aplikacije i prikaz podataka na zaslonu. Kada se aplikacija pokrene, prvi element koji se pojavi na zaslonu jest spiralna traka koja označava proces učitavanja podataka u listu i pokretanje procesa za prikaz liste na zaslonu (eng. *progress bar*). Kada je lista popunjena i učitana, taj element nestaje sa zaslona i prikazuje se lista s naslovima 20 najnovijih članaka.

U procesu učitavanja liste fragment povlači stavke pohranjene u spremnik adaptera i slaže ih u listu. Struktura jedne stavke je definirana na sljedeći način:



Slika 5 - Vizualna struktura stavke u listi

Fragment iz spremnika adaptera vadi stavku po stavku i očitava vrijednosti svake varijable u stavki te ih ispisuje na stavku liste. Slika u lijevom kutu je ikona koja definira kojoj kategoriji pripada članak koji se nalazi na toj stavki. Slika se mijenja ovisno o vrijednosti varijable koja sadržava podatak o kategoriji, a definirana je u adapteru. Sve slike koje se koriste na listi su spremljene u aplikaciji kako bi se smanjilo vrijeme učitavanja liste. Prvotna ideja bila je da se na tome mjestu nalazi slika povezana s člankom, a koja bi se povlačila sa stranice kao i ostali podaci, međutim vrijeme učitavanja bilo je višestruko veće pa je donesena odluka o implementaciji ovog rješenja.

Iznad liste članaka nalazi se izbornik, odnosno funkcionalna traka (eng. *action bar*). Na traci se nalazi logo web-stranice, ime aplikacije i padajući izbornik za filtriranje sadržaja. Sadržaj se može filtrirati po kategorijama, na način da se odabirom jedne od stavki iz padajućeg izbornika fragment ponovno učitava s najsvježijim člancima iz odabrane kategorije. Za ovu se funkcionalnost koristi kontrola Spinner koja ima postavljen svoj adapter za prikaz liste (eng. *ArrayAdapter*) i koja prikazuje listu tekstualnih zapisa koji su definirani u aplikaciji i ne generiraju se svaki puta. Unutar okidača koji se aktivira prilikom pritiska na jednu od stavki (eng. *on click listener*) definirana je *switch* petlja koja postavlja parametre za novi fragment koji se treba učitati. U ovom slučaju mijenja se URL adresa na koju se spaja servis i povlači podatke, odnosno RSS *feed*. To je adresa web-stranice gdje se nalazi već filtrirani RSS *feed* koji prikazuje članke iz određene kategorije. Filtrirani se sadržaj, odnosno kategorija čiji članci trebaju biti prikazani, definira na vrlo jednostavan način. Originalna URL adresa koju servis koristi, a koja sadrži članke iz svih kategorija u ovom slučaju je http://www.fak.hr/feed/. Kako bi dobili članke iz željene kategorije, na kraju te adrese

potrebno je dodati niz „?cat=“ i identifikacijski broj kategorije koji je definiran u Wordpressu, odnosno na serveru. Tako će primjerice za kategoriju „Vijesti“, čiji je identifikacijski broj 3, URL adresa RSS *feeda* biti http://www.fak.hr/feed/?cat=3. Sljedeći dio kôda prikazuje kako radi *switch* petlja:

```
@Override
public void onItemSelected(AdapterView<?>parentView, View
selectedItemView,int position,long id){

switch(position){
case0:
      String urlAll ="http://www.fak.hr/feed/";
      FragmentManagermanagerAll=getSupportFragmentManager();
       FragmentTransaction transactionAll =
      managerAll.beginTransaction();
      RssFragmentfragmentAll=newRssFragment();
      Bundle bundleFragAll=newBundle();
      bundleFragAll.putString("url", urlAll);
      fragmentAll.setArguments(bundleFragAll);
      transactionAll.replace(R.id.fragment_container,fragmentAll);
      transactionAll.commit();
      break;
case1:
      String urlVijesti="http://www.fak.hr/feed/?cat=3";
      FragmentManagermanagerVijesti=getSupportFragmentManager();
      FragmentTransactiontransactionVijesti=
      managerVijesti.beginTransaction();
      RssFragmentfragmentVijesti=newRssFragment();
      Bundle bundleFragVijesti=newBundle();
      bundleFragVijesti.putString("url",urlVijesti);
      fragmentVijesti.setArguments(bundleFragVijesti);
      transactionVijesti.replace(R.id.fragment_container,fragmentVijesti);
      transactionVijesti.commit();
```

Brojevi iza naredbe *case* predstavljaju pozicije na kojima se nalaze stavke u padajućem izborniku, odnosno indekse tih tekstualnih zapisa u listi (eng. *array*). Lista zapisa nalazi se u XML dokumentu za definiranje *stringova* za aplikaciju (*strings.xml*). Kôd unutar petlje isti je kao i kod kreiranja glavnog fragmenta pri otvaranju aplikacije, uz jednu razliku, a to je da se umjesto naredbe za dodavanje fragmenta (*transaction.add ()*), koristi naredba za zamjenu fragmenta (*transaction.replace()*), jer je jedan fragmentu trenutku odabira kategorije već učitan i potrebno ga je zamijeniti novim.

Zadnja operacija fragmenta definiranje je procesa otvaranja cijelog članka kako bi se mogao pročitati sadržaj. Korištenjem elementa *intent*, fragment definira dijelove stavke. Kada korisnik prstom pritisne na jednu od stavki, definirani dijelovi spremaju se

u varijable, odnosno u element *intent* koji služi za otvaranje novog funkcionalnog dijela (*activity*). Za isti je element definirano i ime *activityja* koji se treba otvoriti nakon pritiska na stavku (u ovom slučaju PostOpen):

```
Intent intent =new Intent(getActivity(), PostOpen.class);
intent.putExtra("title", title);
intent.putExtra("published", publishedFinal);
intent.putExtra("comments", comments);
intent.putExtra("content", contentFinal);
startActivity(intent);}
```

Nakon pritiska na stavku otvara se novi zaslon te se učitava sadržaj članka. Cijeli je proces definiran u *activityju* PostOpen, koji prvo preuzima varijable koje su definirane u fragmentu. Vrijednosti iz varijabli zatim ispisuje na za njih predviđena mjesta. Za sadržaj članka učitava se novi element, WebView. Pomoću WebViewa moguće je direktno prikazati cijeli sadržaj članka onako kako je prikazan na web-stranici, ali u formatu prilagođenom za pregled na mobilnim uređajima. WebView služi kao zaslon unutar zaslona, te mu je moguće podesiti margine i način prikaza teksta i slika. Velika je prednost u tome što nema potrebe za dodatnim procesiranjem i formatiranjem teksta sadržaja, koji je obično velik i sadrži veliki broj html oznaka koje definiraju izgled teksta na web-stranici. WebView čita i interpretira html oznake i prikazuje uređeni tekst. Još jedna prednost jest u tome što WebView prepoznaje i slike u člancima na web-stranici i prikazuje ih u prikladnoj veličini na zaslonu mobilnog uređaja. Uz postavljanje nekoliko parametara, mijenjanje veličine slike automatsko je i događa se pri učitavanju. S obzirom na to da WebView ne učitava posebno tekst i svaku sliku članka, vrijeme učitavanja i prikazivanja na zaslonu vrlo je kratko. Parametri koji se moraju definirati za WebView jesu tekst koji treba učitati te definicija html strukture i jezičnog standarda koji se koristi. Ostali parametri koji se mogu definirati služe većinom za kontroliranje na koji je način rezultat prikazan na zaslonu.

Na dnu zaslona nalazi se gumb za otvaranje novog *activityja* koji prikazuje komentare na članak koji je trenutno otvoren. Jedan od podataka koje XML parser očitava iz RSS *feeda* jest i link na web-stranicu koja sadrži RSS *feed* s komentarima na pojedini članak. Prikaz komentara prolazi kroz isti proces kao i prikaz liste članka na

početnom zaslonu aplikacije. Svaki od *activityja* korišten u tom procesu ima zapravo svog dvojnika koji se koristi za prikaz komentara. Servis dohvaća podatke s web-stranice, XML parser ih prevodi i potrebne vrijednosti pohranjuje, a adapter i fragment ih formatiraju, odnosno prikazuju u obliku liste. Ono što ovaj zaslon jedino ne nudi jest pritisak na jednu od stavki, odnosno jedan od komentara.

Članak s *podcastom*

Pri otvaranju zaslona sa sadržajem postoji jedna iznimka. Ukoliko se radi o kategoriji „Repulzije“, sadržaj članka nije isti kao u ostalim slučajevima jer se radi o *podcastu*. U ovome slučaju sadržaj članka ne sastoji se od teksta i slika, već od jedne slike i audiozapisa. Zbog toga postoji i zasebni *activity* koji je nešto drugačiji od *activityja* PostOpen. Umjesto WebView kontrole, ovdje se koristi kontrola MediaPlayer za pokretanje audio zapisa. MediaPlayer je ovdje namješten tako da audio zapis ne preuzima s interneta na uređaj (što bi oduzelo previše vremena), već ga pokreće strujanjem medija. Vrijeme pokretanja gotovo je neuočljivo jer reprodukcija zapisa započinje i prije nego je cijeli zapis očitan od strane kontrole.

Kada se *activity* s *podcastom* otvori, odmah se pokreće servis koji se spaja na oblak s podacima i dohvaća audiozapis koji treba reproducirati. Točna lokacija, odnosno web-adresa s lokacijom audiozapisa parsiranjem se dobiva iz istog RSS *feeda* kao i ostali podaci u aplikaciji. Pomoću *web viewa* na vrhu zaslona prikazuje se slika povezana s epizodom *podcasta* koja se nalazi u otvorenom članku. Ispod slike nalaze se kontrole za manipulaciju zapisom. Kako bi se održala jednostavnost korištenja aplikacije ponuđene su samo kontrole za pokretanje zapisa (eng. *play*), pauziranje zapisa u određenom trenutku (eng. *pause*), potpuno zaustavljanje zapisa (eng. *stop*) i traka za kretanje po zapisu kako bi korisnici brzo i jednostavno mogli birati na koji se dio zapisa žele pomaknuti ukoliko neki dio žele ponovno slušati ili preskočiti.

Budući da se zbog lakšeg i slobodnijeg uređivanja dizajna ne koristi gotova kontrola koja sadrži cijelu funkcionalnost kontrole MediaPlayer, već jednostavni gumbi,

potrebno je svaku od kontrola povezati s jednim od gumba. Petljom je definirano koja se metoda treba aktivirati pritiskom na određeni gumb:

```
publicvoid onClick(View view){
	switch(view.getId()){
		case R.id.ibPlay:
			controls.play();
		break;

		case R.id.ibPause:
			controls.pause();
		break;

		case R.id.ibStop:
			controls.stop();
		break;
	}
}
```

Metode su definirane u servisu za *podcast*. Sljedeći primjer koda prikazuje definicije metoda. Za metode koje pokreću i pauziraju reprodukciju zapisa definirana je samo ta funkcionalnost, dok je za metodu za zaustavljanje audiozapisa dodana i funkcionalnost ponovnog učitavanja zapisa kako bi se mogao reproducirati ispočetka:

```
@Override
publicvoid stop(){
	try{
		mediaPlayer.stop();
		mediaPlayer.reset();
		mediaPlayer.setDataSource(getApplicationContext(),
		Uri.parse(currentUrl.toString()));
		mediaPlayer.prepareAsync();
	}
	catch(IOException e){
		Log.d("service","loadContent", e);
	}
}

@Override
publicvoid play(){
mediaPlayer.start();
}

@Override
publicvoid pause(){
mediaPlayer.pause();
}
```

Budući da je učitavanje i reproduciranje audiozapisa u ovome slučaju asinkrono, moguće je i zatvoriti *activity* s *podcastom* i otvarati ostale članke dok se audiozapis i dalje reproducira. Na dnu zaslona se i u ovome slučaju nalazi gumb za otvaranje komentara na članak, odnosno epizodu *podcasta*.

Objavljivanje komentara

Uz servis za dohvaćanje podataka s web-stranice, najzahtjevniji dio pri radu na aplikaciji bilo je razvijanje funkcionalnosti za objavljivanje komentara na pojedini članak. Kada korisnik pročita članak i postojeće komentare mora mu biti omogućeno ostavljanje vlastitog odgovora i razmišljanja o članku i ostalim objavljenim komentarima. Temeljitim istraživanjem pronađeno je nekoliko knjižnica koje omogućavaju slanje podataka iz mobilne aplikacije na Android sustavu prema WordPress alatu, odnosno serveru na kojem se nalazi web-stranica. Za potrebe ove aplikacije odabrana je knjižnica aXMLRPC[28]. Ova knjižnica sadrži nekoliko klasa s unaprijed definiranim metodama koje se pozivaju u aplikaciji. Osnovna klasa je XMLRPCClient koja sadrži pozive prema ostalim klasama. Jedina klasa koja se poziva u ovoj aplikaciji jest klasa Call. Unutar te klase definirana je metoda call() koja prima dva ulazna parametra – ime metode WordPressa koja se poziva i objekt koji sadrži potrebne parametre za tu metodu. Metode Wordpressa su unaprijed definirane od strane WordPressa pa je u ovoj metodi potrebno definirati samo koja se metoda poziva i dati joj pravilne parametre. WordPressova metoda koju koristimo jest wp_new_comment() za dodavanje novog komentara, a ona prima sljedeće parametre:

- id bloga (vrsta podatka - *integer*) – id bloga je jedinstveni broj , odnosno šifra koja označava o kojem se blogu u WordPressu radi u slučaju da korisnik uređuje više stranica preko WordPressa. U slučaju ove aplikacije uređuje se samo jedna web-stranica, odnosno blog te je ova vrijednost uvijek 1

[28] Copyright (c) 2011-2012 Tim Roes

- korisničko ime (string) – korisničko ime korisnika web-stranice koji želi objaviti komentar
- lozinka (string) – lozinka korisnika web-stranice koji želi objaviti komentar
- id članka (integer) – označava šifru članka na koji se komentar objavljuje. Može se lako očitati iz *shortlink* adrese članka. Primjerice u adresi članka koja izgleda ovako: http://www.fak.hr/?p=48381, id članka predstavlja niz brojeva na kraju adrese (48381)
- struktura komentara (map) – mapa koja sadrži podatke o komentaru:
    - id roditelja komentara (int) – koristi se u slučaju odgovaranja na ranije objavljeni komentar. Budući da web-stranica www.fak.hr ne dopušta „hijerarhijsko“ odgovaranje na komentare, ova je vrijednost uvijek 0
    - sadržaj komentara (string) – tekst koji se želi objaviti
    - autor komentara (string) – ime ili nadimak autora komentara
    - adresa osobne web-stranice autora komentara (string) – može biti prazan parametar
    - adresa elektroničke pošte autora komentara (string) - može biti prazan parametar.

Mapa je tip podatka koji sadrži nizove parova spojenih po principu definicija – vrijednost. U ovom slučaju definicije su imena parametara definirana u WordPressu, a vrijednosti su vrijednosti koje aplikacija šalje svakom od tih parametara.

Prvo što aplikacija radi prije slanja zahtjeva jest dohvaćanje svih potrebnih podataka koje treba slati. Fiksne vrijednosti su id bloga (uvijek 1), korisničko ime, lozinka i id roditelja komentara (uvijek 0). Korisničko ime i lozinka fiksne su vrijednosti jer predstavljaju aplikaciju kao korisnika na web-stranici i korisnici aplikacije uvijek objavljuju komentar preko tih podataka za autentifikaciju. Postoje dva razloga zašto je to tako: web-stranica ima jako mali broj prijavljenih korisnika te većina objavljuje komentare anonimno, odnosno pod nadimkom, a to je u aplikaciji omogućeno postavljanjem nadimka pri slanju komentara na stranicu. Nadimak ostaje spremljen u aplikaciji tako da ga nije potrebno svaki puta unositi, a može se mijenjati po želji. Drugi

je razlog korištenje *captche* na web-stranici. *Captcha* je oblik zaštite web-stranice protiv neželjenih komentara, odnosno spema. Kada korisnik objavljuje komentar preko web-stranice mora upisati potrebnu vrijednost u *captchu* (na web-stranici se koristi jednostavni matematički zadatak gdje je potrebno upisati jedan broj), a komentar je tek nakon toga uspješno objavljen. Aplikacija ne može doći do podatka koja se vrijednost u *captchi* traži, odnosno o kakvim se matematičkom zadatku radi, jer se ona stalno ponovo generira s novim vrijednostima. Iz tog se razloga koristi korisničko ime i lozinka kako bi se web-stranici činilo da komentar dolazi od prijavljenog korisnika, jer u tom slučaju *captcha* nije potrebna i ne blokira objavljivanje komentara. Id članka aplikacija dohvaća ranije u procesu parsiranja RSS *feeda*.

Pri slanju zahtjeva aplikacija prvo inicijalizira varijablu koja predstavlja poziv prema klasi XMLRPCClient te definira vezu prema web-stranici, odnosno dokumentu na serveru xmlrpc.php, koji sadrži funkciju za procesiranje XMLRPC zahtjeva:

```
client =new XMLRPCClient("http://www.fak.hr/xmlrpc.php");
```

Nakon toga vrijednosti se varijabli postavljaju u mapu uparene s imenima parametara. Mapi je definirano da prima tipove podataka *string* za ime parametra i *object* za vrijednost. Tip podatka *object* postavljen je kako bi vrijednost mogla biti string ili integer:

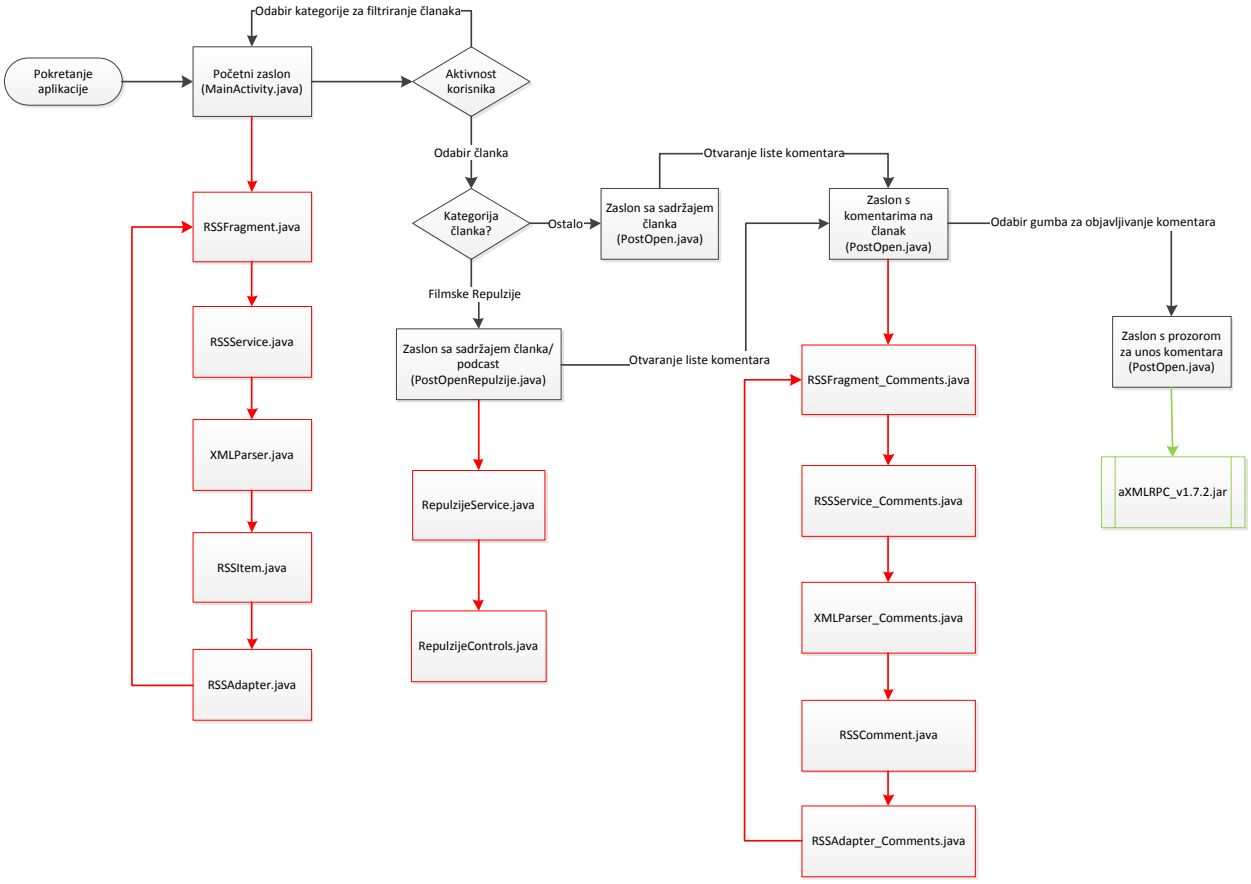
```
Map<String, Object> comment =null;
comment.put("comment_parent",0);
comment.put("content","Sadržaj komentara.");
comment.put("author","Komentator");
comment.put("author_url",null);
comment.put("author_email",null);
```

Sve se varijable nakon toga spremaju u jedan objekt koji se kao niz parametara šalje WordPressovoj funkciji definiranoj u metodi call(). Parametar comment.entrySet() šalje sve parove koji se nalaze u mapi ranije nazvanoj comment:

```
final Object[] params =new Object[]{blog_id, username, password,
post_id, comment.entrySet()};
client.call("wp.newComment", params);
```

5.2. Dijagram rada aplikacije

Sljedeći dijagram prikazuje proces rada aplikacije i klase čije se funkcionalnosti pokreću na pojedinim koracima procesa. Crvenom su bojom označene klase čije se metode pokreću u pozadini, odnosno čiji activityji nemaju grafički prikaz na zaslonu mobilnog uređaja, dok je zelenom bojom označena knjižnica koja se koristi pri objavljivaju komentara na web-stranici.



Slika 6 - Dijagram rada aplikacije

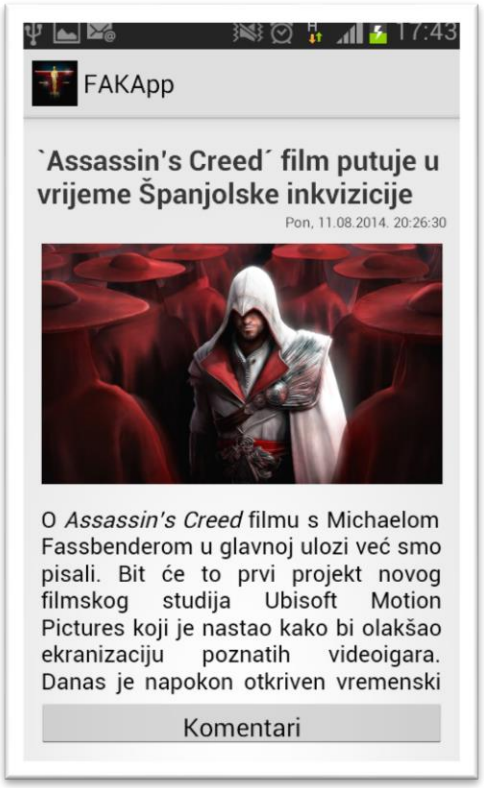
6. Korištenje aplikacije

U ovome je poglavlju objašnjen način korištenja aplikacije kroz slike pojedinih zaslona aplikacije i objašnjenja kako se koriste. Dizajn aplikacije u slikama funkcionalan je i ne predstavlja završni dizajn koji će se koristiti.

Prvi zaslon koji se otvara jest zaslon s listom 20 najnovijih članaka iz svih kategorija. Članci se otvaraju u listi koja se može pomicati povlačenjem prstom po ekranu. U gornjem desnom kutu zaslona, pokraj imena aplikacije nalazi se padajući izbornik za odabir kategorije, odnosno za filtriranje sadržaja po kategoriji. Pritiskom na izbornik otvara se lista stavki za odabir, a pritiskom na jednu od stavki otvara se lista 20 najnovijih članaka iz odabrane kategorije. Kada korisnik pritisne na jednu od stavki u listi članaka otvara se cijeli sadržaj tog članka.

Slika 7 - Početni zaslon (sve kategorije članaka)

Na vrhu zaslona za pregled cijelog članka nalazi se traka s imenom i logom aplikacije. Ispod trake nalazi se naslov i datum objavljivanja članka, a niže i cijeli sadržaj članka s tekstom i slikama. Na dnu zaslona nalazi se gumb za otvaranje komentara na članak. Ukoliko je kategorija članka „Filmske repulzije“, umjesto teksta i slike na zaslonu se nalazi samo glavna slika i gumbi za manipulaciju audiosadržajem, odnosno za preslušavanje epizode *podcasta*.



Slika 8 - Zaslon s punim sadržajem članka

Pritiskom na gumb za komentare otvara se lista komentara na članak. Ukoliko ne postoji još ni jedan komentar, pojavljuje se odgovarajuća poruka na zaslonu. Na dnu ovog zaslona nalazi se i gumb za ostavljanje vlastitog komentara. Kada korisnik pritisne taj gumb otvara se prozor koji se sastoji od dva tekstualna polja u koja korisnik unosi svoj *alias* i sadržaj komentara (pomoću Androidove interne tipkovnice), te gumba za objavljivanje komentara i gumba za odustajanje.

Slika 9 - Zaslon s komentarima na članak

7. Budućnost aplikacije

Nakon što se usavrše još neke funkcionalnosti aplikacije i definira se, te implementira, konačni dizajn, aplikacija će se početi aktivno koristiti. Ciljana grupa su, naravno, posjetitelji web-stranice, a jedan od ciljeva aplikacije je i da privuće nove korisnike. Aplikacija će biti postavljena na Google Play i na samu web-stranicu kako bi je korisnici mogli lako preuzeti na svoje mobilne uređaje.

Novih će zahtjeva od strane voditelja web-stranice sigurno biti, te će biti implementirane neke nove funkcionalnosti. Jedna od takvih je zasigurno dodavanje mogućnosti za pregledavanje video-zapisa, koja trenutno nije implementirana. Plan je također i aplikaciju prilagoditi za korištenje na tabletima, te ostalim većim mobilnim uređajima.

8. Zaključak

Razvijanje mobilne aplikacije za prikaz sadržaja web-stranice pokazao se za jednog programera-početnika relativno lako izvodljivim zadatkom. Slijed izvođenja funkcija u procesu dohvaćanja i prikaza podataka (spajanje na server/web-stranicu, dohvaćanje podataka, parsiranje, prikaz podataka) u aplikaciji logičan je i ne pretjerano kompliciran te sam u tom pogledu aplikacije uspio ispuniti sve zahtjeve koji su bili definirani od strane korisnika, odnosno vlasnika web-stranice za koju je aplikacija razvijana.

Problemi su se javili kod omogućavanje *podcasta* u aplikaciji i omogućavanju objavljivanja komentara na web-stranici preko aplikacije. Najveći problem bilo je neiskustvo u radu s pojedinim kontrolama u aplikaciji poput rada s MediaPlayer kontrolom i slanjem XMLRPC zahtjeva prema serveru.

Aplikacija je jednostavna, relativno brza i laka za korištenje, te je, po mome mišljenju, ispunila glavni zadatak koji je na početku bio postavljen, a to je da pojednostavi način pregleda podataka web-stranice na mobilnim uređajima.

Nadam se da će aplikacija zaživjeti u potpunosti u sljedećih nekoliko mjeseci, te da će je posjetitelji web-stranice znati cijeniti i koristiti, ali još se više nadam da će ovaj rad jednog dana barem djelomično pomoći jednom budućem programeru u izradi slične aplikacije ili barem u implementaciji pojedinih dijelova ove aplikacije u svoj rad. Zadovoljan sam onime što je postignuto, ali bit ću još sretniji ukoliko aplikacija bude korištena redovito.

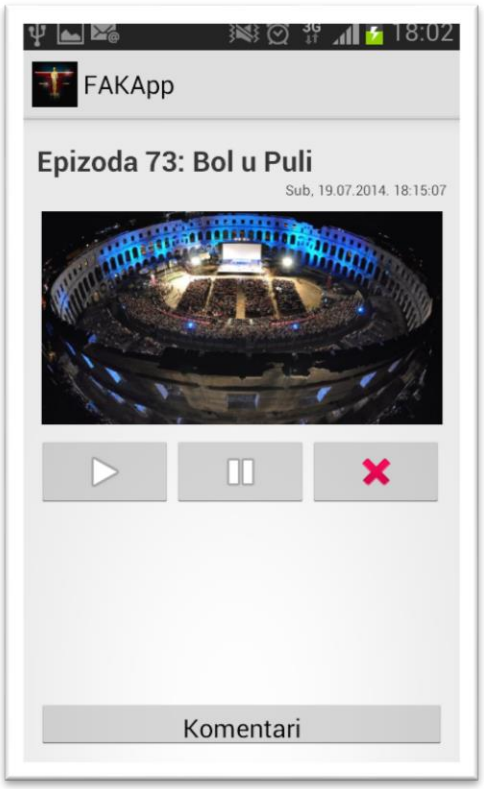
9. Literatura

1) About the Eclipse Foundation: http://www.eclipse.org/org/ (2014.)
2) Android Developers – Intent: http://developer.android.com/reference/android/content/Intent.html (2014.)
3) Android, the world's most popular mobile platform: http://developer.android.com/about/index.html (2014.)
4) AppBrain Stats: http://www.appbrain.com/stats/number-of-android-apps(2014.)
5) Bray, Tim; Paoli, Jean; Sperberg-McQueen, C.M.; Maler, Eve; Yergeau, Francois: Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml/, 26. studenoga 2008. (2014.)
6) Feigin, Ben. Mobile Application Development, The Search for Common Ground in a Divided Market, http://www.cs.cmu.edu/~bam/uicourse/830spring09/BFeiginMobileApplicationDevelopment.pdf (2014.)
7) Hayes, James D. Lazy User Theory and Interpersonal Communication Networks, prosinac 2009., https://etd.ohiolink.edu/!etd.send_file?accession=csu1336150484&disposition=inline (2014.), str. 12-16
8) Introduction: What is Object-Oriented Programming?: http://www.inf.ufsc.br/poo/smalltalk/ibm/tutorial/oop.html (2014.)
9) Learn About Java Technology: http://www.java.com/en/about/ (2014.)
10) Lunden, Ingrid. Gartner: 102B App Store Downloads Globally In 2013, $26B In Sales, 17% From In-App Purchases, 19. rujna 2013., http://techcrunch.com/2013/09/19/gartner-102b-app-store-downloads-globally-in-2013-26b-in-sales-17-from-in-app-purchases/ (2014.)
11) Meier, Reto. Professional Android Application Development. Wiley Publishing, Inc.2009. (2014.)
12) Mobile Applications, ITU-T TechWatch Alert, srpanj2009., http://www.itu.int/dms_pub/itu-t/oth/23/01/T230100000C0004PDFE.pdf (2014.)

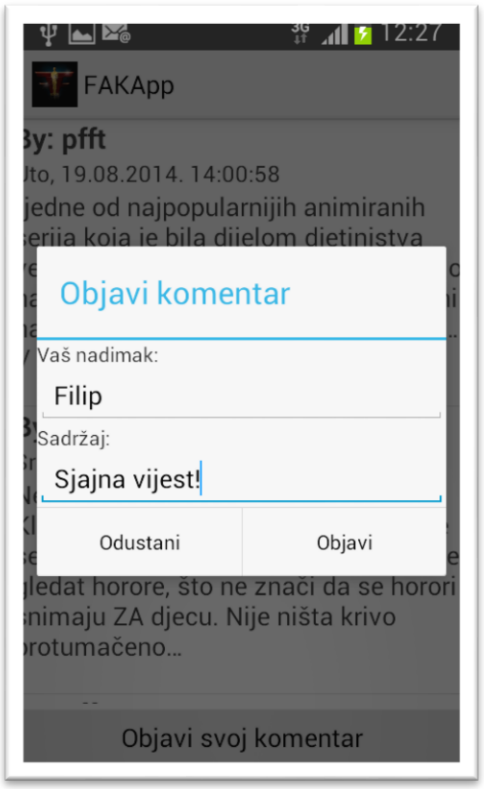
13) Wikipedia – Android (operating system): http://en.wikipedia.org/wiki/Android_(operating_system) (2014.)
14) Wikipedia – Eclipse (software): http://en.wikipedia.org/wiki/Eclipse_(software) (2014.)
15) Wikipedia – Java (programming language): http://en.wikipedia.org/wiki/Java_(programming_language) (2014.)
16) Wikipedia – Lazy User Model: http://en.wikipedia.org/wiki/Lazy_User_Model (2014.)
17) Wikipedia - Mobile App: http://en.wikipedia.org/wiki/Mobile_app (2014.)
18) Wikipedia - Mobile application development: http://en.wikipedia.org/wiki/Mobile_application_development (2014.)
19) Wikipedia - Object oriented programming: http://en.wikipedia.org/wiki/Object-oriented_programming (2014.)
20) Wikipedia – Podcast: http://en.wikipedia.org/wiki/Podcast (2014.)
21) Wikipedia – RSS: http://en.wikipedia.org/wiki/RSS (2014.)
22) Wikipedia – Streaming Media: http://en.wikipedia.org/wiki/Media_streaming (2014.)
23) Wikipedia - WordPress: http://en.wikipedia.org/wiki/WordPress (2014.)
24) Wikipedia – XML: http://en.wikipedia.org/wiki/Xml (2014.)
25) WordPress: http://wordpress.org/about/ (2014.)

10. Prilog – Primjer rada aplikacije

Slika 10 - Početni zaslon (izbornik za filtriranje sadržaja)



Slika 11 - Zaslon za članak s podcastom



Slika 12 - Zaslon s prozorom za objavljivanje komentara

11. Dodatak

Lista izvora koji su korišteni kao pomoć u razvijanju aplikacije:

- aXMLRPCdokumentacija: https://github.com/timroes/aXMLRPC
- Stack Overflow: http://stackoverflow.com/
- WordPress: http://wordpress.org/