SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET U RIJECI

Alen Šišović

RAZVOJ ANDROID APLIKACIJE ZA PRIJENOS DATOTEKA NA WEB STRANICU

Diplomski rad

Rijeka, 2014.

SVEUČILIŠTE U RIJECI

POMORSKI FAKULTET U RIJECI

RAZVOJ ANDROID APLIKACIJE ZA PRIJENOS DATOTEKA NA WEB STRANICU

Diplomski rad

Kolegij: Operacijski sustavi

Mentor: Prof. dr. sc. Božidar Kovačić

Student: Alen Šišović

Matični broj: 14599/E

Studij: Elektroničke i informatičke tehnologije u pomorstvu

Rijeka, veljača 2014.

SADRŽAJ

1. UVOD

S obzirom da je u današnje je vrijeme život bez pametnih telefona gotovo nezamisliv, te da se broj tablet računala svakim danom povećava, broj Android uređaja na tržištu je sve veći i veći. Upravo je zbog toga Android, kao vodeći operacijski sustav posebno zanimljiv razvojnim programerima koji svojim idejama pokušavaju osvojiti dio tržišta. Broj se aplikacija iz dana u dan povećava te je trenutno dostupno nešto više od milijun aplikacija te će taj broj u budućnosti još rasti. Posljedica toga je sve češća implementacija Android operacijskog sustava na razne uređaje što razvojnim programerima otvara vrata jednog velikog tržišta.

Upravo je zbog toga tema ovog diplomskog rada razvoj Android aplikacije, kako bi se pokazalo da je bez ikakvog početnog znanja Java programskog jezika u kratkom roku moguće usvojiti znanja iz područja razvoja Android aplikacija.

Najprije će u radu biti opisan operacijski sustav Android te njegove osnovne značajke. Također će biti opisana njegova arhitektura koja će biti prikazana i blok shemom. Kako se za razvoj Android aplikacija koristi Java programski jezik, isti će biti ukratko opisan te će biti objašnjena Java virtualna mašina koja je posrednik između napisane Java aplikacije i operacijskog sustava.

U trećem poglavlju ovog diplomskog rada će biti opisan razvoj Android aplikacija gdje će biti opisano korištenje razvojnog okruženje Eclipse. Samo razvojno okruženje Eclipse nije dovoljno za razvoj Android aplikacija već je isti potrebno proširiti sa službenim razvojnim paketom Android SDK te ADT koji će također biti opisani. U istom će poglavlju biti opisano od kojih se komponenti sastoji svaka Android aplikacija odnosno osnovna struktura Android aplikacija.

U četvrtom poglavlju će biti opisan razvoj aplikacije za prijenos fotografija na web stranicu. Biti će korištena Java skripta za prijenos datoteka preko HTTP protokola na web stranicu a za izradu aplikacije bit će korišteno razvojno okruženje Eclipse sa instaliranim Android SDK proširenjem. Aplikacija je zamišljena na taj način da korisnik na samom početku, odnosno nakon pokretanja aplikacije ima mogućnost odabira triju posebnih aktivnosti i tipke za gašenje aplikacije, odnosno za prekid njenog životnog ciklusa. Prva je aktivnost da se korisniku pritiskom na tipku ponudi odabir željene fotografije sa uređaja te

da tu istu, pritiskom na tipku pošalje na zadanu web stranicu. Također, sve prenesene fotografije će biti moguće pregledavati u zasebnoj aktivnosti. Treća aktivnost predstavlja tekstualni opis aplikacije. Razvojem navedenih aktivnosti pokušat će se objasniti korištenje osnovnih funkcija koje je moguće koristiti pri izradi Android aplikacija te korištenje nekih od funkcija razvojnog okruženja Eclipse.

Izrađena će aplikacija u obliku Java i XML koda biti dodana kao dodatak na kraju ovog diplomskog rada.

2. OPERACIJSKI SUSTAV ANDROID

Android je prvi otvoreni operacijski sustav za mobilne uređaje (mobilni telefoni, tablet i prijenosna računala i slično). U počecima je razvijan od strane tvrtke Android ali ga je kasnije odlučila kupiti tvrtka Google te su time započeli svoj proboj na tržište pametnih telefona. U studenom 2007. godine Open Handset Alliance izdao je Android SDK, alat za izradu softvera (engl. Software Development Kit) i zanimanje za njega neprestano raste. SDK sadrži alate i sučelja za programiranje aplikacija (engl. Application Programming Interface) skraćeno API-ja koji su potrebni za razvoj aplikacija na Android platformi koristeći programski jezik Java.

Svakim se danom broj izrađenih aplikacija na Google Play-u povećava te je u lipnju 2013. godine dosegnuta brojka od milijun aplikacija, što je više i od Apple-ova App Store-a sa 900 000 dostupnih aplikacija.

2.1 Osnovne značajke

Android operacijski sustav je zamišljen da u potpunosti i na najefikasniji način iskorištava sve resurse i mogućnosti koje mu pruža uređaj na kojemu je instaliran. Otvorenog je koda tako da programer aplikacija može cijeli operacijski sustav prilagoditi svojim potrebama u tolikoj mjeri da isti više nema veze s uobičajenim. Baziran je na Linux jezgri (engl. kernel) koja koristi specifični virtualni stroj dizajniran kako bi se optimiziralo korištenje memorije i ostalih hardverskih resursa uređaja koji ga pokreću. Upravo je zbog svoje otvorenosti koda pojednostavljena implementacija na razne uređaja te se isti sve više koristi ne samo na mobilnim uređajima već i na tablet i prijenosnim računalima, Smart TV-ima i slično.

Uz osnovne mogućnosti koje pruža sam operacijski sustav, korisniku se nudi i veliki broj aplikacija koje korisnik može preuzeti preko Google Play-a te se broj dostupnih aplikacija iz dana u dan sve više povećava. Nakon instalacije novih aplikacija ne pravi se razlika između temeljnih aplikacija i onih instaliranih naknadno. To znači da korisnik može organizirati sustav po vlastitim željama.

Za pohranu podataka koristi se relacijska baza podataka SQLite koja se ne pohranjuje u Androidu kao zasebni proces, već se integrira u samu aplikaciju. Što se tiče povezivanja sa drugim uređajima, koriste se standardne tehnologije koje dolaze skoro sa svakim mobilnim uređajem, GSM/EDGE/CDMA, UMTS, Bluetooth, WiFi, LTE[1] i NFC[2]. Preko Bluetooth-a podržano je upravljanje ostalim uređajima (primjerice televizija, radio) te prijenos datoteka sa jednog uređaja na drugi. Ugrađeni Internet preglednik se temelji na WebKit-ovom engine-u uparenom sa Googleovim Chrome V8 JavaScript engine-om. Iako je većina Android aplikacija pisana u Java programskom jeziku, sam Android operacijski sustav nema svoj Java Virtual Machine pa tako nije moguće izvršavati Java bajt kod. Za pokretanje Java aplikacija, Android koristi Dalvik virtualni stroj.

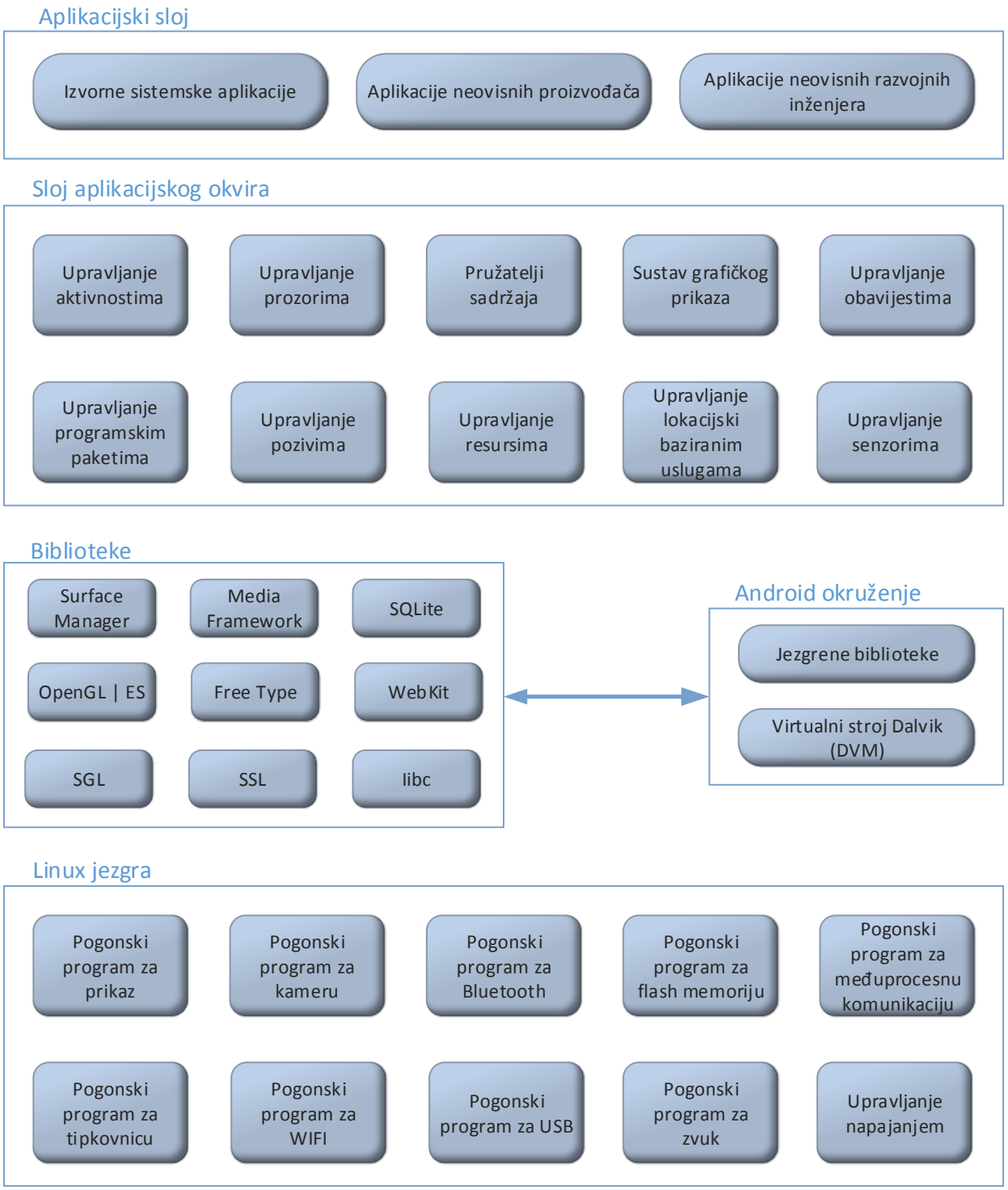
Što se hardverske podrške tiče Android ima ugrađenu podršku za ekran osjetljiv na dodir, GPS, akcelerometar, žiroskop, senzore osjetljive na dodir i blizinu, termometar i grafičku 3D akceleraciju, te također postoji podrška i za multi-touch.

2.2 Arhitektura Android operacijskog sustava

Android operacijski sustav je baziran na Linux 2.6 jezgri (engl. kernel) napisanom u C ili C++ programskom jeziku. S obzirom na to da je otvorenog koda, aplikacije putem posrednika (engl. Middleware) mogu komunicirati i pokretati druge aplikacije primjerice za ostvarivanje poziva, slanje i primanje SMS poruka i email-ova, pokretanja kamere i slično. Iako su C i C++ programski jezici korišteni za radno okružje (engl. Framework), većina aplikacija pisana je u Java programskom jeziku koristeći Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u C ili C++ programskom jeziku, ali se tada koristi Android Native Code Development Kit (NDK). Ovakvim se postupkom omogućuje bolje raspolaganje resursima uređaja i korištenje knjižnica iz jezgre i radnog okružja. Ovakvim se postupkom brzina aplikacije povećavaju i do 10 puta, no pisanje samog koda je složenije. Da se što lakše objasni arhitektura Android operacijskog sustava prikazana je slikom 1. Sastoji se od četiri djela koji će biti objašnjeni u daljnjem tekstu.

[1] Long Term Evolution

[2] Near Field Communication



Slika 1: Arhitektura Android operacijskog sustava

Izvor: Autor prema [1]

2.2.1 Linux jezgra

Na dnu stoga nalazi se Linux 2.6 jezgra koji predstavlja jezgru na kojoj je baziran Android operacijski sustav. Sadrži upravljačke programe (engl. Driver) od kojih su najvažniji programi za međuprocesnu komunikaciju (IPC[3]) koji služi za izmjenu podataka između različitih procesa, upravljački programi za upravljanje napajanjem (Power Managment). Tu se također nalaze i upravljački programi za neke od osnovnih hardverskih komponenti samog sustava poput upravljačkog programa za ekran, kameru, tipkovnicu, Wi-Fi, zvuk i pohranu podataka.

2.2.2 Biblioteke

Iznad jezgre operacijskog sustava nalaze se biblioteke koje su pisane u C/C++ programskom jeziku. One sadrže sav kod koji obavlja glavne funkcije.

- Surface Manager – biblioteka koja nadzire iscrtavanje grafičkog sučelja
- Media Framework – biblioteka temeljena na OpenCORE koja podržava snimanje i reproduciranje poznatih audio/video formata
- SQLite – biblioteka za upravljanje bazama podataka dostupna svim aplikacijama
- OpenGL | ES – biblioteka za sklopovsko ubrzavanje 3D prikaza (ukoliko je moguća) te za visoko optimiziranu 3D softversku rasterizaciju
- FreeType – biblioteka namijenjena iscrtavanju fontova
- WebKit – engine za web preglednike
- SGL – 2D biblioteka korištena za većinu aplikacija
- SSL (Secure Sockets Layer) - biblioteka za sigurnosnu komunikaciju putem Interneta
- libc – sistemska C biblioteka prilagođena za ugradbene sustave zasnovane na Linux OS-u

2.2.3 Android radno okruženje

Slijedi android okruženje (engl. Android Runtime) koji se nalazi u istom sloju kao i biblioteke operacijskog sustava. Sastoji se od dvije važne komponente. Prva su tzv. "Core

[3] Inter-Process communication

libraries" odnosno biblioteke koje sadrže većinu jezgrenih biblioteka programskog jezika Java. To omogućuje razvojnim programerima da izrađuju Android aplikacije uz pomoć Java programskog jezika. Druga komponenta je Dalvik Virtual Machine (DVM) koji pokreće aplikacije kao zasebne procese odnosno kao instance virtualnog stroja. DVM je specijalna virtualna mašina koja pretvara Java class datoteke u svoj vlastiti format (.dex) te je prilagođen za korištenje na uređajima koji se napajaju baterijom, imaju ograničenu procesorsku snagu i radnu memoriju.

2.2.4 Aplikacijski okvir

Aplikacijski okvir (engl. Applicaton framework) dozvoljava upotrebu svih API-ja (Application Programming Interface). Time se omogućilo upravljanje programskim paketima, resursima , pozivima, aktivnostima aplikacije, korištenje podataka od više različitih aplikacija, dohvaćanje i korištenje trenutne lokacije korisnika, prikaz obavijesti te baza pogleda i objekata koji mogu biti korišteni za dizajn aplikacije.

2.2.5 Aplikacije

Na vrhu se nalaze aplikacije. Ovaj sloj je vidljiv krajnjem korisniku i sastoji se od aplikacija koje dolaze sa Android operacijskim sustavom poput programa za pozive, kontakti, SMS programa, kalendar, web preglednika i slično. Tu se nalaze i sve aplikacija koje se mogu naći na Google Play-u odnosno aplikacije koje stvaraju sami korisnici.

2.3 Programski jezik Java

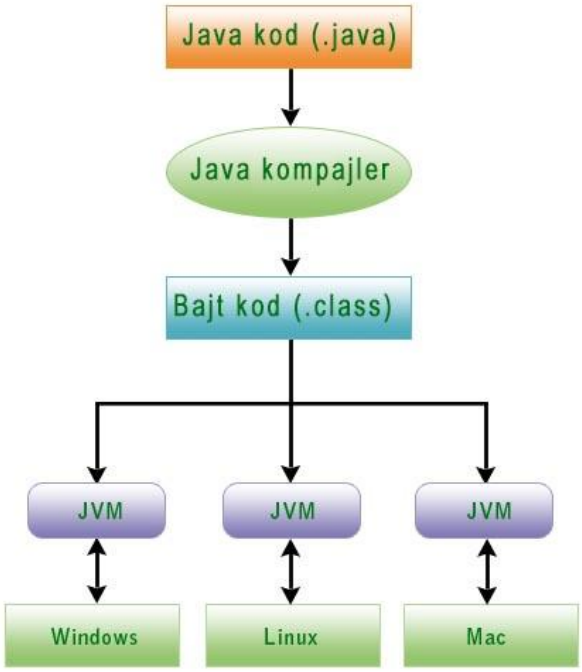
Razvoj Jave započinje u 1991. godine kada je Sun Microsystems pokrenuo projekt razvoja novog programskog jezika koji je namijenjen za male uređaje. Cilj je bio stvoriti jezik koji će trošiti malo memorije, raditi na slabim i različitim procesorima. Java je objektno orijentirani programski jezik što znači da znači da su programi koncipirani poput grupe objekata koji međusobno djeluju. Jedna od bitnijih značajki Java programskog jezika da je neovisan o platformi. To znači da možemo napisani kod izvršavati na bilo kojem drugom računalu bez obzira na njegovu arhitekturu. Java kod prevodi se u tzv. bajtkod

(engl. bytecode) kojeg pokreće operacijski sustav, drugi program ili uređaji pomoću Java interpretera. Bajtkod nije izvršni kod, već skup instrukcija dizajniran za izvršavanje unutar Java izvršnog sustava koji se naziva „Java virtualna mašina“ (Java virtual machine – JVM) koji predstavlja interpreter za bajtkod.

2.3.1 Java virtualna mašina – JVM

Prilikom kompajliranja programskog koda dobivamo tzv. Java bajt kod (class datoteku) i taj kod kao takav nije razumljiv izvršnom uređaju. Zbog toga se koristi Java virtualna mašina (JVM) koji razumije takav kod i Java bajt kod pretvara u jezik razumljiv uređaju. Jednostavno rečeno Java virtualna mašina je posrednik između napisane Java aplikacije i operacijskog sustava. Taj je proces prikazan blok dijagramom na slici 2.

Kako bi se osiguralo izvršenje na različitim operacijskim sustavima postoji više vrsta JVM-a, pa tako postoji poseban JVM za Windows, Linux ili Unix dok je bajt kod za svaki operacijski sustav isti.



Slika 2: Blok dijagram kompajliranja Java koda

Izvor: Autor prema [5]

3. RAZVOJ ANDROID APLIKACIJA

Kako se u zadnje vrijeme Android operacijski sustav sve više i više nameće na tržištu mobilne industrije sasvim je logično da je i interes za razvojem aplikacija baš za taj operacijski sustav sve veći. Razlog tome je to što je Android besplatan i otvorenog koda te se aplikacije pišu u Java programskom jeziku.

Za razvoj Android aplikacija koristi se razvojno okruženje Eclipse kojeg je potrebno proširiti s Android razvojnim paketom nazvanim Android SDK (engl. Software Development Kit). Pored toga za razvoj je dostupan niz besplatnih alata i biblioteka programskog koda, a postoji velika baza razvojnih timova koji razvijaju aplikacije u tom okruženju i na čiju pomoć se može računati tijekom razvoja.

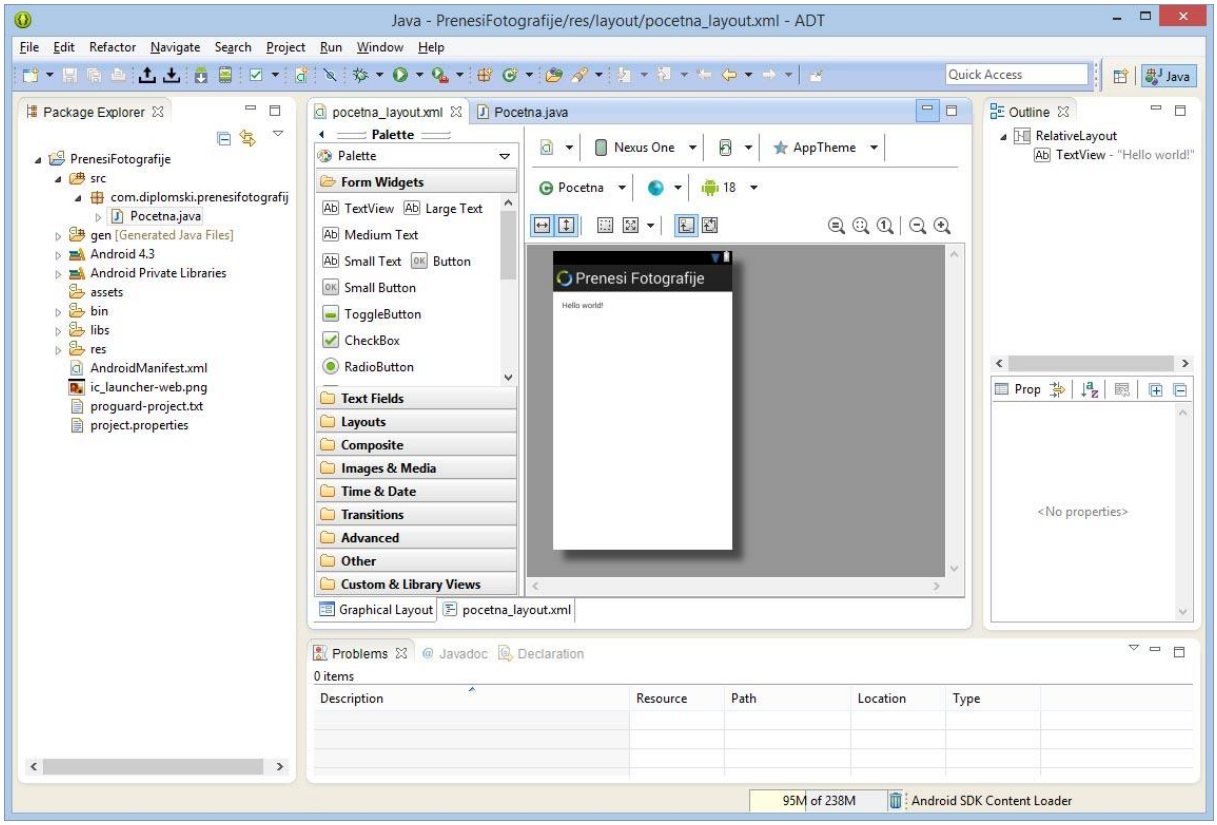
3.1 Razvojno okruženje Eclipse

Kako bi razvojnim programerima omogućilo razvijanje aplikacija za određene skupine programa, programskog okruženja, operacijskih sustava i sličnih platformi razvijen je Android razvojni alat. Njega sačinjavaju skupine gotovih programa preko kojih je omogućena lakša komunikacija sa specifičnim programom ili platformom. Uz to dolazi i alati za lakše otklanjanje grešaka u programu, dizajn sučelja te drugi slični alati koji pomažu programerima u pisanju što kvalitetnijih aplikacija.

Eclipse je programsko okruženje otvorenog koda koje je neovisno o operacijskom sustavu te je bazirana na programskom jeziku Java. Sastoji se od integriranog razvojnog okruženja (IDE) i proširivog plug-in sistema. Eclipse je razvijen od strane Object Technology International kompanije kao Java softver otvorenog koda. Može se koristiti za razvoj aplikacija u Javi i drugim programskim jezicima putem različitih softverskih dodataka. Također je moguće Eclipse nadograditi sa dodacima za programske jezike C, C++, COBOL, Perl, PHP, Python i druge. Google je prepoznao tu prilagodljivost Eclipse-a te je baš zbog toga odabrao Eclipse kao službeno okruženje za izradu aplikacija baziranih na Android operacijskom sustavu.

Razvojno okruženje Eclipse-a je vrlo prilagodljivo, što krajnjem korisniku omogućava uređivanje različitih dijelova programa jednostavnim premještanje, dodavanjem ili

uklanjanjem pojedinih elemenata. Eclipse se sastoji od preglednika i radnog prostora. Na slici 3 je prikazano razvojno okružje Eclipse gdje se sa lijeve strane nalazi preglednik programskog paketa (engl. Package Explorer), radnog prostora za pisanje programskog koda i drugih opcionalnih prozora koje korisnik može odabrati.



Slika 3: Prikaz razvojnog okruženja Eclipse

Izvor: Autor

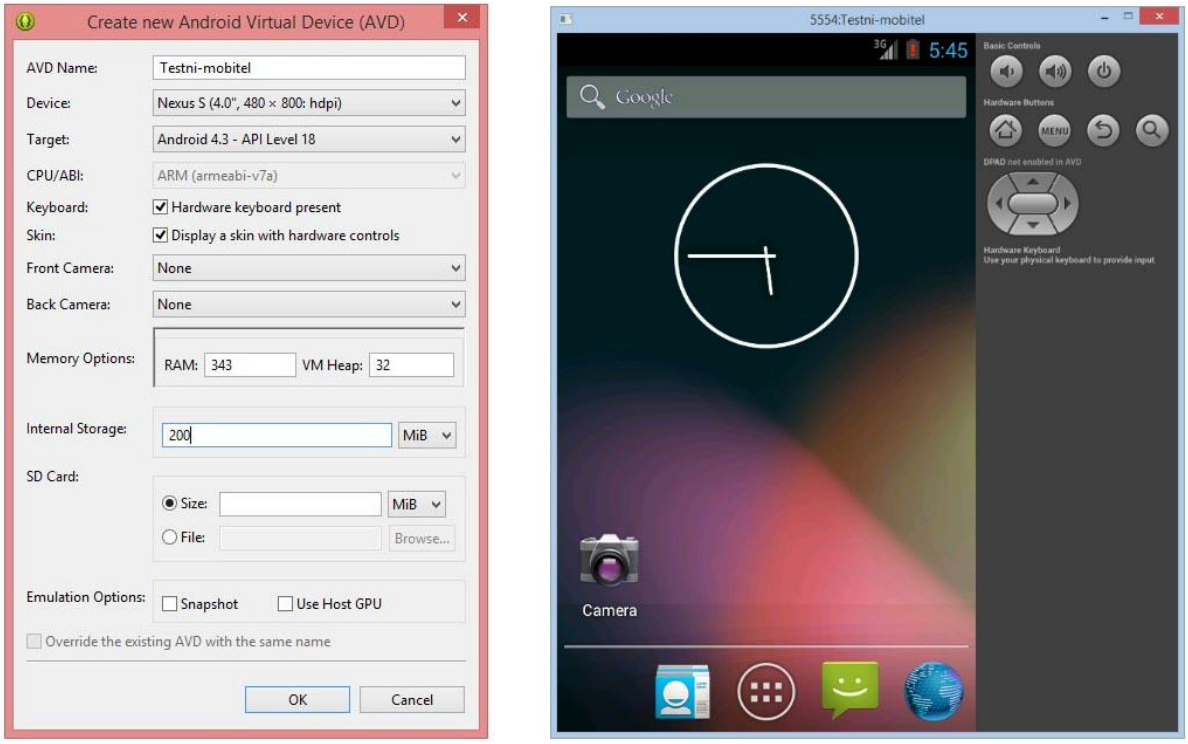
Samo razvojno okruženje Eclipse nije dovoljno za razvoj Android aplikacija već je isti potrebno proširiti sa službenim razvojnim paketom Android SDK te ADT (Android Development Tools).

3.1.1 Android razvojni program (ADT)

ADT omogućava kreiranje, uređivanje i uklanjanje grešaka (engl. debugging) kod Android aplikacija. ADT je paket softverskih razvojnih alata u kojem su uključene sljedeće komponente:

- Android virtualni uređaj (AVD)

Android virtualni uređaj (engl. Android Virtual Device) je emulator koji se koristi za testiranje aplikacija bez fizičkog uređaja. Pogodnost ovog programa je testiranje aplikacija na različitim virtualnim uređajima koji mogu imati različite postavke, verzije operativnog sustava, veličine ekrana, memorije, brzine procesora itd.

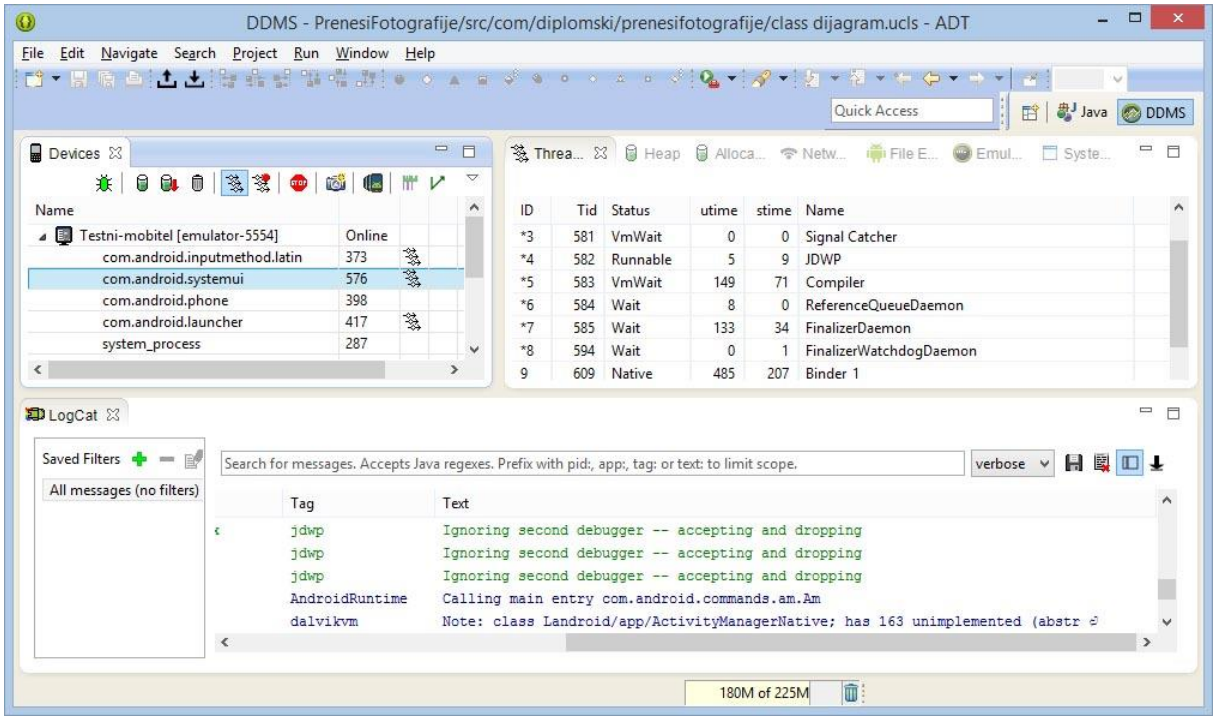


Slika 4: a) Kreiranje novog virtualnog uređaja b) Prikaz virtualnog uređaja

Izvor: Autor

- DDMS (engl. Dalvik Debug Monitor Server)

Služi za pronalaženje i uklanjanje pogrešaka u programskom kodu i za pregled i kontrolu izvršavanja na emulatoru ili pokretnom uređaju. Također pruža mogućnost preusmjeravanja vrata (eng. port redirection), upotrebu, praćenje informacija o procesima, snimanje aktivnog zaslona uređaja, kontrolu lažiranih (eng. spoofing) poziva, SMS poruka i podataka o lokaciji, te mnoge druge usluge. Cijeli se alat ponaša kao poveznica između IDE-a i aplikacija koje se izvršavaju na uređaju. DDMS je prikazan slikom 5.



Slika 5: DDMS (Dalvik Debug Monitor Server)

Izvor: Autor

- AAPT (engl. Android Asset Packaging Tool)

Koristi se za stvaranje i distribuciju Android-ovog programskog paketa u .apk format

- ADB (engl. Android Debug Bridge)

Koristi se za povezivanje koda na emulatoru ili uređaju sa DDMS alatom

- Detaljna dokumentacija o Android-u
- Primjeri koda za demonstraciju korištenja svih verzija API-a

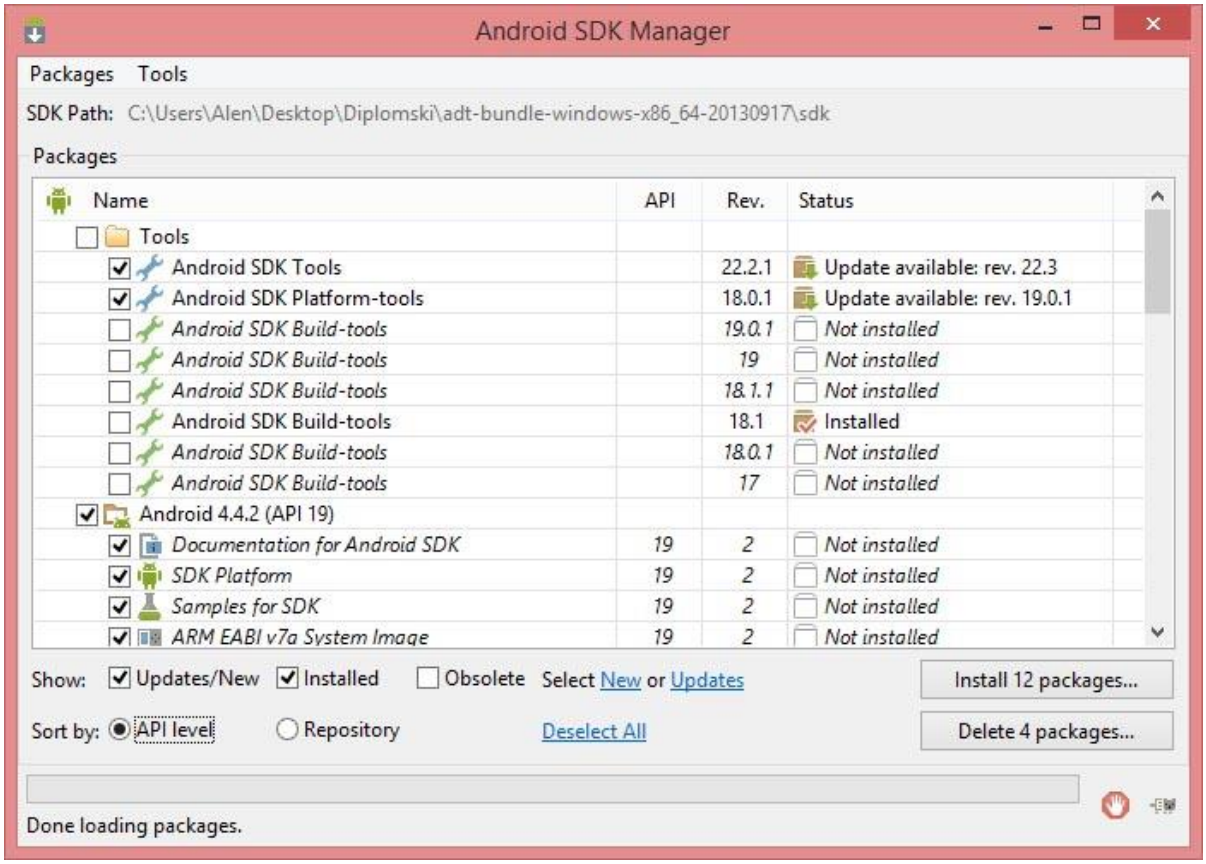
3.1.1.1 Android SDK

Android SDK[4] predstavlja skup alata za programiranje Android aplikacija u Java programskom jeziku. U Android SDK je uključen skup biblioteka, Android emulator za testiranje aplikacija, program za otklanjanje grešaka, dokumentacija za učenje kao i primjeri gotovih Android aplikacija. Android SDK radi na svim operacijskim sustavima,

[4] Software Development Kit

Linuxu te na svim njegovim distribucijama, Mac OS X 10.5.9 pa nadalje, Windowsima XP na više a i na samom Android operacijskom sustavu uz pomoć AIDE (Android IDE). Za razvoj Android aplikacija razvijemo je razvojno okruženje Eclipse koji koristi Android razvojne alate (ADT). Osim razvojnog okruženja Eclipse pisanje aplikacija za Android moguće je u bilo kojem tekstualnom i XML editoru, te kompajliranjem izvornog koda u komandnom okruženju.

Osim izrade Android aplikacije namijenjene zadnjoj inačici Androida možemo i odabrati verziju stariju operativnog sustava te tako ciljati i na korisnike sa starijim verzijama Androida. Sve te mogućnosti mogu se testirati u emulatoru tako da nije potrebno imati mobitele sa različitim verzijama sustava radi testiranja. Jednom kada se izvorni kod kompajlira, Android SDK ga pakira u obliku .apk datoteke koja sadrži izvršni kod kojeg čita Android-ov JVM, Dalvik, te ostale datoteke potrebne za rad aplikacije kao što su slike, dizajn aplikacije, zvukovi i sl.



Slika 6: Android SDK Manager

Izvor: Autor

3.1.1.2 Android NDK

Android aplikacije se najčešće pišu u Javi iako to ne treba uvijek biti slučaj. Ponekad, kada nam za aplikaciju trebaju neke stvari po kojima su ostali programski jezici bolji od Jave, npr. upravljanje memorijom i performansama koristi se Android NDK (engl. Native Development Kit). Android NDK je skup alata za programiranje Android aplikacije kompajliranih u izvorni kod koji razumije ARM ili x86 procesor. Android NDK podržava razvoj aplikacija u C i C++ programskim jezicima te je to glavna razlika u odnosu na Android SDK koji koristi Java programski jezik. Prednost ovakvog načina programiranja je bolje upravljanje memorijskih resursima zbog direktne povezanosti sa uređajem dok je kod SDK Dalvik JVM posrednik preko kojeg se kod izvršava.

3.2 Komponente Android aplikacije

Kako su Android aplikacije su pisane u Java programskom jeziku Android SDK kompajlira programski kod u izvršni kod. Izvršni kod je datoteka sa ekstenzijom .apk te se takva datoteka može instalirati na uređaj sa Android operacijskim sustavom. Svaka se aplikacija sastoji od više dijelova koji će biti opisani u daljnjem tekstu.

3.2.1 Osnovna struktura aplikacije

Kod Android aplikacija bitno je napomenuti da jedna aplikacija može koristiti elemente od neke druge aplikacije uz prethodnu definiciju dozvola od strane prve aplikacije. Time je pisanje istih dijelova u različitim aplikacijama smanjeno, jer je moguće pokrenuti samo dio aplikacije koji je potreban drugoj aplikaciji. Android to omogućava tako da ne postoji samo jedna točka ulaska za svaku komponentu aplikacije nego je svaka aplikacija napravljena od osnovnih komponenti koje sustav može instancirati i pokrenuti po potrebi.

Postoje četiri vrste takvih komponenti, to su:

- Aktivnost
- Usluge
- Primatelj prijenosa
- Dobavljači sadržaja

3.2.1.1 Aktivnost

Zadatak komponente aktivnosti (engl. Activity) je prikaz korisničkog sučelja programa i omogućavanje interakcije korisnika sa programom. Android aplikacija obično sadrži jednu glavnu aktivnost (npr. početni zaslon) i nekoliko aktivnosti koje su međusobno povezane na način da se iz prve, početne aktivnosti pokreću ove ostale na zahtjev korisnika. To može biti, primjerice, početni meni koji se otvara pri pokretanju aplikacije, postavke neke aplikacije ili detalji o aplikaciji. Svaki od njih je nezavisan dio aplikacije iako rade zajedno i dio su iste aplikacije. Svaki od njih je podklasa osnovne klase aktivnost te se pokretanje ostalih aplikacija izvodi po principu „last in, first out". Po pokretanju nove aktivnosti, prekida se prethodna aktivnost i ponovno pokreće po prekidu trenutne aktivnosti. Time se štedi na radnoj memoriji mobilnog uređaja što je i jedan od ciljeva razvoja Android aplikacije. Svaka aktivnost implementirana je kao klasa koja proširuje osnovnu komponentu aktivnost. Koliko će aplikacija imati aktivnosti ovisi o zamišljenom dizajnu i o funkcionalnosti same aplikacije. Svaki od aktivnosti ima svoj poseban izgled koji je zapisan u .xml datoteci a poziva se u samoj aktivnosti. Vizualni sadržaj izgleda ima hijerarhijski organiziran sadržaj elemenata prikaza koji su svi izvedeni iz osnovne klase View. Postoji skup gotovih elemenata prikaza sadržanih u Androidu kao što su gumbi, polja za unos teksta, izbornici, check-boxovi i drugi.

3.2.1.2 Usluge

Za razliku od komponente aktivnost koja je vidljiva korisniku, aktivnost usluge (engl. Service) nema grafičko sučelje nego radi u pozadini. Koristi se za zadatke koji se mogu izvoditi u pozadini te traje neki nedefinirani period vremena. Može se koristiti za dohvat podataka preko mreže ili za izračun nekih podataka koje može isporučiti aktivnosti koji ih treba. Svaka usluga je podklasa osnovne klase usluga. Kao i komponenta aktivnost i ostale komponente, i usluga se izvodi u glavnoj niti procesa aplikacije. Kako u ovakvim slučajevima ne bi blokirali druge komponente korisničkog sučelja, mogu stvoriti novu nit za radnje za koje se zna da zahtijevaju više vremena. Kao primjer komponente usluge je sviranje glazbe u pozadini dok korisnik radi nešto drugo.

3.2.1.3 Primatelji prijenosa

Uloga primatelja prijenosa (engl. BroadcastReceiver) je primanje i reagiranje na emitiranje obavijesti koje može dolaziti od strane sistema uređaja (npr. baterija je prazna, obavijesti sa Interneta, zaslon je uključen), ili od aplikacija (komunikacija s drugim aplikacijama). Takve obavijesti mogu se prikazati na različite načine – kao treperenje pozadinskog svjetla, vibracija pokretnog uređaja, reprodukcija određenog zvuka. Na traci stanja pojavi se ikona koju korisnik može odabrati kako bi pročitao obavijest. Ova komponenta kao ni komponenta usluga nema korisničko sučelje, ali može pokrenuti aplikaciju kao rezultat primljene informacije. Aplikacija može imati više primatelja prijenosa koji mogu odgovarati na sve informacije koje se smatraju važnima. Osnovna klasa svakog od njih je primatelj prijenosa.

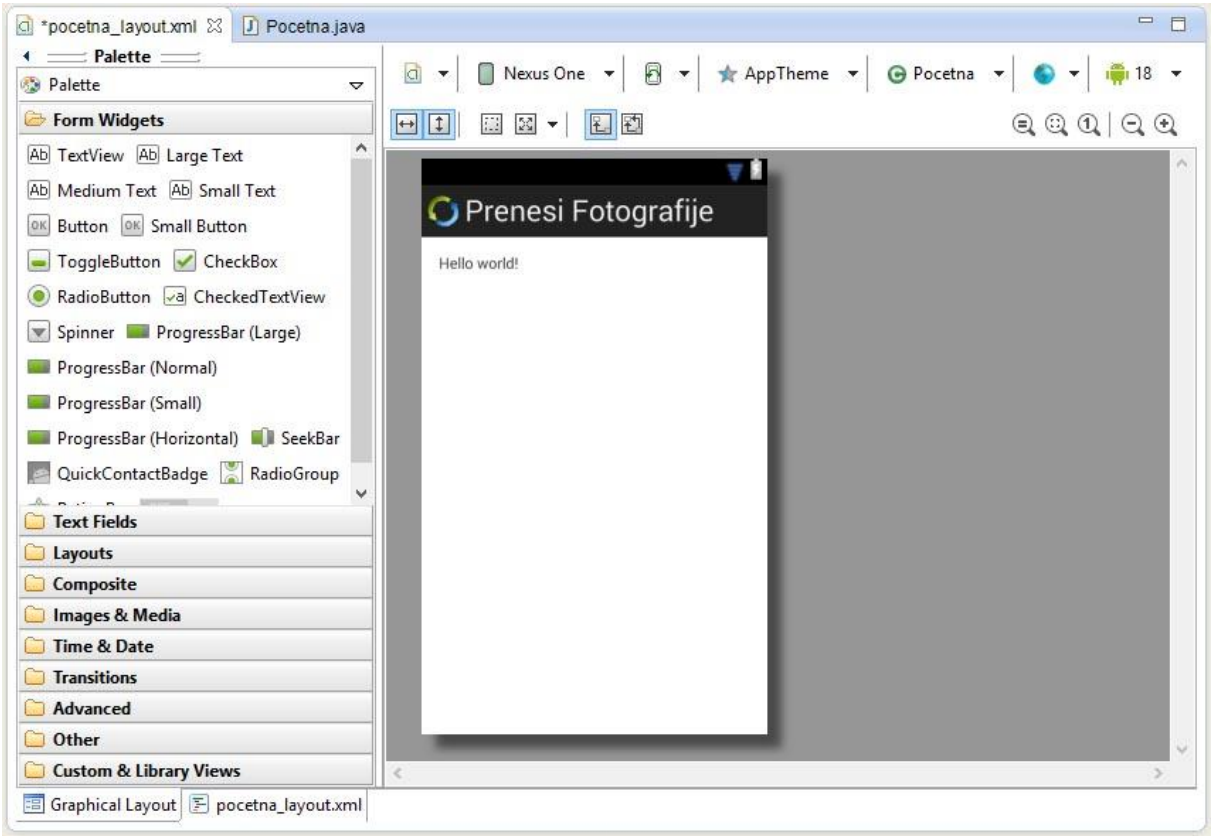
3.2.1.4 Dobavljači sadržaja

Dobavljači sadržaja (engl. content providers) je komponenta aplikacije koja omogućuje aplikacijama da koriste podatke drugih aplikacija. Ti podaci mogu biti pohranjeni u podatkovnom sustavu, bazama podataka SQLite ili na neki drugi smisleni način te je to ujedno i jedini način za izmjenjivanje podataka među aplikacijama. Uz SQLite moguće je koristiti PostgreSQL bazu podataka ili neku drugi način za spremanja sadržaja. Osnovna klasa je ContentProvider koja sadrži standardni skup metoda koje omogućuju aplikacijama preuzimanje ili pohranu tipova podataka s kojima rade. Kako bi se kreirao dobavljač sadržaja najprije se odredi lokacija za spremanje podataka. Tada dobavljač sadržaja daje sadržaj u obliku tablice sa identifikatorom i ostalim tabličnim atributima. Kako bi se pristupilo sadržaju koristi se razlučivač sadržaja (engl. Content Resolver). Upit sadrži adresu (URI) sadržaja, imena polja iz tablice i tip podataka – tekstualni tip (String), cjelobrojni tip (Integer) ili decimalni tip (Float). Svaki dobavljač sadržaja implementiran je kao klasa koja proširuje osnovnu klasu dobavljača sadržaja.

3.2.2 Izgled korisničkog sučelja

Svaka Android aplikacija je zamišljena i razvijena na drugačiji način. Ono što ih sve razlikuje je dizajn, odnosno izgled same aplikacije. Za dizajniranja korisničkog sučelja postoje dva načina; proceduralno i deklarativno. Proceduralni dizajn je kompliciranije rješenje a odnosi se na pisanje Java koda, dok je jednostavniji i učestaliji način deklarativni način, odnosno pisanje XML (engl. *EXtensible Markup Language*) koda kojeg se kasnije poziva u Java klasi. U praksi se za kreiranje grafičkog korisničkog sučelja uglavnom koristi XML. Prednost deklariranja korisničkog sučelja XML datotekom je u tome što se time odvaja prikaz aplikacije od koda koji upravlja njenim ponašanjem.

Prilikom kreiranja nove Android aplikacije dobiva se mogućnost da razvojni programer sam bira način izrade korisničkog sučelja. Prva je mogućnost da korisnik gleda trenutni izgled korisničkog sučelja i da po principu WYSIWYG[5] slaže izgled aktivnosti a druga mogućnost pisanje XML koda.



Slika 7: Izrada izgleda aktivnosti preko Eclipse korisničkog sučelja

Izvor: Autor

[5] What You See Is What You Get

Na slici 7 je prikazan grafički način izrade izgleda za aktivnost „Pocetna.java“ te je to početni izgled pri kreiranju nove Android aplikacije. Prebacivanjem iz ovakvog pogleda u XML dobivamo sljedeći kod:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Pocetna" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Svaka klasa aktivnosti može pozvati svoju ili zajedničku XML datoteku u kojoj je opisan izgled aktivnosti. Sljedeći kod prikazuje pozivanje XML datoteke „pocetna_layout.xml“ u klasi aktivnosti „Pocetna.java“.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.pocetna_layout);
    }
```

Preko XML koda moguće je u potpunosti mijenjati izgled aktivnosti Android aplikacije. Osim definiranja margina, pozicije tipki ili ostalih dostupnih elemenata razvojni programer bira između nekoliko različitih rasporeda (engl. Layouta) pa tako imamo:

- Linearni raspored
- Apsolutni raspored
- Tablični raspored

- Relativni raspored
- Okvirni raspored
- Listajući pregled

Linearni raspored (engl. LinearLayout) predstavlja osnovni raspored elemenata aktivnosti gdje su svi osnovni elementi raspoređeni u samo jednom redu ili samo jednom stupcu. Iako za složenije aplikacije ovakav raspored osnovnih kontrola nije uvijek i najpogodnije rješenje, kod jednostavnijih aplikacija predstavlja najbolje i najjednostavnije rješenja.

Idući raspored elemenata je apsolutni raspored (engl. AbsoluteLayout). Kod njega razvojni programer nije ograničen na neki od podrazumijevanih oblika „redanja“ osnovnih kontrola po pozadini, nego ima potpunu slobodu u njihovom postavljanju na dostupnom području.

Treći po redu je tablični raspored elemenata(engl. TableLayout). Kao što to samo ime govori, koristi se kod razvrstavanja osnovnih kontrola u više redova ili stupaca tablice, odnosno u ćelije tablice. Tako je razvojnom programeru omogućeno pravilno raspoređivanje osnovnih kontrola na samome rasporedu.

Kod relativnog rasporeda (engl. RelativeLayout) elemenata aktivnosti kontrole se mogu postaviti na različita mjesta na dostupnom prostoru za njihovo prikazivanje, ali točno mjesto svake kontrole nije određeno njezinim apsolutnim koordinatama kao kod apsolutnog rasporeda elemenata nego se pozicija osnovnog elementa aktivnosti definira relativnim koordinatama prema drugoj osnovnoj kontroli.

Uz ove navedene, postoji i okvirni raspored (engl. FrameLayout) te se u aplikacijama koristi za rezerviranje mjesta za naknadno postavljanje drugih kontrola tako da se one uvijek pozicioniraju u odnosu na gornji lijevi uglu okvirni raspored elemenata aktivnosti.

Posljednja vrsta raspoređivanja elemenata aktivnosti je listajući pregled (engl. ScrollView) Taj raspored elemenata koristi se u slučajevima kad na fizičkim dimenzijama zaslona jednostavno nije moguće prikazati sve potrebne osnovne kontrole na smislen i pregledan način (bez međusobnog preklapanja i jasnog razdvajanja). Pogled listajućeg pregleda zapravo predstavlja posebnu vrstu rasporeda okvirnog pregleda na koju se u praksi obično postavlja dodatna vrsta osnovnog rasporeda kontrola.

3.2.3 Manifest datoteka

Prije pokretanja neke komponente aplikacije, Android operacijski sustav mora znati od kojih se komponenata aplikacija sastoji. Sve Android aplikacije deklariraju svoje komponente u manifestu koji je spakiran u korijenskom direktoriju paketa. Manifest je strukturirana XML datoteka i uvijek se zove AndroidManifest.xml za sve aplikacije.

Sljedećim kodom je prikazana datoteka AndroidManifest.xml koja se kreira automatski pri izradi novog Android projekta.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.diplomski.prenesifotografije"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.diplomski.prenesifotografije.Pocetna"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Između ostalog, manifest datoteka sadrži slijedeće informacije o aplikaciji:

- imenuje sve Java pakete za aplikaciju (naziv paketa koji aplikacija koristi kao jedinstveni identifikator)
- opisuje komponente aplikacije – aktivnosti, usluge, pružatelji sadržaja, filtre namjera (engl. Intent Filters) i dobavljače sadržaja od kojih je aplikacija sastavljena (imenuje klase koje implementiraju svaku komponentu i objavljuje njihove mogućnosti)

- odredbe o tome koji će procesi sadržavati programske komponente

- deklarira dozvole koje aplikacija mora imati kako bi pristupala zaštićenim dijelovima API-ja i komunicirala s drugim aplikacijama

- deklarira dozvole koje drugi trebaju imati kako bi mogli vršiti interakciju s komponentama aplikacije

- sadrži popis instrumentacijskih klasa koje osiguravaju oblikovanje i ostale informacije dok je aplikacija aktivna – ove deklaracije su prisutne u AndroidManifest.xml datoteci prilikom razvoja i testiranja, te se izbacuju prije njenog objavljivanja

- deklarira minimalnu razinu API-ja koju aplikacija zahtijeva i izlistava biblioteke s kojima aplikacija mora biti povezana

4. ANDROID APLIKACIJA ZA PRIJENOS FOTOGRAFIJA NA WEB STRANICU

U ovome djelu rada radu će biti objašnjena izrada aplikacije za prijenos fotografija sa Android uređaja na web stranicu. Biti će korištena Java skripta za prijenos datoteka preko HTTP protokola na web stranicu. Za izradu aplikacije bit će korišteno razvojno okruženje Eclipse sa instaliranim Android SDK proširenjem.

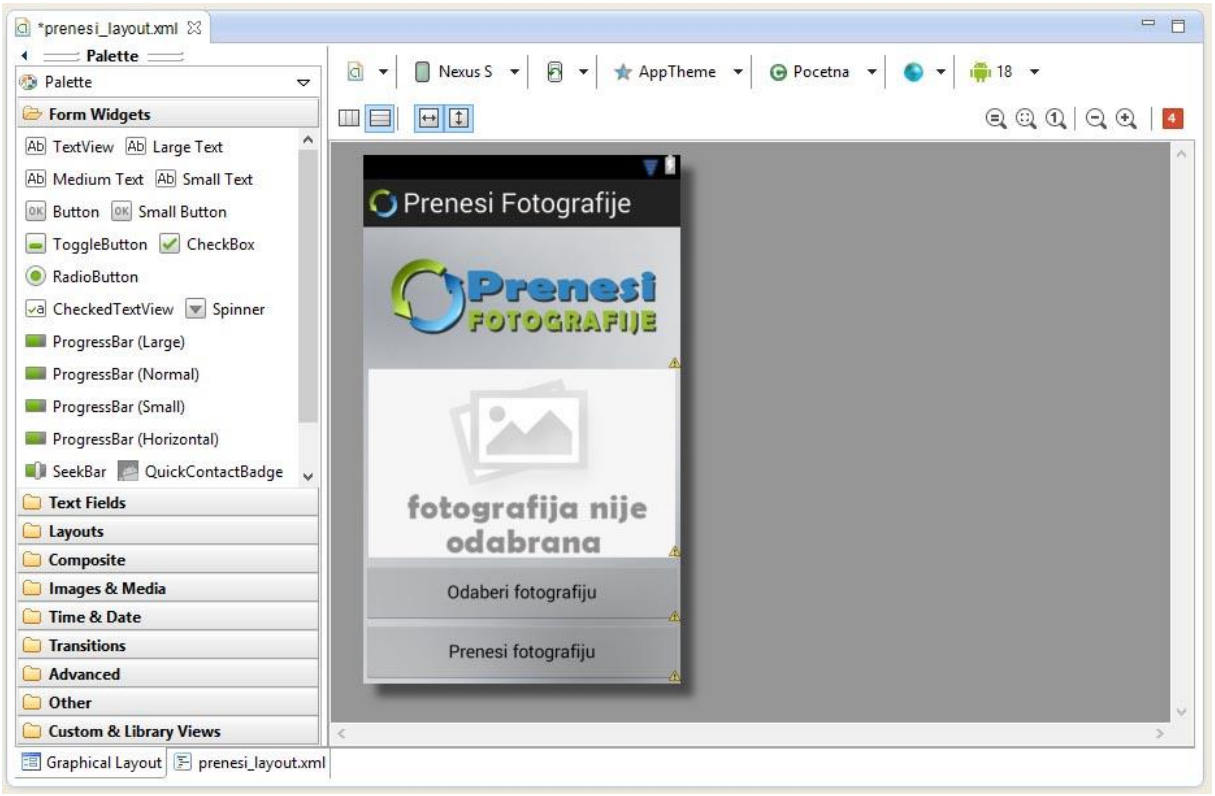
Aplikacija je zamišljena na taj način da korisnik na samom početku, odnosno nakon pokretanja aplikacije ima mogućnost odabira triju posebnih aktivnosti i tipke za gašenje aplikacije, odnosno za prekid njenog životnog ciklusa. Prva je aktivnost da se korisniku pritiskom na tipku ponudi odabir željene fotografije sa uređaja te da tu istu pritiskom na tipku pošalje na zadanu web stranicu. Također, sve prenesene fotografije bi mogao pregledavati na svome uređaju u okviru nove, zasebne aktivnosti. Treća aktivnost predstavlja tekstualni opis programa, ime i prezime autora i mentora, fakultet i slično.

4.1 Aktivnost za prijenos fotografija

Svaka aktivnost sadrži dvije komponente; Izgled aktivnosti u XML formatu i izvršni kod u Java formatu poznat kao klasa aktivnosti. Svaka aktivnost može imati svoj izgled, odnosno posebno napravljenu XML datoteku koja se kasnije poziva u klasi aktivnosti. Ovisno o zamišljenom izgledu, razvojni programer može birati između nekoliko rasporeda (engl. Layout) kao što je prethodno u radu opisano.

Za početak izrade bitno je da razvojni programer složi okvirni dizajn same aplikacije, odnosno da odredi koje elemente želi koristit u svojoj aktivnosti. Za potrebe aktivnosti koja služi za prijenos fotografija sa uređaja biti će korišteno četiri elementa. Prva dva elementa su tipke kojima korisnik može odabrati fotografiju i prenijeti je na web stranicu. Druga dva elementa su „ImageView“ koji imaju mogućnost postavljanja svoje vrijednosti na vrijednost neke varijable ili datoteke. Prvi „ImageView“ će biti postavljen da ukazuje na logo same aplikacije, dok će drugi „ImageView“ biti podešen na sliku koja korisnika obavještava da nije odabrao fotografiju koju želi prenijeti na web stranicu a nakon što odabere sliku „ImageView“ će se postaviti na odabranu fotografiju.

Izgled za određenu aktivnost može biti pisana u XML kodu ili se može koristiti uređivač razvojnog okruženja Eclipse koji predstavlja brz način izrade izgleda ali je pisanje koda u XML-u puno bolje rješenje. Na slici 8 je prikazan izrađeni izgled za aktivnost koja služi za prijenos fotografija na web stranicu.



Slika 8: Izgled aktivnosti za prijenos fotografija

Izvor: Autor

Kako bi svaki od upotrjebljenih elemenata bilo moguće koristiti u Java klasi u kojoj će se pozvati ova XML datoteka svakom od elemenata treba pridodati svoj ID preko kojeg ćemo u Java klasi prepoznavati i pozivati određeni element. Dodjeljivanje ID-a se vrši kod svakog od elemenata te se isti dodaje tako da se unutar prostora elementa doda sljedeći dio koda:

```
android:id="@+id/zeljeni_id"
```

Dobra je praksa da svaki od elemenata započinje kraticom tog istog elementa. Tako se primjerice za tipku (engl. Button) ID započinje slovom „b“ pa bi ID za tipku koja služi za odabir fotografije mogao koristiti „bOdabirF“ što u konačnici daje ukupan kod za tu tipku:

```
<Button
          android:id="@+id/bOdabirF"
          android:layout_width="match_parent"
          android:layout_height="60dp"
          android:text="Odaberi fotografiju" />
```

Nakon što se u cijeloj XML datoteci svim elementima dodjeli ID, sveukupan kod za izgled aktivnosti koja služi za prijenos fotografije na web stranicu izgleda ovako:

```
<?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:background="@drawable/bg"
     android:orientation="vertical" >

     <ImageView
         android:id="@+id/ivLogo"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:src="@drawable/logo_app" />

     <ImageView
         android:id="@+id/ivSlika"
         android:layout_width="match_parent"
         android:layout_height="190dp"
         android:layout_marginBottom="5dp"
         android:src="@drawable/nothing" />

     <Button
         android:id="@+id/bOdabirF"
         android:layout_width="match_parent"
         android:layout_height="60dp"
         android:text="Odaberi fotografiju" />

     <Button
         android:id="@+id/bPrijenosF"
         android:layout_width="match_parent"
         android:layout_height="60dp"
         android:text="Prenesi fotografiju" />

  </LinearLayout>
```

Iz priloženog se koda vidi da je XML jezik prilično jednostavan te razvojnom programeru omogućava da preciznije podesi vrijednosti za poziciju i veličinu pojedinih elemenata određene aktivnosti. Zbog toga je često pisanje XML koda puno efikasnije u odnosu na dostupni vizualni uređivač.

4.1.1 Java skripta za prijenos datoteka

Kako bi se Android aplikaciji omogućilo da prenese odabrane fotografije na web stranicu korištena je skripta za prijenos datoteka na web server. Skripta je pisana u Java programskom jeziku što omogućava prenosivost među raznim platformama te je zato tu skriptu moguće koristit i za Android aplikaciju.

```
//Početak skripte
      {
              String fileName = sourceFileUri;
          HttpURLConnection conn = null;
          DataOutputStream dos = null;
          String twoHyphens = "--";
          String lineEnd = "\r\n";
          String boundary = "*****";
          int bytesRead, bytesAvailable, bufferSize;
          byte[] buffer;
          int maxBufferSize = 1 * 1024 * 1024;
          File sourceFile = new File(sourceFileUri);

          //Početak komunikacije sa web stranicom
          {
                 try {
                     FileInputStream fileInputStream = new
FileInputStream(sourceFile);
                     URL url = new URL(serverUrl);   // Postavljanje URL-
a web stranice
                     conn = (HttpURLConnection) url.openConnection();
                     conn.setDoInput(true);
                     conn.setDoOutput(true);
                     conn.setUseCaches(false);
                     conn.setRequestMethod("POST");
                     conn.setRequestProperty("Connection", "Keep-Alive");
                     conn.setRequestProperty("ENCTYPE", "multipart/form-
data");
                     conn.setRequestProperty("Content-Type",
"multipart/form-data;boundary=" + boundary);
                     conn.setRequestProperty("uploaded_file", fileName);

                     dos = new DataOutputStream(conn.getOutputStream());

                     dos.writeBytes(twoHyphens + boundary + lineEnd);
```

```
dos.writeBytes("Content-Disposition: form-data;
name=\"uploaded_file\";filename=\""
                                                    + fileName + "\"" +
lineEnd);

                    dos.writeBytes(lineEnd);

                    bytesAvailable = fileInputStream.available();

                    bufferSize = Math.min(bytesAvailable,
maxBufferSize);
                    buffer = new byte[bufferSize];

                    bytesRead = fileInputStream.read(buffer, 0,
bufferSize);

                    while (bytesRead > 0)
                    {
                      dos.write(buffer, 0, bufferSize);
                      bytesAvailable = fileInputStream.available();
                      bufferSize = Math.min(bytesAvailable,
maxBufferSize);
                      bytesRead = fileInputStream.read(buffer, 0,
bufferSize);
                     }

                    dos.writeBytes(lineEnd);
                    dos.writeBytes(twoHyphens + boundary + twoHyphens +
lineEnd);
                    //Serverov odgovor (kod i poruka)
                    serverOdgovor = conn.getResponseCode();
                    String serverResponseMessage =
conn.getResponseMessage();

                    Log.i("uploadFile", "HTTP Response is : "
                                + serverResponseMessage + ": " +
serverOdgovor);
                    //Provjerava uspješnost prijenosa
                    if(serverOdgovor == 200)
                    {
                          runOnUiThread(new Runnable()
                              {
                              public void run()
                              {
                                    Toast.makeText(Prenesi.this, "Prijenos
uspješno dovršen", Toast.LENGTH_SHORT).show();
                              }
                             });
                    }

                    fileInputStream.close();
                    dos.flush();
                    dos.close();
                }
                //Zapisivanje grešaka u logcat
                 catch (Exception e) {
```

```
runOnUiThread(new Runnable() {
                        public void run()
                        {
                          Toast.makeText(Prenesi.this, "Iznimka : Pogledaj
LogCat ", Toast.LENGTH_SHORT).show();
                        }
                    });
                    Log.e("Iznimka prilikom prijenosa slike na server",
"Exception : "  + e.getMessage(), e);
                }
                dialog.dismiss();
                return serverOdgovor;
            }
        } // Kraj
```

Skripta je okružena „try and catch“ metodom koja u slučaju da se neki dio iz metode „try“ ne izvrši odnosno da dođe do te iznimke lovi (engl. Catch) tu iznimku. U našem slučaju, ukoliko dođe do neke iznimke ista se zapisuje u LogCat. Zapisivanje iznimaka je posebno korisno kod ispravljanja grešaka prilikom testiranja aplikacije. Također je dobro i da se preko „toast“ funkcije dobije obavijest da je došlo do iznimke te upućuje razvojnog programera da pogleda LogCat.

4.1.1.1 HTTP protokol

HTTP (engl. Hypertext Transfer Protocol) je protokol aplikacijskog sloja za raspodijeljene, surađujuće, hipermedijske informacijske sustave. Njegovo korištenje pri ponovnom pronalaženju međusobno povezanih sredstava dovelo je do nastanka WWW-a. Kao što je to uobičajeno u odnosu klijent – poslužitelj, HTTP se sastoji od upita i odgovora. Klijenta predstavlja aplikacija koju koristi krajnji korisnik (npr. preglednik), a poslužitelj je aplikacija koja se izvodi na računalu na kojem se nalazi web stranica.

HTTP definira osam metoda od kojih se najviše koriste GET i POST. GET traži prikaz nekog izvora i ne izaziva nikakve popratne pojave, dok POST predaje podatke koje treba procesuirati nekom izvoru. Android omogućuje jednostavno korištenje ovih metoda.

HTTP protokolom definira se:

- Forma komunikacije između klijenta i poslužitelja
- Kodiranje znakova
- Kodiranje sadržaja
- Pristup dokumentima
- Pohrana dokumenata
- Sigurnosne aspekte

Prilikom HTTP zahtjeva (engl. request) šalje se odgovor (engl. response) sa poslužitelja, kao što je "200 OK" - nakon čega će server poslati i svoj paket podataka koji najčešće sadrži traženu datoteku ili poruku o grešci. Odmah po ispunjenju zahtjeva klijenta, server će prekinuti komunikaciju.

Sredstva kojima HTTP može pristupiti jednoznačno su označena URI[6]- jima ili URL[7]- ima te se to koristi i u svrhu ovog diplomskog rada

4.1.1.2 PHP skripta

Kako bi Java skripta funkcionirala neophodno je korištenje skripte na web stranici koju pokreće sama Java skripta odnosno naša Android aplikacija. Takva skripta, ovisno o potrebama može biti pisana u različitim jezicima. Tako može biti korištena ASP[8], JSP[9] ili PHP[10] skripta koja prihvaća i obrađuje poslane zahtjeve. U svrhu ovog diplomskog rada biti će korištena PHP skripta na koju se poziva Java klasa same Android aplikacije.

[6] Uniform Resource Identifier

[7] Uniform Resource Locator

[8] Active Server Pages

[9] Java Server Pages

[10] Hypertext Preprocessor

```php
<?php

    $file_path = "preneseno/";

    $file_path = $file_path . basename( $_FILES['uploaded_file']['name']);
    if(move_uploaded_file($_FILES['uploaded_file']['tmp_name'],
$file_path)) {
        echo "Prijenos uspješno dovršen";
    } else{
        echo "Došlo je do pogreške";
    }
 ?>
```

Kod predstavlja korištenu PHP skriptu. Ona pri pokretanju postavlja svoj radni direktorij koji je u ovom slučaju prazan direktorij „preneseno“ na web stranici. Skripta između ostaloga provjerava da li je datoteka koja se šalje važeća, odnosno je li prenesena preko PHP-ovog HTTP POST-a. Ako je sa datotekom sve u redu ista će biti premještena na danu destinaciju.

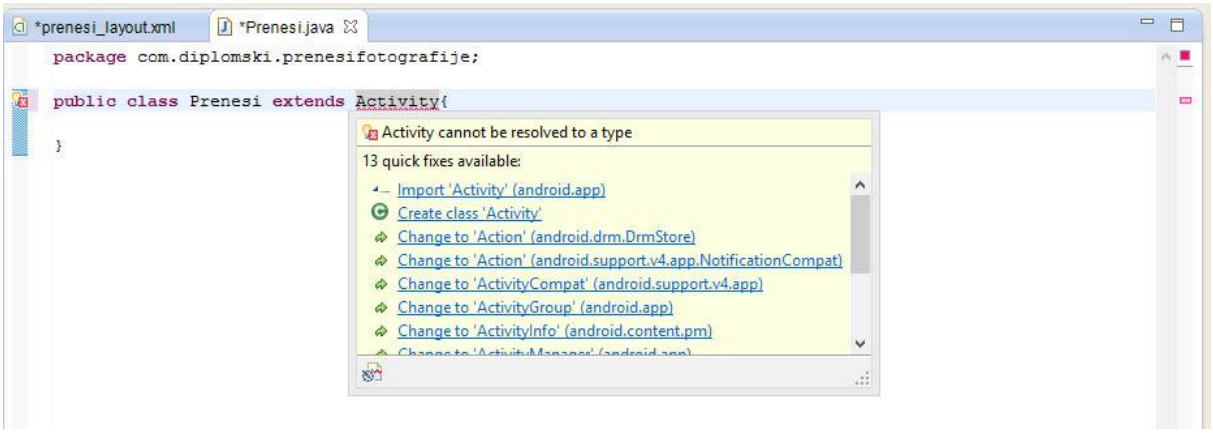
4.1.2 Izrada Java klase za prijenos fotografija

Kako bi Java skripta za prijenos podataka, u našem slučaju fotografija mogla raditi potrebno ju je implementirati u Android aplikaciju. Da bi se to ostvarilo biti će napravljena nova Java klasa koja će kao izgled aktivnosti aplikacije koristiti prethodno napravljenu „prenesi_layout.xml“ datoteku. Prilikom kreiranja nove Java klase dobivamo sljedeći kod:

```java
package com.diplomski.prenesifotografije;

public class Prenesi {

}
```

Prethodni kod je potrebno proširiti tako da može biti korišten kao aktivnost neke aplikacije. To činimo jednostavnim proširivanjem klase „Prenesi“ pomoću funkcije

„Activity“ koju treba uvesti unutar klase kako bi se ona mogla koristiti. Na slici 9 je prikazano kako razvojno okruženje Eclipse samo prepoznaje da funkcija „Activity“ nije prepoznata te nam kao rješenje daje mogućnost dodavanje funkcije „Activity“ u Java klasu.



Slika 9: Pomaganje razvojnog okruženja Eclipse pri pisanju koda

Izvor: Autor

Nakon proširenja klase sa funkcijom aktivnosti te nakon dodavanja funkcije „Activity“ u klasu dobivamo sljedeći kod:

```
package com.diplomski.prenesifotografije;
import android.app.Activity;
public class Prenesi extends Activity{
}
```

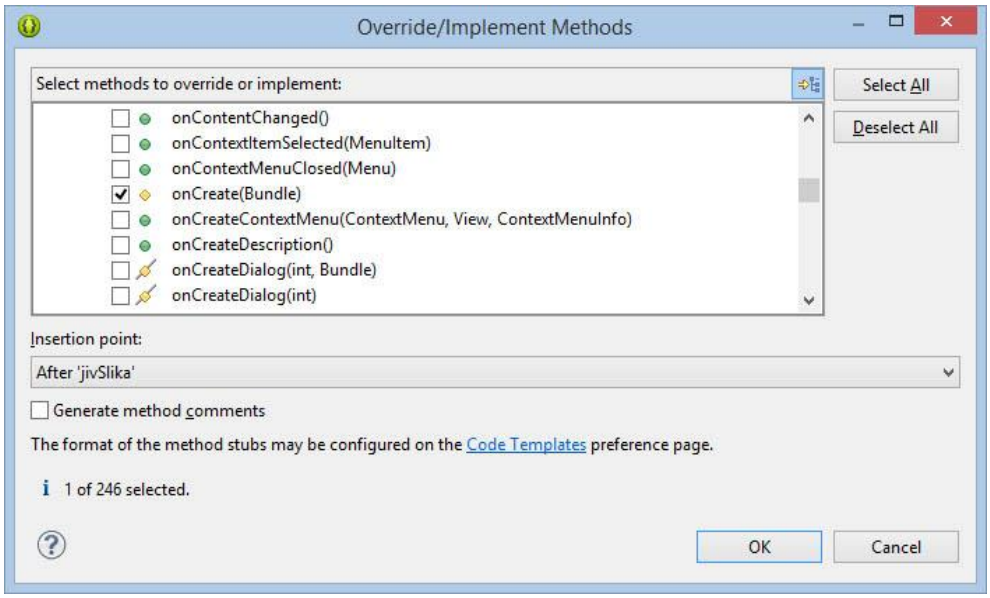
4.1.2.1 Definiranje varijabli

Sada je Java klasa postala aktivnost koju možemo koristiti kako bi prenesi fotografije na Internet. To je prvi i osnovi korak kod izrade aktivnosti ali ipak ovo nije dovoljno. Također je potrebno definirati varijable koje će biti korištene. Definiranje varijabli je vrlo jednostavno i izvodi se na način kao i kod drugih jezika. Najprije je potrebno definirati tip varijable a zatim i samu varijablu. Za našu aplikaciju u planu je

korištenje dviju tipki; Tipke za odabir fotografije i jednu za prijenos te fotografije na Internet. Također biti će korišten i element za prikaz fotografije koji će na početku biti postavljen na početnu fotografiju koju mi odredimo a kasnije nakon što pritiskom na tipku za odabir fotografije odaberemo neku od fotografija sa našeg uređaja element će biti postavljen na tu fotografiju. Kao i kod pisanja ID-a kod kreiranja izgleda pomoću XML datoteke dobro je započeti ime varijable sa kraticom tog elementa. Primjerice, ako je kod XML datoteke tipka za prijenos fotografije imala ID „bPrenesiF“ isti može biti korišten kod Java klase kao ime varijable. To možda i nije najbolje rješenje jer će kasnije ta Java varijabla biti povezana sa varijablom iz XML datoteke te je preglednije kada svaki element, u našem slučaju tipka ima svoje, različito ime. Tako će u ovom radu svi elementi koji budu definirani u Java klasi započinjati sa slovom „j“ što će označavati da se radi upravo o Java varijabli. Sljedeći kod prikazuje definiranje varijabli za elemente tipa „Button“ i „ImageView“:

```
package com.diplomski.prenesifotografije;

import android.app.Activity;
import android.widget.Button;
import android.widget.ImageView;

public class Prenesi extends Activity{

	//Definiranje varijabli

	Button jbOdabirF, jbPrijenosF;
	ImageView jivSlika;

}
```

Nakon definiranja varijabli potrebno je u Java klasi pozvati se na XML datoteku u kojoj se nalazi izgled za određenu aktivnost. To je u našem slučaju datoteka „prenesi_layout.xml“. Kako bi Java klasa znala koju XML datoteku koristiti kao zadani izgled treba joj to dodati u kodu. To se izvodi preko funkcije „setContentView“ koja se poziva u okviru izvođenja metode „onCreate“ koju je potrebno dodati unutar aktivnosti Java klase. Kako bi dodali novu „onCreate“ metodu potrebno je u radni prostor desnom tipkom miša kliknuti i odabrati „Source“ pa „Override/Implement Methods“. Nakon toga nam se otvara prozor iz kojega možemo birati koje od metoda želimo koristiti kao što je prikazano na slici 10.



Slika 10: Odabir metoda za dodavanje u Java klasu

Izvor: Autor

Nakon što odaberemo „onCreate(Bundle)“ metodu dobijemo mogućnost dodavanja naše XML datoteke Java klasi pomoću funkcije „setContentView“ odnosno u našem slučaju dodavanjem sljedećeg dijela koda:

```
setContentView(R.layout.prenesi_layout);
```

Tek sada, nakon dodavanja XML datoteke možemo povezivati elemente iz XML datoteke sa Java varijablama, odnosno Javi reći da tipke napravljene za odabir ili prijenos fotografije. Potrebno je navesti prethodno definiranu varijablu te je postaviti na vrijednost varijable iz XML datoteke kao što je prikazano kodom:

```
// Povezivanje varijabli sa XML varijablama

jbOdabirF = (Button) findViewById(R.id.bOdabirF);
jbPrijenosF = (Button) findViewById(R.id.bPrijenosF);
jivSlika = (ImageView) findViewById(R.id.ivSlika);
```

4.1.3 Implementacija Java skripte za prijenos datoteka

Kako bi Java skripta za prijenos datoteka radila sa aktivnošću Android aplikacije potrebno je izvršiti implementaciju koda u aktivnost. To znači povezivanje svih njenih varijabli sa varijablama aktivnosti kako bi mogla funkcionirati. Pod implementacijom podrazumijevamo definiranje pojedinih tipki odnosno što će koja tipka raditi sa Java skriptom. Također je unijeti sve metode koje skripta koristi. Na početku je potrebno definirati sve varijable koje se javljaju u skripti a nema ih definiranih u trenutnoj aktivnosti.

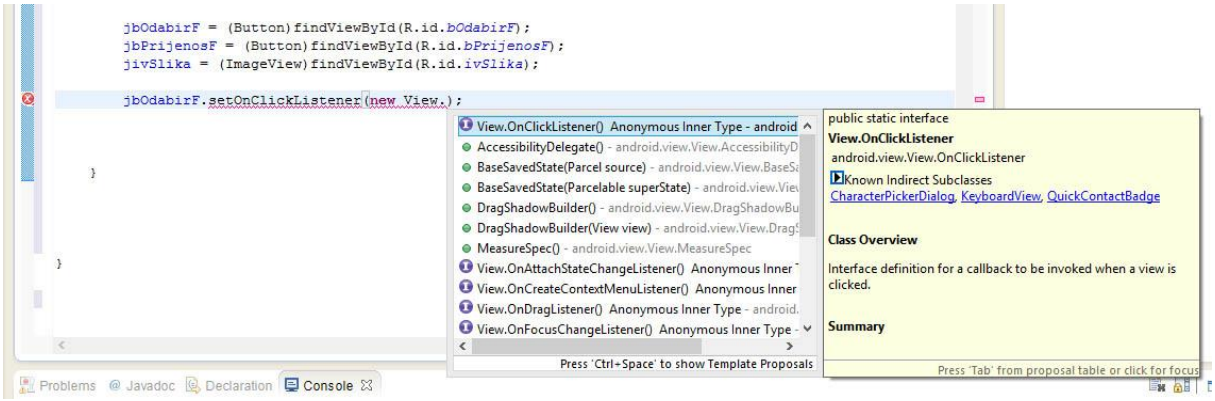
```
//Varijable potrebne za java skriptu

int serverOdgovor = 0;
String serverUrl =
"http://freehost.filesblue.com/prijenos_skripta.php";
String putanja = null;
ProgressDialog dialog;
```

Prva varijabla je cjelobrojna varijabla tipa „int“ koja je postavljena na nulu a koristi se kao varijabla u kojoj će biti zapisan odgovor sa web stranice na koju šaljemo fotografije. Također je potrebno definirati na kojoj se stranici nalazi PHP skripta koja je zadužena za spremanje slika na web stranicu. Ta je varijabla kao i varijabla koja u sebi sadrži putanju fotografije koju korisnik bira tipa „String“ što znači da je njena vrijednost u obliku teksta. Također je i definirana varijabla „dialog“ koja je tipa „ProgresDialog“. Ona se koristi kako bi se korisniku na ekranu uređaja prikazivala poruke sa neodređenim vremenskim periodom trajanja poznate kao „Toast“. Često se koriste kako bi se korisnika obavijestilo o nečemu pa će upravo i u tu svrhu biti korištene u okviru ove Android aplikacije.

4.1.3.1 Dodjeljivanje funkcije tipkama

Kako bi tipke od kojih se sastoji aktivnost bile funkcionalne treba im omogućiti da se prilikom pritiska na njih nešto dešava. To zahtjeva dodavanje nove metode nazvane „onClickListener“ koji se nalazi u grupi metoda „View“ i koji prati koja se tipka pritisnula te definira što određena tipka radi. Kako bi dodali metodu „onClickListener“ koristimo

metodu „setOnClickListener“ koju postavljamo za svaku tipku posebno. Prilikom pisanja koda razvojno okruženje Eclipse pomaže pri odabiru tako da je dovoljno kliknuti na ponuđeni odgovor te Eclipse automatski dopiše ono što smo mislili napisati. Na slici je pokazano kako nam Eclipse pomaže sa dodavanjem metode „onClickListener“.



Slika 11: Predlaganje nastavka koda

Izvor: Autor

Nakon odabira metode „View.OnClickListener()“ Eclipse sam dovršava pisanje koda te otvara novu „onClick“ metodu unutar koje određujemo što zapravo ta tipka pokreće. U konačnici kod metode „onClickListener “ za tipku preko koje korisnik odabire sliku izgleda ovako:

```
jbOdabirF.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                          // TODO Auto-generated method stub
                          //Kod koji mislimo izvšiti
                    }
              });
```

Ukoliko imamo više tipki za svaku je tipku potrebno ponoviti navedeni postupak što u konačnici znači puno istog koda u kojem je jedina razlika kod koji će se izvršiti. Zbog toga je dobro imati samo jednu „onClick“ metodu koja bi provjeravala koja je tipka pritisnuta te na osnovu toga izvodila određeni kod. Kako bi to bilo moguće najprije je

potrebno implementirati „onClickListener“ u aktivnost koju smo prethodno proširili. Implementacija se vrši preko sljedećeg koda.

```
public class Prenesi extends Activity implements OnClickListener {
```

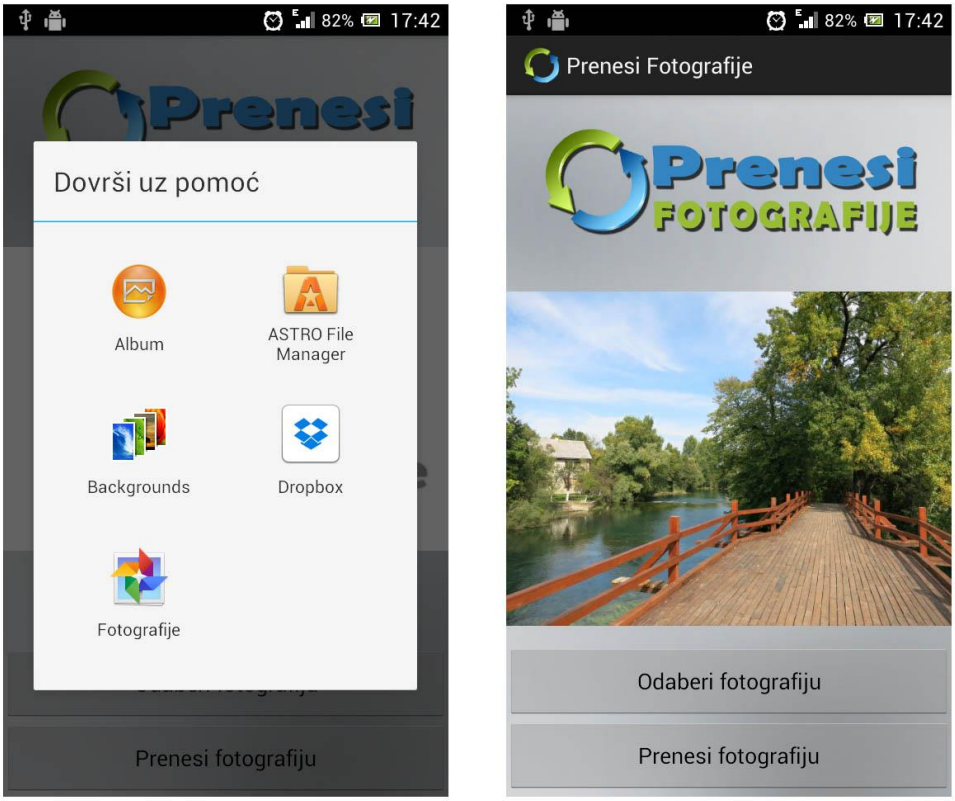
Nakon toga dovoljno je tipku postaviti da radi kao „onClickListener“ te u „onClick“ metodi napraviti jednostavnu „if – else“ petlju koja provjerava koja je od tipki pritisnuta. Ukoliko je pritisnuta tipka za odabir fotografije biti će pokrenut kod za odabir fotografije a ukoliko bude to tipka za prijenos fotografije izvoditi će se kod za prijenos. U konačnici kod je puno pregledniji, jednostavniji i kraći što doprinosi brzini same aplikacije.

```
jbOdabirF.setOnClickListener(this);
jbPrijenosF.setOnClickListener(this);

	}

	@Override
	public void onClick(View tipka) {
		// TODO Auto-generated method stub
		if(tipka==jbOdabirF)
		{
			//Izvrši kod 1
		}
		else if (tipka==jbPrijenosF)
		{
			//Izvrši kod 2
		}
	}
```

Sada je potrebno funkciju „onClick“ povezati sa Java skriptom za prijenos podataka. Treba odrediti koje što čini koja tipka prilikom pritiska na nju. Aplikacija je zamišljena da ukoliko korisnik pritisne tipku „Odaberi fotografiju“ korisnik dobiva mogućnost odabira programa preko kojeg želi odabrati željenu fotografiju te nakon što je odabere da se ta odabrana fotografija postavi na ekran kako bi korisnik mogao vidjeti koju je fotografiju odabrao. Kako bi korisnik mogao odabrati fotografiju Android aplikacija mora to smatrati kao novu namjeru (engl. Intent). Intent služi kako bi aplikacija mogla obavljati neke jednostavne namjere poput pokazivanja neke poruke korisniku ili

jednostavno izvršiti neku akciju. U našem slučaju će pokretati izbornik koji korisniku daje na izbor koji od programa želi koristiti za odabir fotografije.

Nova namjera, odnosno Intent će biti nazvan „odabir slike“. Kako bi aplikacija znala korisniku ponuditi programe kojima je moguće otvoriti fotografije treba to definirati preko funkcije „setType“ te kao tip odabrati „image“ odnosno slike. Također je potrebno definirati da se preko funkcije „startActivityForResult“ pojavi novi prozor koji korisniku daje mogućnost odabira programa za pregled slika kao što je prikazano slikom.



Slika 12: a) Prikaz prozora za odabir programa za pretraživanje fotografija b) Prikazivanje odabrane fotografije

Izvor: Autor

```
@Override
      public void onClick(View tipka) {
            if(tipka==jbOdabirF)
            {
            Intent odabirSlike = new Intent();
            odabirSlike.setType("image/*");
            odabirSlike.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(odabirSlike,
"Dovrši uz pomoć"), 1);
            }
            else if (tipka==jbPrijenosF)
            {
                  //Izvrši kod 2
            }
      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (requestCode == 1 && resultCode == RESULT_OK)
        {
        Uri uriOdabraneSlike = data.getData();
        putanja = getPath(uriOdabraneSlike);
        Bitmap bitmap = BitmapFactory.decodeFile(putanja);
        jivSlika.setImageBitmap(bitmap);
        }
      }
```

Također je potrebno definirati i novu funkciju „onActivityResult“ koja postavlja putanju odabrane slike na prethodno definiranu varijablu „putanja“ te odabranu fotografiju postavlja kao na varijablu „jivSlika“ kako bi korisnik vidio koju je fotografiju odabrao što je prikazano na slici 12 b).

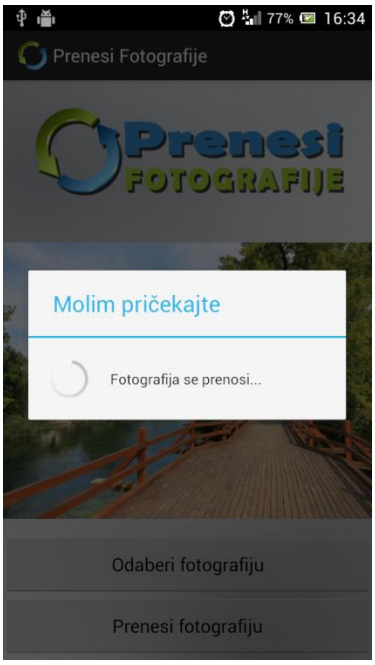
Nakon što je prva tipka koja korisniku dozvoljava da odabere fotografiju koju želi prenijeti na web stranicu u potpunosti funkcionalna, odnosno dozvoljava korisniku da bira program sa kojim želi pregledati fotografije te nakon što odabere tu fotografiju postavlja u aktivnost kako bi korisnik vidio koju je fotografiju odabrao, potrebno je osposobiti i drugu tipku. Druga tipka pokreće Java skriptu za prijenos podataka na web stranicu te je to prikazano sljedećim kodom.

```
else if (tipka==jbPrijenosF)
            {
                  dialog = ProgressDialog.show(Prenesi.this, "Molim
pričekajte", "Fotografija se prenosi...");
                   new Thread(new Runnable() {
                public void run()
                {
                     uploadFile(putanja);
                }
              }).start();

            }
```

To se provjerava preko „if“ skripte, odnosno provjerava se koja je tipka pritisnuta. Najprije je potrebno pokrenuti funkciju „ProgressDialog“ koja korisnika obavještava o trenutnom stanju aplikacije. U našem slučaju je to poruka da se fotografija prenosi. Tek nakon toga pokreće se sama skripta preko funkcije „Thread“ koja pokreće dio koda koji je nazvan „uploadFile“ što predstavlja našu Java skriptu za prijenos datoteka. Kako bi osigurali da „ProgresDialog“ ostane vidljiv korisniku samo dok je prijenos u toku dodan je sljedeći kod na kraju skripte za prijenos:

```
dialog.dismiss();
```

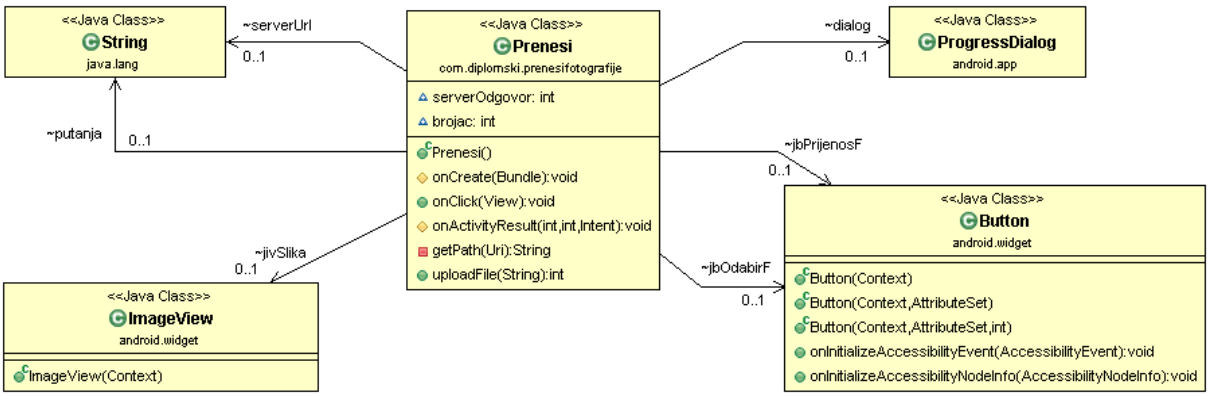
Primjer za funkciju „ProgresDialog“ je prikazan na idućoj slici gdje se korisnika obavještava da se fotografija prenosi. Ta se obavijest, nakon što je fotografija prenesena sama miče sa ekrana.



Slika 13: Obavijest o prijenosu fotografije

Izvor: Autor

Nakon što smo definirali sve varijable koje će koristiti aktivnost za prijenos fotografija na web stranicu, odnosno aktivnost „Prenesi“ možemo je prikazati klasnim dijagramom kao što je prikazano slikom 14. Iz njega možemo vidjeti koje sve funkcije koristi navedena aktivnost. Za prikaz slike se koristi „ImageView“, za tipke „Button“, funkcija „String“ se koristi za odabir putanje fotografije te URL web stranice na koju se prenosi. Za prikaz obavijesti o prijenosu koristi se funkcija „ProgressDialog“.



Slika 14: Klasni dijagram aktivnosti "Prenesi"

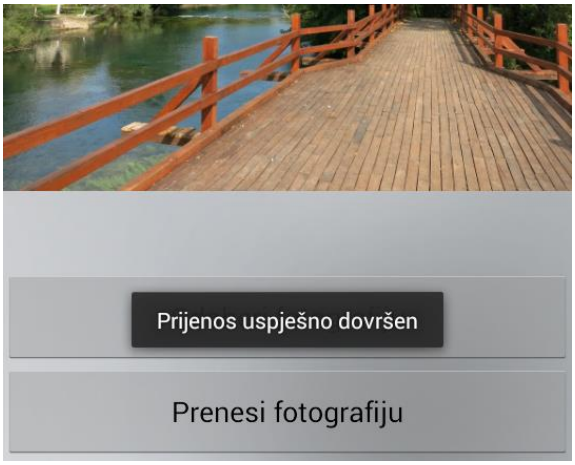
Izvor: Autor

4.1.3.2 *Funkcija „toast“*

Funkcija „toast“ služi za prikazivanje poruke na ekranu na taj način da se prikaže samo tekst koji razvojni programer želi i da se ta poruka nakon nekog određenog vremenskog perioda makne sa ekrana. Java skripta za prijenos koristi dvije „toast“ funkcije koje korisnika obavještavaju o radu aplikacije. Prva „toast“ funkcija se koristi da korisnika obavijesti da je sve u redu sa prijenosom slike na server. Kako se za prijenos koristi HTTP standard obavijest o uspješnom prijenosu se provjerava odgovorom servera koji mora biti jednak 200.

```
//Provjerava uspješnost prijenosa
                    if(serverResponseCode == 200)
                    {
                          runOnUiThread(new Runnable()
                              {
                              public void run()
                              {
                                    Toast.makeText(Prijenos.this,
"Prijenos uspješno dovršen", Toast.LENGTH_SHORT).show();
                              }
                          });
                    }
```

Ukoliko je varijabla „serverResponseCode“ jednaka 200 tada se pojavljuje „toast“ poruka koja korisnika obavještava da je prijenos uspješno dovršen kao što je prikazano slikom.



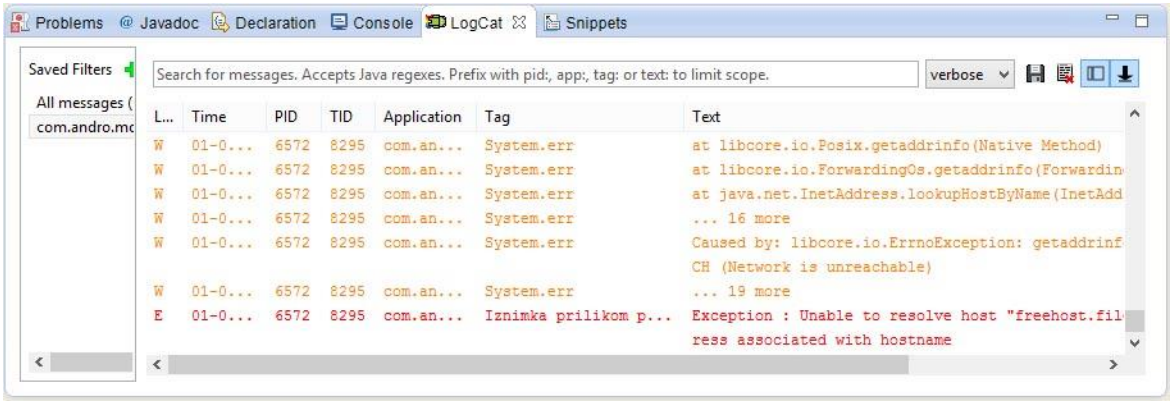
Slika 15: Obavijest o uspješno dovršenom prijenosu

Izvor: Autor

S obzirom da je Java skripta za prijenos datoteka okružena „try and catch“ funkcijom koja provjerava je li kod iz sekcije „try“ uspješno izvršen, u „catch“ dijelu se vrši ispis toast poruke ukoliko dođe do iznimke a i sam zapis u LogCat

```
try {
       .
       .
       .
          }

//Zapisivanje grešaka u LogCat
catch (Exception e) {

                    dialog.dismiss();
                    e.printStackTrace();

                    runOnUiThread(new Runnable() {
                        public void run()
                        {
                          Toast.makeText(Prijenos.this, "Iznimka :
Pogledaj LogCat ", Toast.LENGTH_SHORT).show();
                        }
                    });
                    Log.e("Iznimka prilikom prijenosa slike na server",
"Exception : "  + e.getMessage(), e);
                    }
```

U ovom slučaju fotografija je probana poslati na web stranicu bez uključenog Interneta na samome uređaju te je zbog toga došlo do pogreške pri prijenosu. Ta je greška kao što je i korisnik bio obaviješten zapisana u LogCat-u što razvojnom programeru olakšava pronalaženje greški , odnosno rješavanje problema zbog kojeg je i došlo do iznimke.



Slika 16: LogCat

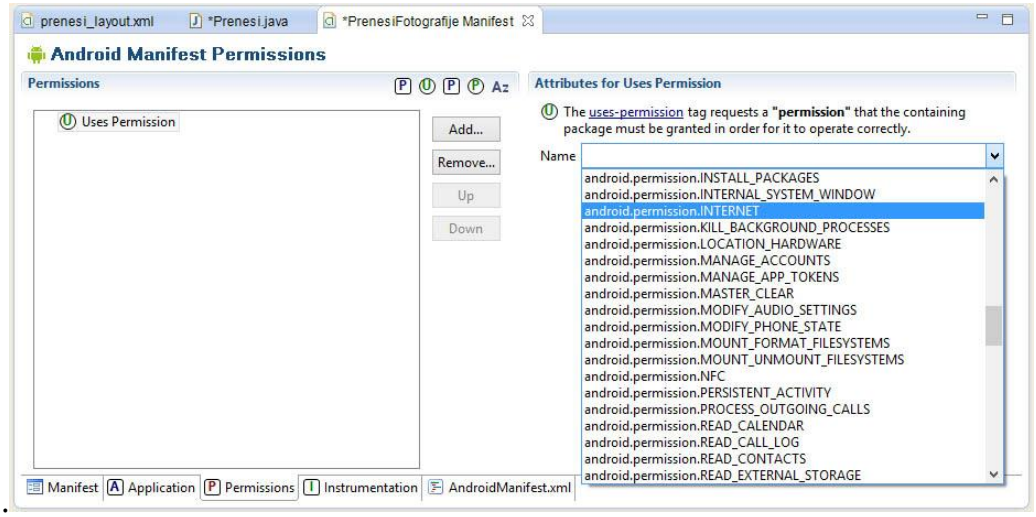
Izvor: Autor

4.1.4 Deklariranje dozvola u Manifest datoteci

Kako bi Android operacijski sustav znao koja sve prava treba dodijeliti određenoj Aplikaciji prilikom razvoja iste potrebno je u Manifest datoteci deklarirati dozvole koje funkcije treba koristiti. U našem slučaju, kada se radi o aplikaciji koja prenosi fotografije na web stranicu potrebno je za aplikaciju odobriti korištenje Interneta. To je moguće preko dva načina. Korištenjem grafičkog sučelja razvojnog okruženja Eclipse ili dodavanjem dijela koda u datoteku „AndroidManifest.xml“. Ukoliko se odlučimo na manualno uređivanje XML datoteke dovoljno je nakon pozivanja funkcije „uses-sdk“ dodati sljedeći kod u datoteku kako bi se aplikaciji dala dozvola.

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

Ukoliko se odabere dodavanje dozvole preko korisničkog sučelja stvar je nešto jednostavnija. Naime u pregledu datoteke „AndroidManifest.xml“ potrebno je odabrati „Permissions“ što označava dozvole te je potrebno dodat novu dozvolu preko tipke „Uses Permission“. Nakon toga se iz izbornika odabire dozvola koju želimo dodati u „AndroidManifest.xml“ te se tada prethodni kod automatski nadodaje na postojeći XML kod. Način dodavanja dozvola preko korisničkog sučelja programa Eclipse prikazan je na slici 17



Slika 17: Dodavanje dozvola u manifest datoteku

Izvor: Autor

4.2 Aktivnost za pregled prenesenih fotografija

Kako bi korisnik znao koje je sve fotografije prenesao na web stranicu biti će izrađena posebna aktivnost aplikacije koja će služiti samo za pregled prenesenih fotografija. Kako bi se to najjednostavnije napravilo biti će korištena PHP skripta u kombinaciji sa HTML-om koja će na samoj web stranici pregledno prikazivati fotografije kako bi se olakšao prikaz tih istih fotografija na Android uređaju. Za Android aplikaciju biti će napravljena aktivnost koja koristi element „WebView“ preko kojeg će se pregledavati prenesene fotografije.

4.2.1 Izrada web stranice

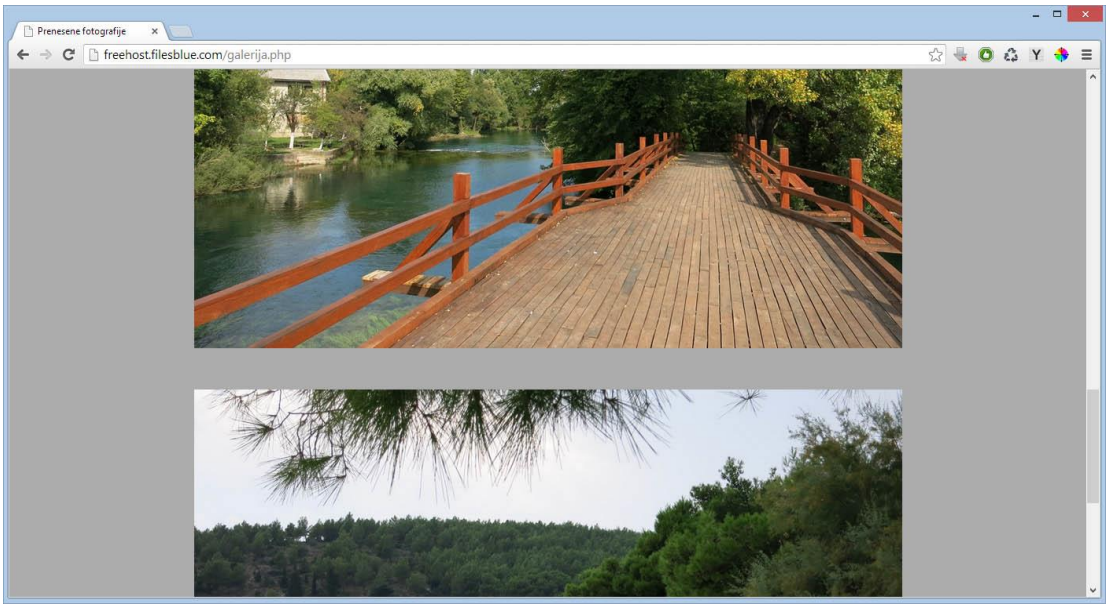
Da bi se pojednostavio posao kod izrade Android aplikacije, prikaz slika na samoj web stranici biti će napravljen što je jednostavnije moguće, odnosno prikaz svih slika vertikalno raspoređenih na web stranici sa po jednom fotografijom u redu. Najprije kako bi bio uopće omogućen prikaz prenesenih fotografija potrebno je korištenje PHP skripte koja se koristi kako bi se fotografije smještene u direktoriju „preneseno“ prikazale na web stranici. Skripta je napravljena tako da pregledava cijeli direktorij preko „for“ petlje te da prikazuje fotografije na web stranici u tabličnom obliku; svaka fotografija u svome redu, čime se dobiva vertikalni pregled svih fotografija. Sljedeći kod prikazuje PHP skriptu koja je korištena za prikaz fotografija na web stranici.

```
<?php

$folder = 'preneseno/';
$filetype = '*.*';
$files = glob($folder.$filetype);

echo '<table>';
for ($i=0; $i<count($files); $i++)
      {
    echo '<tr><td>';
    echo '<img src="'.$files[$i].'" />';
    echo '</td></tr>';
      }
echo '</table>';

?>
```

Sama PHP skripta je dovoljna za prikazivanje fotografija u tabličnom obliku ali se uz korištenje HTML-a i CSS-a može dobiti pregledniji i ljepši pregled. Tako se primjerice može fotografije poravnati na sredinu, odrediti im margine, širinu i slično. U tu svrhu je korišten sljedeći kod:

```
<html>
<head>
<title>Prenesene fotografije</title>
<style type="text/css">
body    {
        margin: 0 auto 20px;
        padding: 0;
        background: #acacac;
        }
td      {
        padding: 0 0 20px;
        }
table   {
        width: 100%;
        }
img     {
        display: block;
        margin: 20px auto 10px;
        max-width: 900px;
        outline: none;
        }
</style>
</head>
<body>
<?php
         //PHP Skripta
?>
</body>
</html>
```

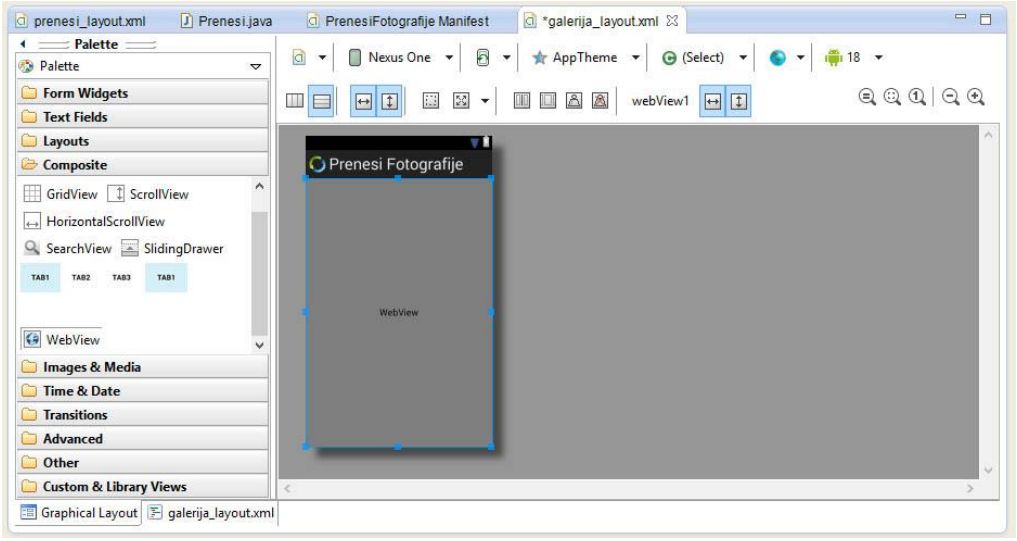
Prethodnim se kodom stranici daje naslov „Prenesene fotografije“ te se preko CSS-a određuje koje će boje biti pozadina, koliki će biti razmak između pojedinih redova tablice, odnosno u našem slučaju razmak između prenesenih fotografija. Također je određena širina fotografija koje se prikazuju na web stranici. Nakon korištenja ovog koda dobivamo sljedeći izgled stranice koja je prikazana slikom 18.



Slika 18: Izgled web stranice
Izvor: Autor

4.2.2 Izgled aktivnosti za pregled fotografija

Najprije je potrebno napraviti novi XML dokument za novu aktivnost Android aplikacije. To se kao i kod prethodnih aktivnosti može učiniti na dva načina; putem korisničkog sučelja programa Eclipse ili preko XML koda. Prvi način je puno jednostavniji te zahtjeva od korisnika da povuče „WebView“ element u radni prostor kao što je prikazano slikom 19.



Slika 19: Izgled aktivnosti za pregled prenesenih fotografija

Izvor: Autor

Isti je rezultat mogao biti ostvaren i pisanjem XML koda što predstavlja nešto kompliciranije rješenje jer razvojno okruženje Eclipse generira isti kod prilikom korištenja grafičkog sučelja. U konačnici kod datoteke „galerija_layout.xml“ izgleda ovako:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <WebView
        android:id="@+id/webView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

4.2.3 Izrada Java klase za pregled fotografija

Kao i kod prethodne aktivnosti Android aplikacije kreiranje XML datoteke koja opisuje izgled same aktivnosti nije dovoljna za njeno samostalno funkcioniranje već je potrebno i kreiranje Java klase za pregled fotografija. Ta će Java klasa biti povezana sa XML datotekom koja opisuje njen izgled, u našem slučaju „galerija_layout.xml“. Nakon kreiranja nove Java klase najprije je, kao i kod prethodne aktivnosti, potrebno ju proširiti sa funkcijom „extends Activity“ kako bi se naša Java klasa mogla koristiti kao aktivnost. Također je preko „onCreate“ metode potrebno povezati Java klasu sa XML datotekom koja je zadužena za izgled. Povezivanje se vrši kao i kod prethodne aktivnosti uz promjenu datoteke koja se poziva.

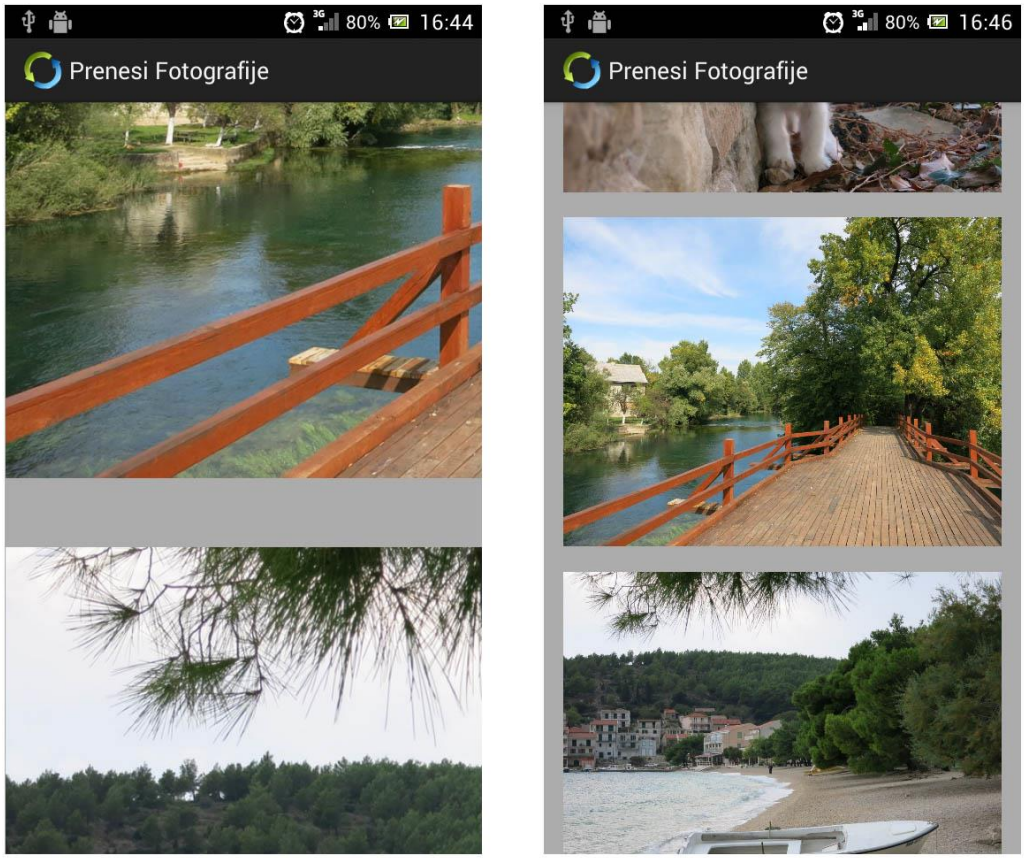
Kako bi se povezalo „WebView“ element sa Java klasnom potrebno je kreirati novu varijablu koja je tipa „WebView“ te će ona u našem slučaju biti nazvana „JWVGalerija“. Nakon što je povežemo sa XML datotekom potrebno ju je postaviti na URL koji ukazuje na PHP datoteku na web stranici koja je zadužena za prikaz prenesenih fotografija kao što je prikazano kodom.

```
private WebView jVWGalerija;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView (R.layout.galerija_layout);

	jVWGalerija = (WebView) findViewById(R.id.wvGalerija);

	jVWGalerija.loadUrl("http://freehost.filesblue.com/galerija.php");

	}
```

Prethodni kod se preko funkcije „loadUrl“ povezuje na našu web stranicu, odnosno na našu PHP skriptu koja je zadužena za prikaz fotografija. Iako je na web stranici prikaz zadovoljavajući, u našoj Android aplikaciji prikazuje se samo dio fotografija, odnosno ekran našeg Android uređaja je premalen za prikaz fotografija cijelih dimenzija. Kako bi se element „WebView“ natjeralo da cijelu web stranicu prikaže u okvirima ekrana našeg Android uređaja potrebno je dodati sljedeći dio koda:

```
//Kod za prikaz u okviru ekrana
jVWGalerija.getSettings().setLoadWithOverviewMode(true);
jVWGalerija.getSettings().setUseWideViewPort(true);
```

Preko ovog koda ulazimo u postavke „WebView“ elementa gdje je prva linija koda, odnosno funkcija „setLoadWithOverviewMode“ korištena kako bi „WebView“ smanjio prikaz web stranice. Iduća linija koda, odnosno „setUseWideViewPort**“** se koristi kako bi web stranica bila prikazana preko cijele površine korištenog elementa „WebView“ što je u našem slučaju cijela površina našeg Android uređaja. Slika 20 prikazuje razliku u prikazu prije i poslije korištenog koda za prikaz u okviru ekrana



Slika 20: a) Prikaz prenesenih fotografija bez koda za prikaz u okviru programa b) Prikaz prenesenih fotografija sa kodom za prikaz u okviru programa

Izvor: Autor

4.3 Aktivnost sa osnovnim podacima o aplikaciji

Osim aktivnosti za prijenos fotografija sa Android uređaja na web stranicu i aktivnosti za pregled prenesenih fotografija, aplikacija je zamišljena i sa aktivnošću koja korisniku daje osnovne informacije o aplikaciji. Zbog toga će biti kreirana i jedna posebna aktivnost koja će sadržavati informacije u tekstualnom obliku te će na ovom primjeru biti pokazano kako i za što koristiti stringove. Ova će aktivnost sadržavati osnovne informacije o aplikaciji kao primjerice koja je svrha aplikacije, ime i prezime autora i mentora, fakultet i slično. Kako je sve navedeno moguće izvesti u cijelosti u XML datoteci, Java klasa će služiti samo za pozivanje XML datoteke kako bi te informacije mogle biti prikazane.

Ova će aktivnost biti sastavljena samo od dvije vrste elemenata. Prva je „ImageView“ za prikazivanje loga, a ostali elementi će biti „TextView“ koji će biti

korišteni za prikaz teksta. Svaki „TextView“ će sadržavati svoj tekst pa to primjerice za jedan od elemenata prikazano kodom izgleda ovako:

```
<TextView
        android:id="@+id/tvOpis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/ivLogo"
        android:layout_marginTop="31dp"
        android:text="Ova je Android aplikacija napravljena u svrhu
diplomskog rada."
        android:textAppearance="?android:attr/textAppearanceLarge" />
```

Kao što je iz koda vidljivo, ukoliko koristimo element „TextView“ za prikaz dugačkog teksta to često zna biti nepraktično i nepregledno. Upravo zbog toga razvojno okruženje Eclipse preporuča korištenje stringova.

4.3.1 Korištenje stringova

Da se izbjegne predugačak kod, često se koriste stringovi u kojima su zapisani podaci te se oni prikazuju pozivanjem u XML kodu. Kako se za prikaz koristi više „TextView“ elemenata koji sadržavaju tekst, kod može izgledati predugačko i nepregledno pa je dobar primjer za korištenje stringova upravo ova aktivnost koja će se koristit za prikaz teksta.

Kako bi kreirali nove stringove potrebno je u direktoriju „values“ urediti datoteku „string.xml“. Kao i kod prethodnik slučajeva to je moguće napraviti preko dva načina; preko korisničkog sučelja razvojnog okruženja Eclipse ili dodavanjem dijela koda u XML datoteku. Slika 21 prikazuje dodavanje novog stringa preko korisničkog sučelja dok sljedeći kod prikazuje dodavanje tog istog stringa kodom.

```
<string name="opis">Ova je Android aplikacija napravljena u svrhu
diplomskog rada.</string>
```

Slika 21: Dodavanje novog stringa preko korisničkog sučelja

Izvor: Autor

Nakon kreiranja stringova, u datoteci koja opisuje izgled aktivnosti, u našem slučaju „informacije_layout.xml“ potrebno je pozvati određene stringove. Sljedeći kod prikazuje pozivanje stringa „opis“ čime se kod skraćuje te samim time izgleda preglednije.

```
<TextView
        android:id="@+id/tvOpis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/ivLogo"
        android:layout_marginTop="31dp"
        android:text="@string/opis"
        android:textAppearance="?android:attr/textAppearanceLarge" />
```

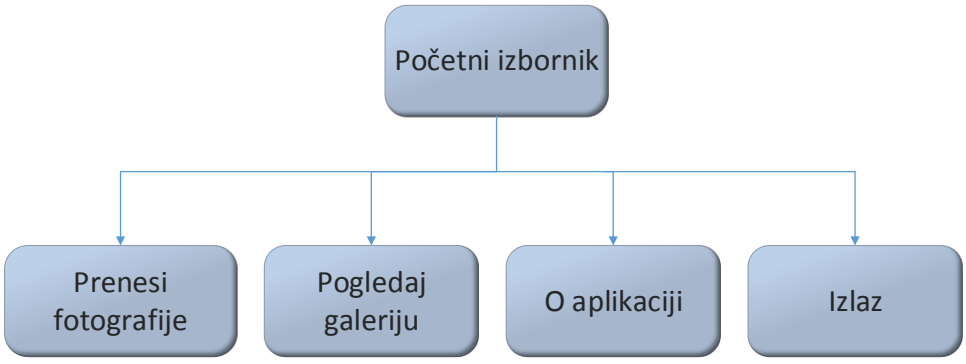
U konačnici, nakon što su dodani svi stringovi u datoteku „strings.xml“ izgled aktivnosti koja će se u našoj aplikaciji koristiti za prikaz informacija je prikazan slikom 22.

Slika 22: Izgled aktivnosti sa osnovnim podacima o aplikaciji

Izvor: Autor

4.4 Početni izbornik

Aplikacija koje će biti izrađena u ovom diplomskom radu je zamišljena da se svima različitim aktivnostima aplikacije pristupa preko jedne glavne aktivnosti odnosno izbornika. Izbornik je zamišljen kao glavna aktivnost koja vodi na ostale, prethodno izrađene aktivnosti a iz kojih se moguće vratiti pritiskom na tipku za povratak na samome uređaju. Struktura aplikacije je prikazana blok dijagramom na slici 23.



Slika 23: Struktura aplikacije za prijenos fotografija

Izvor: Autor

4.4.1 Izgled početnog izbornika

Početni izbornik je zamišljen kao prva aktivnost koju korisnik vidi nakon pokretanja aplikacije. Na njemu se nalaze tipke koje vode do ostalih aktivnosti aplikacije te tipka koja prekida rad cijele aplikacije. Kako bi tipke bilo moguće razmjestiti po ekranu biti će korišten relativan raspored elemenata. Od elemenata će biti korišten „ImageView“ za prikaz loga aplikacije dok će za promjenu aktivnosti biti zadužene četiri tipke odnosno „Button“ elementi.

Da se dobije ljepši izgled glavne aktivnosti biti će korištene ikone koje će zamijeniti normalni izgled tipki. U tu svrhu izrađene su dvije veličine ikona gdje će manje ikonice zamijeniti veće prilikom pritiska na tipku. Kao i sve dodatne datoteke koje su zadužene za izgled aplikacije ikonice moraju biti dodane u „drawable“ direktorij. U istom će tom direktoriju biti spremljen i kod koji će se koristiti za promjene ikone sa veće na manju prilikom pritiska na tipku.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/galerija_mala"
    android:state_pressed= "true"/>
    <item android:drawable="@drawable/galerija"/>
</selector>
```

Prethodni XML kod preko funkcije „item“ uzima navedene datoteke, te ukoliko se radi o pritisnutoj tipki uzima datoteku „galerija_mala“ dok u ostalim slučajevima uzima tipku normalne veličine. Kod je spremljen pod nazivom „galerija_tipka.xml“ te je istog potrebno pozvati u XML datoteci koja opisuje izgled, odnosno u datoteci „pocetna_layout.xml“. Sljedeći kod prikazuje XML kod koji opisuje tipku koja vodi do aktivnosti za prijenos fotografija na web stranicu.

```
    <Button
        android:id="@+id/bPrenesi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/tvPrenesi"
        android:layout_below="@+id/ivLogo"
        android:layout_marginTop="20dp"
        android:background="@drawable/prijenos_tipka" />
```

Kako bi se krajnji korisnik lakše koristio početnim izbornikom, osim ikonica koje će zamijeniti običan izgled tipki biti će i korišten tekstualni opis tipke. To će biti ostvareno preko elementa „ImageView“ koji će se nalaziti ispod same tipke. U konačnici, početni izbornik, uz korištenje ikona za tipke i tekstualnog opisa je prikazan slikom 24.

Slika 24: Izgled početnog izbornika

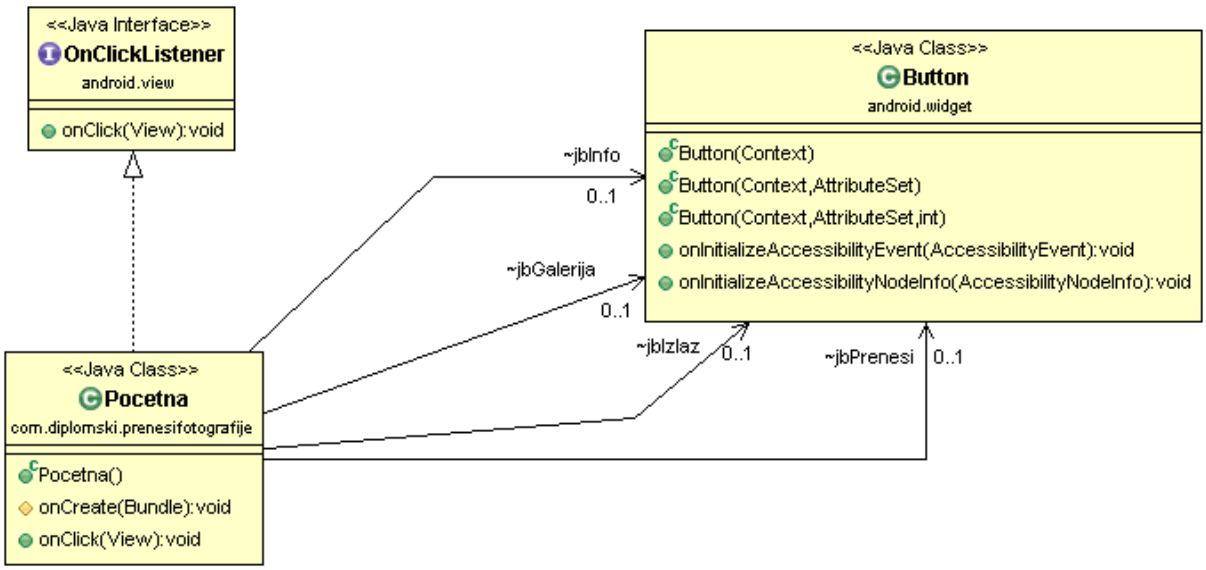
Izvor: Autor

4.4.2 Java klasa početnog izbornika

Da bi početni izbornik mogao raditi potrebno je izraditi novu Java klasu koja će upravljati tipkama, odnosno pratiti koja je tipka pritisnuta te na osnovu toga otvarati novu aktivnost koju korisnik želi. Kako bi to bilo omogućeno u Java klasu je potrebno implementirati metodu „onClickListener“ koja će preko „switch“ petlje pregledavati koja je tipka pritisnuta te onda preko funkcije „Intent“ pokretati navedenu aktivnost. Isti je postupak i kod dodavanja ostalih tipki. Najprije je tipku potrebno deklarirati kao novu Java varijablu te ju zatim povezati sa varijablom iz XML datoteke. Nakon toga preko novog slučaja (engl. case) za određenu tipku pokrenuti određenu aktivnost. Sljedeći kod prikazuje pozivanje klase „Prenesi.java“.

```
public class Početna extends Activity implements View.OnClickListener {

	Button jbPrenesi;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView (R.layout.pocetni_layout);

		jbPrenesi = (Button)findViewById (R.id.bPrenesi);
		jbPrenesi.setOnClickListener(this); }

	@Override
	public void onClick(View tipka)
	{
		switch (tipka.getId())
		{
			case R.id.bPrenesi:
			startActivity (new Intent(new
			Intent("com.diplomski.prenesifotografije.PRENESI"));
			break;
		}
	}
```

Slikom 25 je prikazan klasni dijagram aktivnosti koja se koristi za početni izbornik, odnosno Java klasa „Pocetna.java“. Iz slike se vidi da se koristi funkcija „Button“ pomoću koje su izrađene tipke, dok je za dodjeljivanje funkcija tipkama korištena funkcija „onClickListener“.



Slika 25: Klasni dijagram aktivnosti "Pocetna.java"

Izvor: Autor

Osim što se početni izbornik koristi za pozivanje ostalih aktivnosti aplikacije ima i funkciju gašenja aplikacije. Izlaz iz aplikacije bi bio moguć i bez te funkcije jer se za izlaz može koristiti tipka za povratak na samome Android uređaju ali onda aplikacija i dalje ostaje pokrenuta. To je posebno loše zbog ograničenosti memorije na uređaju te se njenim potpunim gašenjem osigurava brži i fluidniji rad. Sljedeći dio koda prikazuje dio petlje u kojem se provjerava je li pritisnuta tipka za izlaz te izvršavanje gašenja aplikacije.

```
public void onClick(View tipka)
	{
		switch (tipka.getId())
		{
			case R.id.bIzlaz:
			finish();
			break;
		}
	}
```

4.4.3 Dodavanje aktivnosti u Manifest datoteku

Kako bi se iz jedne aktivnosti moglo pokretati ostale koristi se funkcija „Intent“ koja poziva klase zapisane u Android Manifest datoteci. Stoga je sve aktivnosti neke aplikacije potrebno definirati u datoteci „AndroidManifest.xml“. Posebno se definira aktivnost za koju želimo da se pokreće prva te se njoj dodjeljuje kategorija „LAUNCHER“ dok se sve ostale aktivnosti definiraju kao „DEFAULT“ aktivnosti. Kodom je prikazana deklaracija glavne aktivnosti „Pocetna“ i aktivnosti „Prenesi“.

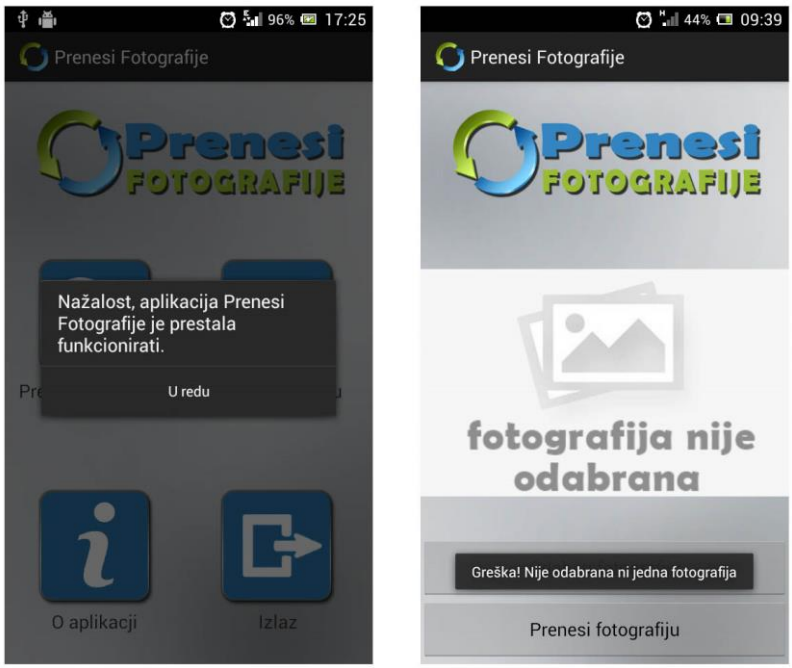
```
<activity
   android:name="com.diplomski.prenesifotografije.Pocetna"
   android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>


<activity
   android:name="com.diplomski.prenesifotografije.Prenesi"
   android:label="@string/app_name" >
  <intent-filter>
     <action android:name=" com.diplomski.prenesifotografije.PRENESI" />
     <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</activity>
```

4.5 Poboljšanja i ispravljanje greški

Nakon što je Android aplikacija gotova, posao razvojnog programera nije završio već je potrebno izvršiti testiranje aplikacije kako bi se pronašlo i uklonilo greške te tako poboljšalo rad aplikacije. Kako bi aplikacija bila što kvalitetnija, na razvojnom programeru je da predvidi što više situacija u kojima se krajnji korisnici mogu naći te podesiti aplikaciju da se te situacije ne događaju.

U slučaju Android aplikacije izrađene za potrebe ovog diplomskog rada to bi bilo primjerice pokušaj prijenosa fotografija bez odabrane fotografije. U tom bi slučaju aplikacija javila krajnjem korisniku da je aplikacija prestala funkcionirati te se nakon toga aplikacija gasi. Slika 26 a) prikazuje poruku koju javlja aplikacija neposredno prije gašenja.

Kako bi se izbjeglo takve situacije, u kod će biti dodana provjera preko varijable „brojac“ kako bi se vidjelo je li odabrana fotografija za slanje ili nije. Ukoliko je fotografija za prijenos odabrana, slanje na web stranicu preko tipke će biti moguće započeti, ali ukoliko nije odabrana ni jedna fotografija tada će korisnika obavijestiti o tome. Korištenje dodatne varijable kao brojača vrlo je često kod programiranja i u ostalim programskim jezicima kako bi se provjerilo neki uvjet.



Slika 26: a)Prestanak funkcioniranja aplikacije b) Obavijest korisniku o grešci

Izvor: Autor

Na početku je potrebno varijablu „brojac“ deklarirati te je postaviti na nulu. Ta će se brojka povećati za jedan u dijelu koda nakon što korisnik odabere fotografiju koju želi prenijeti na web stranicu. Nakon toga preko metode „onClick“ provjerava se koja je tipka u aktivnosti „Prenesi“ pritisnuta. Ukoliko korisnik pritisne tipku „Prenesi fotografiju“ a ni jedna fotografija nije odabrana, odnosno ako je brojač i dalje postavljen na vrijednost 0, korisniku će se javiti greška kako ni jedna fotografija nije odabrana kao što je prikazano slikom 26 b). Ukoliko je fotografija za prijenos odabrana, odnosno ako je brojač postavljen na vrijednost 1 tada se fotografija prenosi na stranicu sa obavijesti da je prijenos u tijeku. Sljedećim je kodom prikazana uporaba brojača u „if“ petlji koja provjerava koja je tipka pritisnuta.

```
int brojac=0;

@Override
      public void onClick(View tipka) {

            //Ako je pritisnuta tipka „Odaberi fotografiju“
            if(tipka==jbOdabirF)
                  {
                  //Odaberi fotografiju
                  }


            //Ako je pritisnuta tipka „Prenesi fotografiju“ ali nije
            odabrana fotografija
            if (tipka==jbPrijenosF && brojac==0)
                  {
                  //Javi grešku
                  Toast.makeText(Prenesi.this, "Greška! Nije odabrana ni
                  jedna fotografija", Toast.LENGTH_SHORT).show();
                  }


            //Ako je pritisnuta tipka „Prenesi fotografiju“ i odabrana
            je fotografija
            else if (tipka==jbPrijenosF && brojac!=0)
                  {
                  //Prenesi fotografiju
                  }
      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data)

        {
        //Odabir fotografije
        brojac++;
        }
```

Jedna od greški je i korištenje aplikacije u vodoravnoj poziciji uređaja (engl. landscape mode). Kako Android aplikacija zahtjeva razvoj posebnog izgleda za različite veličine ekrana tako je moguće i izraditi poseban izgled za vodoravnu orijentaciju. Kako je naša aplikacija rađena samo za portret način prikaza, odnosno okomiti položaj uređaja, aplikacija će prilikom okretanja uređaja izgledati sasvim drugačije nego što bi trebala. Ta je razlika naročito vidljiva kod relativnog razmještaja elemenata koji se preklapaju, ovisno o postavkama izgleda. Slična je stvar i kod linearnog rasporeda elemenata kod kojeg se prikazuje samo gornji dio aktivnosti. Slikom 27 je prikazan izgled početnog izbornika u vodoravnom položaju uređaja.

Slika 27: Greška kod vodoravnog prikaza

Izvor: Autor

Jedno od rješenja zahtijeva izradu posebnog izgleda za različite orijentacije uređaja, što znači da razvojni programer mora izrađivati poseban izgled za vodoravni i okomiti položaj uređaja. Drugo rješenje našeg problema je onemogućavanje promjene orijentacije kod aplikacije. Tako će i kod vodoravnog i kod okomitog prikaza aplikacija uvijek biti ista, odnosno u našem slučaju okomitog izgleda. Da bi se to onemogućilo u svakoj Java klasi aktivnosti potrebno je dodati sljedeći dio koda.

```
//Onemogućavanje rotacije u vodoravni položaj
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

Još jedno od mogućih poboljšanja vezano za dizajn aplikacije je i uklanjanje vodoravne crte sa vrha ekrana koja sadržava logo i ime aplikacije koja ustvari predstavlja naslov aplikacije (engl. title). Time bi se dobio ljepši izgled i veći prostor za aktivnost aplikacije kao što je prikazano na slici 28. To je ostvarivo preko funkcije „requestWindowFeature“ u kojoj je potrebno navesti da ne prikazuje naslov aplikacije kao što to prikazuje sljedeći kod.

```
requestWindowFeature (Window.FEATURE_NO_TITLE);
```

Kako bi kod radio, potrebno ga je dodati u Java klasu aktivnosti i to prije nego što sama klasa pozove XML datoteku s izgledom jer u protivnom neće raditi. Konačan rezultat nakon što je uklonjen naslov je prikazan slikom 28.

Slika 28: Aplikacija prije i poslije micanja naslova

Izvor: Autor

5. ZAKLJUČAK

Android operacijski sustav je zbog svoje popularnosti i sve veće prisutnosti na tržištu sve više popularan kod razvojnih programera. Upravo zbog toga broj dostupnih aplikacija iz dana u dan sve više raste. Međutim, to ne znači da je tržište prezasićeno aplikacijama jer razvojem novih i jačih uređaja raste i potreba za novim aplikacijama.

U ovom diplomskom radu je opisan razvoj Android aplikacije za prijenos slika na web stranicu te je kroz razvoj aplikacije prikazana upotreba nekih od funkcija i metoda koje se koriste kod izrada Android aplikacija te je pokazan rad u razvojnom okruženju Eclipse. Osim toga, na početku rada su opisane glavne karakteristike Android operacijskog sustava, kao i korištenje razvojnog okruženja Eclipse.

Ovim se diplomskim radom pokazalo kako je u kratkom roku moguće naučiti Java programski jezik i tako započeti razvoj Android aplikacija ali kako tržište pametnih telefona i tablet računala ubrzano raste, potrebno je uložiti puno truda kako bi se pratilo sve te promjene te ostalo konkurentan u području izrade Android aplikacija.

LITERATURA

[1] Meier, R.: Professional Android Application Development, Wiley Publishing, Inc., 2009

[2] http://mashable.com/2013/07/24/google-play-1-million/ (20.12.2013.)

[3] http://www.tutorialspoint.com/android/android_architecture.htm (22.12.2013.)

[4] Switkin, D.: Android Application Development, Google Inc.

[5] http://www.devmanuals.com/tutorials/java/corejava/javavirtualmachine.html (24.12.2013)

[6] Kušek, M., Topolnik, M.: Uvod u programski jezik Java, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[7] Lee, W.M.: Beginning Android 4 Application Development, John Wiley & Sons, Inc., 2012

[8] http://developer.android.com/training/index.html (26.12.2013)

[9] http://androidexample.com/Upload_File_To_Server_-_Android_Example/index.php?view=article_discription&aid=83&aaid=106 (26.12.2013)

[10] http://www.mybringback.com/series/android-basics/ (10.1.2014)

[11] Cincar, O.: Android Apps with Eclipse, Apress, 2012

[12] Čarapina, M.: Praćenje tramvajskog prometa na operacijskom sustavu Android, Fakultet elektrotehnike i računarstva, Zagreb, srpanj 2009.

[13] Dujmović, A.: Upravljanje videonadzorom cestovnog prometa mobilnim uređajem na Android platformi, Sveučilište u Dubrovniku, Odjel za elektrotehniku i računarstvo, Dubrovnik, rujan 2010.

[14] Rogers, R., Lombardo, J., Mednieks, Z., Meike, B.: Android Application Development, O'Reilly Media, 2009

[15] Marjanica, A.: GPS kolektor – prototip Android aplikacije za prikupljanje prostornih podataka, Sveučilište u Zagrebu – Geodetski fakultet, Zagreb, rujan 2011.

[16] Glasinović, F.: Android aplikacija kao GNSS kontroler, Geodetski fakultet, Zagreb, rujan 2012.

[17] Kolarić, H.: Razvoj programa za pristup i obradu informacija na pokretnom uređaju s operacijskim sustavom Android, Fakultet elektrotehnike i računarstva, Zagreb, lipanj 2009.

[18] Vukasanović, J.: Spajanje na mrežu u operacijskom sustavu Android, Fakultet elektrotehnike i računarstva, Zagreb, studeni 2009.

POPIS SLIKA

DODATAK

Pocetna.java

```
package com.diplomski.prenesifotografije;

import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;

public class Pocetna extends Activity implements View.OnClickListener {

	Button jbPrenesi, jbGalerija, jbInfo, jbIzlaz;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub

		super.onCreate(savedInstanceState);
		//Micanje vodoravne crte
		requestWindowFeature (Window.FEATURE_NO_TITLE);
		//Odabir izgleda
		setContentView (R.layout.pocetni_layout);
		//Onemogućavanje rotacije u vodoravni položaj

	setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);



		jbPrenesi = (Button)findViewById (R.id.bPrenesi);
		jbGalerija= (Button)findViewById (R.id.bGalerija);
		jbInfo = (Button)findViewById (R.id.bInfo);
		jbIzlaz = (Button) findViewById (R.id.bIzlaz);

		jbPrenesi.setOnClickListener(this);
		jbGalerija.setOnClickListener(this);
		jbInfo.setOnClickListener(this);
		jbIzlaz.setOnClickListener(this);
	}


	@Override
	public void onClick(View tipka)
	{
		// TODO Auto-generated method stub
		switch (tipka.getId())
		{
		case R.id.bPrenesi:
			startActivity (new
Intent("com.diplomski.prenesifotografije.PRENESI"));
			break;
		case R.id.bGalerija:
```

```
                  startActivity (new
Intent("com.diplomski.prenesifotografije.GALERIJA"));
                  break;
            case R.id.bInfo:
                  startActivity (new Intent
("com.diplomski.prenesifotografije.INFORMACIJE"));
                  break;
            case R.id.bIzlaz:
                  finish();
                  break;

            }
      }
}
```

Pocetni_layout.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:background="@drawable/bg"
    android:orientation="horizontal"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/bGalerija"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/bPrenesi"
        android:layout_alignBottom="@+id/bPrenesi"
        android:layout_alignParentRight="true"
        android:layout_marginRight="30dp"
        android:background="@drawable/galerija_tipka" />

    <TextView
        android:id="@+id/tvGalerija"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/bGalerija"
        android:layout_below="@+id/bGalerija"
        android:text="Pogledaj galeriju"
        android:textAlignment="center"
        android:textSize="18sp" />

    <Button
        android:id="@+id/bInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/bIzlaz"
        android:layout_alignBottom="@+id/bIzlaz"
        android:layout_alignLeft="@+id/bPrenesi"
        android:background="@drawable/info_tipka" />

    <TextView
        android:id="@+id/tvInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/bInfo"
        android:layout_below="@+id/bInfo"
        android:layout_marginLeft="16dp"
        android:text="O aplikacji"
        android:textSize="18sp" />

    <ImageView
        android:id="@+id/ivLogo"
        android:contentDescription="logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
```

```
        android:src="@drawable/logo_app" />

    <Button
        android:id="@+id/bIzlaz"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignRight="@+id/tvGalerija"
        android:layout_marginBottom="53dp"
        android:background="@drawable/izlaz_tipka" />

    <TextView
        android:id="@+id/tvExit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/tvInfo"
        android:layout_alignBottom="@+id/tvInfo"
        android:layout_alignLeft="@+id/bIzlaz"
        android:layout_marginLeft="40dp"
        android:text="Izlaz"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/tvPrenesi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/bPrenesi"
        android:layout_marginRight="14dp"
        android:layout_toLeftOf="@+id/tvGalerija"
        android:text="   Prenesi fotografije"
        android:textAlignment="center"
        android:textSize="18sp" />

    <Button
        android:id="@+id/bPrenesi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/tvPrenesi"
        android:layout_below="@+id/ivLogo"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="25dp"
        android:background="@drawable/prenesi_tipka" />

</RelativeLayout>
```

galerija_tipka.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/galerija_mala"
android:state_pressed= "true"/>
    <item android:drawable="@drawable/galerija"/>
</selector>
```

info_tipka.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/info_mala" android:state_pressed=
"true"/>
    <item android:drawable="@drawable/info"/>
</selector>
```

prenesi_tipka.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/prenesi_mala"
android:state_pressed= "true"/>
    <item android:drawable="@drawable/prenesi"/>
</selector>7
```

izlaz_tipka.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/izlaz_mala" android:state_pressed=
"true"/>
    <item android:drawable="@drawable/izlaz"/>
</selector>
```

Prenesi.java

```
package com.diplomski.prenesifotografije;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.HttpURLConnection;
import java.net.URL;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

public class Prenesi extends Activity implements OnClickListener {

      //Definiranje varijabli

      Button jbOdabirF, jbPrijenosF;
    ImageView jivSlika;

    //Varijable potrebne za java skriptu

    int serverOdgovor = 0;
    String serverUrl =
"http://freehost.filesblue.com/prijenos_skripta.php";
    String putanja = null;
    ProgressDialog dialog;
    int brojac=0;

      @Override
      protected void onCreate(Bundle savedInstanceState)
      {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            //Micanje vodoravne crte
            requestWindowFeature (Window.FEATURE_NO_TITLE);
            //Povezivanje sa XML datotekom
            setContentView(R.layout.prenesi_layout);
            //Onemogućavanje rotacije u vodoravni položaj

      setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

            // Povezivanje varijabli sa XML varijablama

            jbOdabirF = (Button)findViewById(R.id.bOdabirF);
```

```
        jbPrijenosF = (Button)findViewById(R.id.bPrijenosF);
        jivSlika = (ImageView)findViewById(R.id.ivSlika);

        jbOdabirF.setOnClickListener(this);
        jbPrijenosF.setOnClickListener(this);

      }

      @Override
      public void onClick(View tipka) {
            //Ako je pritisnuta tipka „Odaberi fotografiju"
            if(tipka==jbOdabirF)
            {
                  Intent odabirSlike = new Intent();
            odabirSlike.setType("image/*");
            odabirSlike.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(odabirSlike,
"Dovrši uz pomoć"), 1);
            }

            //Ako je pritisnuta tipka „Prenesi fotografiju" ali nije
odabrana fotografija
            if (tipka==jbPrijenosF && brojac==0)
            {
                  Toast.makeText(Prenesi.this, "Greška! Nije odabrana ni
jedna fotografija", Toast.LENGTH_SHORT).show();
            }

            //Ako je pritisnuta tipka „Prenesi fotografiju" i odabrana je
fotografija
            else if (tipka==jbPrijenosF && brojac!=0)
            {
                  dialog = ProgressDialog.show(Prenesi.this, "Molim
pričekajte", "Fotografija se prenosi...");
                   new Thread(new Runnable() {
                  public void run()
                  {
                        uploadFile(putanja);
                  }
               }).start();
            }
      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
            if (requestCode == 1 && resultCode == RESULT_OK)
           {
        Uri uriOdabraneSlike = data.getData();
        putanja = getPath(uriOdabraneSlike);
        Bitmap bitmap = BitmapFactory.decodeFile(putanja);
        jivSlika.setImageBitmap(bitmap);
        brojac++;
        }
      }
//
      private String getPath(Uri uriOdabraneSlike) {
            String[] projection = { MediaStore.Images.Media.DATA };
        Cursor cursor = getContentResolver().query(uriOdabraneSlike,
projection, null, null, null);
```

```
        int column_index =
cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        cursor.moveToFirst();
        return cursor.getString(column_index);
      }

      public int uploadFile(String sourceFileUri)
      //Početak skripte
      {
            String fileName = sourceFileUri;
          HttpURLConnection conn = null;
          DataOutputStream dos = null;
          String twoHyphens = "--";
          String lineEnd = "\r\n";
          String boundary = "*****";
          int bytesRead, bytesAvailable, bufferSize;
          byte[] buffer;
          int maxBufferSize = 1 * 1024 * 1024;
          File sourceFile = new File(sourceFileUri);

          //Početak komunikacije sa web stranicom
          {
                try {
                    FileInputStream fileInputStream = new
FileInputStream(sourceFile);
                    URL url = new URL(serverUrl);   // Postavljanje URL-
a web stranice
                    conn = (HttpURLConnection) url.openConnection();
                    conn.setDoInput(true);
                    conn.setDoOutput(true);
                    conn.setUseCaches(false);
                    conn.setRequestMethod("POST");
                    conn.setRequestProperty("Connection", "Keep-Alive");
                    conn.setRequestProperty("ENCTYPE", "multipart/form-
data");
                    conn.setRequestProperty("Content-Type",
"multipart/form-data;boundary=" + boundary);
                    conn.setRequestProperty("uploaded_file", fileName);

                    dos = new DataOutputStream(conn.getOutputStream());

                    dos.writeBytes(twoHyphens + boundary + lineEnd);
                    dos.writeBytes("Content-Disposition: form-data;
name=\"uploaded_file\";filename=\""
                                                + fileName + "\"" +
lineEnd);

                    dos.writeBytes(lineEnd);

                    bytesAvailable = fileInputStream.available();

                    bufferSize = Math.min(bytesAvailable,
maxBufferSize);
                    buffer = new byte[bufferSize];

                    bytesRead = fileInputStream.read(buffer, 0,
bufferSize);

                    while (bytesRead > 0)
                    {
                      dos.write(buffer, 0, bufferSize);
```

```
                        bytesAvailable = fileInputStream.available();
                        bufferSize = Math.min(bytesAvailable,
maxBufferSize);
                        bytesRead = fileInputStream.read(buffer, 0,
bufferSize);
                       }

                    dos.writeBytes(lineEnd);
                    dos.writeBytes(twoHyphens + boundary + twoHyphens +
lineEnd);
                    //Serverov odgovor (kod i poruka)
                    serverOdgovor = conn.getResponseCode();
                    String serverResponseMessage =
conn.getResponseMessage();

                    Log.i("uploadFile", "HTTP Response is : "
                                + serverResponseMessage + ": " +
serverOdgovor);
                    //Provjerava uspješnost prijenosa
                    if(serverOdgovor == 200)
                    {
                          runOnUiThread(new Runnable()
                             {
                             public void run()
                             {
                                   Toast.makeText(Prenesi.this,
"Prijenos uspješno dovršen", Toast.LENGTH_SHORT).show();
                              }
                          });
                    }

                    fileInputStream.close();
                    dos.flush();
                    dos.close();
                }
                //Zapisivanje grešaka u logcat
                 catch (Exception e) {

                    dialog.dismiss();
                    e.printStackTrace();

                    runOnUiThread(new Runnable() {
                        public void run()
                        {
                          Toast.makeText(Prenesi.this, "Iznimka :
Pogledaj LogCat ", Toast.LENGTH_SHORT).show();
                        }
                    });
                    Log.e("Iznimka prilikom prijenosa slike na server",
"Exception : "  + e.getMessage(), e);
                }
                dialog.dismiss();
                return serverOdgovor;
            }
        } // Kraj


}
```

prenesi_layout.xml

```
<?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="match_parent"
     android:background="@drawable/bg"
     android:baselineAligned="false"
     android:orientation="vertical" >

     <ImageView
         android:id="@+id/ivLogo"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:src="@drawable/logo_app" />

     <ImageView
         android:id="@+id/ivSlika"
         android:layout_width="fill_parent"
         android:layout_height="300dp"
         android:layout_gravity="center_horizontal"
         android:src="@drawable/nothing" />

     <Button
         android:id="@+id/bOdabirF"
         android:layout_width="match_parent"
         android:layout_height="60dp"
         android:layout_gravity="bottom"
         android:text="Odaberi fotografiju" />

     <Button
         android:id="@+id/bPrijenosF"
         android:layout_width="match_parent"
         android:layout_height="60dp"
         android:text="Prenesi fotografiju" />

  </LinearLayout>
```

Galerija.java

```
package com.diplomski.prenesifotografije;

import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.webkit.WebView;

public class Galerija extends Activity {

	private WebView jVWGalerija;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView (R.layout.galerija_layout);
		//Onemogućavanje rotacije u vodoravni položaj

	setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

		jVWGalerija = (WebView) findViewById(R.id.wvGalerija);

	jVWGalerija.loadUrl("http://freehost.filesblue.com/galerija.php");

		//Kod za prikaz u okviru ekrana
		jVWGalerija.getSettings().setLoadWithOverviewMode(true);
		jVWGalerija.getSettings().setUseWideViewPort(true);
	}

}
```

galerija_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <WebView
        android:id="@+id/wvGalerija"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

Informacija.java

```
package com.diplomski.prenesifotografije;

import android.app.Activity;
import android.os.Bundle;
import android.view.Window;

public class Informacije extends Activity{

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		//Micanje vodoravne crte
		requestWindowFeature (Window.FEATURE_NO_TITLE);
		setContentView(R.layout.informacije_layout);
	}

}
```

informacije_layout.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/RelativeLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bg"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/ivLogo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:contentDescription="logo"
        android:src="@drawable/logo_app" />

    <TextView
        android:id="@+id/tvOpis"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/ivLogo"
        android:layout_marginTop="31dp"
        android:text="@string/opis"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvStudent"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tvOpis"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="26dp"
        android:text="@string/student"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvMentor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tvStudent"
        android:layout_marginBottom="5dp"
        android:text="@string/mentor"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvFakultet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tvMentor"
        android:layout_marginBottom="5dp"
        android:text="@string/fakultet"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvSmjer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tvFakultet"
        android:text="@string/smjer"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/tvDatum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="@string/datum"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Prenesi Fotografije</string>
    <string name="opis">Ova je Android aplikacija napravljena u svrhu
diplomskog rada.</string>
    <string name="mentor">Mentor: Božidar Kovačić</string>
    <string name="student">Student: Alen Šišović</string>
    <string name="fakultet">Fakultet: Pomorski fakultet u Rijeci</string>
    <string name="smjer">Smjer: Elektroničke i informatičke tehnologije u
pomorstvu</string>
    <string name="datum">Rijeka, veljača 2014</string>

</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.diplomski.prenesifotografije"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.diplomski.prenesifotografije.Pocetna"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <activity
            android:name="com.diplomski.prenesifotografije.Prenesi"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="com.diplomski.prenesifotografije.PRENESI" />
                <category android:name="android.intent.category.DEFAULT"
/>
            </intent-filter>
        </activity>

        <activity
            android:name="com.diplomski.prenesifotografije.Galerija"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="com.diplomski.prenesifotografije.GALERIJA" />
                <category android:name="android.intent.category.DEFAULT"
/>
            </intent-filter>
        </activity>

        <activity
            android:name="com.diplomski.prenesifotografije.Informacije"
            android:label="@string/app_name" >
            <intent-filter>
                <action
android:name="com.diplomski.prenesifotografije.INFORMACIJE" />
```

```
                <category android:name="android.intent.category.DEFAULT"
/>
            </intent-filter>
        </activity>

    </application>

</manifest>
```