Seminarski Rad

Operacijski Sustav Android I Načini Povezivanja S Poslužiteljem

Www.Maturski.Org

Sadržaj

1. Uvod 2
2. Arhitektura Sustava Android 3

2.1 Dalvik Virtualni Stroj 5

3. android Sdk 7

1. Android Emulator 7
2. Dalvik Debug Monitoring Service (Ddms) 8
3. Android Debug Bridge (Adb) 9

4. koncept Aplikacije 10

1. Osnovna Struktura 10
2. Životni Ciklus Aplikacije 11
3. Korisničko Sučelje 14
4. Izbornik 15
5. Događaji Unutar Korisničkog Sučelja 15
6. Dodatne Mogućnosti 15
7. Datoteka Androidmanifest.Xml 16
8. Podatkovni Resursi 18
9. Upotreba Resursa U Kodu 19
10. Upotreba Resursa U Drugim Podatkovnim Resursima 19
11. Upotreba Sistemskih Resursa 19

5. povezivanje S Poslužiteljem 21

5.1 protokol U Dp 21

5.1.1 Aplikacija Koja Koristi Udp Komunikaciju 21

5.2 protokol Tcp 23

5.2.1 Aplikacija Koja Koristi Tcp Komunikaciju 23

5.3 korištenje Protokola Http 25

5.3.1 Aplikacija Koja Koristi Protokol Http 26

1. Http Konekcija 26
2. Xml Parser 27
3. Ostali Formati Zapisa Strukturiranih Podataka 31
4. Protokol Https 32
5. Ostali Načini Komunikacije S Poslužiteljem 32
6. Zaključak 33
7. Literatura 34

1. Uvod

Android Je Programsko Okruženje Za Pokretne Uređaje Bazirano Na Otvorenom Kodu (Eng. Open Source) Koje Se Sastoji Od Kompleta Programske Opreme: Operacijskog Sustava, Programske Međuopreme (Eng. Middleware) I Ključnih Aplikacija Za Pokretne Uređaje. Također, Sadrži Mnoštvo Sučelja Za Izradu Aplikacija (Api, Eng. Application Programming Interface) Koja Razvojnim Inženjerima Omogućuju Samostalan Razvoj I Rad Na Aplikacijama.

Predstavljanje Androida I Njegove Razvojne Okoline Za Pisanje Aplikacija (Sdk, Eng. Software Development Kit) Započelo Je U Studenom 2007. Godine Od Strane Googlea I Grupacije Open Handset Alliance (Oha). Open Handset Alliance Je Konzorcij S Ciljem Razvoja Otvorenih Standarda Za Pokretne Uređaje, Promocija Inovacija I Prilagodba Uređaja Korisniku S Poboljšanom Izvedbom I Pristupačnom Cijenom. Broji Nekoliko Desetaka Članica Među Kojima Se Nalazi I Google.

Shodno Svojoj Politici Oha Ističe Slijedeće Karakteristike Android Programskog Okruženja [1]:

* Otvorenost - Programeru Omogućava Potpunu Slobodu U Razvoju Novih I Već Postojećih Aplikacija, A Proizvođaču Uređaja Slobodno Korištenje I Prilagodbu Platforme Bez Plaćanja Autorskih Prava;
* Sve Aplikacije Su Ravnopravne - Što Znači Da Ne Postoji Razlika Između Osnovnih Jezgrenih Aplikacija Uređaja I Dodatnih Aplikacija. Svim Aplikacijama Omogućen Je Ravnopravni Pristup Resursima Pokretnog Uređaja Što Daje Mogućnost Potpune Prilagodbe Uređaja Specifičnim Potrebama Individualnog Korisnika;
* Automatsko Upravljanje Ž Ivotnim Ciklusom Aplikacije - Omogućava Nadzor Pokretanja I Izvršavanja Aplikacija Na Sistemskoj Razini Optimiziranim Korištenjem Memorije I Snage Uređaja. Krajnji Korisnik Više Ne Brine O Gašenju Određenih Aplikacija Prije Pokretanja Drugih;
* Rušenje Granica "Klasičnih" Aplikacija - Mogućnost Razvoja Novih I Inovativnih Aplikacija Temeljenih Na Međusobnoj Kolaboraciji Tehnologija;
* Brz I Jednostavan Razvoj Aplikacija - Omogućen Je Bogatom Bazom Korisnih Programskih Biblioteka (Eng. Libraries) I Alata Za Izradu Aplikacija;
* Visokokvalitetni Grafički Prikaz I Zvuk - Podržana 2d Vektorska I 3D Opengl (Eng. Open Graphics Library) Grafika, Te Ugrađeni Kodeci Svih Često Korištenih Audio I Video Formata;
* Kompatibilnost S Većinom Sadašnjeg I Budućeg Hardvera - Uključuje Prenosivost Androidovih Aplikacija Na Arm, X86 I Ostale Arhitekture, Te Prilagodljivost Sustava Ulaznim I Izlaznim Komponentama.

2. Arhitektura Sustava Android

Arhitektura Sustava Android [3] Bazira Se Na Linux 2,6 Jezgri (Eng. Kernel) Koja Se Koristi Kao Sloj Apstrakcije Hardvera (Hal, Eng. Hardware Abstraction Layer). Razlog Za Korištenje Jezgre Operacijskog Sustava Linux Je Dokazana Pogonska Podrška (Eng. Driver Model), Mogućnost Upravljanja Memorijom I Procesima, Sigurnosni Model, Mrežni Sustav, Te Dokazana Robusnost, Konstantni Razvoj I Unapređivanje Sustava. Slika 2.1 Prikazuje Arhitekturu Sustava Android.



Slika 2.1: Arhitektura Sustava Android

Najvažniji Elementi Jezgre Su:

• Pogonski Program Za Međuprocesnu Komunikaciju Između Aplikacija I Usluga (Eng. Service) (Eng. Binder (Ipc) Driver) - Omogućava Međuprocesnu Komunikaciju Između Aplikacija I Usluga, Te

• Upravljanje Napajanjem (Eng. Power Management) - Komponenta Androida Temeljena Na Standardnom Linuxovom Sustavu Upravljanja Napajanjem.

Izvorne Programske Biblioteke (Eng. Native Libraries) Pisane Su U Programskim Jezicima C I C++ I Čine Idući Sloj U Arhitekturi Sustava. Neke Od Značajnijih Su:

* Program Za Upravljanje Grafičkim Sučeljem (Eng. Surface Manager) -Biblioteka Odgovorna Za Pravilno Iscrtavanje Različitih Aplikacijskih Komponenti U Vremenu I Prostoru;
* Opengl Es (Eng. Opengl for Embedded Systems) - Biblioteke Koje Se Koriste Za Hardversku 3D Akceleraciju (Ukoliko Je Podržana) Ili Za 3D Rasterizaciju;
* Sgl (Eng. Scalable Graphics Library) - Predstavlja 2d Biblioteke Na Kojima Je Temeljena Većina Aplikacija. Spomenimo Još Da Se 2d I 3D Elementi Mogu Kombinirano Prikazivati U Jednom Korisničkom Sučelju;
* Media Framework - Skupina Kodeka Za Snimanje I Reprodukciju Audio Formata, Video Formata I Nepomičnih Slika. Omogućena Je Od Strane Packetvidea;
* Freetype - Biblioteka Koja Služi Za Vektorsku Rasterizaciju Oblika Pisma (Eng.

Font);

* Ssl (Eng. Secure Socket Layer) - Omogućuje Sigurnosnu Komunikaciju Preko Interneta;
* Sqlite - Programska Biblioteka Koja Implementira Bazu Podataka (Eng.

Database Engine);

* Webkit - Jezgra Preglednika Koji Podržava Javascript I Ostale Standarde Na Malom Uređaju;
* System C Library - Implementacija Standardne C-Ove Sistemske Biblioteke (Libc) Izvedene Iz Operacijskog Sustava Bsd.

Idući Sloj U Arhitekturi Androida Je Radno Okruženje (Eng. Android Runtime) Kojeg Čine Virtualni Stroj Dalvik (Dvm, Eng. Dalvik Virtual Machine) I Jezgrene Biblioteke (Eng. Core Library).

Dvm Je Registarski Baziran Virtualni Stroj, Dok Je Klasični Javin Virtualni Stroj (Jvm, Eng. Java Virtual Machine) Baziran Na Stogu. Više O Dalvik Virtualnom Stroju U Poglavlju 2.1. Jezgrene Biblioteke Pisane Su U Programskom Jeziku Java I Predstavljaju Sve Esencijalne Klase Kao Što Su Klase Za Manipulaciju Kolekcijama, Klase Za Komunikaciju S Okolinom I Slično. Bitna Novost Je I to Što Se Androidove Jezgrene Biblioteke Razlikuju Od Biblioteka U Java Standard Edition (J2se) I Java 2 Micro Edition (J2me).

Sloj Aplikacijskih Okvira (Eng. Application Framework) Napisan Je U Programskom Jeziku Java I Sadrži Proširiv Skup Programskih Komponenti Kojeg Koriste Sve Aplikacije Uređaja. Neki Od Važnijih Elemenata Su:

* Upravljanje Aktivnostima (Eng. Activity Manager) - upravljanje Životnim Ciklusom Aplikacije,
* Upravljanje Programskim Paketima (Eng. Package manager) - Sadrži Informaciju O Aplikacijama Instaliranim Na Sustavu,
* Upravljanje Prozorima (Eng. Window Manager) - Upravljanje Aplikacijskim Prozorima,
* Upravljanje Pozivima (Eng. Telephony Manager) - Sadrži Api-Je Koji Se Koriste Pri Izradi Aplikacija Za Upravljanje Pozivima,
* Pružatelji Sadržaja (Eng. Content Providers) - Omogućuju Zajedničko Korištenje Podataka Od Strane Više Aplikacija,
* Upravljanje Resursima (Eng. Resource Manager) - Služi Za Pohranu Dijelova Aplikacije Koji Nisu Kod (Npr. Slike),
* Sustav Grafičkog Prikaza (Eng. View System) - Sadrži Bazu Gotovih Grafičkih Prikaza I Alata (Eng. Widget),
* Upravljanje Lokacijski Temeljenim Uslugama (Eng. Location Manager) I
* Upravljanje Obavijestima (Eng. Notification Manager) - Upravljanje Obavijestima I Događajima (Npr. Dospijeće Poruke, Nadolazeći Sastanak).

Aplikacijski Sloj Je Zadnji Sloj U Arhitekturi Sustava Android I Č Ine Ga Korisničke Aplikacije Uređaja. Predstavlja Sloj Vidljiv Krajnjem Korisniku. Uključuje Neke Od Standardnih Sistemskih Aplikacija Kao Što Su Web Preglednik, Lista Kontakata, Telefon, Itd.

2.1 Dalvik Virtualni Stroj

Umjesto Upotrebe Standardnog Java 2 Micro Edition (J2me) Kao Mehanizma Za Pokretanje Javinih Aplikacija Na Pokretnim Uređajima, Google Je Razvio Vlastiti Virtualni Stroj Za Android. Dvm Je Najvjerojatnije Razvijen Kako Bi Se Zaobišla Problematika S Dozvolama Korištenjem Sunovog J2me. Svaki Pokretni Uređaj Koji U Sebi Sadrži J2me Mora Kod Suna Licencirati Bilo Kakvu Promjenu Izvornog Koda J2me-A. Virtualni Stroj Dalvik Dobio Je Ime Po Istoimenom Islandskom Gradu U Kojem Su Živjeli Preci Glavnog Razvijatelja Dvm-A Dan Bornsteina.

Osnovna Razlika Između Sun Java Virtualnih Strojeva, Te Dvm-A Je U Tome Što Su Prvo Spomenuti Stožno Bazirani Virtualni Strojevi, Dok Je Dvm Registarski Baziran Virtualni Stroj. Međukod (Eng. Bytecode) Dalvik Virtualnog Stroja Transformira Se Pomoću Alata Dx (Koji Je Sastavni Dio Android Sdk-A) Iz Javinih Klasnih Datoteka (Eng. Java Class File) Prevedenih Javinim Prevoditeljem U Novu Klasu \*.Dex (Eng. Dalvik Executable) Formata. Međukod Kojeg Izvršava Dvm Nije Javin Međukod, Nego Upravo Spomenuti .Dex Oblik. Transformacija Formata Omogućava Bolju Prilagodbu Za Rad Na Manjim Procesorima Boljim Iskorištavanjem Raspoložive Memorije I Procesorske Snage. Rezultat Svega Je Mogućnost Višestrukog Instanciranja Samog Virtualnog Stroja Š To Znači Da Se Svaka Androidova Aplikacija Pokreće Kao Zasebni Proces, S Vlastitom Instancom Virtualnog Stroja Dalvik.

Struktura Datoteke .Dex Prikazana Je Slikom 2.2 [5]. Datoteka Se Sastoji Od Jednostavnog Zaglavlja, Identifikatora Nizova, Tipova, Prototipova, Polja I Metoda, Definicija Klasa I Podataka. Ovime Se Osigurava Da Se Konstante Koje Se Ponavljaju U Klasnim Datotekama Pojavljuju Samo Jednom U Datoteci .Dex Kako Bi Se Sačuvala Memorija.

3. Android Sdk

Paket Softverskih Razvojnih Alata Android Sdk [6] Pruža Podršku Za Razvoj, Testiranje, Pronalaženje I Uklanjanje Pogrešaka (Eng. Debugging) Aplikacija. Uključuje Sljedeće Komponente:

* Android Api-Je;
* Razvojne Alate - Alati Za Prevođenje I Debagiranje. Najznačajniji Je Programski Dodatak (Eng. Plugin) Za Eclipse Ide Naziva Adt (Eng. Android Development Tools Plugin) Koji Omogućuje Jednostavni Pristup Logcatu, Uređivaču Datoteke Androidmanifest.Xml, Kontrolu Dolaznih Poziva, Sms Simulaciju I Slično;
* Emulator - Služi Za Izvršavanje Progama Na Računalu;
* Ddms (Eng. Dalvik Debug Monitoring Service) - Služi Za Kontrolu I Nadzor Pronalaženja I Uklanjanja Pogrešaka U Aplikacijama;
* Aapt (Eng. Android Asset Packaging Tool) - Koristi Se Za Stvaranje I Distribuciju Androidovog Programskog Paketa U .Apk Formatu;
* Adb (Eng. Android Debug Bridge) - Klijentsko-Poslužiteljska Aplikacija Za Instaliranje I Pokretanje Datoteka .Apk Na Emulatoru Ili Uređaju, Te Pristup Komandnom Retku Uređaja. Koristi Se I Za Povezivanje Standardnog Programa Za Pronalaženje I Uklanjanje Pogrešaka (Eng. Debugger) S Kodom Na Emulatoru Ili Uređaju;
* Detaljnu Dokumentaciju;
* Primjerke Koda - Jednostavni Primjerci Koda Za Demonstraciju Korištenja Određenih Api-Ja, Te Mogućnosti Koje Pruža Android.

3.1 Android Emulator

U Android Sdk Uključen Je Emulator Pokretnog Uređaja Koji Omogućava Razvoj I Testiranje Aplikacija. Jedino Ograničenje Emulatora Je Nemogućnost Uspostave Dolaznih I Odlaznih Poziva. Korištenjem Adt-A Kao Programskog Dodatka Za Eclipse Emulator Se Automatski Poziva Prilikom Izvođenja I Procesa Pronalaženja I Uklanjanja Programskih Pogrešaka. Izgled Grafičkog Sučelja Jednog Od Emulatora Prikazan Je Na Slici 3.1.

Android Emulator Je Aplikacija Temeljena Na Qemu (Eng. Open Source Processor Emulator) S Virtualnim Pokretnim Uređajem Temeljenom Na Procesoru Arm (Eng. Advanced Risc Machine). Qemu Je Strojni Emulator Temeljen Na Otvorenom Kodu. Uključuje Cjeloviti Android Sistemski Stog (Eng. System Stack) Sve Do Razine Jezgre Sustava, Skup Unaprijed Instaliranih Aplikacija (Npr. Birač Brojeva) Kojima Se Može Pristupati Korištenjem Vlastitih Aplikacija, Korisničko Grafičko Sučelje, Prilagodljivo Dodjeljivanje Vrijednosti Tipkama, Te Mnoštvo Naredbi I Mogućnosti Za Kontrolu Ponašanja Emuliranog Okruženja.

3.2 Dalvik Debug Monitoring Service (Ddms)

Ddms Je Alat Koji Omogućava Upravljanje Procesima Na Emulatoru Ili Pokretnom Uređaju, Te Pomaže Prilikom Pronalaženja I Uklanjanja Pogrešaka U Programskom Kodu. Pruža Mogućnost Preusmjeravanja Vrata (Eng. Port Redirection), Informacije O Stanju Dretvi I Gomila Na Uređaju, Upotrebu Logcata (Naredba Koja Komprimirane Datoteke Evidencije Događaja Ispisuje U Oovjeku Čitljivom Obliku), 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. Kao Što Je Već Spomenuto, Na Sustavu Android Svaka Aplikacija Pokreće Se Kao Zasebni Proces Koji Se Izvodi Kao Zasebna Instanca Virtualnog Stroja Dalvik. Svaki Od Tih Procesa Povezan Je S Programom Za Pronalaženje I Uklanjanje Programskih Pogrešaka Preko Jedinstvenih Vrata Počevši Uzlazno Od Vrata 8600. Ddms Se Također Spaja I Na Pretpostavljena Lokalna Vrata 8700.

Prilikom Pokretanja Ddms Se Spaja Na Adb Koji Mu Prosljeđuje Informaciju O Tome Da Li Je Uređaj Spojen Ili Odspojen. Kada Je Uređaj Spojen Kreira Se Vm (Eng. Virtual Machine) Nadzorna Usluga Između Adb-A I Ddms-A Čija Je Uloga Obavještavanje Ddms-A Da Li Je Virtualni Stroj Na Uređaju Pokrenut Ili Ugašen. Prilikom Pokretanja Virtualnog Stroja Ddms Posredstvom Adb-A Dobavlja Njegov Jedinstveni I'd I Otvara Vezu Prema Programu Za Pronalaženje I Uklanjanje Pogrešaka Preko Adbd-A (Eng. Adb Deamon) Na Uređaju. Na Taj Način Ostvarena Je Veza Između Ddms-A I Virtualnog Stroja Korištenjem Uobičajenog Wire Protokola.

3.3 Android Debug Bridge (Adb)

Adb Je Klijentsko-Poslužiteljski Alat Koji Služi Za Povezivanje S Androidovim Emulatorom Ili Uređajem. Uključuje Tri Komponente:

* Klijent - Aplikacije (Npr. Ddms) Koje Komuniciraju S Pozadinskim Procesom Preko Poslužitelja,
* Poslužitelj - Pozadinski Proces Koji Se Pokreće Na Računalu Na Kojem Razvijamo Aplikaciju I Upravlja Komunikacijom Između Klijenta I Pozadinskog Procesa Adb-A Koji Se Izvršava Na Emulatoru, Te
* Pozadinski Proces Adb-A Na Emulatoru.

Prilikom Pokretanja Adb-Ovog Klijenta Prvo Se Provjerava Postoji Li Već Aktivni Adb-Ov Poslužitelj. Ukoliko Ne Postoji, Pokreće Se I Spaja Na Lokalna Tcp Vrata 5037 Na Kojima Osluškuje Naredbe Poslane Od Strane Adb Klijenta. Svi Adb Klijenti Koriste Vrata 5073 Za Komunikaciju S Adb Poslužiteljem.

Nakon Toga Se Uspostavlja Komunikacija Poslužitelja I Svih Instanci Emulatora Ili Uređaja. To Se Ostvaruje Na Način Da Poslužitelj Pregledava Sva Neparna Vrata U Rasponu Od 5555 Do 5585. Ukoliko Adb Poslužitelj Prilikom Pretraživanja Na Određenim Neparnim Vratima Naiđe Na Pozadinski Proces Adb-A Uspostavlja Se Veza Na Tim Vratima. Pretraživanje Samo Neparnih Vrata Vrši Se Iz Razloga Š To Je Za Uspostavu Veze Potreban Jedan Par Vrata I to Jedna Vrata Za Vezu S Konzolom I Jedna Vrata Za Vezu Emulatora Ili Uređaja S Adb-Om. Tako Se, Na Primjer, Konzoli Emulatora Povezanog S Adb Poslužiteljem Na Vratima 5555 Pristupa Preko Vrata 5554.

Nakon Uspostave Veze Moguće Je Kontrolirati Emulator Različitim Adb Naredbama.

4. Koncept Aplikacije

Sve Androidove Aplikacije Pisane Su U Programskom Jeziku Java. Aplikacijom Se Smatra Kompletni Kod Paketa Upakiran Pomoću Aapt Alata Koji Kao Rezultat Stvara Datoteku Sufiksa .Apk. Upravo Se U Ovom Formatu Aplikacije Distribuiraju Na Pokretni Uređaj.

Većina Stvorenih Aplikacija Pripada U Jednu Od Slijedećih Kategorija [4]:

* Aktivnost Koja Se Izvršava U Prvom Planu (Eng. Foreground Activity) - Aplikacije Bez Korisne Pozadinske Aktivnosti (Npr. Igre Ili Kartografija). Kada Su Izvan Fokusa Aplikacije Ove Vrste Se Uglavnom Privremeno Zaustavljaju;
* Pozadinska Usluga (Eng. Background Service) - Aplikacije Ograničene Interakcije S Korisnikom Koje Se Uglavnom Izvršavaju U Pozadini (Npr. Aplikacija Za Upravljanje Dolaznim Pozivima);
* Aktivnost S Prekidima - Aplikacije Koje Podrazumijevaju Određeni Stupanj Interakcije S Korisnikom, Ali Se Uglavnom Odvijaju U Pozadini (Npr. Mp3 Svirač (Eng. Mp3 Player)).

Za Razliku Od Većine Aplikacija Na Drugim Sustavima, Androidove Aplikacije Nemaju Samo Jednu Pokretnu Točku (Ovo Konkretno Znači Da Ne Postoji Samo Jedna Main() Funkcija). Razlog Tome Je Karakteristika Sustava Koja Zastupa Ideju Međusobne Interakcije Dvije Ili Više Različitih Aplikacija, A Da Bi to Bilo Ostvarivo Sustav Mora Biti U Mogućnosti Pokrenuti Proces Pojedine Aplikacije I Kada Se Zatraži Rad Samo Nekog Njenog Određenog Dijela, Te Instancirati Javin Objekt Za Dotični Dio.

4.1 Osnovna Struktura

Postoje Četiri Osnovne Komponente Koje Čine Aplikaciju [2]:

* Aktivnost (Eng. Activity) - Predstavlja Komponentu Aplikacije Koja Se Uglavnom Može Poistovjetiti S Jednim Konkretnim Prozorom Aplikacije U Kojem Je Korisnik U Mogućnosti Izvršiti Određenu Radnju. Aplikacija Može Sadržavati Jednu Ili Više Definiranih Aktivnosti, Pri Č Emu Je Jedna Od Aktivnosti Uvijek Definirana Kao Primarna Aktivnost. Prijelaz Između Aktivnosti Odvija Se Tako Što Aktualna Aktivnost Invocira Novu. Iako Više Aktivnosti Tvori Jedno Kompaktno Korisničko Sučelje Treba Imati Na Umu Da Su One Međusobno Nezavisne. Svaka Aktivnost Implementira Se Kao Zasebna Klasa Koja Nasljeđuje Klasu Activity, Te Je Sama Odgovorna Za Čuvanje Svog Stanja U Životnom Ciklusu Aplikacije;
* Namjera (Eng. Intent) - Omogućava Prijelaz Između Zaslona Aplikacija. Ustvari Predstavlja Namjeru Za Obavljanjem Određene Radnje. Ovdje Ćemo Još Spomenuti Broadcast Reciever Koji Predstavlja Dio Koda Koji Se Pokreće Isključivo Ukoliko Je Aktiviran Nekim Vanjskim Faktorom I Intent Filter Koji Opisuje Koje Namjere Je Aplikacija U Stanju Izvršiti;
* Usluga (Eng. Service) - Predstavlja Proces Bez Vidljive Korisničke Interakcije. Uglavnom Se Izvršava U Pozadini Nedefinirani Period Vremena. Služi Za Obnavljanje Podatkovnih Resursa, Vidljivih Aktivnosti I Signalizacijskih Obavijesti. Svaka Usluga Nasljeđuje Klasu Service;
* Pružatelj Sadržaja (Eng. Content Provider) - Omogućava Uzajamno Korištenje Podataka Između Različitih Aplikacija I Njihovih Procesa. Pružatelj Sadržaja Nasljeđuje Klasu Contentprovider.

Aplikacija Ne Mora Sadržavati Sve Spomenute Komponente, A Isto Tako Može Sadržavati I Neke Druge.

V

4.2 Životni Ciklus Aplikacije

Slika 4.1: Stog aktivnosti

Kod Klasične Radne Površine Operacijskog Sustava Windows Ili Linux Postoji Jedan Aktivni Prozor I Ravnopravni Niz Ostalih Aplikacija, A Kontrolu Ž Ivotnog Ciklusa Vrši Korisnik. Android Sam Brine O Ž Ivotnom Ciklusu Aplikacija, Prikaz Se Vrši Na Principu Lifo (Eg. Last in First Out) Stoga Na Koji Se Spremaju Aktivnosti Pokrenutih Aplikacija Kao Što Je Prikazano Na Slici 4.1 [4].

Svaki Zaslon Korisničkog Sučelja Prikazan Je Pomoću Klase Activity. Aplikaciju Čine Jedna Ili Više Aktivnosti. Vrlo Je Važno Napomenuti Da Aplikacija Nije Isto Što I Proces Što Bitno Utječe Na Ponašanje I Životni Ciklus Aplikacije.

Kako Bi Se Očuvala Efikasnost Cijelog Sustava, Procesi I Njihove Aplikacije Će, Ukoliko Je Potrebno, Biti Ugašeni Bez Upozorenja S Ciljem Oslobađanja Memorijskih Resursa Za Aplikacije Višeg Prioriteta (Uglavnom One Koje Korisnik Koristi U Tom Trenu). Slika 4.2 Prikazuje Stablo Prioriteta Procesa [4].



Slika 4.2: Stablo Prioriteta

Definiramo Slijedeće Vrste Prioriteta:

Aktivni Procesi - Ovakvih Procesa Je Uglavnom Malo I Imaju Vrlo Visoki Prioritet Iz Razloga Š To Podržavaju Aplikacije S Kojima Korisnik Ostvaruje Aktivnu Interakciju. Neki Od Ovih Procesa Uključuju Vidljive Aktivnosti,

Broadcast Recievere Koji Izvršavaju Onreceive() Metodu, Te Usluge Koje Izvršavaju Onstart(), Oncreate() Ili Ondestroy() Metodu; Vidljivi Procesi - Predstavljaju Vidljive, Ali Neaktivne Procese. Uglavnom Ih Je Vrlo Malo, Te Se Gase U Krajnjoj Nuždi;

Pokrenuti Uslužni Procesi - Procesi Pokrenutih Usluga S Kojima Korisnik Nema Aktivnu Interakciju, Te Iz Tog Razloga Imaju Nešto Niži Prioritet. I Dalje Ih Se Smatra Aktivnim Procesima, Te Će Biti Ugašeni Samo U Nuždi;

* Pozadinski Procesi - Predstavljaju Procese Nevidljivih Aktivnosti Bez Pokrenutih Usluga. U Pravilu Ih Je Veliki Broj, Te Se Gase Po Principu "Zadnji Viđen Prvi Ubijen" (Lsfk, Eng. Last Seen First Killed);
* Prazni Procesi - Predstavljaju Zapis Pokrenutih Procesa U Privremenoj Memoriji S Ciljem Smanjenja Vremena Ponovnog Pokretanja.

Na Životni Ciklus Aplikacije Utječe Isključivo Sustav, A Ne Sama Aplikacija. Stanje U Kojem Se Nalazi Aplikacija Određuje Njezinu Prioritetnu Razinu. Aktivnost Aplikacije Može Se Nalaziti U Nekom Od Sljedećih Stanja Prikazanih Slikom 4.3 [1]:

* Aktivno - Vidljiva Aktivnost U Fokusu;
* Privremeno Zaustavljeno - Vidljiva Aktivnost Izvan Fokusa;
* Zaustavljeno - Nevidljiva Aktivnost Izvan Aktivne Upotrebe. U Memoriji Se Čuva Njezino Stanje I Ostale Informacije;
* Neaktivno - Stanje Aktivnosti Nakon Gašenja, A Prije Pokretanja.



Slika 4.3: Stanja Aktivnosti

Prijelazi Između Pojedinih Stanja Nevidljivi Su Krajnjem Korisniku, Ali Kontrolirani Na Razini Sustava Upotrebom Odgovarajućih Nadjačavajućih (Eng. Override) Metoda:

Oncreate(Bundle) - Poziva Se Prilikom Pokretanja Aplikacije, Onstart() - Naznačuje Početak Prikaza Aplikacije Korisniku, Onresume() - Poziva Se Prilikom Početka Ili Nastavka Interakcije S Korisnikom, Onpause() - Poziva Se Prilikom Prebacivanja U Pozadinski Način Rada, Onstop() - Poziva Se U Slučaju Dužeg Perioda Nekorištenja Aplikacije, Onrestart() - Naznačuje Ponovo Pozivanje Aplikacije Iz Zaustavljenog Stanja, Ondestroy() - Poziva Se Trenutak Prije Gašenja Aplikacije,

Onsaveinstancestate(Bundle) - Opcionalna Metoda Koja Se Poziva U Slučaju Čuvanja Stanja Instance I

Onrestoreinstancestate(Bundle) - Poziva Se Prilikom Reinicijalizacije Aktivnosti Iz Stanja Prethodno Pohranjenog Onsaveinstancestate() Metodom.

4.3 Korisničko Sučelje

Postoje Dva Načina Dizajniranja Korisničkog Sučelja: Proceduralno I Deklarativno. Proceduralni Dizajn Odnosi Se Na Pisanje Java Koda, A Deklarativni Na Pisanje Xml (Eng. Extensible Markup Language) Koda. U Praksi Se Za Kreiranje Grafičkog Korisničkog Sučelja Uglavnom Koristi Xml.

Kreiranjem Sučelja Aktivnosti Dobivaju Svoju Funkcionalnost, Tj. Vidljivost Na Zaslonu Uređaja I Na Taj Se Način Omogućava Interakcija S Korisnikom. Osnovne Jedinice Korisničkog Sučelja Su Objekti Pogled (Eng. View) I Grupe Pogleda (Eng. Viewgroup) [2]:

* Pogled - Objekt Čija Podatkovna Struktura U Sebi Nosi Zapis Izgleda I Sadržaja Određenog Pravokutnog Područja Na Zaslonu, Te Upravlja Iscrtavanjem Elemenata, Pomicanjem Sadržaja Na Zaslonu (Eng. Scrolling) I Ostalim Faktorima Koji Utječu Na Izgled Definiranog Dijela Zaslona. U Hijerarhijskom Stablu Objekti Pogled Su Listovi Stabla. Android Raspolaže S Već Gotovim Skupovima Objekata Ove Vrste Kao Što Su Gumbi, Kvadratići Za Odabir (Eng. Checkbox) I Slično. Ovi Objekti Nasljeđuju Klasu View;
* Grupe Pogleda - Posebna Vrsta Objekta Pogled Koja Sadrži I Upravlja Skupinom Zavisnih Objekata Pogleda I Grupe Pogleda Čime Je Omogućena Kompleksnost Prikaza Korisničkog Sučelja. Objekti Ove Vrste Su Inastance Klase Viewgroup.

Hijerarhijska Ovisnost Objekata Pogleda I Grupe Pogleda Prikazana Je Slikom 4.4. Iscrtavanje Elemenata Stabla Započinje Od Korijena Stabla Tako Š To Aktivnost Prvo Poziva Svoju Setcontentview() Metodu I Android Sustavu Predaje Referencu Na Korijenski Objekt. Svaki Podčvor Iscrtava Se Sam Pozivanjem Draw() Metode I to Pod Uvjetom Da Čvor Sam Postavlja Zahtjev Za Lokacijom I Veličinom, Ali Roditeljski Č Vor (Grupa Pogleda) Donosi Konačnu Odluku O Veličini Prostora Za Iscrtavanje Podčvora I Njegovoj Lokaciji Na Zaslonu. Svaka Grupa Pogleda Je Odgovorna Za Renderiranje Prikaza Svojih Podčvorova.

4.3.1 izbornik

Za Pristup Izborniku Aplikacije Najčešće Se Koristi Tipka Menu Koja Se Nalazi Na Samom Uređaju (Ukoliko Je Hardverski Podržano) Ili Se Izbornik Može Prikazati Ukoliko Se Odabere Neka Od Komponenti Aplikacije. Strukturu Izbornika, Koja Je Hijerarhijska, Nije Potrebno Ručno Konfigurirati Nego Se Isto Postiže Definiranjem Povratnih (Eng. Callback) Metoda Oncreateoptionsmenu() Ili Oncreatecontextmenu() S Popisom Stavki Izbornika Za Određenu Aktivnost.

Upravljanje Događajima Također Se Odvija Automatski Pozivanjem Metoda Onoptionsitemselected() Ili Oncontextitemselected() Od Strane Sustava.

4.3.2 događaji Unutar Korisničkog Sučelja

Kako Bi Se Znalo Da Je Korisnik Uopće Odabrao Neku Od Aktivnosti Ili Stavku U Izborniku Sučelja Potrebno Je Definirati Način Na Koji Korisnik Vrši Interakciju S Određenim Komponentama Sučelja. Postoje Dva Načina:

* Definiranje Slušača Događaja (Eng. Event Listener) - Najčešći Način Osluškivanja Događaja. View Klasa Sadrži Kolekciju Ugniježđenih Sučelja On<event>listener Od Kojih Svako Sadrži Povratnu Metodu On<event>() Pri Čemu Događaj Može Biti Ili Odabir Određenog Elementa Ili Dodir Ekrana Ili Nešto Treće;
* Nadjačavanje (Eng. Override) Postojeće Povratne Metode Za View Klasu - Koristi Se U Slučaju Implementiranja Vlastite View Klase I Osluškivanja Posebnih Događaja Koji Se U Njoj Pojavljuju.

4.3.3 dodatne Mogućnosti

Osim Osnovnih Komponenti Aplikacija Može Sadržavati Mnoštvo Dodatnih Elemenata I Njihovih Varijacija. Spomenut Ćemo Adaptere, Te Stilove I Teme.

Adapteri Su Vrsta Podklase Viewgroup Koja Se Koristi Za Prikaz Gotovih Pohranjenih Podataka (Za Razliku Od Iscrtavajućih Grupa I Nizova).

Kao Dio Aplikacijskog Sustava Postoje Već Gotovi Stilovi I Teme, No Također Je Moguće Stvoriti I Vlastite. Stil Je Skup Jednog Ili Više Atributa Za Oblikovanje Zasebnih Elemenata U Prikazu (Npr. Definiranje Određene Veličine Prikaza I Boje Slova). Tema Je Skup Jednog Ili Više Atributa Za Oblikovanje Koji Se Primjenjuju Na Jednu Ili Sve Aktivnosti U Aplikaciji (Npr. Definiranje Određenih Pozadinskih Boja, Izgleda Teksta Za Određenu Aktivnost).

4.4 Datoteka Androidmanifest.Xml

Svaka Aplikacija Obavezno Mora Sadržavati Datoteku Androidmanifest.Xml Koja Ju Opisuje. Ova Datoteka Nalazi Se U Korijenskom Direktoriju Paketa, Te Sadrži Slijedeće Informacije O Aplikaciji:

Naziv Paketa -Služi Kao Jedinstveni Identifikator Aplikacije,

Opis Komponenti - Aktivnosti, Usluge, Pružatelji Sadržaja, Filtere Namjera (Eng.

Intent Filters), Broadcast Recievers Itd.,

Odredbe O Tome Koji Će Procesi Sadržavati Programske Komponente, Deklaracija Dozvola Za Pristup Aplikacijskim Komponentama Od Strane Drugih Aplikacija,

Minimalna Razina Android Api-Ja Koju Zahtjeva Aplikacija,

Popis Biblioteka S Kojima Aplikacija Treba Biti Povezana I

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.

<?xml Version="1.0" Encoding=mutf-8m?> <manifest>

<uses-Permission /> <permission /> <permission-Tree /> <permission-Group /> <instrumentation /> <uses-Sdk /> <application> <activity> <intent-Filter> <action /> <category /> <data /> </intent-Filter> <meta-Data /> </activity> <activity-Alias>

<intent-Filter> . . . </intent-Filter> <meta-Data /> </activity-Alias>

<service>

<intent-Filter> . . . </intent-Filter>

<meta-Data/> </service> <receiver>

<intent-Filter> . . . </intent-Filter>

<meta-Data /> </receiver> <provider>

<grant-Uri-Permission />

<meta-Data /> </provider> <uses-Library /> </application>

</manifest>

Tablica 4.1: Struktura Androidmanifest.Xml Datoteke

U Tablici 4.1 [7] Prikazani Su Osnovni Elementi Datoteke Androidmanifest.Xml Čije Značenje Sljedeće:

* <action> - Opisuje Akciju Dodijeljenu <intent-Filter> Elementu. Obavezan Dio <intent-Filter> Polja. Ukoliko Se Ne Definira Intent Objekti Ne Prolaze Kroz Filter;
* <activity> - Obavezni Deklarirajući Element Za Svaku Aktivnost Aplikacije. Podržava Podelement <intent-Filter> Kojim Se Specificiraju Pojedine Namjere Aktivnosti. Pokretanje Aktivnosti Koja Nije Definirana U Androidmanifest.Xml Datoteci Rezultira Dojavom Greške Pri Izvođenju (Eng. Runtime Error);
* <activity-Alias> - Zamjenska Aktivnost. Ciljanu Aktivnost Predstavlja Kao Nezavisni Entitet S Vlastitim Skupom Filtera Namjera Koji Umjesto Definiranja Namjera Za Izravno Pokretanje Ciljane Aktivnosti Određuju Namjere Koji Ju Pokreću Preko Zamjenske Aktivnosti;
* <application> - Korijenski Element S Opisom Dubine Stabla Kojeg Čine Komponente Paketa. Također Može Sadržavati Globalne I/ili Unaprijed Zadane Aplikacijske Atribute Kao Što Su Tema, Ikona, Dozvole Itd. Manifest Datoteka Može Sadržavati Samo Jedan Element Ove Vrste;
* <category> - Vrsta Kategorije Intent Filtera. Sadržan U Bloku <intent-Filter>;
* <data> - Određuje Vrstu Podatka Intent Filtera. To Mogu Biti Opcionalni, Ali Međusobno Zavisni Atributi Mime Ili Uri. Sadržan U Bloku <intent-Filter>;
* <grant-Uri-Permission> - Određuje Garantirane Dozvole Podatkovnih Podskupova Pružatelja Sadržaja. Sadržan U Bloku <provider>;
* <instrumentation> - Deklarira Koje Su Instrumentacijske Komponente Raspoložive Za Testiranje Funkcionalnosti Paketa. Sadržan U Bloku <manifest>;
* <intent-Filter> - Određuje Vrstu Namjere Na Koju Reagiraju Aktivnost, Usluga Ili Broadcast Receiver. Sadržan U Blokovima <activity>, <activity-Alias>, <service> I <receiver>;
* <manifest> - Definira Komponente, Sigurnosne Postavke I Testne Klase Koje Čine Aplikaciju. Ovo Je Korijenski Element Androidmanifest.Xml Datoteke;
* <meta-Data> - Par Ime-Vrijednost Dodatnih Metapodataka Koji Mogu Biti Isporučeni Roditeljskoj Komponenti. Sadržan U Blokovima <activity>, <activity-Alias>,<service> I <receiver>;
* <permission> - Deklarira Sigurnosne Dozvole Nad Komponentama Kojima Pristupaju Druge Aplikacije;
* <permission-Group> - Određuje Grupaciju Povezanih Dozvola;
* <permission-Tree> - Određuje Stablo Dozvola;
* <provider> - Definira Pružatelja Sadržaja. Sadržan U <application>;
* <receiver> - Definira Broadcast Reciever Koji Omogućuje Prihvat Namjera I U Slučaju Kada Aplikacija I Njene Komponente Nisu Pokrenute;
* <service> - Definira Usluge. Sadržan U <application> Bloku, A Može Sadržavati <intent-Filter> I <meta-Data>;
* <uses-Library> - Specificira Biblioteku S Kojom Aplikacija Mora Biti Povezana. Sadržan U <application> Bloku;
* <uses-Permission> - Dio Sigurnosnog Modela Koji Sadrži Sigurnosne Dozvole Dodijeljene Od Strane Korisnika, A Odnose Se Na Funkcije Aplikacije. Dozvole Se Definiraju Prilikom Instaliranja Aplikacije;
* <uses-Sdk> - Definira Razine Android Api-Ja Na Kojima Se Može Izvoditi Aplikacija.

Dozvole Se Definiraju Zbog Hardverski Ograničenog Pristupa Memoriji Jednog Procesa Od Strane Drugog, Te Zato Što Je Svakoj Aplikaciji Dodijeljen Specifični Korisnički Id. Neke Od Najčešće Korištenih Dozvola Su:

* Internet - Pristup Internetu,
* Readcontacts - Dozvola Za Čitanje Podataka Kontakata,
* Writecontacts - Dozvola Za Pisanje Podataka Kontakata,
* Receivesms - Nadzor Pristigle Sms Poruke,
* Access Coarse Locator - Grubo Bežično Lociranje Ili Lociranje Preko Bazne Stanice I
* Access Fine Location - Finije Lociranje Korištenjem Npr. Gps-A.

Uređivanje Androidmanifest.Xml Može Biti Ručno Pisanjem Xml Koda Ili Korištenjem Uređivača Koji Dolazi Kao Dio Adt-A.

4.5 Podatkovni Resursi

Pod Pojmom Podatkovnih Resursa Podrazumijevaju Se Vanjski Podaci (Slike, Video, Teme, Xml Kod I Slično) Koji Se Koriste U Aplikaciji, A Nisu Dio Programskog Koda. Podatkovni Resursi U Aplikacijskom Paketu Smješteni Su U Vlastitom Direktoriju Pod Nazivom Res/. Ovo Odjeljivanje Se Prvenstveno Radi Zbog Lakšeg Upravljanja, Ažuriranja I Manipulacije Podacima. Podatkovni Resursi Prevode Se Pomoću Alata Aapt Ovisno O Tipu I Poddirektoriju U Kojem Se Nalaze. Ovisnost Tipa Podatkovnog Resursa I Lokacije U Paketu Prikazana Je U Tablici 4.2.

Važno Je Napomenuti Da Se Xml Datoteke Pretvaraju U Binarni Zapis Kako Bi Se Zbog Veće Efikasnosti Mogle Brže Učitati.

|  |  |
| --- | --- |
| Lokacija | Podatkovni Resurs |
| /res/layout/ | Datoteke Prikaza |
| /res/drawable/ | Slike |
| /res/anim/ | Animacije |
| /res/values/ | Jednostavne Vrijednosti (Stilovi, Konstantni Nizovi, Boje...) |
| /res/raw/ | Sirovi (Eng. Raw) Podaci (Mp3 Ili Video) |

Tablica 4.2: Tipovi Podatkovnih Resursa I Njihova Lokacija U Paketu

Dozvoljeni Znakovi U Zapisu Podatkovnih Resursa Su Mala Slova, Brojevi, Točka I Podvlaka. Prilikom Prevođenja Stvara Se Posebna Klasa Naziva R Koja Sadrži Identifikatore Podatkovnih Resursa Za Razliku Od Standardnih Java Podatkovnih Resursa Koji Su Referencirani Po Ključevima Tipa String. Ova Klasa Sastoji Se Od Niza Podklasa, Svaka Za Različitu Vrstu Podataka.

4.5.1 Upotreba Resursa U Kodu

Podaci Se U Kodu Pozivaju Preko Identifikatora Sintaksama R.Resource Type.Resource Name Ili Android.R.Resource Type.Resource Name Pri Eemu Nije Potrebno Navoditi Ime Paketa. Resource Type Je Podklasa Koja Sadrži Određenu Vrstu Podatkovnog Resursa, A Resource Name Je Ime Resursovog Atributa Definiranog U Xml Datoteci Ili Ime Datoteke, Bez Sufiksa, Za Podatkovne Resurse Definirane Od Strane Nekog Drugog Podatkovnog Tipa. Razlika Između Ove Dvije Spomenute Sintakse Je U Tome Što Se Sintaksa Android.R.Resource Type.Resource Name Koristi Za Već Postojeće Podatkovne Resurse U Sustavu.

4.5.2 Upotreba Resursa U Drugim Podatkovnim Resursima

Podaci Se Mogu Referencirati I Iz Drugih Resursa (Npr. Xml Datoteke). Ovo Se Ponajviše
koristi Prilikom Izrade Stilova I Datoteka Prikaza. Za Referenciranje Jednog Resursa Iz
drugog koristi se notacija @ u sintaksi

Attribute="@[Packagename:]Resourcetype/resourceidentifier ".

4.5.3 Upotreba Sistemskih Resursa

Odnosi Se Na Upotrebu Resursa Koje Već Koriste Neke Izvorne Aplikacije. Korištenje Ovih Podatkovnih Resursa Sli No Je Korištenju Vlastitih Resursa S Razlikom Da I'm Se Pristupa U Klasi Android.R.

5. Povezivanje S Poslužiteljem

Komunikacija Pokretnog Uređaja I Udaljenog Poslužitelja Odvija Se Na Razini Arhitekture Klijent - Poslužitelj (Eng. Client - Server).

Postoje Razli Iti Na Ini Kojima Se Može Ostvariti Komunikacija Između Klijenta I Poslužitelja. Izbor Određene Kombinacije Protokola I Na Ina Povezivanja Ovisi O Vrsti Usluge.

Razmatranje Na Ina Povezivanja Zapo Et Ć Emo Na Transportnom Sloju Usporedbom Protokola Udp (Eng. User Datagram Protocol) I Tcp (Eng. Transmission Control Protocol). Transportni Sloj Rješava Nedostatke Ip (Eng. Internet Protocol) Sloja. Sadrži Rješenja Za Kontrolu Pogrešaka I Kontrolu Toka. Izbor Protokola Transportnog Sloja Određuje Se Ovisno O Korisnikovim Zahtjevima. Transportni Sloj Omogućava Više Transportnih Veza Preko Jedne Mrežne Veze Tzv. Multipleksiranje. Takvu Konekciju Definiraju Dvije Priključnice (Eng. Socket) Koje Se Sastoje Od Tri Parametara: Vrste Protokola (Udp Ili Tcp), Ip Adrese I Broja Vrata (Eng. Port).

Nakon Transportnog Sloja Promatrat Ć Emo Povezivanje Korištenjem Protokola Aplikacijskog Sloja, Konkretno Http (Eng. Hyper Transfer Protocol) Konekciju U Sklopu Čega Ćemo Obradit Parsiranje Jednostavnog Xml Dokumenta Koji Se Nalazi Na Udaljenom Poslužitelju.

5.1 Protokol Udp

Udp Je Standardni Internetski Protokol Opisan U Specifikaciji Rfc 768. To Je Jednostavni Protokol Koji Osigurava Beskonekcijski Orijentiranu Uslugu S Nepouzdanim Prijenosom Podataka, Bez Kontrole Toka I Kontrole Zagušenja. Paketi Nisu Numerirani, A Zaštitna Suma U Zaglavlju Nije Obavezna Tako Da Se Ne Vrši Provjera Pogreške Prilikom Prijenosa. O Pouzdanosti Prijenosa, Ako Je Potrebno, Brine Sama Aplikacija. Ukoliko Iz Nekog Razloga Dođe Do Odbacivanja Paketa, Ne Vrši Se Dojava O Grešci. Udp Protokol Zapravo Služi Kao Omotač (Eng. Wrapper) Između Aplikacije I Ip Protokola.

5.1.1 Aplikacija Koja Koristi Udp Komunikaciju

U Nastavku Je Prikazan Primjer Jednostavne Udp Komunikacije Android Klijenta S Udaljenim Poslužiteljem. Aplikaciju Čine Dvije Klase: Klasa Aktivnosti Udpconnection I Klasa Koja Predstavlja Klijent. Klijent Je U Mogućnosti Poslati I Primiti Jedan Udp Paket. Za Potrebe Testiranja Aplikacije Udp Poslužitelj Ostvaren Je Pomoću Programa Netcat Na Udaljenom Poslužitelju Sintaksom Nc -U -L -P 6666.

Kod 5.1: Klasa Udpexample (Main Activity)

Package Hr.Fer.Tel.Android;

Import Android.App.Activity; Import Android.Os.Bundle;

Public Class Udpexample Extends Activity {

/\*\* Called When the Activity Is First Created. \*/

@override

Public Void Oncreate(Bundle Icicle) { Super.Oncreate(Icicle); Setcontentview(R.Layout.Maiw);

// Start the Client Client C = New Client(); New Thread(C).Start();

Klasa Udpexample Obavlja Inicijalizaciju Udp Klijenta. Metoda Oncreate() U Klasi Udpexample Nadjačava Metodu Istog Naziva Iz Klase Activity. Ako Se Aktivnost Ponovo Inicijalizira U Varijabli Icicle Nalazit Ć E Se Podaci Prethodno Spremljeni Metodom Onsavelnstancestate(Bundle). Pozivom Metode Setcontentview(R.Layout.Main) Definira Se Prikaz Na Zaslonu. Zatim Se Klijent Inicijalizira I Pokreće U Novoj Dretvi.

Kod 5.2: Klasa Client

Package Hr.Fer.Tel.Android;

Import Java.Net.Datagrampacket; Import Java.Net.Datagramsocket; Import Java.Net.Inetaddress;

Import Android.Util.Log;

Public Class Client Implements Runnable {

Public Static Final String Serverip = "161.53.71.194"; // Remote Host Ip

Public Static Final Int Serverport = 6666;

@override

Public Void Run() { Try {

// Retrieve the Servername Inetaddress Serveraddr = Inetaddress.Getbyname(Serverip);

Log.D("Udp", "C: Connecting...");

/\* Create New Udp Socket \*/

Datagramsocket Socket = New Datagramsocket(); // Sending Packet

Byte[] Bufs = ("Hello From Client!\n").Getbytes();

/\* Create Udp Packet \*/

Datagrampacket Packets = New Datagrampacket(Bufs,

Bufs.Length,

Serveraddr, Serverport); Log.D("Udp", "C: Sending: '" + New String(Bufs) +

);

/\* Sending Out the Packet \*/

Socket.Send(Packets);

Log.D("Udp", "C: Sent.");

// Receiving Packet

Byte[] Bufr = New Byte[10];

Datagrampacket Packetr = New Datagrampacket(Bufr,

Bufr.Length);

/\* Receiving the Packet \*/ Socket.Receive(Packetr); Log.D("Udp", "C: Received: '" + New String(Packetr.Getdata())

+ );

} Catch (Exception E) {

Log.E("Udp", "C: Error", E);

}

}

J

Klasa Client Implemetira Sučelje Runnable Što Je Nužno Kako Bi Se Mogla Odvijati U Posebnoj Dretvi. Komunikacija S Udp Serverom Obavlja Se Korištenjem Klasa Datagramsocket I Datagrampacket. Klasa Inetaddress Koristi Se Za Pohranu Ip Adrese Servera. U Ovom Jednostavnom Primjeru Klijent Šalje Znakovni Niz "Hellofrom Client!\n", Te Prima 10 Okteta Podataka I Sprema Ih U Polje.

5.2 Protokol Tcp

Tcp Je Standardni Konekcijski Orijentirani Internetski Protokol Opisan U Osnovnoj Specifikaciji Rfc 793. Osigurava Uspostavu Konekcije I Dvosmjerni Prijenos Kontinuiranog Niza Podataka Pakiranjem Okteta Podataka U Segmente Koje Prenosi Protokol Mrežnog Sloja. Svakom Oktetu Dodjeljuje Se Slijedni Broj (Eng. Sequence Number) Č Ime Se Osigurava Retransmisija Podataka U Slučaju Gubitka Ili Pogrešnog Prijenosa Na Principu Potvrde Prijama Ispravnog Okteta Na Prijamnoj Strani. Također, Omogućava Simultano Korištenje Od Strane Više Procesa Definiranjem Dodatne Adresne Informacije Tzv. Broja Vrata (Eng. Port Number) Koji Jednoznačno Određuje Ip Korisnika.

5.2.1 Aplikacija Koja Koristi Tcp Komunikaciju

U Nastavku Je Prikazan Kod Jednostavne Tcp Komunikacije Android Klijenta S Udaljenim Poslužiteljem. Klijent Šalje I Prima Po Jedan Paket Podataka.

Za Potrebe Testiranja Aplikacije Tcp Poslužitelj Ostvaren Je Pomoću Programa Netcat Na Udaljenom Poslužitelju Sintaksom Nc -L -P 6666.

Kod 5.3: Klasa Tcpconnection (Main Activity)

Package Hr.Fer.Tel.Android;

Import Android.App.Activity; Import Android.Os.Bundle;

Public Class Tcpconnection Extends Activity {

/\*\* Called When the Activity Is First Created. \*/

@override

Public Void Oncreate(Bundle Icicle) { Super.Oncreate(Icicle); Setcontentview(R.Layout.Maiw);

Thread Cthread = New Thread( New Client());

Cthread.Start();

}

Klasa Tcpconnection Ponaša Se Isto Kao I Klasa Udpexample.

Kod 5.4: Klasa Client

Package Hr.Fer.Tel.Android;

Import Java.Io.Bufferedreader; Import Java.Io.Bufferedwriter; Import Java.Io.Inputstreamreader; Import Java.Io.Outputstreamwriter; Import Java.Io.Printwriter; Import Java.Net.Inetaddress; Import Java.Net.Socket;

Import Android.Util.Log;

Public Class Client Implements Runnable {

Public Static Final String Serverip = "161.53.71.194"; Public Static Final Int Serverport = 6666;

Public Static String Str;

Public Void Run() {

Try {

Inetaddress Serveraddr =

Inetaddress.Getbyname(Serverip);

Log.D("Tcp", "C: Connecting...");

Socket Socket = New Socket(Serveraddr, Serverport); // Sending Message

String Message = "Hello From Client!";

Log.D("Tcp", "C: Sending: '" + Message + "'"); Printwriter Out = New Printwriter( New Bufferedwriter( New

Outputstreamwriter(Socket.Getoutputstream())), True);

Out.Println(Message); Log.D("Tcp", "C: Sent."); Log.D("Tcp", "C: Done.");

// Getting Message

Log.D("Tcp", "C: Getting Message.."); Bufferedreader in = New Bufferedreader(New

Inputstreamreader(Socket.Getinputstream()));

String Str = In.Readline();

System.Out.Println("C: Received: '" + Str + "'");

} Catch (Exception E) {

Log. E("Tcp", "C: Error", E);

J

Kod Tcp Komunikacije Situacija Je Slična Kao I Kod Udp Komunikacije Uz Manje Izmjene. Za Komunikaciju Se Koristi Klasa Socket, A Podaci Se Ne Enkapsuliraju U Datagrame Nego Se Prenose Kao Struja Podataka. Metode Za Rad Sa Strujom Podataka Su Printwriter(), Bufferedwriter() I Outputstreamwriter() Koje Služe Za Slanje, Te Bufferedreader() I Inputstreamwriter() Za Primanje Znakovnog Niza.

5.3 Korištenje Protokola Http

Www (Eng. World Wide Web) Usluga U Mreži Je Izvedena Na Modelu Klijent-Poslužitelj Protokolom Http. Http Je Standardni Internetski Protokol Aplikacijskog Sloja Koji Definira Format I Način Razmjene Poruka Između Klijenta I Poslužitelja. Poruke Koje Se Prenose Formirane Su U Obliku Zahtjeva I Odgovora. Http Asocijacija Ostvaruje Se Preko Tcp Veze. Poslužitelj Konstantno Osluškuje Tcp Zahtjeve Na Dobro Poznatim Vratima 80 (Ukoliko Nije Drugačije Definirano), Dok Klijent Pokreće Tcp Vezu Sa Slobodno Odabranih Vrata Na Lokalnom Računalu Na Ip Adresu Poslužitelja I Tcp Vrata 80. Nakon Uspostave Tcp Veze, Dolazi Do Izmjene Zahtjeva I Odgovora.

5.3.1 Aplikacija Koja Koristi Protokol Http

U Narednim Primjerima Bit Ć E Prikazana Izvedba Uspostave Http Veze I Načini Obrade Jednostavne Xml Datoteke S Udaljenog Poslužitelja.

5.3.1.1 Http Konekcija

Kod 5.5: Metoda Za Ostvarenje Http Veze

Private Inputstream Openhttpconnection(String Urlstring)

Throws Ioexception

{

Inputstream in = Null; Int Response = -1;

Url Url = New Url(Urlstring); Urlconnection Conn = Url.Openconnection();

If (!(Conn Instanceof Httpurlconnection))

Throw New Ioexception("Not an Http Connection");

Try{

Httpurlconnection Httpconn = (Httpurlconnection) Conn; Httpconn.Setallowuserinteraction( False); Httpconn.Setinstancefollowredirects( True); Httpconn.Setrequestmethod("Get"); Httpconn.Connect();

Response = Httpconn.Getresponsecode(); If (Response == Httpurlconnection.Http Ok) { In = Httpconn.Getinputstream();

}

}

Catch (Exception Ex) {

Throw New Ioexception("Error Connecting");

}

Return In;

}

Metoda Openhttpconnection(String Urlstring) Otvara Http Vezu Prema Poslužitelju, Te Vraća Sadržaj Na Adresi Urlstring Kao Struju Podataka. Klase Url I Urlconnection Obavljaju Spajanje Na Zadani Urlstring. Korištenjem Instanceof Operatora Provjerava Se Da Li Je Otvorena Konekcija Uistinu Http. Spajanje Na Url Obavlja Se Korištenjem Http Metode Get. Uspješnost Zahtjeva Provjerava Se Pomoću Koda Odgovora (Eng. Response Code). U Slučaju Uspješnog Zahtjeva Kod Odgovora Mora Imati Vrijednost Httpurlconnection.Httpok, Tj.Brojčanu Vrijednost 200.

5.3.1.2 Xml Parser

Xml Je Jezik Č Ija Je Glavna Namjena Olakšavanje Pohrane I Razmjene Podataka. Sax (Eng. Simple Api for Xml) I Dom (Eng. Document Object Model) Su U Javi Dva Najvažnija Api-A Za Pristup Xml Dokumentima, Također Podržanih U Android Sdk. Sax Api Temelji Se Na Serijskom Č Itanju Xml Dokumenta. Upotrebom Dom Api-A Dokument Se Č Ita Odjednom I Pohranjuje U Memoriju U Obliku Stabla, Gdje Su Čvorovi Stabla Komponente Dokumenta.

Dom Se Najčešće Upotrebljava Kod Aplikacija Gdje Se Određenom Dokumentu Pristupa Uzastopno I Nesekvencijalno, Dok Se Sax Koristi Za Sekvencijalno I Jednoprolazno Čitanje Dokumenta.

U Nastavku Ćemo Obraditi Sljedeći Primjer Xml Dokumenta:

Kod 5.6: Xml Primjer

<?xml Version="1.0" Encoding="Utf-8"?> <popis> <tramvaj Id="1"> <broj>14</broj> <smjer>+</smjer> <n>454600</n> <e>155100</e> </tramvaj>

</popis>

Dom Api

Parsiranje Xml Dokumenta Dom Metodom Ostvaruje Se Pomoću Ugrađenih Klasa Documentbuilder Koja Izvornu Xml Datoteku Pretvara U Objekt Document, Te Documentbuilderfactory Stvara Instance Klase Documentbuilder. Nakon Učitavanja Dokumenta U Document Objekt Moguće Je Izdvojiti I Obraditi Željene Elemente. Prilikom Dom Parsiranja Najprije Se Uspostavlja Http Veza I Preuzima Xml Datoteka. U Nastavku Su Prikazani Važniji Elementi Koda Za Parsiranje Dokumenta.

Kod 5.7: Dio Koda Za Parsiranje Xml-A Dom Metodom

(... )

Doc.Getdocumentelement().Normalize(); // Retrieve All the <item> Nodes

Nodelist Itemnodes = Doc.Getelementsbytagname("Tramvaj");
for (Int I = 0; I < Itemnodes.Getlength(); {

Node Itemnode = Itemnodes.Item(I);

If (Itemnode.Getnodetype() == Node.Element Node){ // Convert the Node Into an Element Element Itemelement = (Element) Itemnode;

String Strnumber = Getinnertext(Itemelement,

"Broj");

String Strdirection = Getinnertext(Itemelement,

"Smjer");

String Strn = Getinnertext(Itemelement, "N"); String Stre = Getinnertext(Itemelement, "E");

Kod 5.8: Dio Koda Za Parsiranje Xml-A Dom Metodom

Private String Getinnertext(Element Itemelement, String Tagname){

String Result = "";

// Get All the <tagname> Element Under the <itemelement> Nodelist Numbernodes = Itemelement.Getelementsbytagname(Tagname);

// Convert a Node Into an Element

Element Numberelement = (Element) Numbernodes.Item(0);

// Get All the Child Nodes Under the <broj> Element Nodelist Textnodes = ((Node) Numberelement).Getchildnodes();

// Retrieve the Text of the <tagname> Element Result = ((Node) Textnodes.Item(0)).Getnodevalue();

Return Result;

}

Kod Parsiranja Dom Metodom, Osnovne Klase Koje Koristimo Su Node I Element I Predstavljaju Reprezentacije Čvorova U Xml Dokumentu. Osnovne Metode Koje Koristimo Su Getelementsbytagname() Koja Dohvaća Listu Čvorova Određene Oznake (Eng. Tag), Getnodetype() Kojom Provjeravamo Tip Čvora, Te Item(Index) Kojom Dohvaćamo Podčvor S Indeksom Index.

Metoda Getinnertext() Je Pomoćna Metoda Napisana Za Potrebe Ovog Specifičnog Primjera. Ta Metoda Za Neki Objekt Tipa Element I Oznaku Tagname Dohvaća Sadržaj Prvog Podčvora Koji Je Oznake Tagname.

Sax Api

Parsiranje Xml Dokumenta Sax Metodom Ostvaruje Se Pomoću Ugrađenih Klasa Saxparser Koja Stvara Omotač Oko Objekta Sax Xmlreader, Te Klase Saxparserfactory Stvara Instance Klase Saxparser.

Aplikacija Je Izvedena Pomoću Tri Klase Iz Kojih Će Biti Prikazani Najvažniji Dijelovi.

Kod 5.9: Dio Koda Za Sax Parsiranje Iz Klase Xmlparser

// Create a Url We Want to Load the Tram Info From Url Url = New Url("Http://.../zeljeni.Xml");

// Get a Saxparser From the Saxparserfactory Saxparserfactory Parserfactory = Saxparserfactory.Newi^stance();

Saxparser Parser = Parserfactory.Newsaxparser();

// Get the Xmlreader of the Saxparser We Created Xmlreader Reader = Parser.Getxmlreader();

// Create a New Contenthandler and Apply It to the Xml-

Reader

Tramxmlparser Tramparser = New Tramxmlparser(); Reader.Setcontenthandler(Tramparser);

// Parse the Xml-Data From Our Url Reader.Parse( New Inputsource(Url.Openstream()));

// Get the Parsed Data From the Tram Xml Parser Tramdataset Tramdataset = Tramparser.Getparseddata();

Kod 5.10: Kod Za Sax Parsiranje Iz Klase Tramxmlparser

Public Class Tramxmlparser Extends Defaulthandler {

/\*\* The Fields Representing the Xml Tags for Each Tram, Used in Parsing \*/

Private Boolean Number = False;

Private Boolean Direction = False;

Private Boolean N = False;

Private Boolean E = False;

/\*\* The Object in Which the Data From the Xml Will Be Stored \*/ Private Tramdataset Tramdataset = New Tramdataset();

Public Tramdataset Getparseddata() { Return This.Tramdataset;

}

@override

Public Void Startdocument() Throws Saxexception { This.Tramdataset = New Tramdataset();

}

@override

Public Void Enddocument() Throws Saxexception {

}

/\*\* Gets Called On Opening Tags Like:

* <tag>
* Can Provide Attribute(S), When Xml has Attributes, Like:

\* <tag Attribute="Attributevalue">\*/ @override

Public Void Startelement(String Namespaceuri, String Localname, String Qname, Attributes Atts) Throws Saxexception { If (Localname.Equals("Tramvaj")) {

// Extract an Attribute

String Attrvalue = Atts.Getvalue("Id"); Tramdataset.Setid(Integer. Parseint(Attrvalue)); } Else If (Localname.Equals("Smjer")) {

This.Direction = True; } Else If (Localname.Equals("Broj")) {

This.Number = True; } Else If (Localname.Equals("N")) {

This.N = True; } Else If (Localname.Equals("E")) { This.E = True;

}

/\*\* Gets Called On Closing Tags Like:

\* </tag> \*/

@override

Public Void Endelement(String Namespaceuri, String Localname, String Qname)

Throws Saxexception { If (Localname.Equals("Smjer")) {

This.Direction = False; } Else If (Localname.Equals("Broj")) { This.Number = False;

} Else If (Localname.Equals("N")) {

This.N = False;

} Else If (Localname.Equals("E")) {

This.E = False;

}

}

/\*\* Gets Called On the Following Structure:

\* <tag>characters</tag> \*/

@override

Public Void Characters( Char Ch[], Int Start, Int Length) { If ( This .Number) {

Tramdataset.Setnumber(Integer. Parseint( New String(Ch,

Length)); Length)); Length));

}

Start, Length)) );

}

If ( This.Direction) {

}

If ( This.N) {

Tramdataset.Setlatitude( New String(Ch, Start,

}

If ( This.E) {

Tramdataset.Setlongitude( New String(Ch, Start,

}

Tramdataset.Setdirection( New String(Ch, Start,

kod 5.11: Kod Za Sax Parsiranje Iz Klase Tramdataset

public

class Tramdataset { /\*\* The Number of the Tram \*/ Private Int Number;

/\*\* The Direction of the Tram, Can Be '+' Or '-' \*/ Private String Direction = Null;

/\*\* Geographic Latitude (N), Parallels \*/ Private String Latitude = Null;

/\*\* Geographic Longitude (E), Meridians \*/ Private String Longitude = Null;

/\*\* The I'd of the Specific Tram Since There Can Be Multiple Trams with the Same Tram Number \*/ Private Int I'd = 0;

Public Int Getnumber() { Return Number;

}

Public Void Setnumber(Int Number) { This.Number = Number;

}

(... )

Number of Get/set Methods.

Public String Tostring() {

Return "Tramvaj Broj = " + This.Number + "\nsmjer = "

+ This.Direction + "\nn: " + This.Latitude +

"\ne: " + This.Longitude

+ "\nid = " + This.Id;

}

}

Kod Sax Parsiranja Osnovne 2 Klase Su Tramxmlparser Koja Obavlja Parsiranje Xml Dokumenta Te Tramdataset U Koju Se Spremaju Parsirani Podaci. Tramxmlparser Sadrži Metode Koje Predstavljaju Radnje Koje Parser Mora Obaviti Kada Naiđe Na Ključne Dijelove Xml Dokumenta Kao Recimo Otvaranje Oznake I Zatvaranje Oznake.

5.3.1.3 Ostali Formati Zapisa Strukturiranih Podataka

Od Ostalih Formata Zapisa Strukturiranih Podataka Spomenut Ćemo Još Json (Eng. Javascript Object Notation) I Csv (Eng. Comma Separated Values).

json Je Vrlo Jednostavan Jezik Za Razmjenu Podataka Ekstenzije .Json. Koristi Se Prilikom Programiranja Ajax (Eng. Asynchronous Javascript + Xml) Web-Aplikacija Kao Alternativa Xml-U.

Csv Je Zapis Koji Se Najčešće Koristi Kod Tabličnih Struktura Zapisa. Podaci Su Međusobno Odvojeni Zarezom. Svaka Linija Cvs Zapisa Odgovara Jednom Retku Tablice.

5.4 Protokol Https

Protokol Https (Eng. Hyper Text Transfer Protocol Secure) Kombinacija Je Protokola Http I Kriptografskih Protokola Ssl (Eng. Secure Socket Layer) Ili Tls (Eng. Transport Layer Security). Sintaksa Protokola Https Identična Je Protokolu Http S Time Da Je Inicijalna Postavka Dobro Znanih Vrata 443 (Umjesto 80). Koristi Se Kao Sigurna Verzija

Protokola Http.

Kod Komunikacije S Poslužiteljem Protokolom Https Neće Biti Detaljno Prikazan Zbog Velike Sličnosti S Protokolom Http.

5.5 Ostali Načini Komunikacije S Poslužiteljem

Osim Spomenutih Protokola Udp, Tcp, Http I Https, Klijent I Poslužitelj Mogu Komunicirati Preko Ftp (Eng. File Transfer Protocol), Protokola Telnet, Protokola Ssh (Eng. Secure Shell) I Drugih Protokola Za Koje Postoji Već Ostvarena Aplikacijska Programska Podrška Za Android, Te Ih Nećemo Detaljnije Obrađivati.

6. Zaključak

U Sklopu Seminarskog Rada Napravljen Je Pregled Cjelokupne Arhitekture Sustava Android, Osvrt Na Paket Softverskih Razvojnih Alata (Android Sdk), Objašnjen Je Koncept Aplikacije, Te Realizirana Praktična Izvedba Različitih Načina Povezivanja S Poslužiteljem. Govoreći O Arhitekturi Sustava Posebna Pažnja Posvećena Je Slojevima Sustava I Njihovim Elementima, Te Virtualnom Stroju Dalvik Koji Je Istaknut Kao Novitet U Arhitekturi Sustava Za Male Uređaje. U Sklopu Poglavlja O Paketu Softverskih Razvojnih Alata Dan Je Detaljan Opis Najvažnijih Komponenti I Njihovih Funkcija. Obrađeni Su Emulator Pokretnog Uređaja, Ddms I Adb. Na Razini Aplikacije Opisani Su Njezina Osnovna Struktura, Životni Ciklus, Način Razvoja I Princip Prikaza Elemenata Korisničkog Sučelja, Datoteka Androidmanifest.Xml, Te Podatkovni Resursi I Njihova Upotreba. Na Kraju Su Realizirani Praktični Primjeri Povezivanja Android Klijenta S Udaljenim Poslužiteljem Upotrebom

Protokola Udp, Tcp I Http. Korištenjem Protokola Http Obrađeno Je Dom I Sax

Parsiranje Jednostavnog Xml Dokumenta. Za Realizaciju Praktičnog Dijela Zadatka Korišteni Su Eclipse Ide, Android Sdk, Android Adt I Udaljeni Linux Poslužitelj. Aplikacije Su Testirane Na Emulatoru.

S Obzirom Na Činjenicu Da Je Android Tek Kratko Vrijeme Zastupljen U Svijetu Pokretnih Uređaja Iza Sebe Ima Izrazito Bogatu Bazu Stručne Literature I Podrške Na Internetu. Zahvaljujući Otvorenosti Programskog Koda Razvojnim Programerima Omogućen Je Uvid U Već Postojeće Aplikacije I Primjerke Koda. Na Internetu Je Moguće Pronaći Primjerke Različitih Vrsta Aplikacija Ili Dijelova Koda. Razvojnim Programerima U Korist Ide I Bogata Baza Biblioteka I Alata Za Izradu Aplikacija.

Također, Android Sdk Vrlo Dobro Funkcionira S Eclipse Ide. Zamjerke Su Siromašan Skup Grafičkih Sučelja Za Emulator, Te Nepravilno Funkcioniranje Emulatora Ukoliko Se Pokreće Automatski Pozivom Iz Eclipsea. To Znači Da Je Za Ispravan Rad Emulator U Pravilu Trebalo Pokrenuti Prije Pokretanja Eclipsea, Jer U Suprotnom Dolazi Do Neispravnog Rada Aplikacije Na Emulatoru.

U Sklopu Diplomskog Zadatka Bit Ć E Potrebno Realizirati Uslugu Praćenja Tramvajskog Prometa Korisniku U Pokretu Na Operacijskom Sustavu Android. Za Ostvarenje Istog Koristit Ć E Se Postojeći Simulator Tramvajskog Prometa. Ovisno O Implementaciji Simulatora Trebat Će Razmotriti Najbolji Način Spajanja Na Poslužitelj I Dohvat Podataka S Njega. Vrlo Vjerojatno Će Se Raditi O Kombinaciji Protokola Http I Dohvata Podataka U Nekom Od Spomenutih Zapisa.

Osnovna Vizija Klijentske Aplikacije Za Pokretni Uređaj Je Izvedba Korisničkog Sučelja S Vizualnim Prikazom Položaja Svih Ili Ž Eljenih Tramvaja Na Karti Na Temelju Podataka Dobivenih Od Udaljenog Poslužitelja.

7. Literatura

[1] E. Burnette, Hello, Android, 2008. [2] N. Gramlich, Andbook!

Http://andbook.Anddev.Org/

[3] Mike, Androidology Part 1 Of 3 - Architectureoverview

Http://www.Youtube.Com/watch?v=mm6ju0xhuw8, 2007.

[4] R. Meier, Professional Android Application Development, 2009.

[5] D. Bornstein, Dalvik Virtual Machine Internals.

Http://www.Youtube.Com/watch?v=ptjedozexpm, 2008.

[6] Android Deveopers, Tools Overview.

Http://developer.Android.Com/guide/developing/tools/index.Html, 2009.

[7] Android Developers, The Androidmanifest.Xml File.

Http://developer.Android.Com/guide/topics/manifest/manifest-Intro.Html, 2009.

Www.Maturski.Org