Poglavlje 1

Android

1.1 Uvod

Android nam je poznat kao *OS za mobilne uređaje* baziran na modificiranoj verziji Linux-a, od 2005. u vlasništvu Google-a. Ali Android je zapravo mnogo više od toga: skup open source softwarea koji uključuje OS, middleware i klučne aplikacije, zajedno sa API library-ma za pisanje mobilnih aplikacija koje mogu oblikovati izgled i funkciju mobilnih uređaja. Google definira Android kao *prvu potpuno otvorenu platformu za mobilne uređaje*, sav software potreban za moblni uređaj ali bez proprieatary zapreka koje bi kočile inovacije.
U skladu s time, Android je open i free, objavljen pod open source Apache licence i svi ga mogu slobodno downloadati i modifiicrati.[1]

Ta otvorenost je velika prednost Androida; prihvatile su ga velike kompanije takozvanog OPEN HANDSET ALIANCE-a (udruženje od preko 30 tehnoloških kompanija uključujući Motorolu, HTC, TMobile i ostale koje razvijaju uređaje koji koriste Android).

Druga velika prednost Androida je da se *sve aplikacije tretiraju jednako*, dok Windows Mobile i Appleov iPhone, koji su bazirani na propieatary software-u, daju veći prioritet svojima (native) aplikacijama i ograničavaju komunikaciju između native i third party aplikacija. Kod Androida su sve aplikacije ravnopravne i sve se mogu izbrisati, zamjeniti i nadograditi. Također, srušene su granice među aplikacijama - aplikacije lako, bez ograničeja, komuniciraju među sobno i razmjenjuju podatke.

Iduća velika prednost je da kad jednom napišemo aplikaciju za Android, ona se može pokrenuti na svim uređajima koji koriste Android (pa-

[1] Bitno je napomenuti da aplikacije razvijene za Android ne moraju biti open source, ali bilo bi lijepo da jesu.

metni telefoni, tableti, e-book reader, MP4 playeri, Internet TV,...) bez razmišaljanja o hardware-u. To je postignto korištenjem *Dalvik i ART virtualnih mašina*.

Sve to doprinjelo je *brzom i laganom* razvoju aplikacija (prilično složena aplikacija može se napisati za 30-ak minuta), razvoju velike zajednice developera širom svijeta (što omogućava pronalaženje odgovora na neka pitanja, ali i olakšava pronalaženje suranika). Uz to, Android ima i *odličnu dokumentaciju* i *nema troškova* za *razvoj i distribuciju aplikacija* (sve potrebno se besplatno downloada). Android tržište aplikacijama (Android Market) je online trgovina aplikacijama koja je instalirana na svim Android uređajima i omogučuje jednostavano preuzimanje aplikacija direktno na uređaj (tu se nalaze i besplatne aplikacije kao i one koje se plaćaju iako je pristup plaćenim aplikacijama u nekim zemljama zabranjen ili restringiran[2]).

Android podržava i koristi:

- Bluetooth, SMS, MMS, Multitouch screen (može propoznati više točki dodira odjednom), Flash,...
- MP3, MP4, WAV, MIDI, JPEG, PNG, GIF, BMP,...
- fotoaprat/kamera, GPS, digitalni kompas,...
- WebKit preglednik
- SQLite relacijsku bazu podataka (brza i efikasna, što je esencijalno za uređaje čiji su kapaciteti limitirani njihovom kompaktnom prirodom)

1.2 Dalvik virtual machine

Dalvik virtualna mašina je specijalizirana virtualna mašina dizajnirana za Android i optimizirana je za mobilne urđaje s limitiranom memorijom i CPU-om. Aplikacije za android pišu se u Java programskom jeziku ali se ne izvršavaju u tradicionalnoj Java virtualnoj mašini (stack based virtual machine) već u Dalvik virtualnoj mašini (register based virtual machine). Kompajlirane Java klase se transformiraju u Dalvik executables (.dex) i izvršavaju kao posebni proces u vlastitoj instanci Dalvika (to omogučava između ostalog da ako se jedna aplikacija sruši ne utječe na druge). Sav pristup hardware-u i low level sistemskim funkcionalnostima (memorija,

[2]Ima zemalja gdje se smije prodavati ali ne i preuzimati i obratno

threading, sigurnost,...) ide preko Dalvika i osigurava da developeri ne moraju brinuti o tome za koji uređaj rade.

Tvorac Dalvika je Dan Bornstein koji joj je dao ime po ribarskom selu Dalvik na Islandu odakle su mu pretci.

1.3 ART

ART (Android Run Time) je nasljednik Dalvik virtualnih mašina za Android. ART, poput Dalvika, izvršava .dex datoteke, stoga kod koji je napisan za Dalvik može raditi pod ART-om, ali neke tehnologije koje su radile za Dalvik ne rade na ART-u. Prednosti ART-a nad Dalvikom su: Ahead Of Time comiplation (AOT) koji ubrzava performanse neke aplikacije, poboljšani Garbige collector (GC), jasnije se iskazuju greške u aplikaciji.

ART polako zamjenjuje Dalvik, ali prijelaz još nije načinjen u potpunosti. Android verzije 5.0 još uvijek koriste Dalvik kao početnu postavku ali omogučuju prelazak na ART. Verzija 6.0 također koristi Dalvik kao primarnu virtualnu mašinu. Poćevši s verzijom 7.0. ART dolazi u prvi plan.

1.4 Optimiziran memory i process managment

Vlastiti run time i virtualnu mašinu, kao i Android, koriste i Java i .NET. Za razliku od njih, Android run time također upravlja i životnim vijekom procesa. Android osigurava dobar rad aplikacije zaustavljanjem i terminiranjem procesa ako je potrebno, da bi se oslobodili resursi za aplikacije višeg prioriteta. Prioritet se određuje ovisno o aplikaciji s kojom je korisnik u interakciji. Sve aplikacije dakle trebaju biti pripremljene da budu naglo ugašene, ali i da se mogu lako restartati u isto stanje.

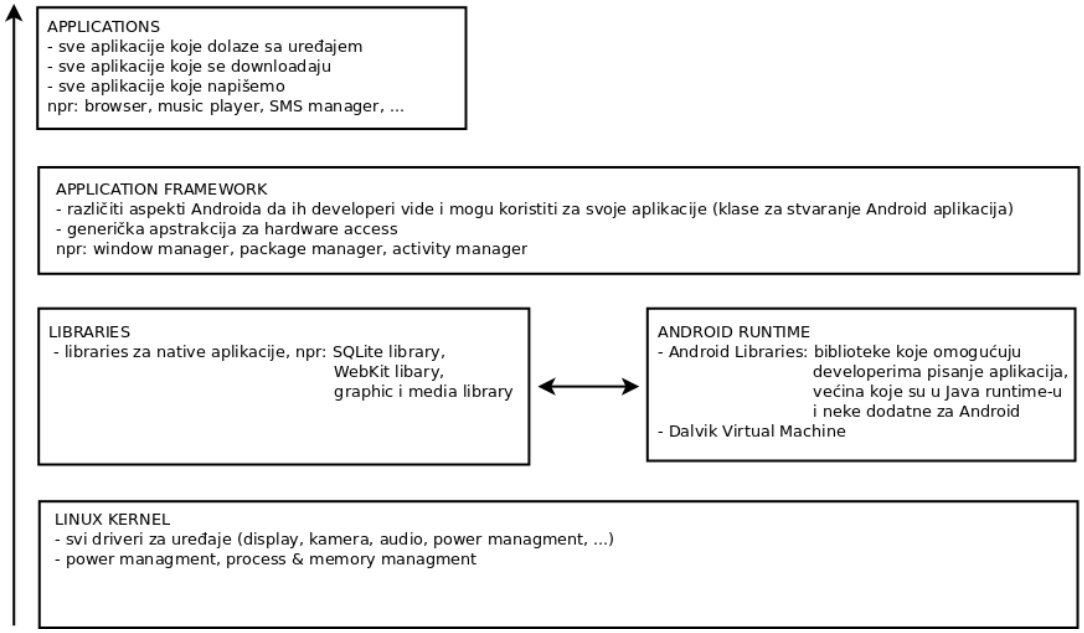
1.5 Arhitektura Androida

Arhitektura android sustava sastoji se od 5 djelova koji su smješteni u 4 sloja (slika 1.1):

- Linux Kernel
- Libraries, Android Run Time
- Application Framework

- Applications

U Android Run Time dijelu nalaze se Android biblioteke koje omogućavaju developerima pisanje aplikacija, većina kojih su u Java run time-u i još neke dodatne za Android. Dakle, Dalvik i Android run time sjede na Linux jezgri koja upravlja low level hardware interakcijama (driveri, momory management), a skup API-ja (Application Programming Interface, to su razne klase biblioteka) daje pristup svim hardware-skim komponentama. Aplikacije se izvršavaju u Android run time koristeći klase iz Aplication Framework-a.



Slika 1.1: Arhitektura Android sustava prikazana po slojevima

Napomena: Moguće je pisati i C/C++ aplikacije direktno u Linux OS-u, ali za to nema potrebe, osim ako želite npr. modificirati Dalvika. To neće biti naša tema, za više informacija zainteresiranima: `https://groups.google.com/forum/#!forum/android-internals`.

Napomena: Android je unificiran OS za pametne telefone i tablete. Od vrezije 3.0 ugrađene su neke specifične stvari za tablete, u verziji 4.0 to je prenešeno i na pametne telefone te su dodane nove opcije za obje vrste uređaja. Mi ćemo se baviti Androidom 4.0 i više (sve što se napravi za starije verzije možemo prenjeti na novije, ali ne i obratno).

Poglavlje 2

IDE

Za razvoj Android aplikacija danas se koristi Android Studio koji u sebi već sadrži osnovne pakete za razvoj, a lako se može nadopuniti drugim paketima. Ranije je preporučeni IDE za razvoj bio Eclipse koji se i dalje može koristiti, ali je izgubio razvojnu podršku (update) za Android. Projekti izrađeni u Eclipse-u se lako mogu prenijeti u Android Studio.

Android Studio je podržan za Linux, Windows i Mac OS X operativne sustave. Ovdje će biti opisan postupak instalacije na Linux sustavima.

2.1 Build sustav - Gradle

Do sada se tokom studija spominjao samo pojam kompajliranja programa. Kompajliranje je proces prevođenja programskog koda u objektni kod. Danas kompajleri uz kompajliranje vrše i linkanje, koje objektni kod pretvara u datoteku koja se može izvršiti na računalu. Buildanje projekta, uz kompajliranje i linkanje, uključuje i neke dodatne radnje, na primjer izradu installera. Buildanje još omogućuje da se provode testovi kompajliranog koda, moguće je definirati korake koje treba učiniti prije ili poslje kompajliranja nekog dijela projekta, odrediti gdje će se izvršni program nalaziti i još mnogo toga. Procesom buildanja moguće je stvarati statičke i dinamičke pomoćne biblioteke koje se kasnije mogu samostalno koristiti. Razlika među njima je u tome što kad statičku biblioteku unesemo u neki kod, ona se u cjelosti linka prilikom kompajliranja, dok se kod dinamičke datoteke linka samo dio koji će se koristiti u izvršnom programu. U slučaju razvoja android aplikacija, build sustav služi kako bi uzeo izvorne programske datoteke (.java ili .xml) na njih primjenio neke alate kompajliranja i linkanja (na primjer .java datoteku pretvori u .dex) te grupira sve te datoteke u jednu komprimiranu .apk datoteku s kojom Android sustav zna raditi.

Build sustav koji se koristi u Android Studiju naziva se Gradle. Ovaj sustav je razvijen promatrajući druge build sustave, te integriranjem njihovih najboljih karakteristika. Gradle je baziran na JVM (Java Virtual Machine) build sustavima što znači da se mogu pisati skripte za buildanje u Java programskom jeziku.

2.2 Android Studio vs. Eclipse

Najveće razlike između Android Studija i Eclipsea su u sljedećim segmentima:

- *Build alati*: Android Studio koristi sve popularniji Gradle sustav buildanja. Gradle builda projekte pomoću Apache Ant i Apache Maven ali uvodi i Groovy DSL (Domain-Specific Language) koji omogućuje skriptiranje buildova što na kraju omogućuje automatizaciju uploada beta .apk datoteka na TestFlight u svrhu testiranja. S druge strane Eclipse koristi Apache Ant kao osnovni build sustav koji koristi vrlo robusni XML koji je poznat Java developerima.

- *Napredno autozavršavanje koda i refaktorizacija*: Oba IDE-a koriste standardno Java autozavršavanje koda, ali u slučaju Android Studija, Google je unio dodatne stvari vezane za Android i njegovu refaktorizaciju.

- *Organizacija projekta*: Eclipse je projekte organizirao u workspace koji se odredio pri pokretanju. Takav način rada je ograničavajući jer dopušta da se koriste samo projekti iz određenog workspace-a. Ako bi htjeli radit sa projektima iz drugog workspace-a moramo ugasiti Eclipse i prilikom ponovnog pokretanja preći u drugi workspace. Android Studio taj problem rješava korištenjem modula. Moduli mogu biti svaki dio nekog projekta, npr. jedan modul je aplikacija na kojoj se upravo radi, drugi modul je neka biblioteka koja je skinuta sa interneta i želi se uključiti u trenutni projekt, treći modul je integrirani SDK... Svaki od modula može imati svoje Gradle build filove i zahtjevati svoje pakete za ovisnost.

- *IDE performanse i stabilnost*: Eclipse je IDE koji je razvijen u svrhu razvoja Java aplikacija, zbog toga zahtjeva puno RAM-a i CPU-a kako bi radio bez greške. Kada se u Eclipse-u razvijaju aplikacije za Android često se dogodi da se IDE sruši jer nije predviđen za tu vrstu razvoja. Zna se dogoditi da nakon nekoliko sati neprekidnog rada u

Eclipse-u, IDE prestane raditi pa ga se treba ponovno pokrenuti. S druge strane, Android Studio je kada se počeo koristiti bio još u beta verziji razvoja, a to sa sobom donosi probleme sa stabilnosti IDE-a, koji su sada bitno popravljeni ali ne i još u potpunosti uklonjeni.

2.3 Android Studio

Android Studio je integrirana okolina za razvoj (integrated development environment - IDE) Android aplikacija. Preuzimanje je besplatno i omogućeno je Apache 2.0 licencom. Za razvoj aplikacija Android Studio zahtjeva instalaciju Java Development Kit-a (JDK) 7 ili više. Od Android Studia 2.3.0 nadalje, Android Studio dolazi sa Built in Javom.

Androi Studio je dostupan za preuzimanje na adresi `https://developer.android.com/sdk/index.html`.

Instalacija

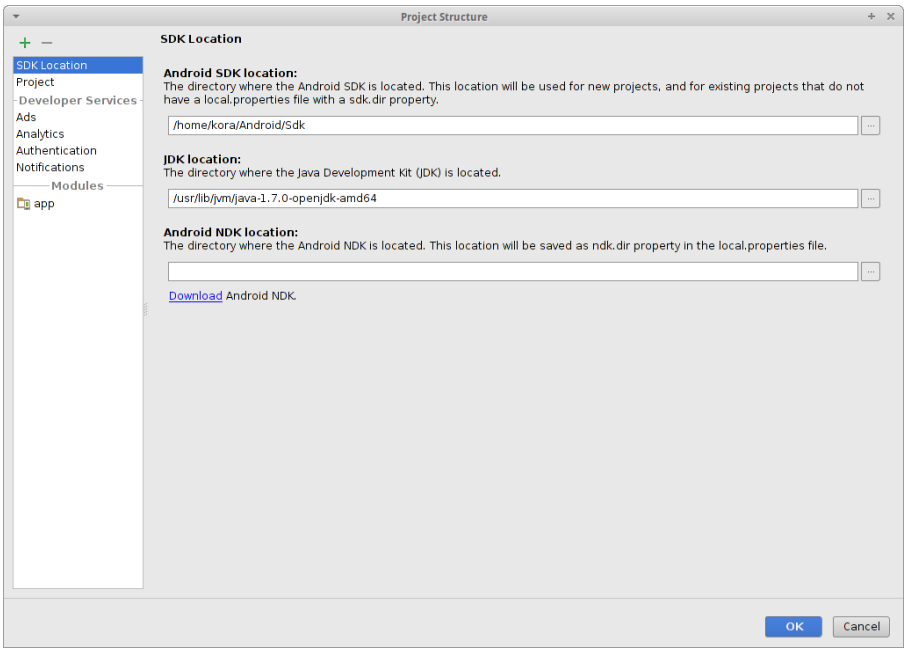
Nakon pruzimanja paketa za Linux operativni sustav potrebno je preuzeti paket raspakirati na željenu lokaciju, nazovimo to ANDROIDHOME. U toj mapi se nalaze upute za daljnju instalaciju.

Prije nego što započnemo podešavanje Android Studija (u starijim verzijama, prije 2.3.0), potrebno je instalirati JDK na sustav. JDK se može besplatno pruzeti sa interneta na `http://www.oracle.com/technetwork/java/javase/downloads/index.html`. Treba je instalirati prema navedenim uputama (za linux se upute mogu naći ovdje: `http://www.wikihow.com/Install-Oracle-Java-JDK-on-Ubuntu-Linux`).

Podešavanje Android Studija

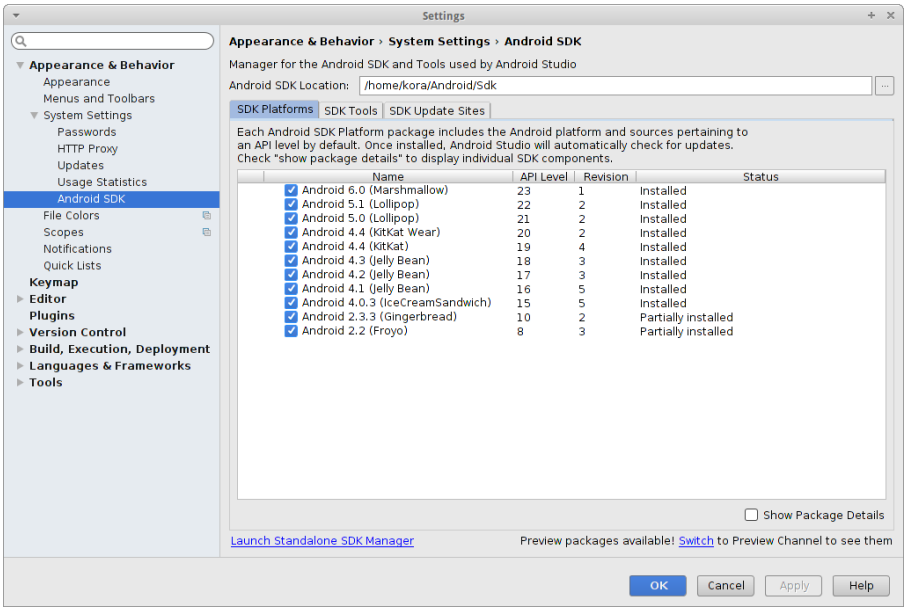
Otiđemo u ANDROIDHOME/bin te pokrenemo studio.sh skriptu (ili putem terminala se pozicioniramo u ANDROIDHOME/bin i upisemo ./studio.sh). Naravno, možemo si postaviti i desktop ikonu za direktno pokretanje skripte. Ukoliko smo instalirali vlastitu Javu i želimo je koristiti umjesto defaultne, ili nemamo defaultnu, tada trebamo podesiti da Studio zna koju Javu treba koristiti. U $File- > Projectstructure$ unesemo lokaciju jave u za to predviđeno polje (slika 2.1), uglavnom je to /usr/bin/jvm/ i upišemo verziju koju smo instalirali, te pohranimo postavke.

Sljedeći korak je instalirati podršku za sve verzije Androida za koje mislimo razvijati aplikacije (u našem slučaju su to sve verzije). Navigiramo

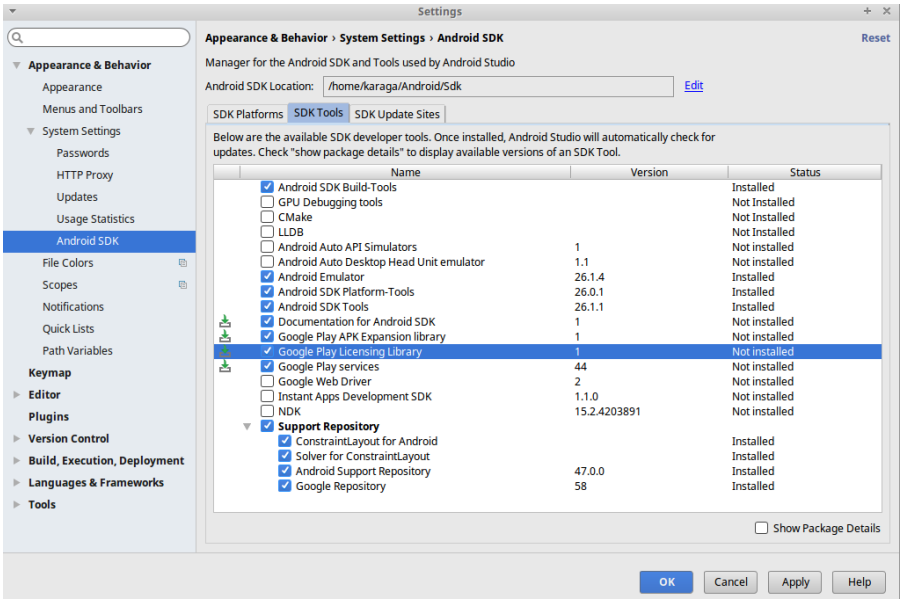


Slika 2.1: Podešavanje lokacije JMV u Android Studiju

u $File->Settings->Appearance\&Behavior->SystemSettings->AndroidSDK$ odemo na karticu $SDKPlatforms$ te označimo sve ponuđene verzije (slika 2.2, kako se pojavljuju nove verzije, biti će i one ponuđene na popisu). Zatim na kartici $SDKTools$ označimo kao što je prikazano na slici 2.3. Kliknemo gumb za instalaciju i čekamo da se sve instalira.



Slika 2.2: Instalacija SDK Platforms



Slika 2.3: Instalacija SDK Tools