Programiranje za Android

Duda, Alen

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

"Dr. Mijo Mirković"

ALEN DUDA

PROGRAMIRANJE ZA ANDROID

Završni rad

Pula, 2015.

Sveučilište Jurja Dobrile u Puli

Fakultet ekonomije i turizma

"Dr. Mijo Mirković"

ALEN DUDA

PROGRAMIRANJE ZA ANDROID

Završni rad

JMBAG: 0303012620, redoviti student
Studijski smjer: Informatika

Predmet: Programiranje
Mentor: doc. dr. sc. Krunoslav Puljić

Pula, rujan 2015.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Alen Duda, kandidat za prvostupnika Informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student:

U Puli, __. __. 2016. _____

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom _____
_____

koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SADRŽAJ

UVOD

Kupci se najčešće odlučuju za kupovinu smartphonea zbog njihovih naprednijih mogućnosti u odnosu na konvencionalne mobilne uređaje. Sve te mogućnosti ne bi mogli koristiti ukoliko ne postoji adekvatan software koji im omogućava glatku izvedbu. Što je kvalitetniji software, što je veći broj različitih aplikacija – veće su šanse da da se uređaj može iskoristiti do maksimalnog potencijala.

U današnje vrijeme tržištem dominiraju mobilni uređaji koje pokreće Android operativni sustav o kojem će riječi biti u ovom radu. Rad će biti fokusiran na programiranje za Android platformu jer, koliko god aplikacija bilo dostupno za različite svrhe, uvijek ima mjesta za nove i poboljšane verzije.

Ovaj rad će dati općeniti pregled stvaranja programa u razvojnom okruženju Android Studio, koji je službeni razvojni alat za Android platformu. Cilj rada je upoznati čitatelja s alatom Android Studio i procedurom izrade jednostavne aplikacije.

Rad se, osim uvoda, zaključka i pratećih popisa literature, slika i priloga, sastoji od šest poglavlja u kojima se razrađuje tema. U prvom će poglavlju biti riječi općenito o Androidu te kratko o njegovoj povijesti nastanka i razvoja.

Drugo poglavlje bavi se razvojnim alatom Android Studio koji se koristi za programiranje na Android platformi. Riječi će biti ukratko o nastanku i razvoju, sučelju te najčešće korištenim mogućnostima programa.

U trećem će poglavlju biti opisano stvaranje novog projekta Android aplikacije, najvažnijim datotekama i strukturi aplikacije. Također će se obraditi najvažniji elementi sučelja i upravljanja ponašanjem aplikacije.

Četvrto će poglavlje biti fokusirano na pokretanje i debugiranje Android aplikacija korištenjem fizičkih uređaja i stvaranjem emulatora – virtualnih Android uređaja.

Peto poglavlje bavit će se objavljivanjem aplikacija na Google Play trgovinu te mogućnostima zarade, što je jedan od ciljeva programiranja.

U šestom poglavlju praktično će se prikazati spomenute tehnike pomoću jednostavne aplikacije za konverziju stranih valuta u kune.

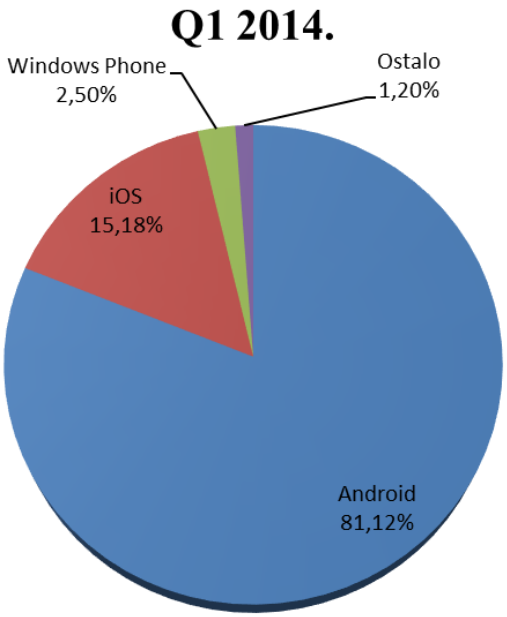
1 ANDROID OPĆENITO

Android se često poistovjećuje s Android OS-om. Međutim, osim operacijskog sustava temeljenog na Linux kernelu[1], Android se sastoji od cjelokupnog frameworka[2] za pokretanje, izradu i dijeljenje aplikacija. Najvažniji dio za developere zasigurno je Android SDK[3] koji sadrži sve alate potrebne za razvoj i pakiranje aplikacija, uključujući alate za kompajliranje[4] i debugiranje[5].

Android je cjelokupna platforma otvorenog koda dizajnirana prvenstveno za mobilne uređaje. Odvaja hardware uređaja od softwarea kojeg pokreće te tako omogućuje velikom broju uređaja pokretanje istih aplikacija, što stvara bogati ekosustav i za korisnike i programere. Programerima su dostupni svi alati i framework-ovi za što lakše i brže stvaranje mobilnih aplikacija, čak i ako nemaju fizički telefon pri ruci. Korisnicima je bitno da uređaj radi bez dodatnih podešavanja i istovremeno daje razne mogućnosti personalizacije uređaja. Proizvođačima uređaja Android pruža sve osim specifičnih drivera[6] za određene dijelove hardwarea. (Gargenta 2011, 1-2)

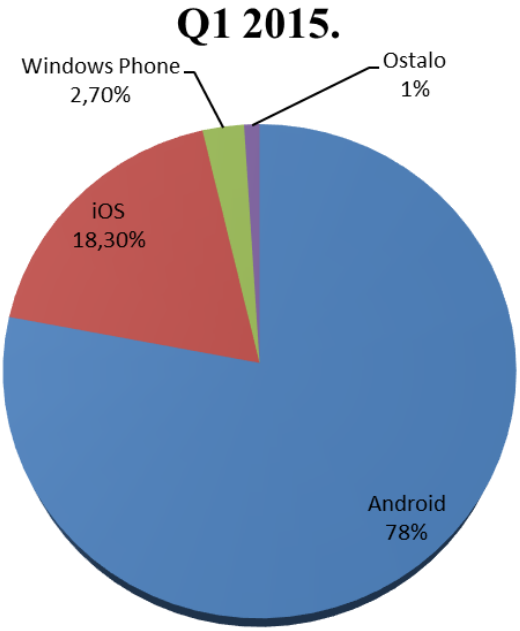
Android uređaji su već nekoliko godina najprodavaniji na tržištu mobilnih uređaja, što se vidi sa grafova 1 i 2 koji prikazuju udio operacijskih sustava u odnosu na proizvedene mobilne uređaje za prvi kvartal 2014. i 2015. godine.

[1] Jezgra operativnog sustava koja spaja aplikacije s hardwareom
[2] Univerzalno okruženje za razvoj softwarea, često dolazi s pratećim programima i bibliotekama
[3] eng. Software Development Kit, paket programa za razvijanje softwarea
[4] eng. compiler – program za prevođenje izvornog koda iz nekog programskog jezika u strojni jezik tako da se može izvoditi neovisno o kodu
[5] Proces otklanjanja grešaka u kodu programa koje mogu dovesti do neželjenog ponašanja
[6] Pomoćni programi pokretači bez kojih hardware teško ili uopće nije moguće u potpunosti iskoristiti



Grafikon 1. Tržišni udio mobilnih operativnh sustava u prvom kvartalu 2014.

Izvor: Izrada autora uz pomoć podataka s http://www.idc.com/prodserv/smartphone-os-market-share.jsp



Grafikon 2. Tržišni udio mobilnih operativnih sustava u prvom kvartalu 2015.

Izvor: Izrada autora uz pomoć podataka s http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Grafovi 1 i 2 navode da Android uređaji zauzimaju više od tri četvrtine tržišta mobilnih uređaja, dok sljedeći po redu iOS zauzima manje od jedne petine. Stoga i ne čudi što je sve više programera zainteresirano za stvaranje aplikacija upravo za Android platformu.

1.1 Povijest

Google je 2005. godine kupio tvrtku Android, Inc. te se smatralo da se time želi probiti na tržište mobilnih uređaja. Međutim, Eric Schmidt, Google-ov CEO[7], izjavio je kako je Android zamišljen kao platforma za mnogobrojne telefone i uređaje. Cilj je bio stvoriti ravnopravno tržište za sve proizvođače uređaja uklanjanjem ograničenja dostupnih operacijskih sustava te zaraditi pružajući reklame i ostale usluge. (Gassner, Developing Android Apps Essential Training 2015)

U tu svrhu 2007. godine osnovana je tvrtka Open Handset Alliance (OHA) kojeg čine Google i brojne druge tvrtke uključujući proizvođače uređaja, aplikacija i čipova te pružatelje mobilnih usluga. Nedugo nakon osnivanja, OHA 2008. godine predstavlja Android zajedno sa SDK verzijom 1.0. U to vrijeme na tržištu još nije bilo uređaja s Android platformom, što je značilo da nije nužno imati uređaj za razvoj aplikacija za Android. HTC uskoro izdaje prvi komercijalni Android uređaj pod imenom G1 (poznat i kao HTC Dream). (Gargenta 2011, 3-4)

1.2 Verzije sustava

HTC Dream bio je prvi komercijalni uređaj kojeg je pogonio Android OS 1.0 izdan u rujnu 2008. godine. Od tada do danas, Android OS doživio je mnoge inkarnacije do najnovije stabilne verzije 5.1.1. Glavne značajke većih verzija sustava su sljedeće:

- **1.0 (rujan 2008.)** – Web preglednik, podrška za kamere, Google usluge (pretraga, karte, sinkronizacija, dopisivanje), reprodukcija multimedije
- **1.1 (veljača 2009.)** – poboljšane karte, prikaz i skrivanje numeričke tipkovnice, spremanje privitaka elektronske pošte
- **1.5 (Cupcake, travanj 2009.)** – prva verzija nazvana po desertu; podrška za video, početni zaslon, widgeti[8], copy-paste operacije, slike kontakata, animirane tranzicije, automatska rotacija zaslona
- **1.6 (Donut, rujan 2009.)** – poboljšana podrška za gestikulacije s više prstiju, integrirana aplikacija za kameru i galeriju slika
- **2.0 (Éclair, listopad 2009.)** – podrška za više Google računa, Bluetooth 2.1, Microsoft Exchange i više veličina ekrana te pretraživanje SMS i MMS poruka

[7] eng. Chief Executive Officer, glavni izvršni direktor
[8] Programčići prilagođeni izvođenju kao dio pozadinskog zaslona

- **2.2 (Froyo, svibanj 2010.)** – poboljšana brzina i korištenje memorije, novi JavaScript engine za Chrome preglednik, USB dijeljenje mrežne veze i Wi-Fi hotspot
- **2.3 (Gingerbread, prosinac 2010.)** – posljednja verzija namijenjena isključivo mobilnim telefonima; poboljšano oslobađanje memorije, audio i video izvođenje, copy-paste poboljšanja i NFC[9] podrška
- **3.0 (Honeycomb, veljača 2011.)** – optimiziran za tablet uređaje; Fragments API i akcijska traka za modernizaciju aplikacija
- **4.0 (Ice Cream Sandwich, listopad 2011.)** – ujedinjenje telefonskih i tabletnih SDK-ova; poboljšan video i omogućeno mijenjanje postavki pokretača programa
- **4.1 (Jelly Bean, veljača 2012.)** – prvi od tri Jelly Bean izdanja, uveo poboljšanja u brzini izvedbe sučelja
- **4.2 (studeni 2012.)** – poboljšanja kamere, više korisnika na tabletima, jedinstven dizajn sučelja početnih ekrana
- **4.3 (srpanj 2013.)** – smanjena potrošnja Bluetootha, bolja grafika za video igre
- **4.4 (KitKat, listopad 2013.)** – bolje upravljanje memorijom i energijom, bolja podrška za NFC i upravljanje spremljenim podacima, razna SMS i multimedijska poboljšanja
- **5.0 (Lollipop, studeni 2014.)** – material design – novi način dizajniranja sučelja, redizajnirane notifikacije, ART runtime zamijenio Dalvik[10]

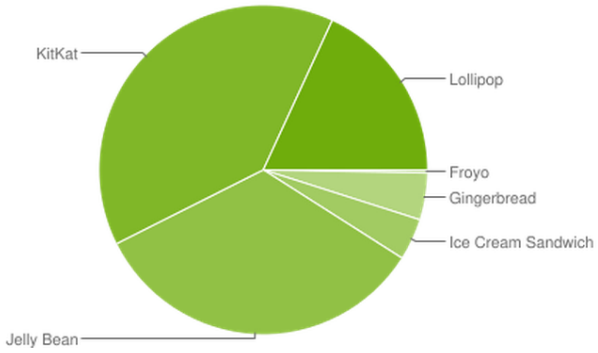
Do verzije 4.0 developeri su morali stvarati odvojene verzije aplikacija za mobilne telefone, koje je pokretao Gingerbread, i za tablete, koje je pokretao Honeycomb. Od verzije 4.0 nadalje mobiteli i tableti koriste zajednički SDK što uvelike olakšava razvoj aplikacija (Gassner, Developing Android Apps Essential Training 2015).

Novi uređaji pogonjeni Androidom neprekidno izlaze na tržište te je stoga nemoguće očekivati da svi imaju istu verziju operacijskog sustava. Naprotiv, vrlo je mnogo različitih uređaja s različitim verzijama Androida te raznim hardverskim sposobnostima, što je samo po sebi problem kod optimizacije aplikacija jer je potrebno prilagoditi izvorni kod za ciljane verzije operacijskog sustava. Google je toga itekako svjestan te redovito izdaje statistički

[9] eng. Near Field Communication – mogućnost međusobne komunikacije i prijenosa podataka kompatibilnih uređaja koji se gotovo dodiruju
[10] Virtualni strojevi za izvođenje Java aplikacija

pregled korištenja različitih verzija njegovog sustava. Tako je moguće vidjeti koje su verzije najzastupljenije i samim tim ciljati željeno tržište. Stanje krajem kolovoza 2015. godine prikazano je na slici 1.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%



Slika 1. Distribucija verzija Androida trenutno u upotrebi

Izvor: https://developer.android.com/about/dashboards/index.html 29.08.2015.

Slika 1 jasno prikazuje da je najzastupljenija verzija Android KitKat, a da Lollipop-u popularnost polako počinje rasti. Gingerbread i Ice Cream Sandwich koriste većinom stariji uređaji za koje nisu dostupne novije verzije. Ne može se zanemariti ni tri verzije Jelly Bean-a koje zajedno čine trećinu svih uređaja koji pristupaju Google Play aplikaciji po kojoj se mjeri ova statistika.

Navedeni statistički podaci brzo se mijenjaju i važna je osobina programera mogućnost brze prilagodbe na te, ponekad drastične, promjene. Srećom, relativno je lako pratiti trenutno stanje uz pomoć Google-ovih online alata prilagođenih Android developerima što uvelike olakšava odluku koje je verzije Androida najpametnije ciljati te tako potencijalno smanjiti veličinu i kompleksnost izvornog koda.

2 ANDROID STUDIO

Od nastanka Androida do kraja 2014. godine, aplikacije za Android razvijale su se pomoću Eclipse editora i dodatka ADT – Android Development Tools. Eclipse ADT je bio službeno podržani način stvaranja aplikacija za Android i sadržavao je sve alate potrebne za razvoj Android aplikacija, od kojih neki alati još ne postoje za Android Studio.

Slično kao Eclipse ADT, i Android Studio IDE [11] je baziran na popularnom Java razvojnom alatu. U ovom slučaju radi se o besplatnoj (Community Edition) inačici programa IntelliJ IDEA tvrtke JetBrains. Android Studio je najavljen u svibnju 2013. godine na Google-ovoj I/O konferenciji. Nakon godinu i pol razvoja i javnih beta testiranja objavljena je verzija 1.0 i od tada Android Studio postaje službeni razvojni alat. Kao i njegov prethodnik, sadrži editor i Android SDK sa gotovo svim pratećim pomoćnim alatima koji su nužni za razvoj, testiranje i objavljivanje Android aplikacija.

Kako je pisan u Java programskom jeziku na kojem se temelji i programiranje za Android, za uspješno funkcioniranje Android Studija nužno je na računalu imati instaliran najnoviji Java Development Kit (JDK) kojeg sam Android Studio preporučuje instalirati prije svoje instalacije. (Gassner, Android Studio Essential Training 2015)

U vrijeme pisanja ovog rada, najnovija stabilna verzija programa Android Studio je 1.3.1 objavljena 7. kolovoza 2015.

2.1 Sučelje

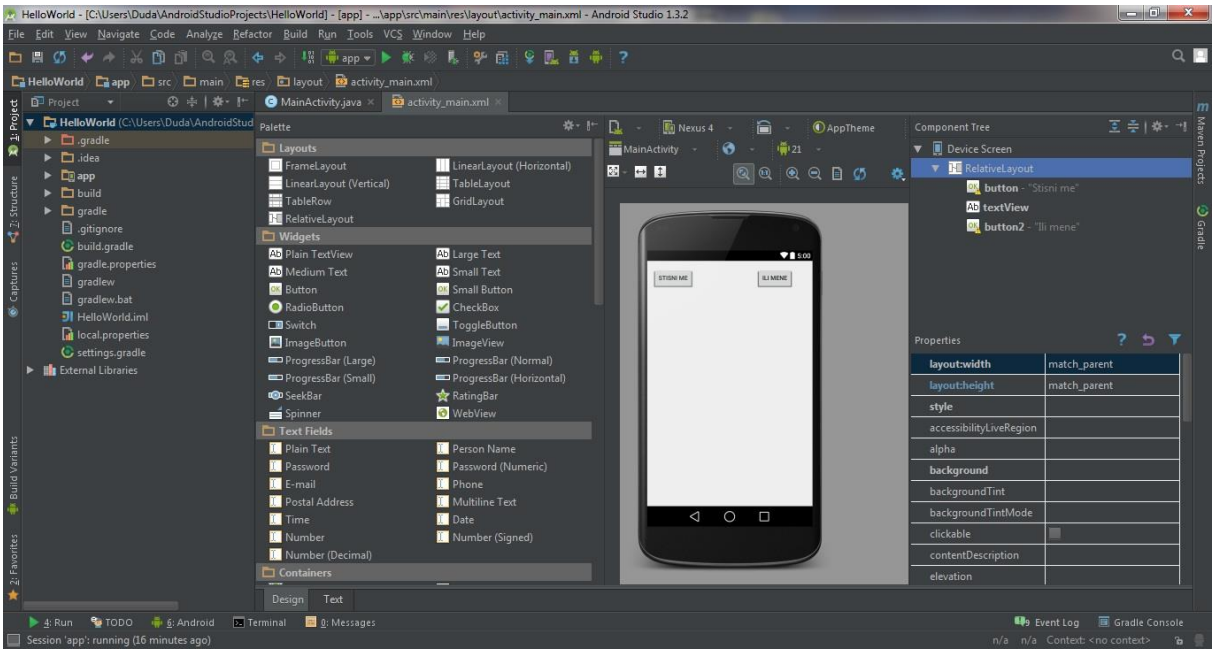
Android Studio, očekivano, jako nalikuje na program na kojem se temelji – IntelliJ IDEA. Ipak, bile su nužne određene preinake kako bi se sučelje prilagodilo izradi mobilnih aplikacija. Sučelje je vrlo prilagodljivo te je moguće imati istovremeno prikazan velik broj prozora, ovisno o veličini zaslona. Prozore koje ne može uredno prikazati, Android Studio će sakriti ispod trenutno aktivnih prozora.

Slika 2 prikazuje tipično sučelje Android Studia s otvorenim projektom. Osim standardne alatne trake s najkorištenijim funkcijama te iscrpnog izbornika, sučelje sadrži datotečni pregled projekta gdje je moguće vidjeti i otvoriti sve datoteke koje su dio aktivnog projekta te stvoriti nove. Sredinom sučelja dominira editor koda ili dizajner sučelja, ovisno je li trenutno otvorena datoteka Java ili XML[12] formata. Konkretno, na prikazanoj slici 2 vidljiv

[11] eng. Integrated Development Environment – integrirano razvojno okruženje
[12] eng. eXtensible Markup Language – vrlo prilagodljivi set instrukcija za opisivanje podataka i sučelja

je dizajner sučelja, uključujući pretpregled dizajna aplikacije, popis komponenti sučelja koje je moguće odvući na ekran te prikaz postojećih komponenti kojima je moguće urediti atribute bez tekstualnog uređivanja XML koda.

Pri dnu ekrana nalaze se prozori koji se najčešće automatski prikazuju ovisno o kontekstu – tijekom pokretanja i debugiranja aplikacije prikazuju se poruke kompajlera i uređaja. Moguće je i brzinski izvršiti naredbe komandne linije pomoću ugrađene terminal aplikacije.



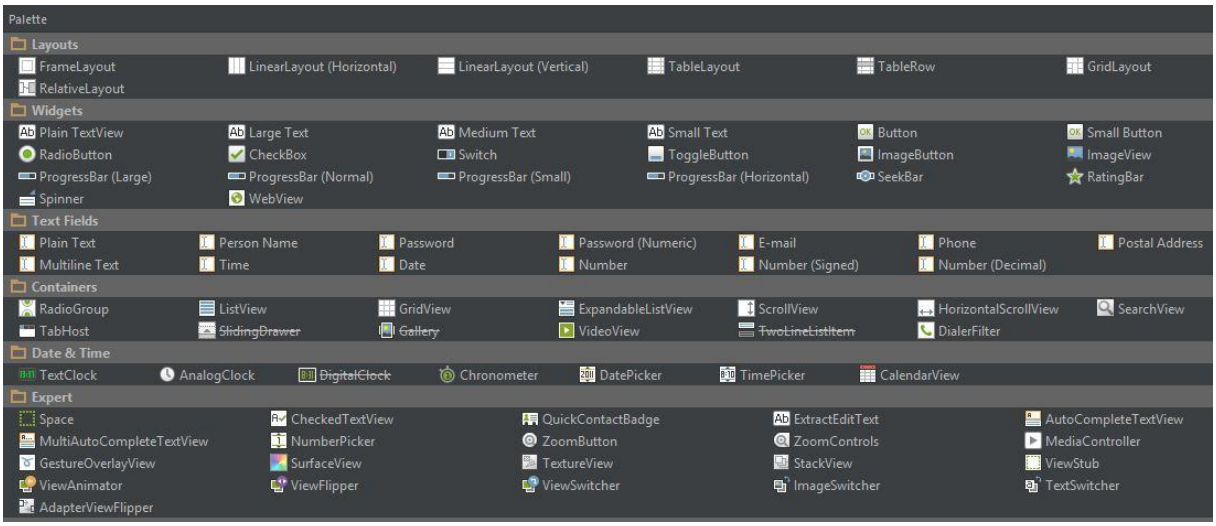
Slika 2. Sučelje Android Studia

Izvor: snimka ekrana autora

2.2 Editor sučelja

Sredinom Android Studia dominira njegov WYSIWYG [13] editor sučelja aplikacija. Sastoji se od palete s mnogim kontrolama (kontrole rasporeda, prikazi teksta, slika, Web stranica, sata, kontrole izbora itd. – slika 3) koje je moguće mišem odvući na virtualni pregled sučelja desno od palete. Na tom pregledu se kontrole pozicioniraju u odnosu na ostale kontrole i rubove ekrana te im se mogu uređivati osnovni atributi. Istovremeno se u pozadini generira XML kod koji opisuje raspored elemenata i njihove atribute. (Gassner, Developing Android Apps Essential Training 2015)

[13] eng. What You See Is What You Get – način dizajniranja sučelja u kojem je moguće stvarati sučelje vizualnim kontrolama, a izvorni se kod automatski generira u pozadini. Vrijedi i obrnuto – editiranjem koda vide se rezultati bez potrebe za vanjskim programima.



Slika 3. Paleta kontrola

Izvor: snimka zaslona autora

Dostupne su kontrole unutar palete organizirane u nekoliko kategorija:

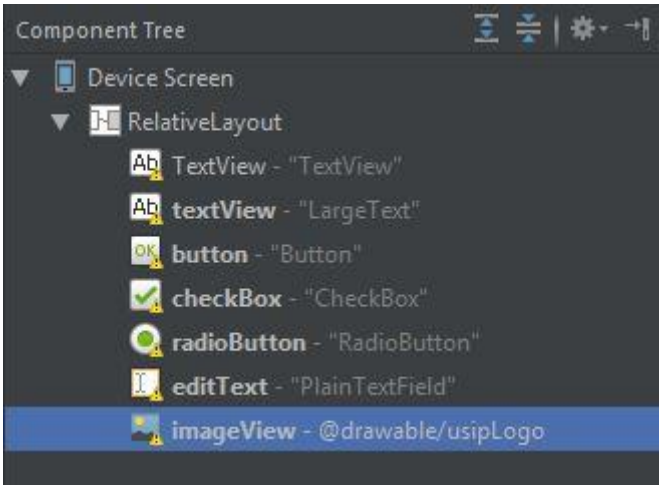
- Layouts – kontejnerski objekti koji diktiraju razmještaj kontrola na ekranu. Korijen sučelja je layout koji može sadržavati druge layoute stvarajući hijerarhiju.
- Widgets – najčešće kontrole koje se nalaze u većini dizajna. U njih spadaju gumbi, pregledi teksta, slika i Web stranica, elementi formulara, trake napretka i slično.
- Text Fields – polja za unos teksta koja se razlikuju po tipu podataka za koje su predviđeni.
- Containers – kontejneri koji grupiraju slične kontrole koje imaju zajednička svojstva.
- Date & Time – sve vezano za datum i vrijeme (sat, kalendar, zaporni sat…).
- Expert – specijalizirane kontrole koje se koriste samo u posebnim slučajevima.
- Custom – sadrže kontrole vlastite izrade, često iz drugih projekata. (Zapata 2013, 48)

Virtualnom uređaju za pregled sučelja moguće je odabrati model, orijentaciju njegovog zaslona i verziju Android sustava za koji će se prikazivati pregled. Ponuđeno je nekoliko modela uređaja s najčešćim veličinama i rezolucijama ekrana te je čak moguće odabrati više uređaja odjednom kako bi se osigurao što bolji prikaz izgleda sučelja aplikacije na što većem broju različitih uređaja. Primjer virtualnog pregleda s nekoliko različitih kontrola poslaganih u relativnom razmještaju prikazan je na slici 4.

Slika 4. Pregled dizajna sučelja i njegove opcije

Izvor: snimka zaslona autora

Desno od pregleda dizajna sučelja nalazi se popis svih kontrola koje se koriste, zajedno s njihovim najvažnijim atributom (npr. prikazani tekst ili putanja do slike). Popis kontrola korištenih na primjeru sučelja sa slike 4 može se vidjeti na slici 5.

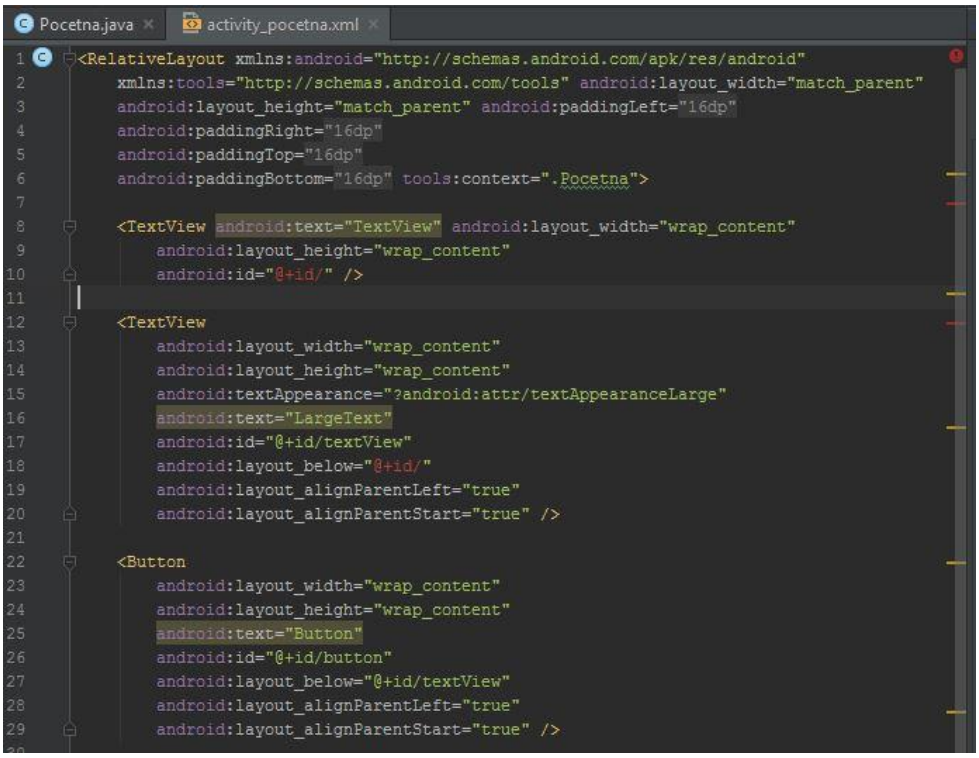


Slika 5. Popis korištenih kontrola

Izvor: snimka zaslona autora

Ispod popisa korištenih kontrola nalazi se editor svojstava pomoću kojega se za odabranu kontrolu može odrediti mnoga svojstva bez potrebe za ručnim pisanjem XML koda. Ipak, iskusnim programerima je draže prijeći u XML način editora i ručno definirati željeni atribut. Tada paletu kontrola zamjenjuje moćni editor izvornog koda s mnogim naprednim

opcijama, uključujući pametno automatsko nadopunjavanje i olakšanu navigaciju među povezanim datotekama te razne preporuke za moguće poboljšanje koda i praćenje najboljih praksi. Izgled tekstualnog editora prikazan je na slici 6. (Gassner, Developing Android Apps Essential Training 2015)



Slika 6. XML editor

Izvor: snimka zaslona autora

2.3 Prateći programi

Kao i Eclipse prije njega, Android Studio dolazi s nekoliko posebnih programa koji olakšavaju česte radnje u programiranju. Neki od njih su isti oni programi koji su pratili prethodnika, neki su unaprijeđeni, a nekima je funkcionalnost integrirana u Android Studio. Najvažniji su SDK Manager, AVD Manager i Device Monitor koje je moguće pronaći u Tools (alati) izborniku pod stavkom Android.

2.3.1 SDK Manager

Software Development Kit Manager je alat integriran u Android Studio koji služi za kontrolu instalacije SDK-a. Pomoću navedenog alata moguće je pregledati instalirane Android platforme, nadograditi ih, instalirati nove platforme ili posebne komponente. npr. za podršku Google Play usluga.

SDK Manager prikazuje popis dostupnih paketa sa sljedećim svojstvima:

- Name – naziv paketa ili skupa paketa
- API – verzija API-ja kad je paket dodan
- Rev – broj revizije ili verzije paketa
- Status – stanje paketa na računalu. Može biti: instaliran, nije instaliran, dostupna nadogradnja, nekompatibilan ili zastario.

Na tom je popisu moguće označiti željene pakete koji će biti instalirani, nadograđeni ili obrisani. Važno je imati instalirane SDK verzije na kojima se testiraju aplikacije. Označeni se paketi instaliraju nakon prihvaćanja licenci. (Zapata 2013, 68-70)

2.3.2 AVD Manager

Android Virtual Device Manager je integrirani alat koji služi za upravljanje virtualnim Android uređajima koji se pokreću uz pomoć Android emulatora. Omogućuje pregled, stvaranje i brisanje virtualnih uređaja. Uređajima je moguće mijenjati naziv, konfigurirati hardverske parametre poput veličine i gustoće zaslona, memorije i slično, odabrati verziju Android OS-a i još mnogo opcija o kojima će više riječi biti u kasnijem poglavlju posvećenom pokretanju aplikacija. (Zapata 2013, 70-72)

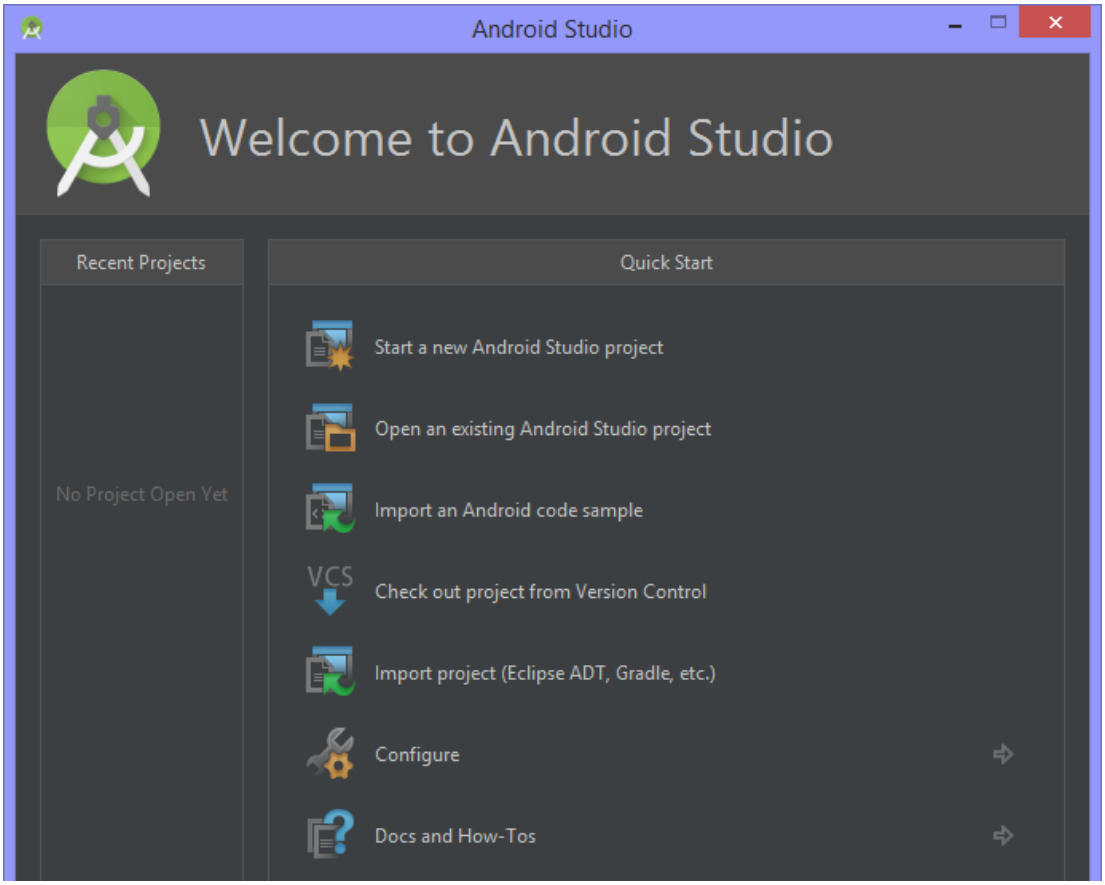
2.3.3 Device Manager

Device Manager je alat za pregled stanja uređaja koji su spojeni na računalo. Omogućuje praćenje memorije i pokrenutih procesa na uređaju te korištenje mreže. Prikazuje se i LogCat prozor kojeg je moguće filtrirati te tako pratiti željenu aplikaciju. U zasebnom odjeljku nalazi se upravitelj datoteka s kojim se pojedine datoteke prenose, kopiraju, preimenuju i brišu. Također je omogućeno detaljno upravljanje opcijama emulacije virtualnih uređaja, poput postavki poziva. Za vrijeme integracije u Eclipse ovaj je alat bio poznat pod nazivom DDMS[14] pogled te je starijim developerima vrlo dobro poznat. (Gassner, Android Studio Essential Training 2015)

[14] Dalvik Debug Monitor Server

3 STVARANJE APLIKACIJE

Stvaranje nove aplikacije najčešće započinje stvaranjem novog projekta u Android Studiu. Već na zaslonu dobrodošlice (prikazanom na slici 7) programer može odabrati želi li započeti novi projekt, otvoriti postojeći projekt ili ga stvoriti uvozom već postojećih projektnih datoteka iz drugih izvora. Ovo će se poglavlje baviti time kako stvoriti i prilagoditi novu aplikaciju.

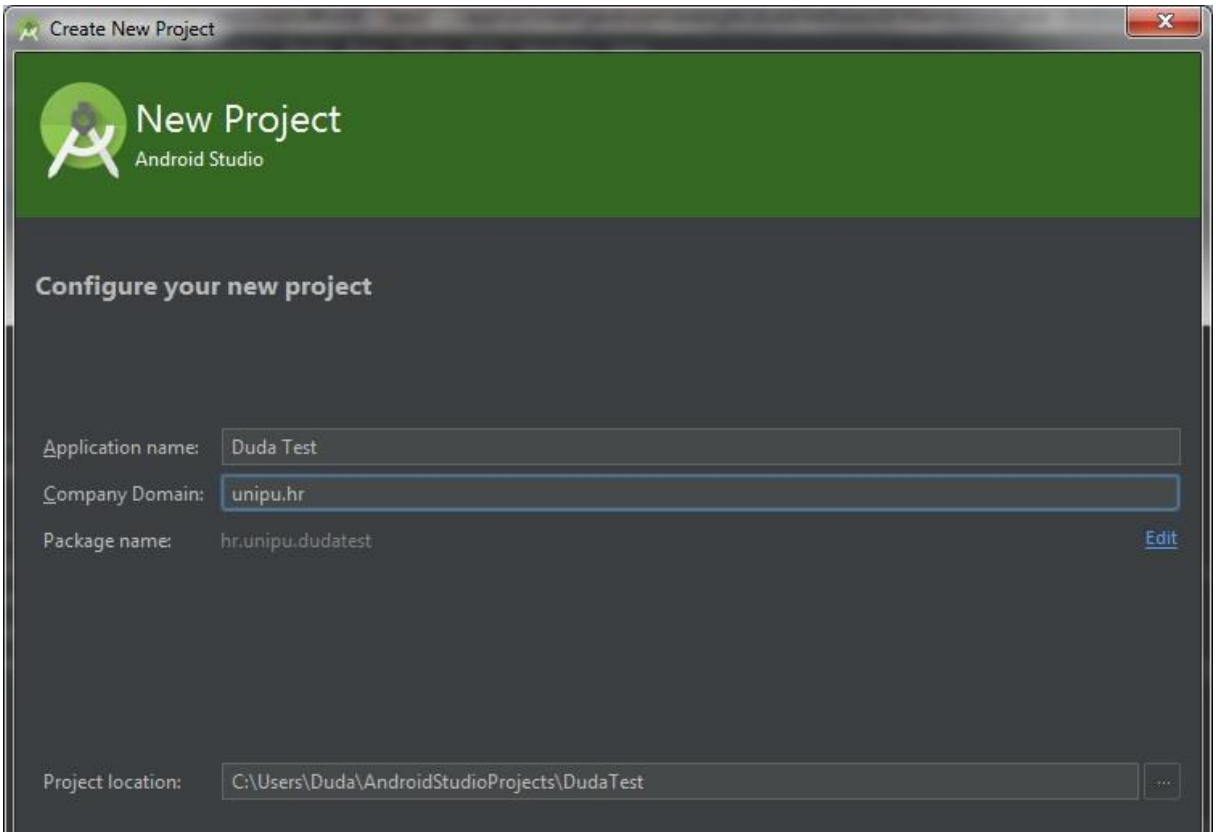


Slika 7. Početni zaslon Android Studia

Izvor: snimka ekrana autora

3.1 Osnovni podaci

Čarobnjak za izradu novog projekta traži ispunjavanje određenih polja kako bi stvorio i podesio novu aplikaciju. Sam čarobnjak je prvi korak ka stvaranju nove aplikacije i prikazat će se u cjelosti na sljedećim slikama uz prateći tekst.

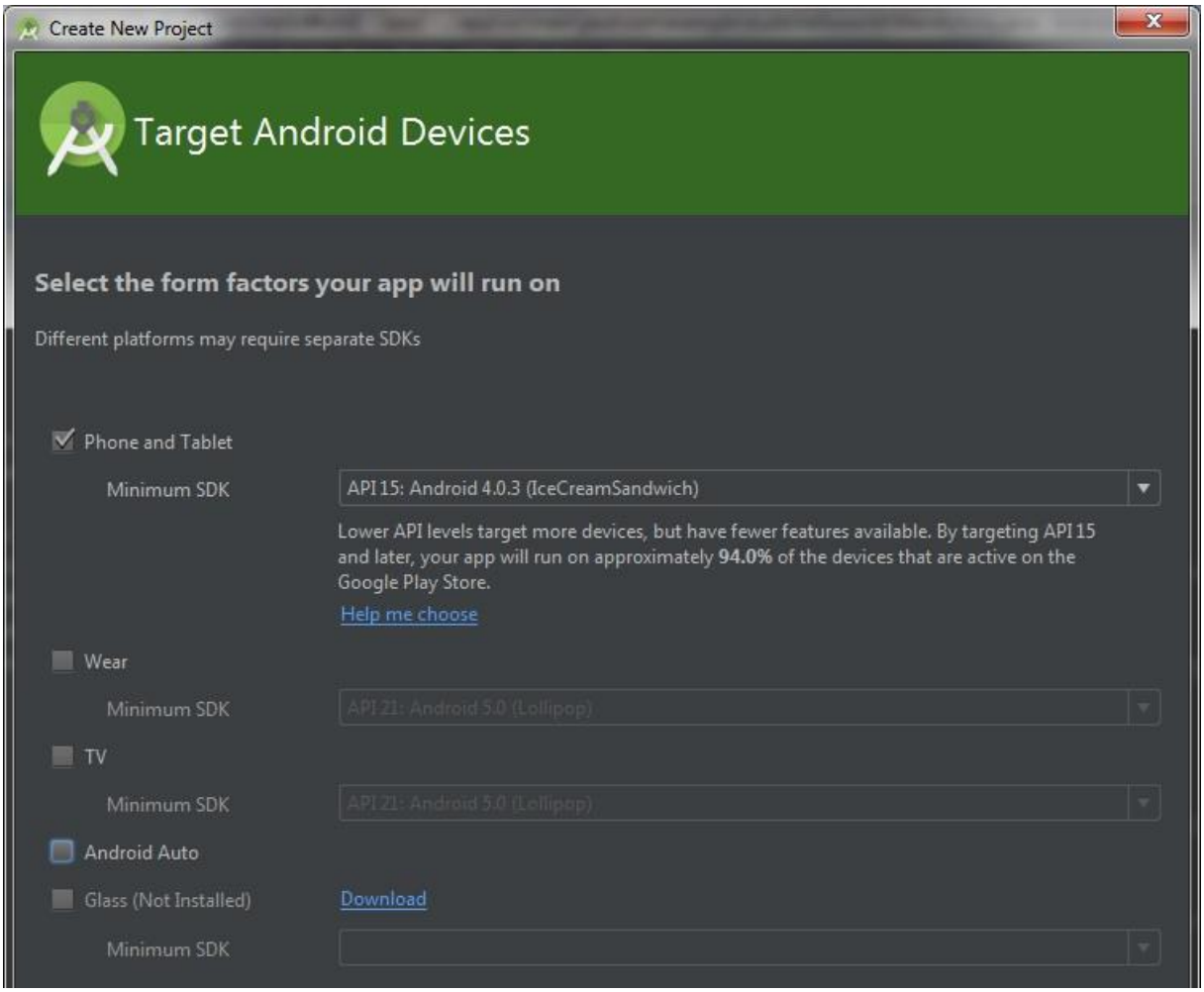


Slika 8. Izbor naziva i lokacije nove aplikacije

Izvor: Snimka zaslona autora

Na slici 8 prikazan je prvi korak čarobnjaka za izradu novog projekta. Tu se od programera očekuje da ispuni osnovne podatke o aplikaciji, kao što su:

- Naziv aplikacije – ime koje korisnici vide u pokretaču i Google Play trgovini
- Naziv domene – Web stranica autora ili tvrtke, koristi se za generiranje jedinstvenog imena paketa
- Naziv paketa – jedinstveni identifikator aplikacije, obično se koristi obrnuta domena tvrtke s imenom aplikacije što uvelike smanjuje rizik konflikta imena s ostalim aplikacijama
- Lokacija projekta – mjesto na lokalnom računalu gdje će se spremati projektni podaci

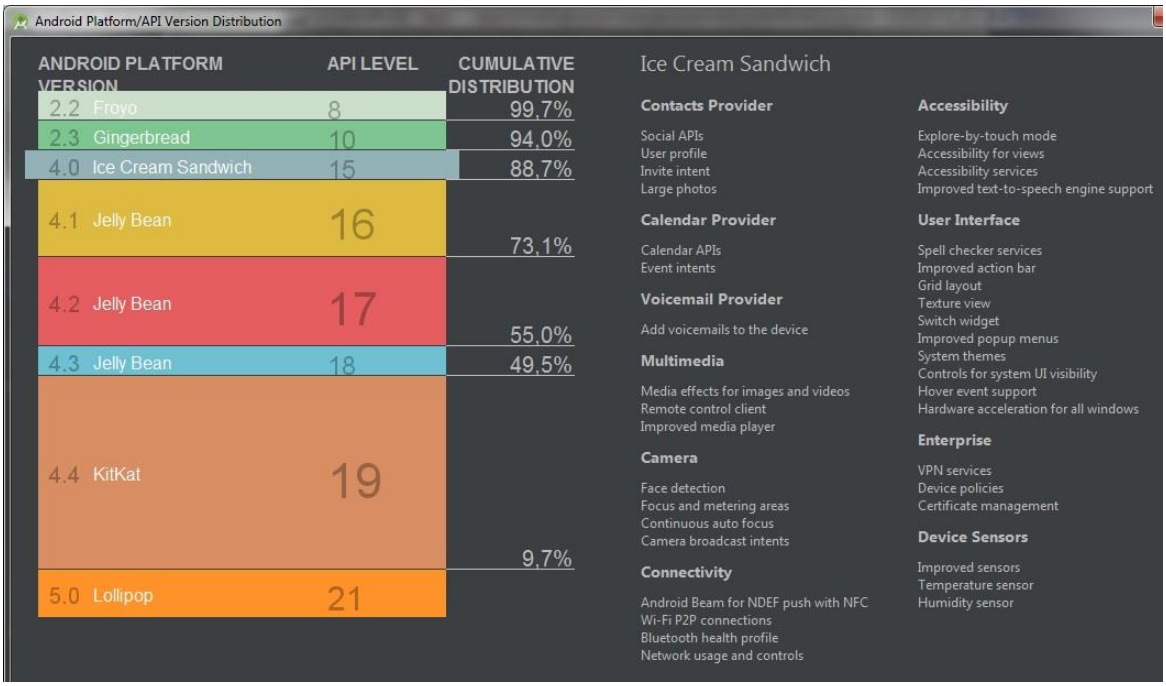


Slika 9. Izbor željenih platformi za aplikaciju

Izvor: snimka zaslona autora

Slika 9 prikazuje drugi korak čarobnjaka za izradu novog projekta koji se bavi ciljanjem aplikacije za određene uređaje i verzije. Moguće je odabrati bilo koju kombinaciju ciljnih uređaja i verzije Android OS-a, uključujući mobilne telefone i tablete, pametne ručne satove (Android Wear), pametne televizore, pametne naočale (Google Glass) te Android Auto, za koji još nije izdan SDK. Važno je odabrati minimalnu podržanu SDK verziju koja može instalirati i pokrenuti aplikaciju koja se izrađuje. Nužno je naći ravnotežu između široke podrške i novijih mogućnosti te tako odabrati odgovarajuću verziju API[15]-a. Tome, osim ranije spomenutih Google-ovih Web stranica, pomaže i ugrađeni statistički prikaz verzija Androida koji je detaljniji i prilagođeniji developerima jer prikazuje dostupne mogućnosti za odabranu verziju sustava što možemo vidjeti na slici 10.

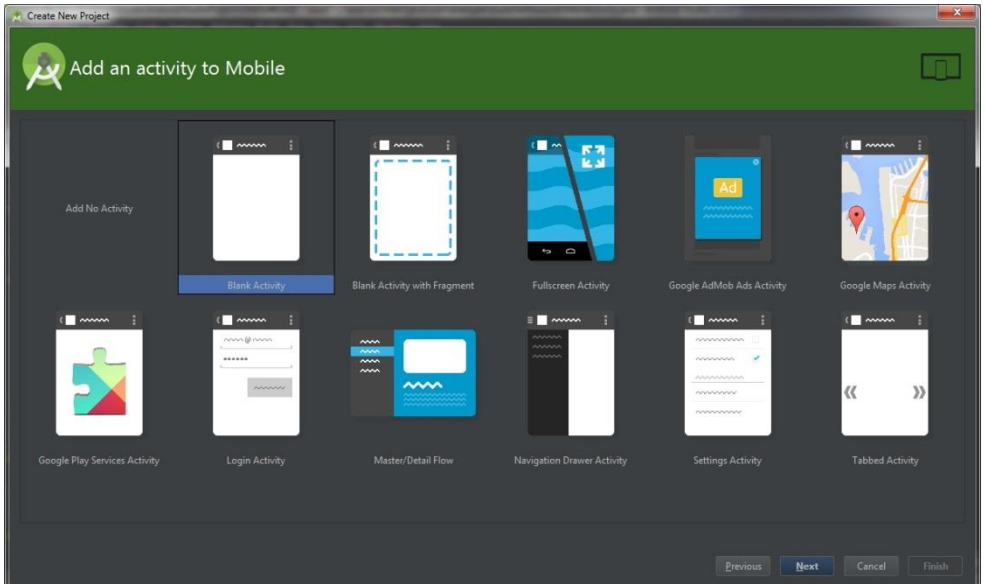
[15] eng. Application Programming Interface – sučelje za programiranje aplikacija



Slika 10. Prikaz distribucija Android verzija unutar Android Studia

Izvor: snimka zaslona autora

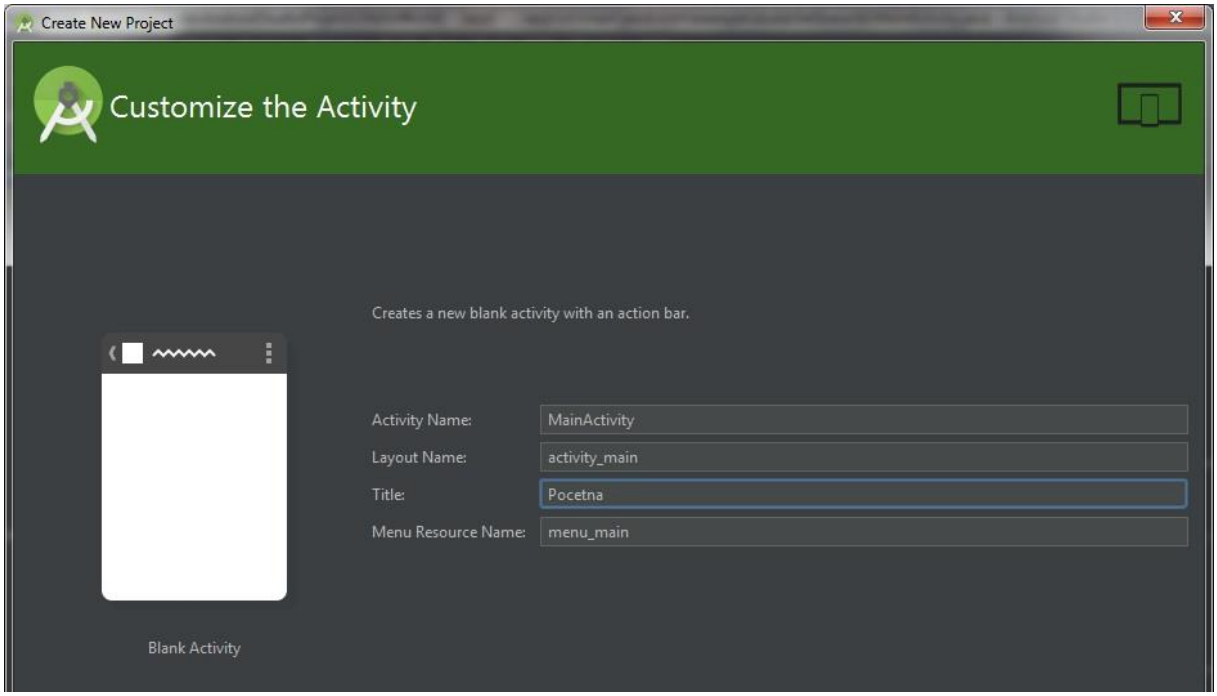
Sljedeći korak (slika 11) daje mogućnost stvaranja početne aktivnosti te nudi razne predloške kako bi ubrzao stvaranje sučelja. Kako verzije Androida i Android Studia napreduju, tako raste i broj ponuđenih predložaka za aktivnosti. Aktivnosti su temeljni dio Android aplikacije te će o njima više riječi biti u kasnijem poglavlju.



Slika 11. Odabir početne aktivnosti koristeći predloške

Izvor: snimka zaslona autora

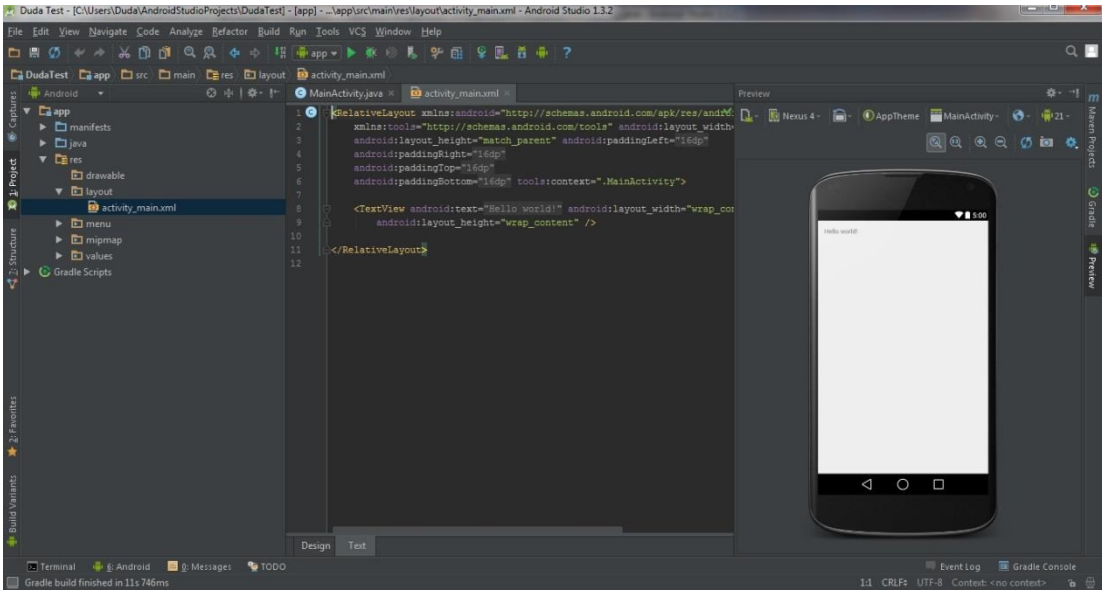
Nakon stvaranja početne aktivnosti, potrebno je odrediti njen naziv, naslov te koje će resurse koristiti za dizajn i izbornik. Sve to radi se u koraku koji je prikazan na slici 12.



Slika 12. Podešavanje nove aktivnosti

Izvor: snimka zaslona autora

Nakon svih prethodno prikazanih koraka, Android Studio koristi unešene podatke kako bi napravio strukturu projekta (više o tome u sljedećem poglavlju) te stvorio osnovno sučelje aplikacije koristeći odabrani dizajn početne aktivnosti. Otvara se editor XML koda i pretpregled na zaslonu mobilnog uređaja s tekstom „Hello World!“. Sa sučeljem koje je prikazano na slici 13 projekt je spreman i moguće je krenuti u kodiranje i dizajn aplikacije.



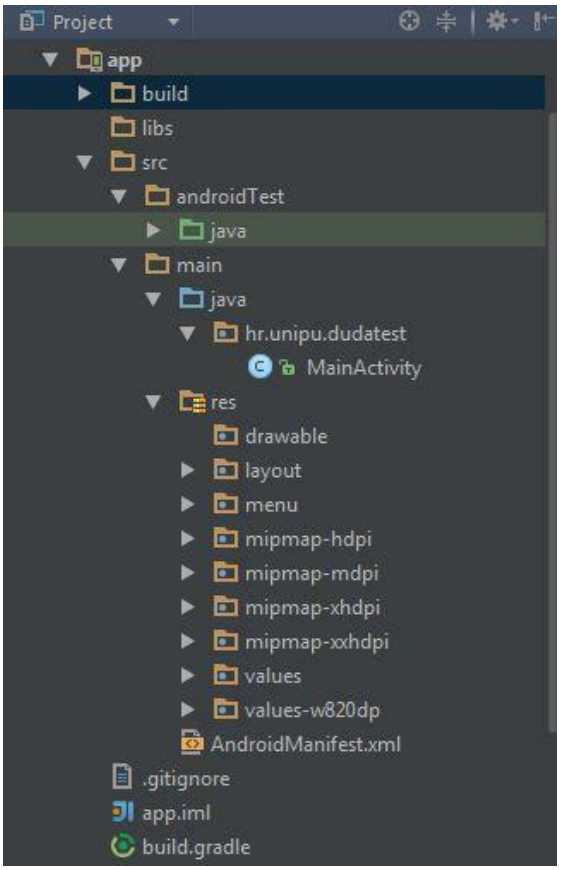
Slika 13. Sučelje s novonastalim projektom

Izvor: snimka zaslona autora

3.2 Ključne datoteke i struktura projekta

S lijeve strane sučelja Android Studia nalazi se preglednik projekta. Pomoću njega moguće je vidjeti otvorene projekte i sve datoteke koje tim projektima pripadaju. Kontekstni izbornik pruža opcije pretraživanja, dodavanja, preimenovanja i brisanja elemenata projekta. Također, moguće je vršiti analizu i refaktoriranje[16] koda, kao i uspoređivanje datoteka i otvaranje u posebnom prozoru preglednika datoteka.

[16] izmjena izvornog koda s ciljem boljeg praćenja najboljih praksi, primjerice izdvajanje ponavljajućih djelova koda u funkciju/metodu ili korištenje konstanti umjesto ponavljanja istih vrijednosti što olakšava izmjenu



Slika 14. Folder struktura Android projekta

Izvor: snimka zaslona autora

Koristeći preglednik projekta može se vidjeti struktura projekta, što je prikazano na slici 14. Direktorij s imenom aplikacije sadrži sve datoteke projekta strukturirane po poddirektorijima. Najvažnije datoteke i direktoriji projekta su:

- **build/** : folder koji sadrži kompajlirane resurse nakon izgradnje aplikacije i klase koje generiraju Android alati, kao što je R.java klasa koja sadrži reference na resurse aplikacije
- **libs/** : sadrži biblioteke koje koristi izvorni kod aplikacije
- **src/main/** : direktorij sa izvornim kodom aplikacije, sadrži najkorištenije datoteke
    - **java/** : sadrži Java klase organizirane po paketima. Stvaranjem novog projekta stvara se i njegova glavna aktivnost koja se nalazi u paketu s imenom aplikacije
    - **res/** : sadrži resurse projekta poput nekih slika i XML datoteka koje opisuju izgled sučelja i izbornike

- **drawable/** : sadrži slike koje aplikacija koristi, kategorizirane po različitim pikselnim gustoćama ekrana. Također sadrži ikonu aplikacije koja se stvorila s projektom
- **layout/** : sadrži XML definicije pogleda i njihovih elemenata
- **menu/** : sadrži XML definicije izbornika aplikacije
- **values/** : sadrži XML datoteke koje definiraju skupove uređenih parova ime →vrijednost. To mogu biti boje, stringovi ili stilovi. Više je direktorija vrijednosti organizirano po različitim veličinama ekrana kako bi za njih bolje prilagodili sučelje (npr. povećanje komponenti i veličine teksta na tabletima)
- **AndroidManifest.xml** : esencijalna datoteka Android projekta koja se s njim automatski generira. Deklarira osnovne informacije potrebne Android sustavu da pokrene aplikaciju – ime paketa, verziju, aktivnosti, dopuštenja, namjere i potreban hardware. Manifest je prikazan na slici 15

- **build.gradle** : skriptna datoteka za gradnju aplikacije

Navedeni se direktoriji stvaraju automatski s novim projektom, iako nije nužno za svaki projekt da sadrži sve navedene foldere – za manje projekte nema potrebe npr. definirati posebne nijanse željenih boja ili kompleksne varijante sučelja. (Zapata 2013, 31-34)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hr.unipu.dudatest" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Duda Test"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="Duda Test" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Slika 15. AndroidManifest.xml novostvorene aplikacije

Izvor: snimka zaslona autora

3.3 Aktivnosti

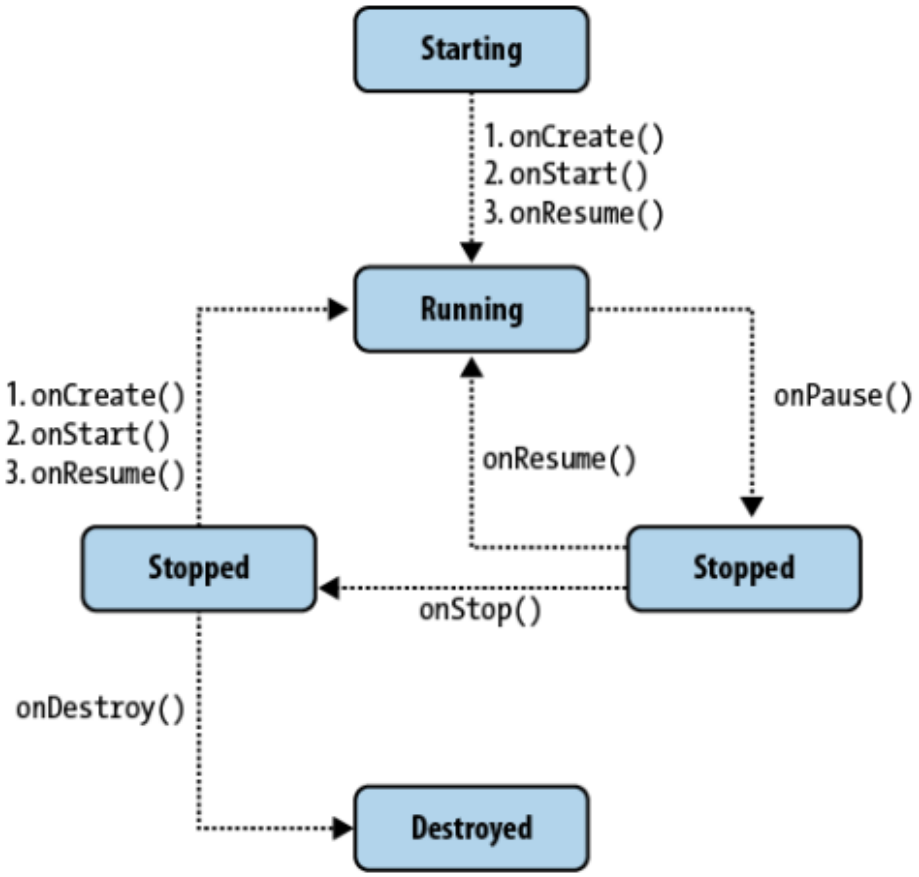
Stvaranje Android aplikacije obično počinje stvaranjem glavne aktivnosti (**Activity**). Nakon toga postavljaju se servisi (**Services**), primatelji emitiranja (**BroadcastReceivers**) i pružatelji sadržaja (**ContentProviders**) i svi se zajedno povežu pomoću namjera (**Intents**). Zajednički naziv za navedene djelove je komponenta.

Navedene komponente su one koje se najčešće koriste. Aktivnosti su odgovorne za korisničko sučelje, a servisi implementiraju operacije koje se pokreću u pozadini. Primatelji emitiranja čekaju događaje sustava, dok pružatelji sadržaja spremaju podatke aplikacije. (Hellman 2014, 75)

Aktivnost je obično jedan ekran kojeg korisnik vidi na uređaju. Aplikacija se obično sastoji od više aktivnosti pa se one izmjenjuju kako korisnik koristi aplikaciju. To čini aktivnost vizualno najeksponiranijom komponentom.

Kao što se Web projekt sastoji od više stranica, tako se aplikacija sastoji od više aktivnosti. Dok Web projekt ima početnu stranicu, aplikacija ima glavnu (main) aktivnost koja se prikazuje po pokretanju. Kao što je moguće prebacivanje s jedne Web stranice na drugu putem navigacije i poveznica, tako je moguće i mijenjanje prikazane aktivnosti na mobilnom uređaju.

Aktivnosti mogu imati nekoliko stanja i na programeru je da odredi kako će se aplikacija ponašati ovisno o stanju određene aktivnosti. Životni ciklus aktivnosti i tranzicijske metode koje se javljaju prikazani su na slici 16.



Slika 16. Životni ciklus aktivnosti

Izvor: Gargenta, Marko. Learning Android. Sebastopol: O'Reilly Media, Inc., 2011. str.49

3.3.1 Početno stanje

Dok aktivnost još uvijek ne postoji u memoriji, nalazi se u početnom stanju. Kako se pokreće, aktivnost prolazi kroz mnošto povratnih metoda dok ne prijeđe u sljedeće stanje. Taj je proces jedna od najskupljih operacija kad je u pitanju korištenje procesora, a samim tim i baterije uređaja. Upravo to je razlog zašto se aktivnosti ne uništavaju automatski čim više nisu prikazane, jer postoji mogućnost da ih korisnik ponovno želi vidjeti.

3.3.2 Stanje izvođenja

Aktivnost u stanju izvođenja je ona koja je trenutno prikazana na zaslonu i omogućuje korisniku interakciju. Ta aktivnost ima fokus – sve korisnikove interakcije, bilo dodiri, tipkanje ili klikanje – prenose se upravo toj aktivnosti. Iz tog razloga u nekom trenutku može postojati samo jedna aktivnost u stanju izvođenja. Kako bi korisnik dobio dojam da je sučelje vrlo responzivno, aktivnost u stanju izvođenja ima najveći prioritet korištenja memorije i resursa uređaja.

3.3.3 Stanje pauze

Kad aktivnost nema fokus, ali je još uvijek vidljiva na zaslonu – kaže se da je u stanju pauze. To nije tipičan scenarij jer su zasloni uređaja (isključujući tablete) najčešće mali pa aktivnosti zauzimaju cijeli ekran ili ne zauzimaju uopće. Najčešći je slučaj dijaloški okvir koji iskoči ispred aktivnosti te ona tako postane pauzirana. Sve aktivnosti prođu kroz stanje pauze prije nego ih se zaustavi.

Pauzirane aktivnosti i dalje imaju visok prioritet korištenja memorije i ostalih resursa. Razlog tome je što su i dalje vidljive na zaslonu i ne može ih se ukloniti bez da korisniku to izgleda neprirodno.

3.3.4 Zaustavljeno stanje

Kad aktivnost nije vidljiva, a i dalje je prisutna u memoriji – kaže se da je u zaustavljenom stanju. Zaustavljene se aktivnosti mogu vratiti u stanje izvođenja i opet postati vidljive, ili ih se može uništiti i izbrisati iz memorije.

Sustav drži aktivnosti u zaustavljenom stanju u slučaju da ih korisnik želi ponovno pokrenuti i mnogo je brže nastaviti zaustavljenu aktivnost nego ponovno pokrenuti novu. Kako su svi djelovi aktivnosti već u memoriji, potrebno ju je samo dovesti u prvi plan. Zaustavljene aktivnosti mogu biti uklonjene iz memorije u bilo kojem trenutku.

3.3.5 Uništeno stanje

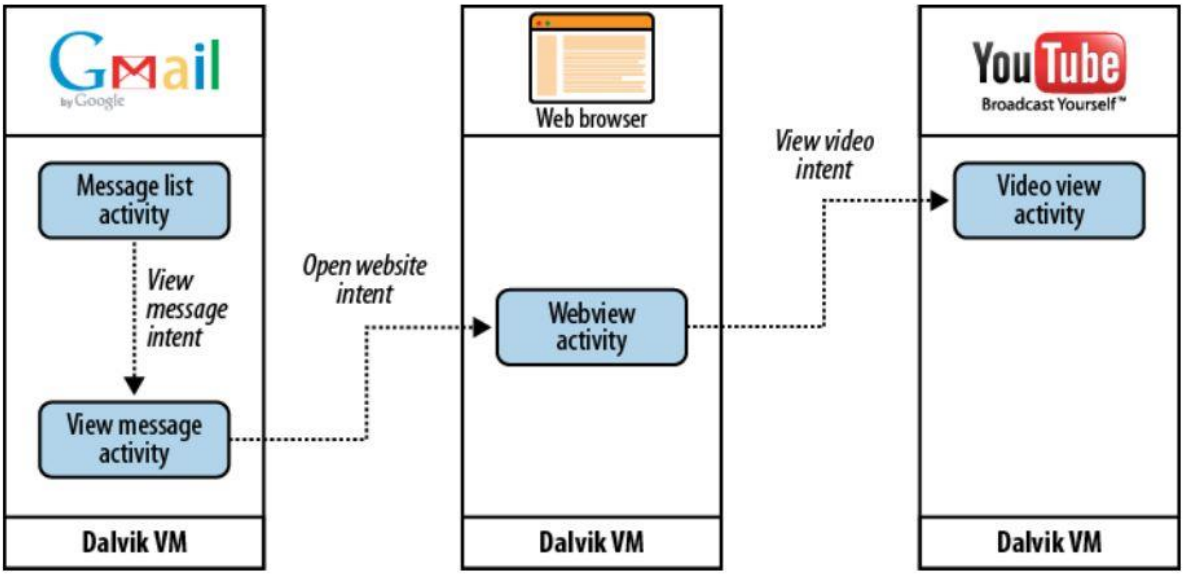
Uništena aktivnost više ne postoji u memoriji. Prije uništenja moguće je obaviti neke akcije, npr. spremiti podatke. Nema garancije da će aktivnost biti zaustavljena prije uništenja, kao ni da pauzirana aktivnost neće biti uništena. Stoga je najbolje važne akcije poput spremanja podataka obaviti na putu prema stanju pauze radije nego prema uništenom stanju. (Gargenta 2011, 48-50)

3.4 Namjere

Namjere su poruke koje se šalju među glavnim komponentama. Uzrokuju pokretanje aktivnosti, pokreću ili zaustavljaju servise ili se jednostavno emitiraju. Asinkrone su što znači da kod koji ih je poslao ne mora čekati da se izvrše.

Namjere mogu biti implicitne i eksplicitne. Pomoću eksplicitnih namjera pošiljatelj jasno označava koja komponenta treba biti primatelj. Kod implicitnih namjera pošiljatelj bira tip primatelja. Primjerice, ako aktivnost šalje namjeru kako je potrebno prikazati Web stranicu, svaka instalirana aplikacija može poslužiti kao primatelj. Ukoliko više aplikacija

može izvršiti željenu radnju, Android će ponuditi korisniku odabir jedne i mogućnost postavljanja zadane aplikacije za buduće radnje istog tipa. To omogućuje korisniku korištenje bilo koje aplikacije za određeni tip radnji, uključujući pozive, slanje i primanje SMS poruka i slično. Slika 17 prikazuje kako se namjere mogu koristiti za prebacivanje između raznih aktivnosti unutar jedne ili više aplikacija. (Gargenta 2011, 51)



Slika 17. Korištenje namjera za prebacivanje među raznim aktivnostima

Izvor: Gargenta, Marko. Learning Android. Sebastopol: O'Reilly Media, Inc., 2011. str. 51

3.5 Servisi

Servisi se pokreću u pozadini i nemaju kontrole korisničkog sučelja, iako mogu obavljati iste radnje kao aktivnosti. Korisni su za akcije koje treba izvoditi dulje vrijeme, neovisno što je trenutno na ekranu. Tako je, primjerice, moguće slušati glazbu dok se na zaslonu ne prikazuje multimedijski svirač.

Za razliku od aktivnosti, servisi imaju mnogo jednostavniji životni ciklus. Servis je ili pokrenut ili zaustavljen i programer ima gotovo potpunu kontrolu nad tim. Stoga je potrebno voditi računa o pokretanju servisa tako da oni nepotrebno ne troše zajedničke resurse uređaja, poput memorije i procesora. (Gargenta 2011, 51-52)

3.6 Pružatelji sadržaja

Pružatelji sadržaja su sučelja za razmjenu podataka među aplikacijama. Po zadanim postavkama Android drži sve podatke aplikacija odvojene jedne od drugih. Iako mala količina

podataka može biti razmijenjena koristeći namjere, pružatelji sadržaja mnogo su prikladniji za dijeljenje trajnih podataka između npr. baze podataka i aplikacije.

Android sustav stalno koristi pružatelje sadržaja za svakodnevne radnje. Tako poseban pružatelj sadržaja omogućuje prikaz podataka o kontaktima aplikacijama kojima je to potrebno. Slično je i za multimedijske datoteke, postavke uređaja i slično. Ovakav način podjele aplikacije s korisničkim sučeljem i samih podataka omogućuje veliku fleksibilnost pri odabiru načina prikaza istih podataka.

Pružatelji sadržaja jednostavna su sučelja sa standardnim metodama za pregled, unos, mijenjanje i brisanje podataka, što odgovara većini metoda za rad s bazama podataka. (Gargenta 2011, 52-54)

3.7 Primatelji emitiranja

Primatelji emitiranja su implementacija mehanizma za čekanje na određene događaje širom Android sustava. Primatelj je kod koji čeka izvršenje dok se događaj kojeg očekuje ne pojavi. Kaže se da je primatelj pretplaćen na neki događaj.

Android sustav stalno emitira događaje. Primanje SMS-a, pražnjenje baterije i paljenje sustava su sve emitirani događaji koje prate neke aplikacije. Sami primatelji emitiranja nemaju vizualno sučelje niti su pokrenuti u memoriji. Kad ih pokrene primanje ciljanog događaja, mogu izvršiti neki kod, poput pokretanja aktivnosti, servisa i slično. (Gargenta 2011, 54)

4 TESTIRANJE I DEBUGIRANJE

Kako je već rečeno u ranijem poglavlju, uz Android Studio dolazi i Android Emulator. To je program koji koristi hardware računala na kojem je pokrenut i sliku Android sustava kako bi imitirao Android uređaj te tako omogućio testiranje aplikacija.

Emulator može birati između više AVD[17]-ova koji se stvaraju i definiraju pomoću AVD Managera. Moguće je imati mnogo virtualnih uređaja s raznim rezolucijama ekrana i verzijama Android sustava. Na taj se način jedna aplikacija može testirati na reprezentativnom broju različitih scenarija bez potrebe za posjedovanjem fizičkih uređaja. AVD Manager prikazan je na slici 18.

Slika 18. Pregled AVD-ova u AVD Manageru

Izvor: snimka zaslona autora

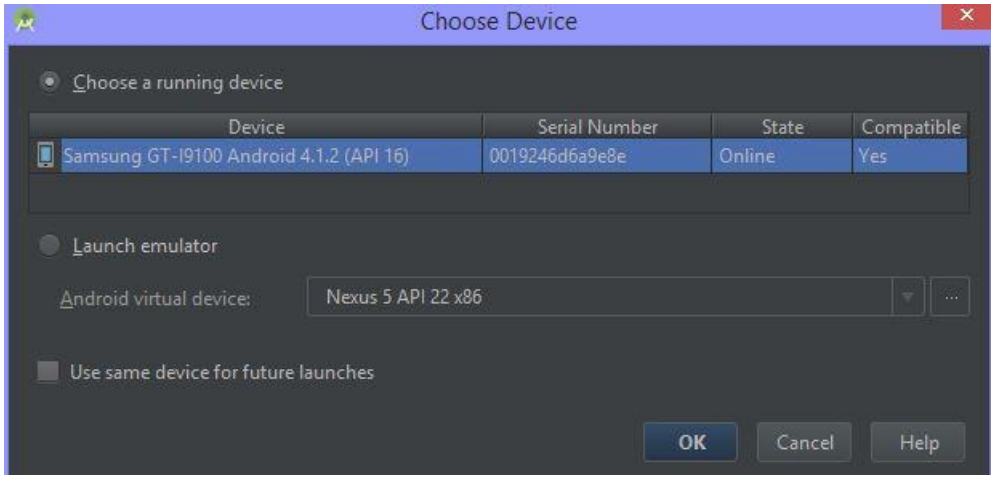
4.1 Stvaranje i mijenjanje virtualnog uređaja

AVD Manager pruža odabir mnogih opcija za virtualne uređaje. Osim naziva, uređaja na kojem se bazira, rezolucije i Android sustava, moguće je odabrati hardverske opcije poput veličine dijagonale zaslona, skaliranje virtualnog zaslona na fizički, količine radne memorije i memorije za spremanje podataka, emulacije kamera i brzine mreže. Za ubrzavanje izvedbe moguće je koristiti grafičku karticu računala ili zapisivanje stanja uređaja na disk pri gašenju kako bi se ubrzalo sljedeće paljenje. Zasloni virtualnih uređaja okruženi su slikovnim okvirima kako bi što više ličili na pravi uređaj, iako je to moguće isključiti.

[17] eng. Android Virtual Device – virtualni Android uređaj

4.2 Pokretanje aplikacije

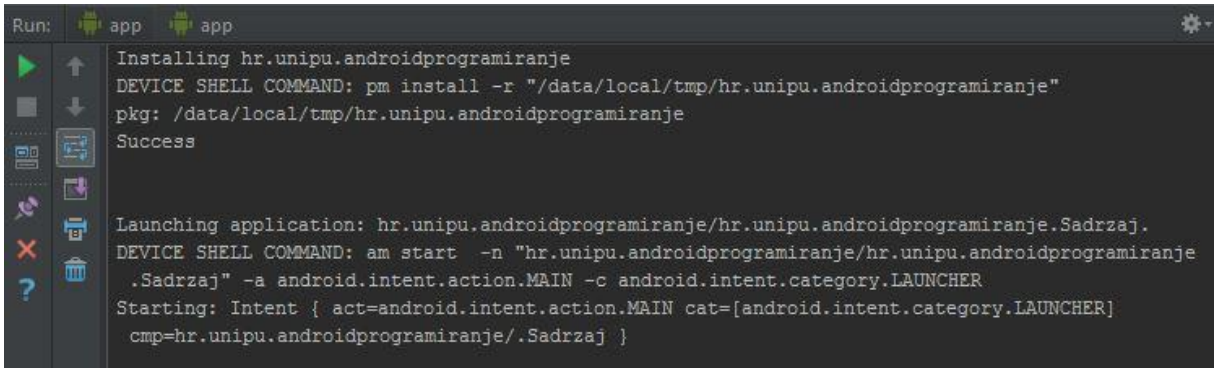
Korištenjem virtualnog ili fizičkog Android uređaja, pokretanje i testiranje aplikacije iz Android Studia počinje na isti način – odabirom opcije Run iz istoimenog izbornika ili klikom zelene strelice na traci s alatima. To otvara popis mogućih uređaja koji mogu pokrenuti aplikaciju. Može se birati između fizičkih uređaja spojenih USB kablom na računalo, pokrenutih virtualnih uređaja u emulatoru te dostupnih virtualnih uređaja koji trenutno nisu pokrenuti. Odabir uređaja prikazan je na slici 19.



Slika 19. Izbor ciljanog Android uređaja

Izvor: snimka zaslona autora

Odabirom željenog uređaja u pozadini se pokreće alat Android Debug Bridge, poznatiji kao adb. Pomoću njega Android Studio prenosi i instalira aplikaciju na ciljani uređaj te se ona počinje izvoditi. Istovremeno se na dnu sučelja Android Studia pojavljuje prozor Run koji prikazuje poruke o izvođenju naredbi. Po pokretanju aplikacije na uređaju prozor Run zamjenjuje se prozorom Android. Run prozor prikazan je na slici 20.



Slika 20. Run prozor tijekom pokretanja aplikacije

Izvor: snimka zaslona autora

4.3 LogCat

LogCat je Androidov sustav bilježenja poruka koje generira sustav i pokrenute aplikacije. Prikazan je na dnu sučelja Android Studia i sastavni je dio prozora Android. Zabilježene poruke imaju nekoliko razina važnosti po kojima ih je moguće filtrirati. Od općenitih do kritičnih, te su razine sljedeće: verbose (detaljnije poruke), debug (poruke namijenjene otklanjanju greški i tijekom razvoja), information (informacije), warning (upozorenja) i error (greške). (Gargenta 2011, 81)

Korištenje LogCat-a za praćenje poruka aplikacije vrlo je korisno tijekom razvoja. Moguće je zapisati specifične poruke u LogCat koristeći metode klase Log. Koja se metoda koristi ovisi o razini poruke koja se želi zapisati, npr. Log.d() zapisuje debug razinu poruke. (Gassner, Android Studio Essential Training 2015)

4.4 Debugiranje

Za pokretanje aplikacije u debug načinu koristi se ikona Debug koja se nalazi odmah uz Run ikonu. Aplikacija se pokreće na uobičajeni način, ali se po pokretanju na nju vezuje debugger. Na dnu sučelja Android Studia prikazuje se debug prozor u kojem je prikazan ispis LogCat-a, slično kao u Android prozoru.

Prednost pokretanja aplikacije u debug načinu je mogućnost zaustavljanja izvođenja aplikacije na određenom mjestu u izvornom kodu. Takvo mjesto, poznato pod nazivom breakpoint[18], dodaje se klikom pored linije izvornog koda koja je u pitanju. Također je moguće uvjetno zaustavljanje izvođenja aplikacije dodajući uvjete u točku prekida. U trenutku izvođenja te linije, debugger zaustavlja aplikaciju i prikazuje vrijednosti varijabli koje aplikacija koristi. Pritom je moguće selektivno izvršavati liniju po liniju koda istovremeno prateći vrijednosti relevantnih varijabli. (Gassner, Android Studio Essential Training 2015)

[18] eng. točka prekida

5 PLASIRANJE NA GOOGLE PLAY TRGOVINU

Iako nije jedini način za distribuciju gotovih aplikacija, Google Play trgovina je najveća i najpopularnija opcija koja sadrži preko milijun i pol aplikacija. Za mogućnost izdavanja aplikacija na Google Play potrebno je posjedovati izdavački račun i apk[19] paket željene aplikacije. Također je bitno pratiti build.gradle datoteku aplikacije u kojoj se nalazi brojčana oznaka verzije aplikacije. Ta se verzija razlikuje od prikazane po tome što se uvijek povećava za 1 te tako omogućuje Android sustavu praćenje dostupnih ažuriranja. S druge strane, prikazana verzija se smatra tekstualnim podatkom i moguće ju je oblikovati po želji. (Gassner, Distributing Android Apps 2015)

5.1 Pakiranje i zaštita aplikacije

Nakon implementacije i testiranja aplikacije na reprezentativnom broju različitih uređaja, potrebno ju je zapakirati. Pakiranjem se dobiva apk datoteka koja je u stvari komprimirana zip datoteka i u sebi sadrži više direktorija i datoteka. U to se ubraja manifest aplikacije, potpisani certifikat autora, platformni kod za ciljanu Android verziju, kompajlirane Java klase, resursi i ostali sastavni djelovi aplikacije.

Stvaranje apk datoteke počinje iz izbornika Build → Generate Signed APK. To otvara dijaloški okvir u kojem je potrebno odabrati ključnu datoteku. Moguće je stvoriti novu ili koristiti postojeću. Ključna datoteka (koja ima ekstenziju .jks) sadrži ključ i certifikat autora aplikacije zaštićene lozinkom. Moguće je odabrati vijek trajanja ključa koji je preporučen na 25 godina. Po izradi i odabiru ključa i certifikata, bira se lokacija apk datoteke. Potom se generira apk datoteka koju je moguće distribuirati.

Uz Android SDK dolazi i besplatni alat imena ProGuard čija je zadaća optimizirati izvorni kod i smanjiti razumljivost Java paketa. Uz njegovu pomoć moguće je smanjiti veličinu Android paketa i onemogućiti neželjenu rekonstrukciju aplikacija iz izvornog koda. Također se može koristiti za automatsko uklanjanje debug linija koda iz završne verzije aplikacije, što uklanja potrebu za ponovnom analizom izvornog koda i ručno uklanjanje potencijalno kompleksnih debug naredbi. ProGuard se uključuje u build.gradle datoteci prije stvaranja apk datoteke. (Gassner, Distributing Android Apps 2015)

[19] APK – Android PacKage

5.2 Distribucija

Apk datoteku moguće je pokrenuti na bilo kojem uređaju te će ona instalirati aplikaciju. Ta činjenica omogućuje distribuciju aplikacije bez potrebe za objavljivanjem u bilo kojoj službenoj trgovini – jednostavno je moguće objaviti apk datoteku, primjerice na Web stranici, i svatko zainteresiran može pristupiti aplikaciji.

Ipak, za komercijalnu distribuciju logični je izbor Google Play trgovina. Korisnici više vjeruju takvom sustavu zbog automatskih i ručnih provjera prenesenih aplikacija. Također je veća vjerojatnost da će aplikacija biti pronađena zbog kategorija i pretraživanja. Aplikacije se provjeravaju kako ne bi sadržavale maliciozni kod, ali ne provjerava se sadržaj aplikacije. To čini objavljivanje na Google Play puno bržim od nekih trgovina koje detaljnije provjeravaju pristigle aplikacije.

Da bi aplikaciju objavio na Google Play, programer osim apk datoteke treba pružiti dodatne informacije o njoj. Potrebno je odabrati naziv za prikaz, sastaviti kraći i dulji tekstualni opis aplikacije, barem dva slikovna prikaza izvođenja aplikacije na raznim uređajima te joj definirati cijenu. Također je potrebno stvoriti nekoliko promotivnih slika raznih veličina kako bi se aplikaciju lakše automatski promoviralo unutar Google Play trgovine. Moguće je i povezati videozapis korištenja aplikacije koji će biti prikazan među ostalim grafičkim prikazima. Prije objavljivanja potrebno je još ručno opisati aplikaciju birajući spada li u općenite aplikacije ili igre, kojoj kategoriji pripada te ispuniti upitnik o sadržaju kojim se dobiva certifikat odobrenja sadržaja.

Nakon objave dostupne su razne mogućnosti praćenja uspjeha aplikacije. Moguće je vidjeti broj instalacija te filtrirati prikaz po verziji Android sustava, nazivu uređaja, jeziku i zemlji korisnika. (Gassner, Distributing Android Apps 2015)

5.3 Mogućnosti zarade

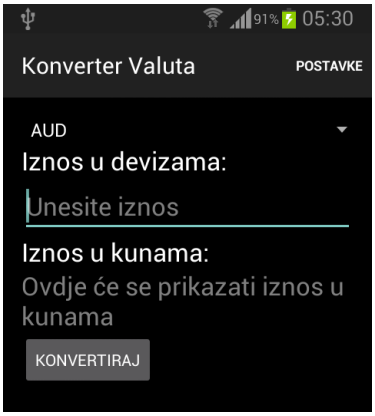
U odnosu na aplikacije za iOS, većina Android aplikacija je besplatna. Iako je moguće naplaćivati samu aplikaciju, najčešće prihodi dolaze od oglašavanja i prodaje unutar aplikacije. Za naplatu prihoda potrebno je posjedovati Google Wallet račun i unutar aplikacije omogućiti korištenje Google-ovih API-ja za naplatu i/ili oglašavanje. (Hellman 2014, 408)

6 PRIMJER APLIKACIJE

Kroz prethodna poglavlja pokazalo se što čini Android aplikaciju te kako stvoriti novu. Ovo poglavlje će spomenute tehnike prikazati pomoću jednostavne Android aplikacije za konverziju valuta.

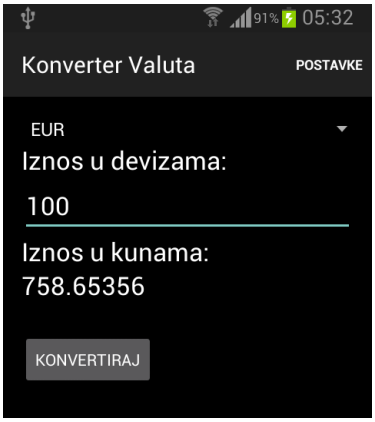
6.1 Opis funkcionalnosti

Aplikacija ima dva zaslona – početni zaslon i zaslon postavki. Na početnom se zaslonu nalazi padajući izbornik pomoću kojeg korisnik može odabrati jednu od šest ponuđenih stranih valuta. Ponuđene valute uključuju australski, kanadski i američki dolar, švicarski franak, britansku funtu i euro. Nakon odabira valute, korisnik dodirom na predviđeno polje unosi iznos strane valute koji želi konvertirati u kune. Potom, pritiskom na gumb Konvertiraj aplikacija množi iznos valute s interno spremljenim tečajem te valute te prikazuje rezultat na predviđenom mjestu. Te su aktivnosti prikazane na slikama 21 i 22.



Slika 21: Početni zaslon aplikacije.

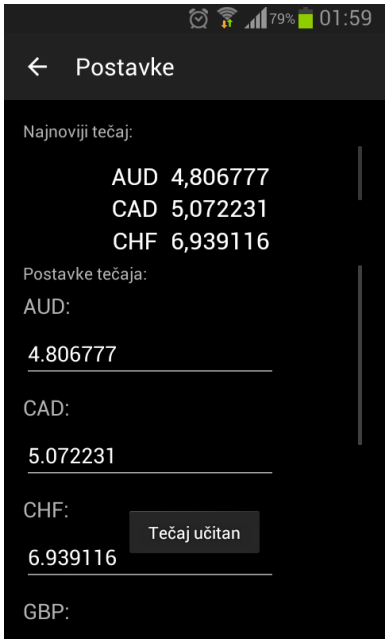
Izvor: snimka zaslona autora



Slika 22: Konverzija EUR u HRK.

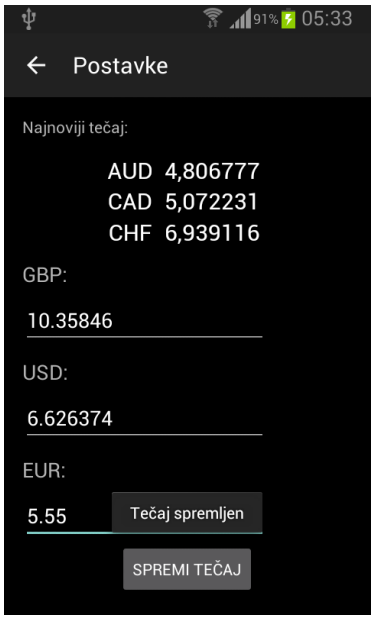
Izvor: snimka zaslona autora

Odabirom opcije Postavke korisniku se prikazuje drugi zaslon aplikacije gdje mu je omogućeno uređivanje tečaja svih valuta koje se mogu konvertirati pomoću aplikacije. Pri vrhu zaslona pomoću integriranog Web prozorčića prikazana je Web stranica s najnovijim tečajem spomenutih šest valuta. Ispod Web prikaza nalazi se formular s poljima za upis novog tečaja za svaku valutu. Tečaj je spremljen u internu datoteku s postavkama aplikacije te se učitava automatski prilikom pokretanja aplikacije. Mijenjanjem prikazanog tečaja i dodirom gumba Spremi tečaj pohranjuju se izmjene. Zaslon postavki prikazan je na slikama 23 i 24.



Slika 23: Zaslon postavki automatski učitava tečaj.

Izvor: snimka zaslona autora



Slika 24: Spremljeni tečaj vrijedi za buduće konverzije.

Izvor: snimka zaslona autora

Promijenjeni tečaj primjenjuje se za buduće konverzije dok se ponovno ručno ne izmijeni. Za potrebe demonstracije, tečaj eura postavljen je na 5,55 kuna. Rezultat konverzije s takvim tečajem prikazan je na slici 25.

Slika 25: Primjena novog tečaja.

Izvor: snimka zaslona autora

6.2 Korištene kontrole i komponente

Budući da se aplikacija sastoji od dva zaslona, za njenu izradu bilo je potrebno stvoriti dvije aktivnosti te svakoj dodijeliti XML datoteku sa dizajnom sučelja. S početne aktivnosti na aktivnost postavki kreće se pomoću eksplicitne namjere sadržane unutar funkcije koja se pokreće kao odgovor na dodir opcije Postavke. Izvorni kod funkcije prikazan je na slici 26.

```
public void goSettings(MenuItem item) {
    Intent settingsIntent = new Intent(this, SettingsActivity.class);
    startActivity(settingsIntent);
}
```

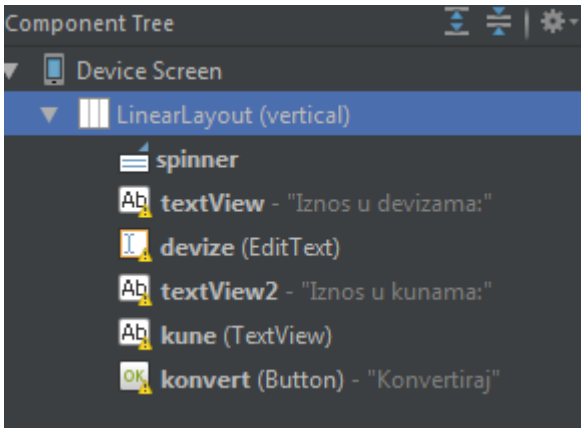
Slika 26: Funkcija za prikaz aktivnosti postavki.

Izvor: snimka zaslona autora

Za dizajn sučelja korištena je kombinacija LinearLayout i RelativeLayout kontejnera. Budući da zaslon postavki ima podosta sadržaja, korišten je kontejner ScrollView koji omogućuje korisniku da povlači prstom sadržaj koji trenutno nije vidljiv na zaslonu. Padajući izbornik je napravljen pomoću Spinner kontrole kojoj je dodijeljen niz (eng. Array) String[20]

[20] tip podataka korišten za prikaz kraćih ili dužih nizova slova

vrijednosti s troslovnim nazivima valuta. Za prikaz teksta korištene su textView kontrole s različitim postavkama veličine teksta, dok su polja predviđena za unos podataka napravljena pomoću editText kontrola. Aplikacija ima i dva gumba koji su stvoreni koristeći kontrolu Button. Prikaz korištenih kontrola početnog zaslona prikazan je na slici 27.



Slika 27: Korištene kontrole na početnom zaslonu aplikacije.

Izvor: snimka zaslona autora

Prikazane kontrole Android Studio je označio žutim trokutom kako bi skrenuo pozornost na njih zbog nepoštovanja najbolje prakse. Naime, prikazani tekst je fiksno napisan u XML kodu, dok je ustaljena praksa koristiti datoteke s tekstualnim resursima kako bi se olakšalo mijenjanje teksta i prevođenje aplikacije na druge jezike.

6.3 Ključni djelovi koda

Svim bitnim kontrolama nužno je dodijeliti jedinstveni identifikator (ID) pomoću kojeg je moguće pristupiti određenoj kontroli iz Java koda te upravljati njenim ponašanjem i događajima vezanim uz nju. Primjerice, padajućem izborniku dodijeljen je ID „spinner“ te mu je moguće pristupiti korištenjem metode findViewById. Tada se spremi u varijablu i dodijeli mu se posebna metoda koja reagira na specifične događaje, u slučaju padajućeg izbornika to je izbor neke od ponuđenih stavki. U ovom slučaju izbor određene valute iz padajućeg izbornika postavlja globalnu varijablu tecaj u vrijednost koju pročita iz interne datoteke postavki. Izvorni kod za spomenute akcije prikazan je na slici 28.

```
// postavljanje varijable padajućeg izbornika
Spinner dropdown = (Spinner) findViewById(R.id.spinner);

// čekanje na izbor valute
dropdown.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        Log.d("MainActivity", "odabrana valuta je  " + parent.getItemAtPosition(position));

        // postavljanje globalne varijable tečaj ovisno o izabranoj valuti
        try {
            switch ((int) parent.getItemIdAtPosition(position)) {
                case 0:
                    tecaj = postavke.getFloat(AUDTECAJ, (float) 4.806777);
                    break;
                case 1:
                    tecaj = postavke.getFloat(CADTECAJ, (float) 5.072231);
                    break;
                case 2:
                    tecaj = postavke.getFloat(CHFTECAJ, (float) 6.939116);
                    break;
                case 3:
                    tecaj = postavke.getFloat(GBPTECAJ, (float) 10.358460);
                    break;
                case 4:
                    tecaj = postavke.getFloat(USDTECAJ, (float) 6.626374);
                    break;
                case 5:
                    tecaj = postavke.getFloat(EURTECAJ, (float) 7.586536);
                    break;
            }

        } catch (Exception e) {
            Log.d("MainActivity", "onItemSelected error: "+e);
        }
    }
```

Slika 28: Izvorni kod za upravljanje padajućim izbornikom.

Izvor: snimka zaslona autora

Na sličan se način pristupa ostalim poljima za unos i prikaz podataka u aplikaciji. Kako je aplikaciji zadatak konvertirati stranu valutu u domaću ovisno o tečaju, algoritam za to je vrlo jednostavan: potrebno je pomnožiti količinu odabrane valute s njenim tečajem te to prikazati na predviđenom polju. Taj kod nalazi se u metodi koja se poziva prilikom odabira gumba Konvertiraj i prikazan je na slici 29.

```
// funkcija koja konvertira iznos deviza u kune, pokreće se klikom na gumb Konvertiraj
public void konvertiraj(View view) {

    // postavljanje varijabli iz tekstualnih polja
    TextView devizeView = (TextView) findViewById(R.id.devize);
    TextView kuneView = (TextView) findViewById(R.id.kune);

    // pretvaranje unosa deviza u decimalni broj
    String devizeText = devizeView.getText().toString();

    // za slučaj da se klikne Konvertiraj bez upisa iznosa
    try {
        devize = Float.parseFloat(devizeText);
    } catch (Exception e) {
        Log.d("MainActivity", "konvertiraj greška: " +e.getMessage());
    }

    // konvertiranje množenjem upisanog iznosa deviza sa srednjim tečajem
    float iznos = devize * tecaj;

    Log.d("MainActivity", "iznos = " + iznos );

    // ispis rezultata
    String iznosText = Float.toString(iznos);
    kuneView.setText(iznosText);

}
```

Slika 29: Izvorni kod metode za konverziju valute.

Izvor: snimka zaslona autora

Aktivnost postavki odmah pri pokretanju učitava postavke kako bi ih mogla prikazati na predviđenim mjestima. Također pri pokretanju pronalazi Web kontrolu i učitava željenu stranicu. Kod pokrenut pri stvaranju aktivnosti postavki prikazan je na slici 30.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);
    // učitavanje web stranice s tečajem
    WebView webview = (WebView) findViewById(R.id.webView);
    webview.loadUrl("http://tania.unipu.hr/~aduda/hnb/");
    // učitavanje postavki
    postavke = getSharedPreferences("tecajevi", MODE_PRIVATE);
    ucitajPostavke();
}
```

Slika 30: Izvorni kod koji se izvodi prilikom pokretanja aktivnosti postavki.

Izvor: snimka zaslona autora

Metoda ucitajPostavke() odgovorna je za prikaz spremljenih tečajeva u poljima za izmjenu kako bi se korisniku olakšalo mijenjanje pojedinog ili svih tečajeva. Procedura je već poznata: poljima se dodijele varijable pomoću ID-eva, zatim se u posebne varijable učita tečaj svake valute iz spremljenih postavki. Ukoliko nema spremljenih postavki, primjerice prilikom prvog pokretanja aplikacije na uređaju, dodjeljuje se svakoj valuti zadana vrijednost tečaja.

Tada se vrijednosti tečaja iz varijabli prikazuju na pojedinim poljima za izmjenu i prikazuje se poruka korisniku da je uspješno učitan tečaj. Izvorni kod metode za učitavanje postavki prikazan je na slici 31.

```
private void ucitajPostavke() {

    try {
        // postavljanje varijabli za rad s edit poljima
        EditText editAUD = (EditText) findViewById(R.id.editAUD);
        EditText editCAD = (EditText) findViewById(R.id.editCAD);
        EditText editCHF = (EditText) findViewById(R.id.editCHF);
        EditText editGBP = (EditText) findViewById(R.id.editGBP);
        EditText editUSD = (EditText) findViewById(R.id.editUSD);
        EditText editEUR = (EditText) findViewById(R.id.editEUR);

        // čitanje postavki iz datoteke
        Float tecajAUD = postavke.getFloat(AUDTECAJ, (float)4.806777);
        Float tecajCAD = postavke.getFloat(CADTECAJ, (float)5.072231);
        Float tecajCHF = postavke.getFloat(CHFTECAJ, (float)6.939116);
        Float tecajGBP = postavke.getFloat(GBPTECAJ, (float)10.358460);
        Float tecajUSD = postavke.getFloat(USDTECAJ, (float)6.626374);
        Float tecajEUR = postavke.getFloat(EURTECAJ, (float)7.586536);

        // postavljanje vrijednosti tečaja u edit polja
        editAUD.setText(tecajAUD.toString());
        editCAD.setText(tecajCAD.toString());
        editCHF.setText(tecajCHF.toString());
        editGBP.setText(tecajGBP.toString());
        editUSD.setText(tecajUSD.toString());
        editEUR.setText(tecajEUR.toString());

        // javlja uspješno učitavanje
        Toast.makeText(this, "Tečaj učitan", Toast.LENGTH_SHORT).show();

    } catch (Exception e) {
        Log.d("postavke", "ucitajPostavke greska: " +e);
    }

}
```

Slika 31: Izvorni kod metode za učitavanje postavki.

Izvor: snimka zaslona autora

Ukoliko želi promijeniti tečaj, korisnik treba upisati nove vrijednosti tečaja za željene valute i odabrati gumb Spremi tečaj. To poziva metodu za spremanje tečaja u datoteku postavki. Koristi se posebna Java klasa za rad s postavkama koja je dio Android SDK-a. Pomoću te klase i njenih metoda spremaju se pročitani podaci te zapisuju u datoteku. Tada se korisniku javlja poruka o uspjehu spremanja tečaja. Budući da je pristupanje kontrola i čitanje iz njih već nekoliko puta prikazano, slika 32 prikazuje samo dio metode spremiTecaj() koji je odgovoran za spremanje i zapisivanje tečaja u datoteku postavki.

```
        // spremanje iz varijabli u datoteku postavki
        SharedPreferences.Editor editor = postavke.edit();

        editor.putFloat(AUDTECAJ, Float.parseFloat(editTecajAUD));
        editor.putFloat(CADTECAJ, Float.parseFloat(editTecajCAD));
        editor.putFloat(CHFTECAJ, Float.parseFloat(editTecajCHF));
        editor.putFloat(GBPTECAJ, Float.parseFloat(editTecajGBP));
        editor.putFloat(USDTECAJ, Float.parseFloat(editTecajUSD));
        editor.putFloat(EURTECAJ, Float.parseFloat(editTecajEUR));

        editor.commit();

        // javlja uspješno spremanje
        Toast.makeText(this,"Tečaj spremljen", Toast.LENGTH_LONG).show();

    } catch (Exception e) {
        Log.d("postavke", "spremiTecaj greska: "+e);
    }
```

Slika 32: Dio metode za spremanje tečaja odgovoran za spremanje postavki u datoteku.

Izvor: snimka zaslona autora

ZAKLJUČAK

Programiranje za Android počinje prilagođavanjem tržištu i tehnologiji. Nastavlja se stvaranjem sučelja aplikacije u XML-u i kontroli ponašanja u Androidovoj verziji Java programskog jezika. Aplikacija se, idealno, testira na različitim virtualnim i fizičkim uređajima kako bi se utvrdila kompatibilnost s raznim verzijama Android OS-a i veličinama te gustoćama ekrana. Ukoliko su svi uvjeti zadovoljeni, aplikacija se zapakira i plasira na tržište gdje programeru može donijeti zaradu i popularnost te ga tako potaknuti na izradu novih i boljih aplikacija.

Kako je trenutno vodeća platforma za mobilne uređaje, isplati se naučiti programirati za Android operativni sustav. Uz Google-ovu podršku i stalne nadogradnje, Android Studio ima svjetlu budućnost za razvoj aplikacija za uređaje koje pokreće Android, bili to mobilni telefoni, tableti, pametni ručni satovi, televizori, naočale ili koju god nam tehnologiju budućnost donese.

LITERATURA

Knjige:

1. Gargenta, Marko. *Learning Android.* Sebastopol: O'Reilly Media, Inc., 2011.
2. Hellman, Erik. *Android Programming - Pushing the Limits.* Chichester: John Wiley & Sons Ltd, 2014.
3. Zapata, Belén Cruz. *Android Studio Application Development.* Birmingham: Packt Publishing Ltd., 2013.

Internet stranice:

4. Gassner, David. *Android Studio Essential Training.* 19. 02 2015. http://www.lynda.com/Android-Studio-tutorials/Android-Studio-Essential-Training/361465-2.html (pristup 29. 08 2015).
5. Gassner, David. *Developing Android Apps Essential Training*. 04 2015. http://www.lynda.com/Android-tutorials/Developing-Android-Apps-Essential-Training-Revision-Q2-2015/369905-2.html (pristup 29. 08 2015).
6. Gassner, David. *Distributing Android Apps.* 11. 06 2015. http://www.lynda.com/Android-tutorials/Distributing-Android-Apps-Revision-Q2-2015/375926-2.html (pristup 10. 09 2015).

POPIS SLIKA I GRAFIKONA

Slike:

Grafikoni:

PRILOG

Radu se u digitalnom obliku prilaže cjelokupan izvorni kod Android aplikacije Konverter valuta izrađene u sklopu pisanja rada.

SAŽETAK

Android OS je trenutno najpopularniji operacijski sustav za mobilne uređaje, ali je tržište fragmentirano na veliki broj uređaja s raznim hardverskim specifikacijama – od malih mobitela i satova do tableta i pametnih televizora.

Android Studio relativno je nedavno postao službeni alat za razvoj Android aplikacija. Izrađen na temeljima Java razvoja i prilagođen mobilnom razvoju uspješno obavlja zadaću kojoj je namijenjen zahvaljujući intuitivnom sučelju, bogatim editorima i mnoštvu pratećih programa.

Aplikacije za Android složene su u unaprijed definiranu datotečnu strukturu projekta u kojoj se unaprijed zna gdje se koji tip datoteke očekuje. Dizajn sučelja se opisuje pomoću XML-a, dok se ponašanjem aplikacije upravlja Androidovom verzijom Java programskog jezika.

Zbog velikog broja različitih uređaja, aplikacije se testiraju na različitim veličinama ekrana i gustoćama rezolucije. Osim fizičkih uređaja, za testiranje se koriste emulatori koji pokreću virtualne Android uređaje.

Gotove aplikacije prolaze kroz proces optimizacije i skrivanja izvornog koda te postaju APK datoteke s pratećim ključem jedinstvenim autoru aplikacije. Te datoteke uz prateći tekstualni i slikovni opis mogu se distribuirati krajnjim korisnicima izravno ili uz pomoć centraliziranog repozitorija, kao što je Google Play trgovina. Aplikacije mogu biti besplatne ili ih je moguće naplaćivati. Besplatne aplikacije mogu prikazivati oglase ili omogućiti prodaju raznih sadržaja unutar aplikacije i tako autoru donijeti zaradu kroz duže vrijeme.

Ključne riječi: Android, programiranje, aplikacija, Android Studio, mobilni uređaj

SUMMARY

The Android OS is currently the most popular operating system for mobile devices, but the market is fragmented with a large number of devices of varying hardware specifications – ranging from mobile phones and smart watches to tablets and smart TVs.

Android Studio has quite recently become the official tool for Android application development. Built on the foundation of Java development and adapted for mobile development, it successfully accomplishes set goals thanks to its intuitive interface, rich editors and a variety of integrated tools.

Android applications have a predefined folder structure where each file type has its place. The user interface is designed using XML, while application behaviour is determined using a special Android Java dialect.

Owing to a large number of different devices, applications need to be tested on a variety of screen sizes and resolution densities. Apart from phisical devices, applications are tested using emulators which can run virtual Android devices.

Finished applications go through a process of source code optimization and obfuscation, turning them into APK files with dedicated keys unique to their author. These files, with additional graphic and text descriptions, can be distributed to end users directly or by using centralised repositories, such as Google Play Store. Applications can be free or they can be paid. Free applications can integrate advertisment displaying or in-app purchases, therefore enabling long term profit to the author.

Key words: Android, development, application, Android Studio, mobile device