Razvoj aplikacija za operacijski sustav Android

Haberl, Silva

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Silva Haberl

RAZVOJ APLIKACIJA ZA OPERACIJSKI SUSTAV ANDROID

Diplomski rad

Voditelj rada:
Dr.sc. Goran Igaly

Zagreb, rujan,2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Android je započeo jasnom vizijom njenih graditelja 2003. godine u Palo Alto, Kaliforniji. Prema Andy Rubinu, jednom od osnivača Androida cilj Android-a bio je izgraditi ”pametnije mobilne uređaje koji sadrže podatke o lokaciji korisnika i njegova druga svojstva”. Njegova prva pojava u tehnologiji bila je 2005, kada je Google, tehnološka kompanija vrijedna više milijardi dolara, kupila Android. U to vrijeme, rijetko tko je znao za Android i Googlove namjere sve do 2007. godine kada je Google objavio svjetsku prvu otvorenu platformu za mobilne uređaje.

Poglavlje 1

Android

1.1 Android (operacijski sustav)

Android je otvoreni operacijski sustav američke tvrtke Google. Ovaj operacijski sustav je modularan i prilagodljiv, te se osim u pametnim telefonima i tabletima koristi i u drugim naprednim uređajima, primjerice pametnim televizorima, pametnim satovima, čitačima elektroničkih knjiga, automobilima te multimedijskim izvođačima (*playerima*). Android je zasnovan na jezgri Linux 2.6 i napisan u programskom jeziku C/C++. Obzirom na otvorenost izvornog programskog koda, aplikacije putem softvera srednjeg sloja (*middlewarea*) imaju mogućnost komuniciranja i pokretanja drugih aplikacija primjerice za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Iako je C programski jezik primjenjivan za radno okružje (framework), većina aplikacija pisana je u programskom jeziku Java rabeći Android Software Development Kit (SDK). Postoji mogućnost pisanja aplikacija i u programskom jeziku C, no tada se upotrebljava Android Native Code Development Kit (NDK) (Androidov razvijateljski kit u izvornom kodu).

Od 2008. godine do danas objavljeno je više Android inačica. U sljedećoj tablici dane su brojčane oznake inačica, nazivi, te godina i mjesec izdavanja;

Inačica	Naziv	Godina (i mjesec) izdavanja
1.0	ApplePie	rujan 2008.
1.1.	Banana bread	veljača 2009.
1.5	Cupcake	travanj 2009.
1.6	Donut	rujan 2009.
2.0/2.1	Eclair	listopad 2009.
2.2	Froyo	svibanj 2010.
2.3.x	Gingerbread	prosinac 2010.
3.x	Honeycomb	veljača 2011.
4.0.x	Ice Cream Sandwich	listopad 2011.
4.1./4.2/4.3	Jelly Bean	srpanj 2012., 4.2 studeni 2012.
4.4	Kit Kat	rujan 2013.
5.0/5.1	Lollipop	lipanj 2014.
6.0.x	Marshmallow	listopad 2015.
7.0	Nougat	kolovoz 2016.

1.2 Android televizor (Android TV)

Android TV je pametna TV platforma koju je razvio Google. Bazira se na operacijskom sustavu Android. Može biti implementiran u TV i u samostalne digitalne medijske izvođače. Postoji više od 600 optimiziranih aplikacija za Android TV. Trenutno je Android TV baziran na inačici Nougat 7.0. Google je jamac nadogradnji operacijskog sustava za Android TV, koje su ponekad značajne. Android TV je idealna platforma za igre jer jamči dobar hardver čime omogućuje gladak rad zahtjevnih aplikacija te dobre performanse u igrama koje danas još uvijek nisu na razini vrhunskih igara za pametne telefone.

1.3 Android u automobilu (Android Auto)

Android Auto omogućuje jednostavniji način korištenja mobitela u automobilu. Kako bi korisnici mogli biti koncentrirani na vožnju, Android Auto je dizajniran da radi na zaslonu mobitela ili zaslonu u automobilu. Sadrži preglednije ikone i prilagođen je jednostavnom korištenju tokom vožnje glasovnim odabirom i jednostavnim sučeljem. Trenutna glavna aplikacija za Android Auto je Google Maps koja pokazuje put do odredišta.

1.4 Android pametni sat (Android Wear)

Android pametni sat može se koristiti individualno. Mogućnosti korištenja su široke: pregled i primanje poziva, odgovor na poruke direktno sa sata, pregled sastanaka, osobni

fitness podatci, kontakti te slušanje muzike.

1.5 Android u hladnjaku

Hladnjaci postaju s vremenom sve pametniji odnosno mogu pružati informacije za koje je ranije bilo nužno korištenje dodatnih računala. Tako na primjer Android aplikacija u Samsungovom hladnjaku omogućuje pretragu recepata, kreiranje popisa za kupnju, pregled roka trajanja pojedinih namirnica u hladnjaku i slično.

Poglavlje 2

Android igre

2.1 Promjene na tržištu

Igre su bile veliko tržište davno prije pojave iPhonea i Androida. Međutim, pogledi su se promijenili. Današnje igre nisu usmjerene samo na djecu. Ozbiljni poslovni ljudi često se mogu vidjeti kako igraju popularne igre na svojim pametnim telefonima u javnosti, novine objavljuju priče uspješnih mladih programera igrica (*developera*) koji zarađuju bogatstvo na tržištu mobilnih aplikacija. Programeri igrica moraju stalno pratiti i usvajati razne promjene i prilagođavati im se. Pametni telefoni su svugdje. Cijene hardvera konstantno padaju, novim uređajima raste snaga i brzina, pa postaju idealni za igranje igara. Prije Androida i iPhonea, za igranje video igre bilo je potrebno kupiti uređaj za igranje igara ili osobno računalo. Sada ta funkcionalnost dolazi besplatno na pametnom telefonu, tabletu i drugim uređajima. Nema dodatnog troška i uređaj za igranje dostupan je u bilo kojem trenutku. Također nije potrebno kupiti igricu već se može dohvatiti sa tržišta igara, besplatno. Korisnik pametnog telefona stalno je priključen na internet što otvara novi svijet u igranju igara na mobitelima. Korisnik može biti u bilo kojem trenutku u kontaktu sa protivnikom iz svijeta, prijateljem ili strancem. Društvene mreže također imaju utjecaja na mobilne igre. Igre imaju funkcionalnost objavljivanja zadnjih rezultata igre na društvenim mrežama ili obavještavanju prijatelja o zadnjem postignutom rezultatu u igri. Veliko tržište niskih zahtjeva na igre privlačno je mnogim neovisnim programerima i ljudima kojima je to hobi. U slučaju Androida, ta granica je posebno niska: potreban je SDK i možete odmah programirati. Nije potreban mobilni uređaj jer je moguće pokrenuti igru na emulatoru. Android okruženje dopušta inovaciju i isprobavanje novih funkcionalnosti. Google Play omogućuje objavu igara s malo truda i pruža pristup do široke publike koja je spremna isprobati nove ideje.

2.2 Vrste igara

- **Uobičajene igre (Casual Games)**
  Najveći segment na Google Playu su takozvane uobičajene igre. Imaju nekoliko zajedničkih karakteristika. Obično su jako dostupne i jednostavne, pa imaju široku publiku.

- **Slagalice (Puzzle Games)**
  Igre poput vrlo poznatog Tetrisa

- **Akcijske i arkadne igre**
  Ova grupa igara obično koristi puni potencijal Android platforme na trenutnoj generaciji hardvera.

- **Obrana tornja (*Tower-Defense*) igre**
  Podgrupa strateških igara u kojima je cilj igrača obraniti svoj teritorij od neprijatelja.

- **Društvene igre**
  U ovu kategoriju spadaju društvene mrežne igre kao što su *Sim Social* i igre koje su dio društvene mreže Facebook: *FarmVille, Mafia Wars*. Također, obuhvaćaju klasične igre poput šaha, domina i ostale tradicionalne igare koje uključuju više igrača.

2.3 Utjecaj razvoja igara

Pojam igrifikacija

Igrifikacija (*gamification*) je učenje kroz igru ili savladavanje problema iz stvarnog svijeta putem igre. Moderne oblici igrifikacije možemo prepoznati u neigračkom okruženju. Na mnogim mjestima susrećemo se sa avatarima, raznim razinama napredovanja, nagradama, zadacima, značkama. Primjer toga su razni forumi koji sukladno broju postova dodjeljuju različite titule. Tvrtke koje se ozbiljno bave igrifikacijom idu dalje od toga. Razni programski dodaci i aplikacijska sučelja omogućuju integraciju između velikih poslovnih aplikacija i društvenih mreža kao što su Facebook, Twitter i platformi za izradu igrifikacijskih aktivnosti. Cilj igrifikacije je angažirati ljude, zaposlenike, studente, za posao i

učenje ili rješavanje nekog konkretnog problema. Riječ igrifikacija prvi puta je korištena 2002. godine. Tek 2010. se češće počela upotrebljavati. Na temelju nekih istraživanja edukacijskih ustanova zaključak je bio da je ono što učenje čini učinkovitijim igrajući igre je igračeva aktivnost, motivacija, interaktivnost i angažiranost. Posljedica igranja je veća inteligencija što rezultira boljim učenjem. Ponašanje u igri, ponavlja se u drugim životnim područjima.
Jedan od primjera igre u tehnologiji koja je doprinijela razvoju znanosti, zove se Foldit koja je izdana 2008. godine. i sastavni je dio istraživanja sveučilišta u Washingtonu. Foldit je besplatna mrežna igra, tipa slagalice. Cilj je složiti protein koristeći alate igre uz moguću suradnju s drugim igračima ili uz natjecanje. Učeći molekularnu biologiju, igrači pomažu napretku znanosti. Najbolji rezultati i pripadna rješenja analiziraju istraživači koji odlučuju postoji li u stvarnom svijetu u prirodnom obliku takav protein. Predviđanje strukture proteina značajno je u bioinformatici, molekularnoj biologiji i medicini. Poznavanje strukture proteina omogućuje njihovo bolje razumijevanje. To vodi do poboljšanja u liječenju bolesti i rješenju drugih problema kao što su otpad i onečišćenje. U rujnu 2011., igrači Foldita su uspjeli dokučiti kristalnu strukturu Mason-Pfizer virusa, virusa majmuna koji uzrokuje HIV/AIDS, što je objavljeno u časopisu *Nature Structural and Molecular Biology*. Znanstvenici nisu mogli dokučiti tu strukturu više od desetljeća, dok su igrači Foldita to uspjeli u tri tjedna igrajući i slažući proteine. Igra je dostupna na 9 jezika i moguće ju je igrati na operacijskim sustavima Windows, Linux i Mac OS.

Najčešći elementi Android igara su sljedeći:

- **Avatari**. Virtualne reprezentacije igrača.
- **Skupljanje bodova**. Numerička vrijednost koja pokazuje napredak igrača.
- **Timovi**. Igrači surađuju da bi postigli zajednički cilj.
- **Bitke**. Korisnik se natječe s računalom ili nekim drugim igračem u inteligenciji ili nekoj drugoj vještini.
- **Neotključan sadržaj**. Kako korisnici postižu određene rezultate otvaraju im se nove mogućnosti unutar igre.
- **Misije**. Postoje ograničenja i ciljevi koje igrač mora postići u danom vremenu.
- **Virtualna dobra**. Igrači postupno dobivaju dobra koja su ima korisna u novim razinama igre ili ih nagrađuju.
- **Ploča s rezultatima igrača**. Prikazi koji pokazuju na stanje igrača u odnosu na druge igrače i njihov napredak.

Slika 2.1: Prikaz igre Foldit

Poglavlje 3

Android Studio

3.1 Uvod u Android Studio

Android Studio je službeno razvojno okruženje (IDE) za razvoj Android aplikacija koje se temelji na integriranom razvojnom okruženju Intelij IDEA. Nudi mnogobrojne funkcionalnosti kao što su:

- Gradle sustav za izgradnju (*build*)
- Brzi emulator različitih funkcionalnosti
- Jedinstveno razvojno okruženje za sve Android uređaje
- Pokretanje aplikacija bez stvaranja novog APK-a
- Integracija sustava za verzioniranje koda (GitHub)
- Alati za testiranje i knjižnice (frameworks)
- Podrška za C++ i NDK
  i druge

Preuzimanje je besplatno i omogućeno je Apache 2.0 licencom. Za razvoj aplikacija Android Studio zahtijeva instalaciju Java Development Kit-a (JDK). Android Studio je podržan za Linux, Windows i Mac OS X operacijske sustave. Android Studio je dostupan za preuzimanje na adresi https://developer.android.com/sdk/index.html.

3.2 Instalacija Android Studija

Nakon pruzimanja paketa za operacijski sustav Linux potrebno je paket raspakirati na željenu lokaciju.
1. Otvoriti *.zip* dokument koji smo preuzeli na adekvatno mjesto za naše aplikacije, kao što je */usr/local/* za korisnički profil, ili */opt/* za shared users.
2. Da bismo pokrenuli Android Studio, otvorimo namjenski program (*terminal*), navigiramo do direktorija *android-studio/bin* i izvršimo *studio.sh*
3. Odaberemo hoćemo li otvoriti postojeći projekt ili ne, kliknemo **OK**.
4. Android Studio Wizard vodi nas kroz ostatak instalacije, koji obuhvaća preuzimanje Android SDK komponenti koje su potrebne za razvoj.
Pokretanje na 64-bitnoj Fedori (operacijski sustav Linux) zahtijeva instalaciju nekih 32-bitnih biblioteka (*libraries*). Sljedećom naredbom napravimo instalaciju:
sudo yum install zlib.i686 ncurses-libs.i686 bzip2-libs.i686

3.3 Struktura projekta

Strukturu projekta možemo okvirno podijeliti u tri dijela:
Moduli Android aplikacije

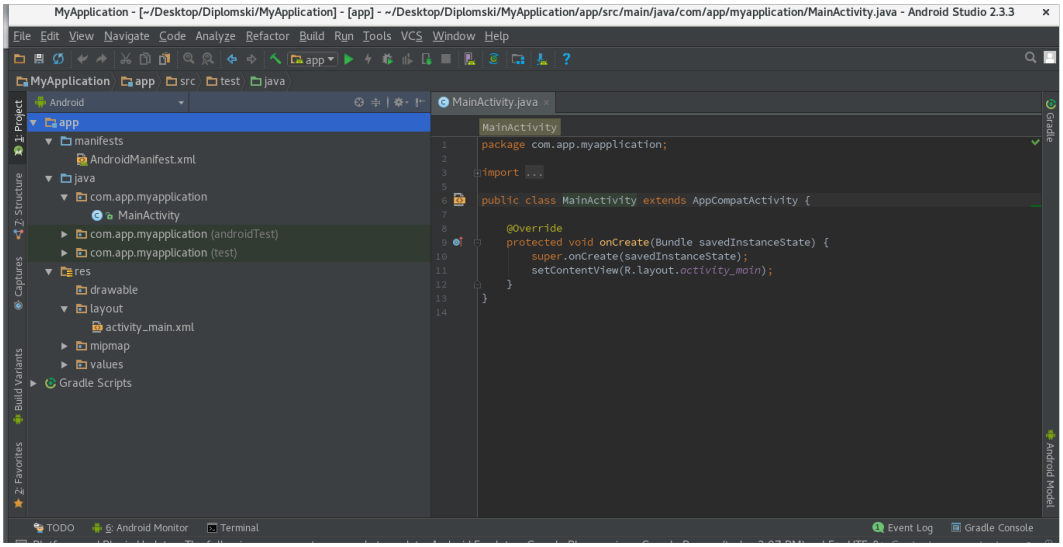
Ovaj dio projekta pruža kontejner za kod aplikacije, datoteke resursa i postavke aplikacije po nivoima kao što je module-nivo, dokument za kompiliranje i Android Manifest dokument. Kada kreiramo novi projekt, prema početnim postavkama naziv modula je ”app”.

Moduli knjižnica (libraries)

Ovaj dio projekta pruža kontejner za kod koji se može više puta koristiti. Za razliku od modula aplikacije pri kompiliranju generira arhivu koda umjesto APK pa ne može biti instaliran na uređaj.

Moduli Google App Engine

Pruža kontejner za Google Cloud (backend) kod. Google Cloud modul za razvoj unutar Android Studija omogućuje savladavanje koda aplikacije i backend koda u istom projektu. S lijeve strane sučelja Android Studija nalazi se preglednik projekta. Pomoću njega moguće je vidjeti otvorene projekte i sve datoteke koje tim projektima pripadaju. Također moguće je vršiti analizu i refaktoriranje koda, kao i uspoređivanje datoteka i otvaranje u posebnom prozoru preglednika datoteka.



Slika 3.1: Struktura projekta

Konfiguracijski dokumenti projekta

Project (Slika 3.1)

- **ime-modula/**
- **AndroidManifest.xml/**: osnovna 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.
- **java/**: sadrži Java klase organizirane po paketima. Stvaranjem novog projekta stvara se i njegova glavna aktivnost koja se nalazi u paketu s imenom aplikacije. Također sadrži kod za lokalne testove koji se pokreću na JVM (Java Virtual Machine).
- **res/**: sadrži resurse projekta poput 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.

  - **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 bismo za njih bolje prilagodili sučelje.

- **build.gradle(module)** Konfiguracije za kompajliranje pojedinih modula.
- **build.gradle(project)** Konfiguracije za kompajliranje koje se primjenjuju na sve module.

Navedeni se direktoriji stvaraju automatski s novim projektom, iako nije nužno za svaki projekt da sadrži sve navedene dokumente.

3.4 Sustav za verzioniranje koda - Git (osnove git-a)

Android Studio podržava različite sustave za verzioniranje koda uključujući Git, GitHub, CVS, Mercurial, Subversion i Google Cloud repozitorij koda. U ovom radu koristit ćemo GitHub.

Verzioniranje koda

Sve na GitHubu je pohranjeno u Git, trenutačno najbolji sustav za verzioniranje koda. Verzioniranje koda nam omogućava da isprobavamo i radimo pogreške u kodu neovisno o krajnjem rezultatu projekta.

Očuvanje koda na jednom mjestu

Na GitHubu imamo mjesto za trajno čuvanje koda starih projekata, te nam je omogućen rad s više udaljenih računala na istom projektu.

Suradnja

U toku razvoja koda na git-u, možemo pozvati suradnike da se priključe našem projektu. Jednom kad napravimo repozitorij za dani projekt, dobili smo URL za svaki dokument u projektu. Svi git repozitoriji se temelje na spremanju verzije (*commit*) - trenutno zabilježenom stanje projekta u vremenu.

Prvi git repozitorij

Svaki direktorij može postati git repozitorij. Ne mora uopće postojati udaljeni server i neki

centralni repozitorij kojeg koriste ostali koji rade na projektu. Za stvaranje novog direktorija čije je ime *projekt* izvršavamo sljedeće naredbe u namjenskom programu (terminalu):

- *mkdir projekt*
- *cd projekt*
- *git init*

Prva naredba stvara direktorij *projekt*, druga nas pozicionira u njega. Treća naredba inicijalizira .git direktorij u *projekt*.
Prije ili kasnije dogodit će se situacija da u direktoriju s repozitorijem imamo datoteke koje ne želimo spremiti u povijest projekta. To su, na primjer konfiguracijske datoteke za različite editore ili datoteke koje nastaju kompajliranjem. U tom slučaju treba dati do znanja gitu da takve datoteke ne treba nikad snimati. Otvorimo datoteku naziva **.gitignore** i unesemo sve što ne treba biti dio povijesti projekta.

Lokalni repozitorij

Lokalni repozitorij je direktorij stvoren lokalno na našem računalu.

Udaljeni repozitorij

Udaljeni repozitorij je direktorij s kojeg preuzimamo kod i njegove promjene i na kojeg šaljemo vlastite promjene. Za nas je njegov naziv *origin* i ima svoju adresu.

Kloniranje repozitorija

Kloniranje je postupak kojim kopiramo cijeli repozitorij s udaljene lokacije na naše lokalno računalo. S tako kloniranim repozitorijem možemo nastaviti rad kao s repozitorijem kojeg smo inicirali lokalno. Za razliku od kopije direktorija, novi (lokalni) repozitorij ostaje ”svjestan” da je on kopija nekog udaljenog repozitorija. Klonirani repozitorij čuva informaciju o repozitoriju iz kojeg je kloniran. Ta informacija će mu kasnije omogućiti da na udaljeni repozitorij šalje svoje izmjene i od njega preuzima izmjene drugih suradnika.

Preuzimanje izmjena s udaljenog repozitorija

Za preuzimanje izmjena s udaljenog repozitorija potrebno je u namjenski program (terminal) na mjestu gdje se nalazi lokalno repozitorij, izvršiti naredbe:

- *git fetch*
- *git merge origin/master*

Kratica za prethodne dvije naredbe je *git pull*.

Aktivno mijenjanje udaljenog repozitorija

Prebacivanje lokalnih izmjena na udaljeni repozitorij ovisi imamo li za to ovlasti ili ne. Udaljeni repozitorij mora biti tako konfiguriran da bismo mogli raditi *git push*. Ukoliko nemamo ovlasti sve što možemo napraviti je obavijestiti njegovog vlasnika da pogleda naše izmjene i da ih preuzme na sebe, ako mu odgovaraju. Taj proces se zove *pull request*. Ukoliko imamo ovlasti onda je ono što trebamo napraviti *git push origin master*.

3.5 Gradle sustav za izgradnju (*build*)

Kompajliranje je proces prevođenja programskog koda u objektni kod. Izgradnja (*build*) projekta, uz kompajliranje i linkanje, uključuje i neke dodatne radnje, na primjer izradu instalacijske procedure (*installera*). U slučaju razvoja android aplikacija, sustav za izgradnju 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 grupirao sve te datoteke u jednu komprimiranu .apk datoteku s kojom Android sustav zna raditi. Android Studio koristi Gradle kao osnovu za izgradnju (build). Gradle je razvijen je promatrajući druge sustave za izgradnju, te integriranjem njihovih najboljih karakteristika. Gradle je baziran na JVM (Java Virtual Machine) sustavima za izgradnju, što znači da se mogu pisati skripte za izgradnju u programskom jeziku Java.

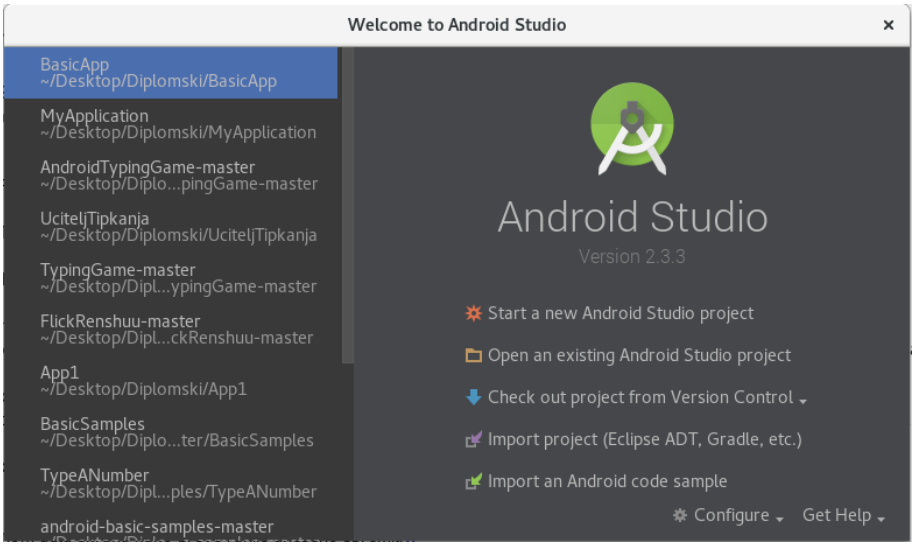
Poglavlje 4

Razvoj minimalne aplikacije

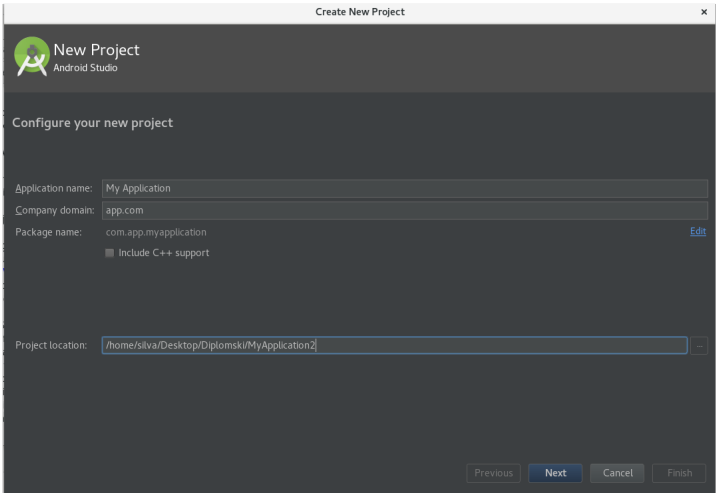
Kreiranje novog projekta

1. U dijalogu *Welcome to Android Studio* kliknemo na **Start a new Android Studio project**, ako imamo već otvoren projekt, kliknemo **File**, zatim **NewProject**.

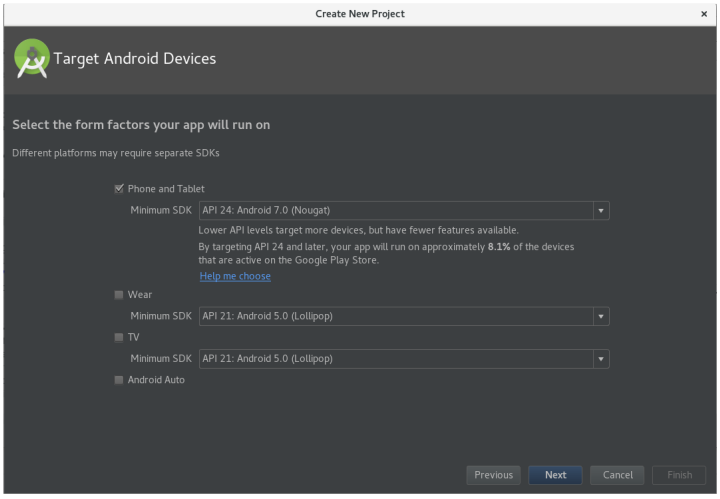
2. U Create New Project popunimo osnovnim informacijama naziv aplikacije i ime kompanije, te mjesto gdje želimo pospremiti projekt na računalu.



Slika 4.1: Kreiranje novog projekta - 1. korak



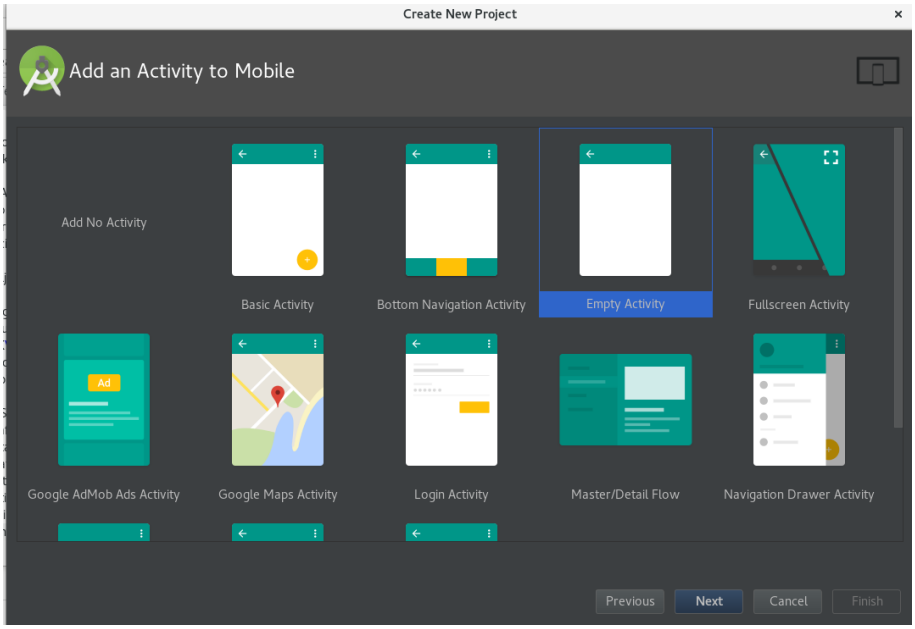
Slika 4.2: Kreiranje novog projekta - 2. korak



Slika 4.3: Kreiranje novog projekta - 3. korak

3. Kliknemo **Next**. Ostavimo već odabranu formu na kojoj želimo pokretati našu aplikaciju, a to je: Phone and Tablet, API24, Android 7.0 (Nougat). Kliknemo **Next**.

4. Odaberemo **Empty Activity** i kliknemo **Next**.



Slika 4.4: Kreiranje novog projekta - 4. korak

5. Ostavimo već odabrane postavke i kliknemo **Finish**.

Dokumenata i mapa ima puno, no neke od njih nećemo trebati koristiti.

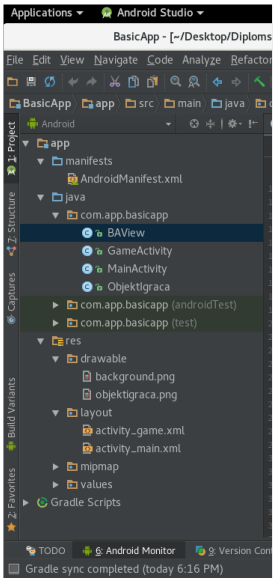
Pokretanje aplikacije

U gornjoj alatnoj traci izaberemo gumb **Run app**. Aplikacija će se otvoriti na emulatoru. Android emulator je besplatan i jednostavan za testiranje Android aplikacije. Da bismo zaustavili testiranje aplikacije u gornjoj alatnoj traci kliknemo **stop**.

Poglavlje 5

Razvoj osnovnog dijela aplikacije (igre)

U ovom poglavlju ćemo objasniti razvoj osnovnog dijela aplikacije (igre) koju ćemo nazvati BasicApp.



Slika 5.1: Struktura projekta igre BasicApp

Slika 5.2: Početni zaslon

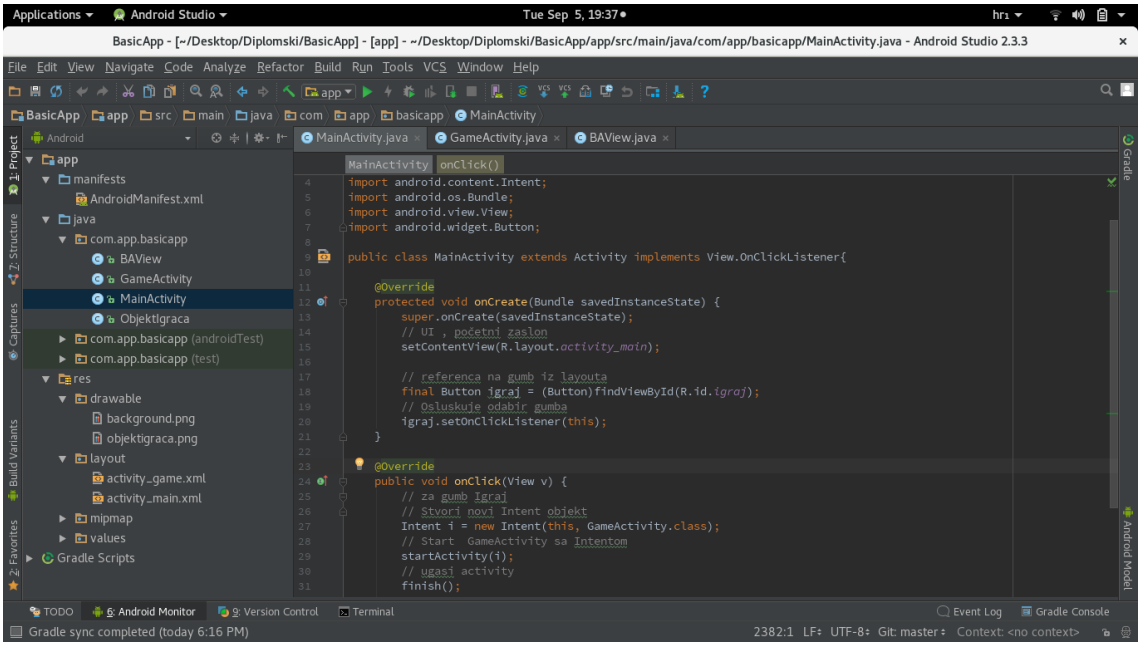
5.1 Struktura projekta igre u Android Studiu

Kreiranje početnog zaslona igre

Najjednostavniji dio igre je početni zaslon. Potrebna nam je slika koja obilježava igru, najbolji rezultat i gumb za start igre. Dodamo sliku *background.jpg* u projektnu mapu *drawable*, najbolji rezultat ćemo ispisivati u TextView koji sadrži svoj ID koji služi za povezivanje s funkcionalnošću iz Java koda. Na kraju dodamo *Button* kao gumb za pokretanje igre.
Kada kreiramo projekt, Android Studio otvara dva dokumenta koja možemo mijenjati. To su: *MainActivity.java* i *activity_main.xml*.

Kodiranje funkcionalnosti

Nakon izrade izgleda početnog zaslona moramo napisati funkcionalnost u Java kodu kako bi gumb za početak igre mogao raditi. U *MainActivity.java* ulaz u igru je u *OnCreate* metodi. Unutar nje se nalazi metoda *setContentView()* koja prikazuje izgled iz activity_main.xml na zaslonu.*findViewById* dohvaća referencu na gumb sa našeg izgleda *activity_main.xml* početnog zaslona. *SetOnClickListener()* osluškuje kada će netko kliknuti gumb. Za osluškivanje kada će gumb biti kliknut, moramo implementirati *onClickListener* sučelje i implementirati *onClick()* metodu. Za prijelaz iz jedne aktivnosti u drugu u igri (ovdje iz MainActivity u GameActivity) stvaramo Intent objekt. Prelazimo u GameActivity koju ćemo nadalje opisati. Kreiranjem nove aktivnosti u Android Studiu dobivamo dva dokumenta *GameActivity.java* i *activity_main.xml*. Dodavanje nove aktivnosti moramo



Slika 5.3: MainActivity.java

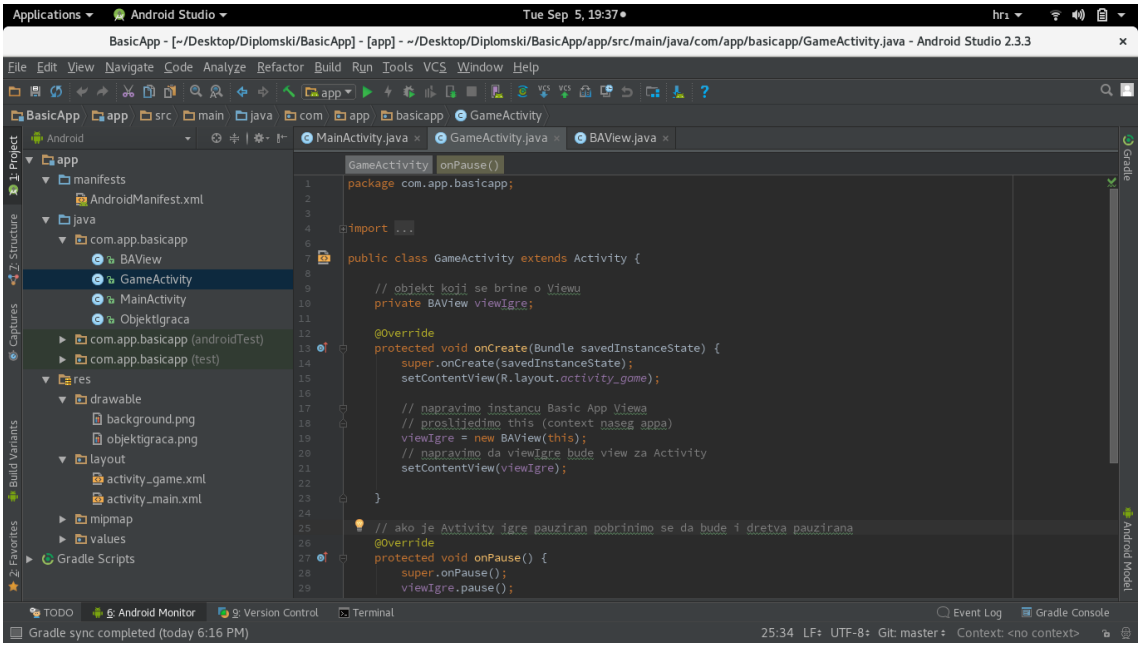
zabilježiti u dokumentu *AndroidManifest.xml*.

Kodiranje i petlja igre (game loop)

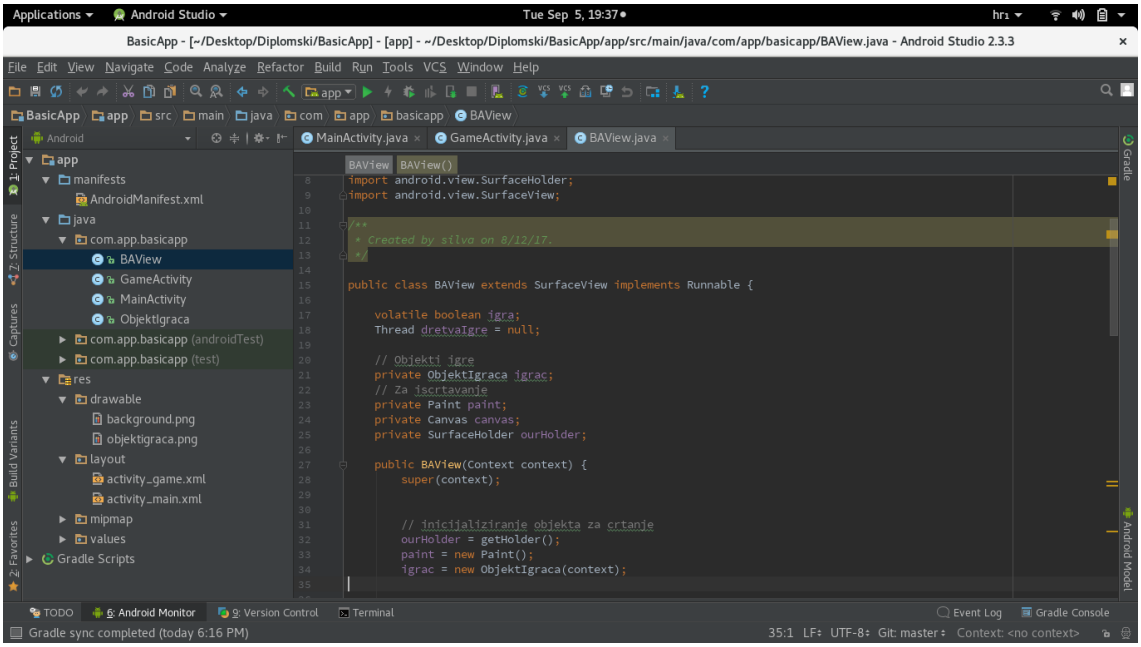
Stvorimo novu klasu *BAView* (prema imenu BasicApp) koja nam omogućuje izgled ekrana (View) i koristi se kao instanca objekta BAView. Takav ekran dinamički je iscrtan i omogućuje brzi unos igračevih naredbi. Koristimo ga u GameActivity.java. BAView u metodi run koristi metode *obnovi* (podatke igrača), *crtaj* (slike), *kontrole* (zaustavljanje rada dretve, broj obnavljanja slike u minuti). Petlja igre (game loop), određena varijablom *igraj*, istovremeno upravlja sa ponašanjem igrača i ostalim sistemskim zahtjevima, jer smo implementira *Runnable* interface kreirajući novu dretvu. Klasa *android.view.SurfaceView* omogućuje iscrtavanje slika, teksta, linija ali i ponašanje igrača. *Context* koji je predan kao parametar konstruktoru nove klase BAView je trenutno stanje igre u GameActivity klasi.

Objekt u igri

Kreiramo klasu *ObjektIgraca* kao model koji je odvojen od ostatka igre. Igrač ima svoja svojstva: sliku, x i y koordinate i brzinu. Osim konstruktora i metoda za dohvaćanje vrijednosti svojstava igrača (metode *getteri*) potrebna nam je metoda *obnovi* koja povećava x koordinatu objekta u igri. Vizualno, slika objekta igrača pomiče se u desno na ekranu.



Slika 5.4: GameActivity.java



Slika 5.5: BAView.java

Slika 5.6: Igrač se kreće u desnu stranu

Iscrtavanje objekta na platnu

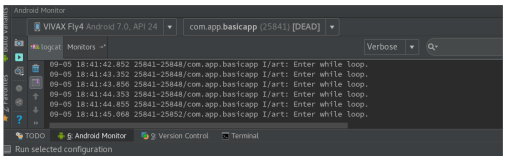
Za iscrtavanje objekta koristimo klasu *Canvas*. Napravimo virtualno platno i smjestimo ga u GameActivity klasu. Možemo upravljati pikselima objekta kroz objekt *Paint*.
Koristimo *SurfaceHolder* klasu za manipulaciju platna, kada da ono bude vidljivo. Za inicijalizaciju platna najbolje mjesto je klasa BAView nakon čega možemo instancirati igrača i pozvati metodu *obnovi*.

Pokretanje igre (deploy)

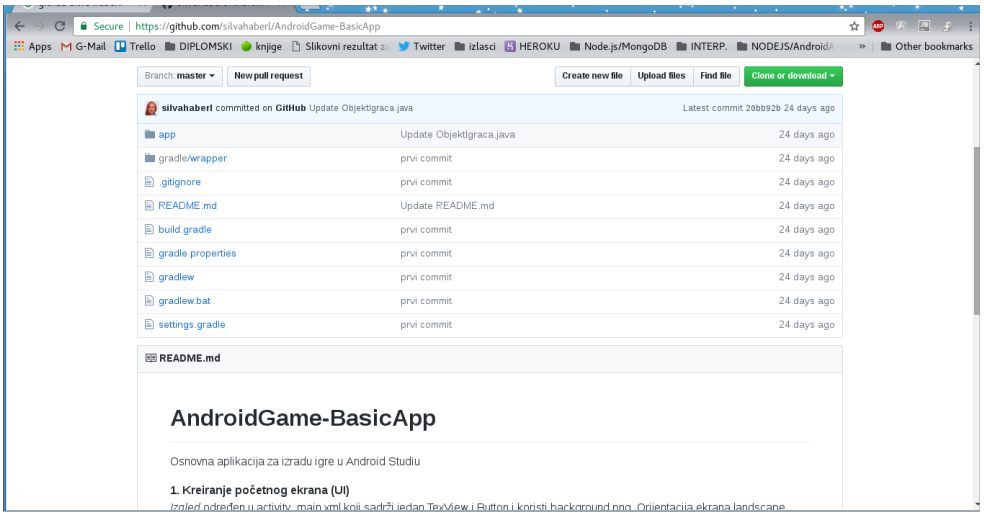
Igru ćemo pokrenuti na Vivax Android telefonu. Postupak kako omogućiti povezivanje sa uređajem:
1. Kliknemo na **Settings** u izborniku.
2. Odaberemo **Developer** opciju.
3. Kliknemo na **USB Debugging**
4. Spojimo Android uređaj na USB ulaz računala.
5. Na dnu Android Studija vidjet ćemo uređaj koji je vidljiv računalu za spajanje.
6. Kliknemo **Play** u alatnoj traci i **OK** da pokrenemo aplikaciju na odabranom uređaju.

Repozitorij koda igre na GitHubu

Pri razvoju aplikacije koristimo repozitorij koda na GitHubu. Od dokumenata bitnijih za razumijevanje programa važni su: *app* u kojem se nalazi izvorni kod programa i *Readme.md* dokument za čitanje objašnjenja funkcionalnosti i strukture igre. Poveznica na repozitorij BasicApp projekta: **https://github.com/silvahaberl/AndroidGame-BasicApp**



Slika 5.7: Pokretanje na Vivax Android uređaju, API 24



Slika 5.8: Repozitorij koda BasicApp na GitHubu

Poglavlje 6

Testiranje aplikacije

Strategija testiranja

Strategija testiranja uključuje:

- odabir ciljanih uređaja (pametni telefon, tablet, različite vrste ekrana, različiti procesori (CPU), memorije)
- testiranje korisničkog iskustva (*usability testing*) - navigacija između ekrana, povratna informacija kod interakcije s aplikacijom
- funkcijsko testiranje (*functional testing*) predstavlja testiranje ispravnosti pojedinih funkcionalnosti aplikacije i odgovara li aplikacija korisničkim zahtjevima
- testiranje kompatibilnosti (*compatibility testing*) podrazumijeva validaciju aplikacije na različitim uređajima s različitim operacijskim sustavima, veličinama ekrana i različitim rezolucijama. Također provjerava kako aplikacija radi s ostalim aplikacijama.
- operativno testiranje (*operational testing*) podrazumijeva testiranje aplikacije u slučaju da se baterija isprazni, dostupnost aplikacije u slučaju da korisnik primi poziv, poruku.

Uređaji

Nakon definirane strategije testiranja mobilne aplikacije potrebno je odrediti način testiranja, odnosno uređaje na kojima će se provoditi testiranje.

Fizički uređaji

Testiranje aplikacije na ciljanoj skupini uređaja je najpouzdanije i najtočnije, posebno za testiranje korisničkog iskustva (*user experience*).

Emulatori

Emulator predstavlja softver koji se pokreće na računalu i oponaša fizički uređaj. Prije samog pokretanja emulatora potrebno je instalirati Android SDK (Software Development Kit) i definirati AVD (Android Virtual Device), kojim se definira hardver kao što je RAM, ima li uređaj zaslon osjetljiv na dodir, fizičku tipkovnicu, kameru. Moguće je kreirati više AVD-ova za potrebe testiranja na više uređaja.

Cloud servisi za testiranje

Cloud servisi za testiranje predstavljaju vanjske servise koji nude uslugu najma uređaja, čime se testiranje mobilnih aplikacija znatno unaprijedilo. Uređajima na kojima će se raditi testiranje može se pristupiti na jednostavan način kroz web preglednik.

Poglavlje 7

Pakiranje aplikacije za objavu (Google Play)

7.1 Trgovina Google Play

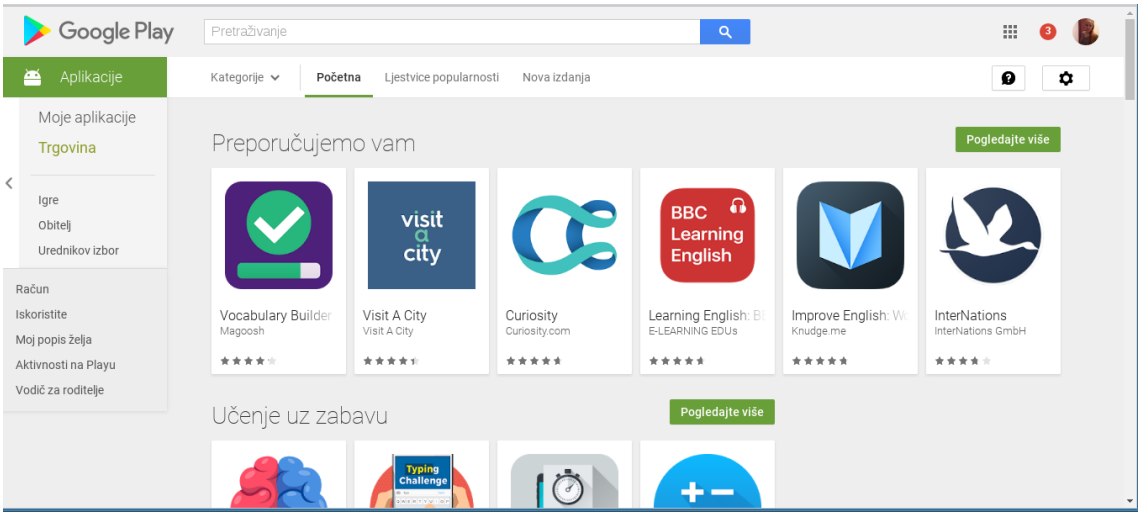
Trgovina Google Play (prijašnji Android Market) je platforma za distribuciju aplikacija za Android osnovana od kompanije Google 2008. godine. Osim aplikacija, s trgovine se mogu skinuti i drugi sadržaji, kao što su muzika novine, knjige i TV program. Besplatne aplikacije na Google Playu su dostupne cijelom svijetu, dok se aplikacije koje se plaćaju mogu skinuti samo u nekim zemljama. Registracija na trgovini se plaća 25 dolara jednokratno. U veljači 2017. Google Play broji preko 2.7 milijuna Android aplikacija.

7.2 Priprema i objava aplikacije

Objavljivanje aplikacije je proces koji omogućuje da izrađena aplikacija bude dostupna krajnjim korisnicima. Dvije su glavne faze u objavljivanju aplikacije:

Priprema za puštanje u produkciju: u ovoj fazi se aplikacija izgrađuje (*builda*) u konačnoj verziji.

Puštanje u produkciju: druga faza podrazumijeva da se aplikacija objavi javno i distribuira korisnicima, bilo u besplatnom obliku ili kroz naplatu.
Ova faza sastoji se od pet radnji koje su podijeljene na sljedeći način:



Slika 7.1: Google Play trgovina

Prikupljanje materijala i resursa

Minimalno što moramo u ovoj fazi napraviti je pribavljanje kriptografskog ključa za digitalni potpis aplikacije te izrada ikone za aplikaciju.

Kriptografski ključ - Android sustavi zahtijevaju da je svaka aplikacija, koja se instalira na korisnikov uređaj, digitalno potpisana. To znači da u sebi sadrži javni certifikat za koji programer ima privatni ključ. Taj certifikat se koristi kako bi se programera moglo prepoznati kao autora aplikacije te se na taj način uspostavlja odnos zasnovan na povjerenju među instaliranim aplikacijama.
Ikona - Predstavlja našu aplikaciju korisniku. To je važan vizualni element koji krajnji korisnik susreće svakim pokretanjem naše aplikacije. Za objavljivanje aplikacije na Google Play potrebno je izraditi ikonu u visokoj rezoluciji.
Licencirani ugovor s krajnjim korisnikom (End-user License Agreement (EULA)) - Treba razmotriti i sastavljanje neke vrste ugovora koju korisnik prihvaća prilikom instalacije.
Ostalo - radnje vezane za marketing te reklamiranje. Pri objavljivanju aplikacije na Google Play trebamo imati neki kraći tekst koji opisuje aplikaciju i predstavlja je korisniku.

U fazi konfiguriranja aplikacije potrebno je napraviti sljedeće:

- Odabrati prikladno ime za aplikaciju

- Isključiti logove i debugiranje
- Očistiti projekt od suvišnih datoteka
- Pregledati manifest i Gradle postavke
- Provjeriti kompatibilnost
- Provjera URL adresa
- Implementirati Licencu

Izgradnja aplikacije (build)

Android Studio i Gradle sustav za izgradnju (build) nam omogućuju da izgradimo aplikaciju ako je spremna za objavu u obliku *.apk* (Android Package) datoteke. Ovaj proces pretpostavlja da smo već pripremili certifikat i privatni ključ koji su potrebni. Ako nemamo pripremljen privatni ključ, Android Studio omogućuje njegovu izradu.

Priprema eksternih servisa i resursa

Ako koristimo neki server za uspješan rad naše aplikacije moramo provjeriti da je osiguran od hakerskih napada.

Testiranje

Izgrađenu verziju aplikacije (bulid) koju pripremamo za puštanje u produkciju treba detaljno testirati pod uvjetima koji su realni u svijetu ciljanih korisnika.

S obzirom da želimo aplikaciju distribuirati najširem mogućem tržištu, u kontroliranim uvjetima (biramo hoće li biti besplatna ili ne, u kojim zemljama će biti dostupna, te na kojim Android uređajima, kao što su Android pametni sat, Android Auto i slično, se može pokrenuti) , potrebno ju je staviti na Google Play. Objava na Google Play-u sastoji se od tri osnovna koraka:

- priprema promotivnih materijala (screenshotowa, grafika, videa, promotivnih tekstova)
- konfiguracija opcija - u kojim zemljama želimo distribuirati aplikaciju, na kojim jezicima, po kojoj cijeni, je li aplikacija namjenjena za sve uzraste i dobne skupine korisnika ili samo neke i slično.

- objava aplikacije - nekoliko minuta nakon postavljanja svega na Google Play razvojnu konzolu (*https://market.android.com/publish/Home*, logiramo se pomoću Googleovog korisničkog računa) i pritiska tipke *Publish*, aplikacija će biti dostupna za preuzimanje (*download*).

7.3 Kreiranje privatnog ključa i APK

1. Kreiranje privatnog ključa s kojim ćemo potpisati aplikaciju.

Bez privatnog ključa aplikaciju ne možemo staviti na Google Play Store. Ovaj ključ koristimo kroz cijeli životni vijek aplikacije, tj. prilikom dodavanja nadogradnji (updatea).
Kako bismo kreirali privatni ključ koristit ćemo keystore unutar naredbenog retka (command-line utility). Privatni ključ vrijedi 10 000 dana, što je dovoljno da pokrije životni vijek aplikacije.

keytool -genkey -v -keystore BasicApp.keystore -alias BasicApp -keyalg RSA -keysize 2048 -validity 10000

Ova naredba će stvoriti BasicApp.keystore datoteku koju možemo smjestiti bilo gdje na računalu i koju moramo čuvati jer će nam trebati prilikom prvog potpisivanja aplikacije, ali i prilikom svake nadogradnje. Svaka verzija aplikacije mora biti potpisana sa istim privatnim ključem.

2. Kreiranje nepotpisanog APK

Pomoću naredbe build izradimo nepotpisani APK:

cordova build –release android

Novi APK možemo pronaći u:

platforms/android/build/outputs/apk/android-release-unsigned.apk .

Kako bismo si olakšali idući korak APK možemo odmah preimenovati u *BasicApp.apk*.

3. Potpis APK koristeći privatni ključ

Potpisivanje APK-a obavlja se pomoću alata jarsigner. Kopiramo privatni ključ BasicApp.keystore (moramo znati lozinku privatnog ključa) i APK basicapp.apk u istu mapu i pokrenemo naredbom:

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore BasicApp.keystore BasicApp.apk BasicApp

4. Optimizacija APK-a

Ovo je zadnj korak u pripremi našeg APK-a za objavu. Alat zipalign će optimizirati naš APK tako da bolje iskorištava resurse mobilnog uređaja.

Pokrenemo naredbu:

zipalign -v 4 BasicApp.apk BasicApp.apk

7.4 Postavljanje APK-a na Google Play Store

1. Registracija

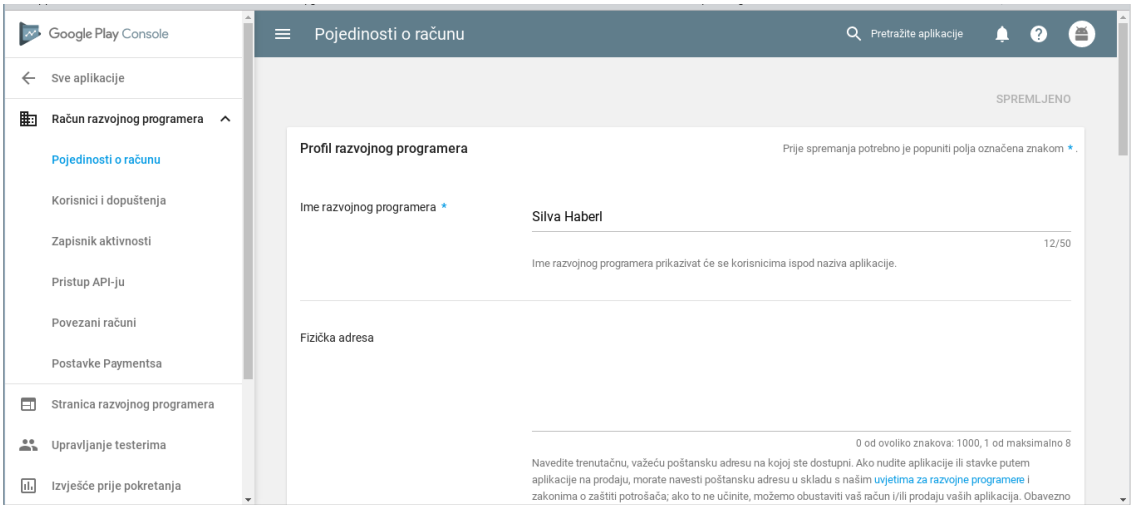
Posjetimo Google Play Console i registriramo se.

2. Prijava

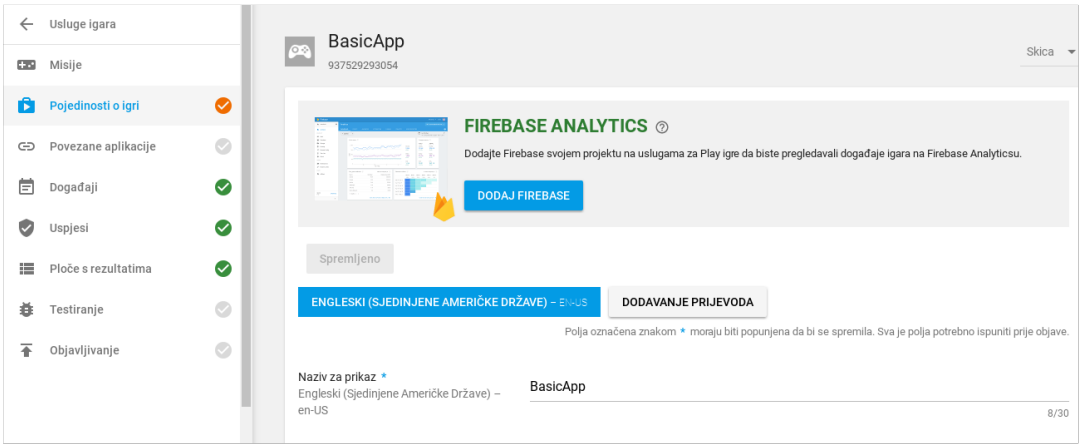
Prijavimo se u *Google Developer Console* i dodamo aplikaciju tako da kliknemo *Add new application*. Navedemo jezik i naziv aplikacije pa odaberemo *Upload APK*.

3. Dodamo APK

APK -možemo odmah dodati u produkciju, ali isto tako možemo pokrenuti i alfa tj. beta test ako želimo aplikaciju testirati među kontroliranom grupom osoba prije objave produkcijske verzije koja bi bila dostupna svima. Kada pokrenemo alfa, tj. beta test korisnike možemo pozvati unosom njihovih adresa elektroničke pošte ili im poslati URL na alfa, tj. beta verziju aplikacije.



Slika 7.2: Registracija



Slika 7.3: BasicApp na Google Play (neobjavljen)

4. Dodamo osnovne informacije

Odaberemo *Store Listing* kako bi dodali osnovne informacije o aplikaciji.

5. Unesemo grafičke elemente.

Unesemo slike naše aplikacije, također promotivni video aplikacije ako ga imamo. **6. Unos detalja**

Unesemo ostale detalje: tip aplikacije, kategoriju, naše podatke (e-mail adresu i slično), itd. Kliknemo na *Save draft* u gornjem desnom uglu ekrana kako bi sve ranije uneseno spremili.

7. Upitnik

Ispunimo kratak upitnik kako bi naša aplikacija dobila pripadajuću ocjenu (*Content Rating)* te navedemo koje države želimo pokriti svojom aplikacijom. Možemo dodati ocjenu (*Pricing and Distribution*). Svaka stavka iz tog izbornika mora biti zelena.

8. Objavimo aplikaciju

Kada smo ispunili sve navedeno, *Ready to publish*, spremni smo za objavu aplikacije *Publish app*.

7.5 Nadogradnja (*update*)

Svaki sljedeći put kada želimo nadograditi aplikaciju moramo proći gore navedene korake od 1 do 5, osim koraka 2 jer keystore već imamo. Prije nadograđivanja u *config.xml* navedemo novi broj inačice (*version number*) koji mora biti veći od onoga koji je u tom trenutku nalazi na Google Play Store.

Sažetak

U ovom radu opisan je početak razvoja Androida kao operacijskog sustava i njegovo korištenje u tehnologiji od mobitela, preko kućanskih aparata do automobila. Razvoj Androida kao operacijskog sustava omogućilo je razvoj igara. Igre služe osim za zabavu i u edukacijske svrhe, za razvoj znanosti, medicine te napretku u poslovnom svijetu. U skladu s time opisan je pojam ”igrifikacije”. Opisano je razvojno okruženje za razvoj Android aplikacija. Prikazan je postupak razvitka jednostavne igre, te postupak njenog objavljivanja na Google Play trgovinu.

Summary

This thesis describes the beginning of the development of Android as an operating system and its use in cellular technology, through home appliances to cars. The development of Android as an operating system enabled the development of games. Games serve other than entertainment and educational purposes, for the development of science, medicine and business development. Accordingly, the term ”gamification” is described. A development environment for developing Android applications is described. The procedure for developing a simple game is presented and the procedure for its release on the Google Play Store.

Životopis

Silva Haberl rođena 29.ožujka 1989. u Zagrebu, Hrvatska. Odrasta uz oca, majku i brata. Svoje obrazovanje započinje u osnovnoj školi Malešnica, a nastavlja u X.gimnaziji "Ivan Supek", smjer matematika, koju završava 2007. godine. Nakon završetka srednjoškolskog obrazovanja upisuje se na preddiplomski studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta Sveučilišta u Zagrebu. Godine 2013. upisuje diplomski studij Računarstva i matematike na istom fakultetu. Tijekom visokoškolskog obrazovanja pokazuje interes za programiranje web i mobilnih aplikacija. Aktivno sudjeluje na radionicama za mlade početnike u programiranju u tehnologijama: Android, Ruby on Rails, iOS. Kraće razdoblje radi kao web programer u programskom jeziku Python. Posebnu pažnju usmjerava na izradu mobilnih aplikacija u Androidu.

Bibliografija

[1] Belen Cruz Zapata, *Android Studio Application Development,* 2013.

[2] Rick Rogers, *Learning Android Game Programming: A Hands-On Guide to Building Your First Android Game*, 2012.

[3] Jeff Friesen, *Learning Java for Android Development*, 2014.

[4] Roy Sandberg, Mark Rollins, *The Business of Android Apps Development*, 2013.

[5] Peter Bell, Brent Beer, *Introducing GitHub*, 2014.