SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Tim AIR1909**

DISCOUNT LOCATOR - KOTLIN

**PROJEKT IZ KOLEGIJA „Analiza i razvoj programa“**

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Alen Peršić**

**Kristijan Maoduš**

**Patrik Fumić**

**Valentino Poljak**

**Projekt pisanja tutoriala:**

**- GitHub:** https://github.com/AIR-FOI-HR/AIR1909

**- ZenHub**: https://app.zenhub.com/workspaces/air1909-5dbc2c4ee93ebe000191a845/board?repos=218985829

**Kotlin Discount Locator projekt (za tutorial)**

**- GitHub:** https://github.com/AIR-FOI-HR/AIR1909-DL

**- ZenHub**: https://app.zenhub.com/workspaces/air1909-dl-5dbc2ca6de0163000119b590/board?repos=218985934

DISCOUNT LOCATOR - KOTLIN

**PROJEKT IZ KOLEGIJA „Analiza i razvoj programa“**

Mentor/Mentorica:

Doc. dr. sc. Zlatko Stapić

Varaždin, studeni 2019.

**Sadržaj**

[1. Uvod 1](#_Toc25184375)

[2. Popis funkcionalnosti 2](#_Toc25184376)

[3. Baza podataka 9](#_Toc25184377)

[4. Razvoj aplikacije 10](#_Toc25184378)

[4.1. Instalacija razvojnog okruženja 10](#_Toc25184379)

[4.2. Postavljanje Android SDK-a 13](#_Toc25184380)

[4.3. Preporučene postavke Android Studio razvojnog okruženja 18](#_Toc25184381)

[4.4. Kreiranje novog projekta 21](#_Toc25184382)

[4.5. Kreiranje entiteta baze podataka 24](#_Toc25184383)

[Popis literature 31](#_Toc25184384)

[Popis slika 32](#_Toc25184385)

[Popis tablica 33](#_Toc25184386)

[Prilozi (1, 2, …) 34](#_Toc25184387)

1. Uvod

Tema ovog projekta je Android aplikacija za lociranje popusta, pod nazivom „*Discount Locator*“. Umjesto klasičnog rada u Java programskom jeziku, u ovome projektu koristili smo Kotlin programski jezik i u nastavku će biti detaljno opisan postupak kreiranja potpunog Android projekta od same instalacije razvojnog okruženja do pojedinačnih implementacija raznih featurea koji će kasnije sačinjavati gotov projekt. Cilj ovog rada je da student koji će pratiti isti bude u stanju razviti cijelu Android aplikaciju u Kotlin programskom jeziku uz pomoć Android Studio razvojnog okruženja.

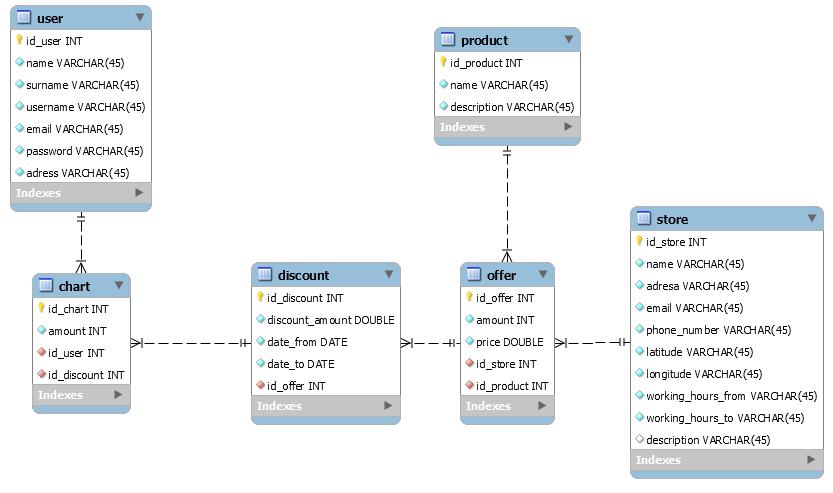
1. Popis funkcionalnosti

.

Tablica 1. Popis funkcionalnosti

|  |  |  |  |
| --- | --- | --- | --- |
| Broj funkc. | Naziv  funkcionalnosti | Opis funkcionalnosti | Slika ekrana |
| 1. | Registracija (modularno) | Registriranje korisnika – zapis imena, prezimena, lozinke, korisničkog imena, email-a i godine rođenja korisnika u bazu podataka |  |
| 1.1. | Registracija (Facebook) | Registracija korisnika putem njihovog Facebook korisničkog računa |  |
| 1.2. | Registracija (Google) | Registracija korisnika putem njihovog Google korisničkog računa |  |
| 2. | Prijava (modularno) | Mogućnost prijave korisnika u sustav sa svojim podacima za prijavu. Izvedena će biti prijava korisničkim imenom i lozinkom te brza prijava pomoću otiska prsta |  |
| 2.1 | Prijava (korisničko ime i lozinka) | Prijava je moguća korištenje korisničkog imena i lozinke kojim se korisnik registrirao, te je to osnovna vrsta prijave koju korisnik uvijek može koristiti u slučaju da ostale vrste ne rade ili ako korisnik to preferira. Kada se korisnik prvi puta prijavi sa svojim podacima u našu bazu podataka će se povući njegovi podaci i kreirati njegov korisnički profil u našoj bazi te će se korisniku u budućnosti omogućiti prijava sa tim podacima unutar same aplikacije i korištenje brzog načina prijave. |  |
| 2.2 | Prijava (Otisak prsta) (modularno) | Prijava putem otiska prsta bit će opcija brze prijave za korisnike koji imaju čitač otiska prsta na svojem mobilnom uređaju. Kada se korisnik prijavi u sustav može u unutar svojeg profila aktivirati mogućnost korištenja otiska prsta te će mu se u budućnosti nuditi ta opcija za prijavu |  |
| 3. | Početni zaslon aplikacije | Ovo je osnovni zaslon koji svaki korisnik vidi kada otvori aplikaciju |  |
| 3.1 | Pregled trgovina | Pregled svih trgovina iz baze podataka te je moguće te trgovine na razne načine sortirati ili filtrirati |  |
| 3.1.1 | Sortiranje trgovina | Mogućnost sortiranja trgovina po abecednom redu, kategorijama artikala koje trgovine prodaju, udaljenosti od trenutne lokacije korisnika |  |
| 3.1.2 | Filtriranje trgovina | Mogućnost filtriranja trgovina prema prvom slovu naziva trgovine |  |
| 3.1.3 | Pregled proizvoda odabrane trgovine | Pritiskom na određenu trgovinu otvara nam se popis svih proizvoda te trgovine s njihovim cijenama |  |
| 3.2 | Pregled svih proizvoda | Mogućnost pregleda svih proizvoda koji su zapisani u bazi podataka, moguće ih je sortirati te filtrirati na razne načine |  |
| 3.2.1 | Sortiranje proizvoda | Mogućnost sortiranja proizvoda prema kategorijama, datumu dodavanja, cijeni, popustu |  |
| 3.2.2 | Filtriranje proizvoda | Mogućnost filtriranja proizvoda prema prvom slovu proizvoda ili nazivu proizvoda. |  |
| 3.2.3 | Pregled odabranog proizvoda | Pritiskom na određeni proizvod otvara se novi prozor u kojem je moguće vidjeti detalje o određenom proizvodu te trgovine koje taj proizvod prodaju. |  |
| 3.2.4 | Favoriti | Mogućnost da korisnik doda neku trgovinu kao favorita. |  |
| 3.5 | Korisnički račun i postavke korisničkog računa | Ekran korisničkog računa sadrži razne informacije o korisniku te također i postavke koje korisnik može mijenjati |  |
| 3.5.1 | Postavke korisničkog računa | Pritiskom na postavke otvara se novi prozor u kojem je moguće mijenjati jezik aplikacije, odjaviti se prikazati uvjete korištenja |  |

1. Baza podataka



Na slici iznad nalazi se osnovni ERA model naše aplikacije. Sada ćemo pojasniti svaku od tablica modela kako bi bilo lakše shvatiti model podataka.

User – Tablica koja se odnosi na sve registrirane korisnike aplikacije sa osnovnim podacima svakoga od njh

Product – Tablica product predstavlja sve proizvode koje su trenutno u poduci aplikacije (neovisno o tome koja trgovina ima proizvod u ponudi)

Store– Predstavlja sve trgovine koje koriste aplikaciju, te njihove osnovne podatke

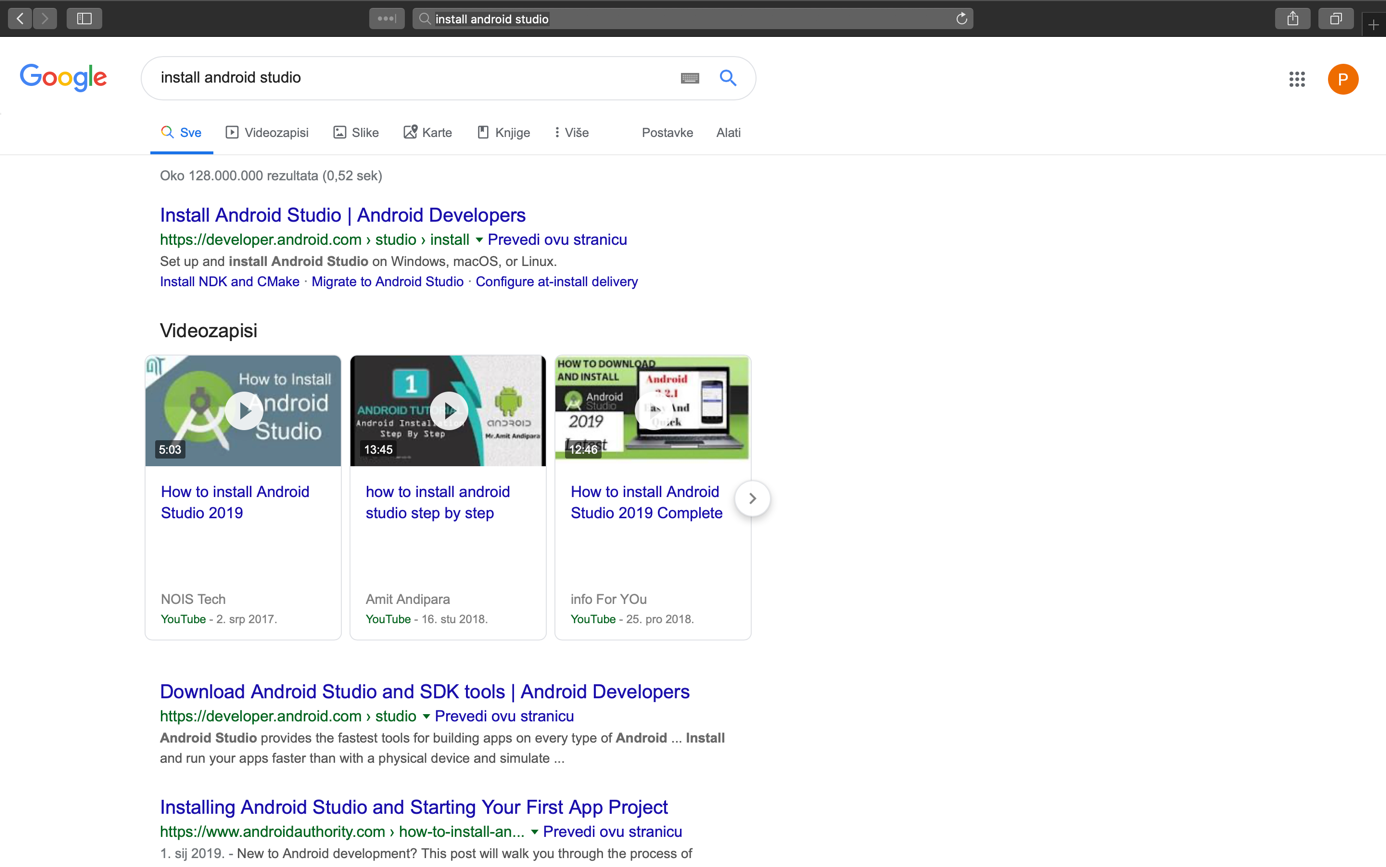
Offer – Tablica koja sadrži ponudu proizvoda jedne trgovine, sa definiranom količinom i cijenom(1 kom) proizvoda

Discount – Tablica predstavlja koli činu i trajanje popusta određenog proizvoda

Chart – Tablica chart jest košarica svih proizvoda koje je korisnik odabrao za kupnju

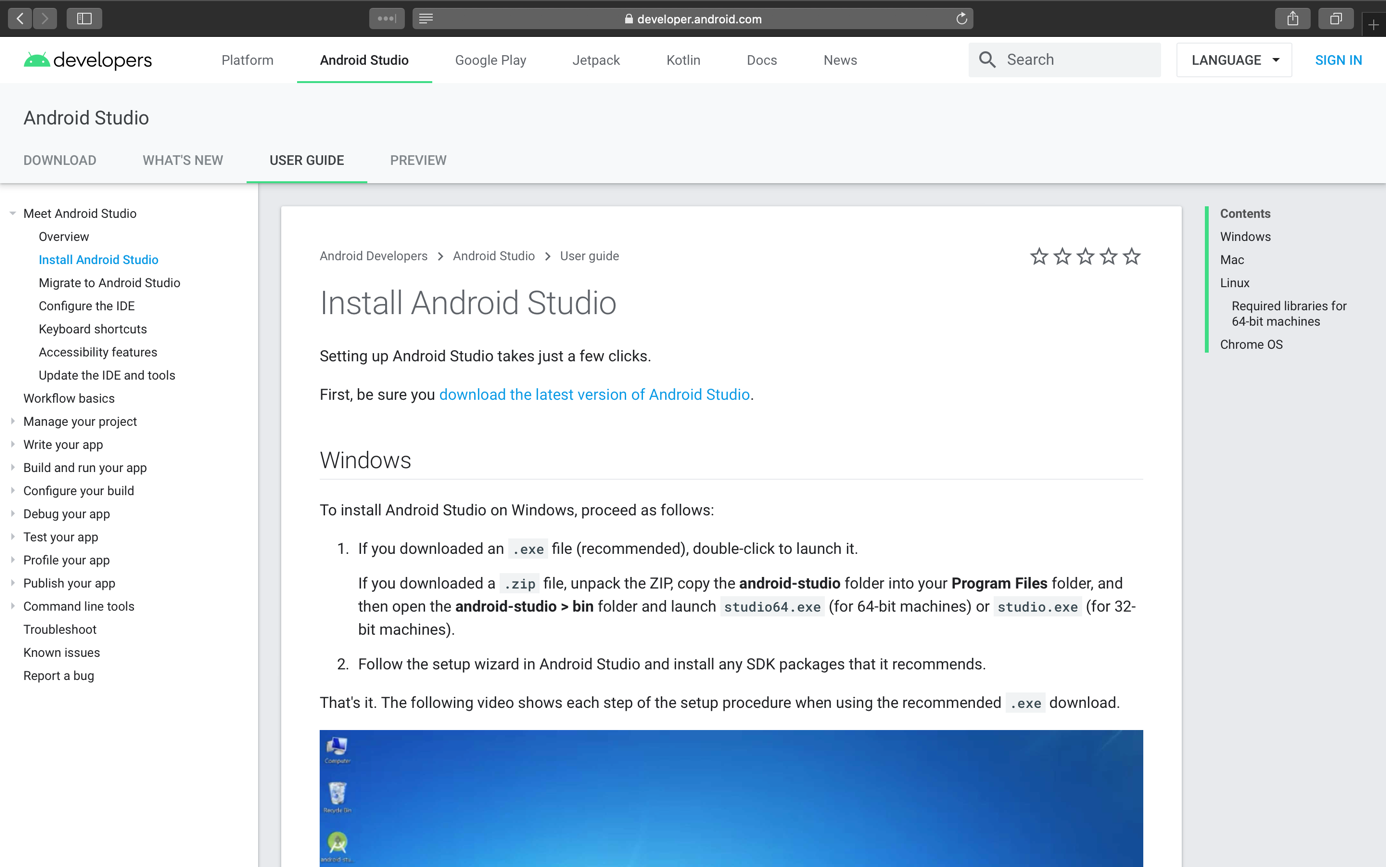
1. Razvoj aplikacije
   1. Instalacija razvojnog okruženja

Prije samog početka rada u Kotlin programskom jeziku potrebno je instalirati razvojno okruženje uz pomoć kojeg ćemo kasnije kreirati naš projekt i isprogramirati potrebne funkcionalnosti. Za potrebe ovog rada koristit ćemo Android Studio. Kako bi instalirali Android Studio potrebno je u bilo kojem internet pregledniku otvoriti Google i utipkati „install android studio“.



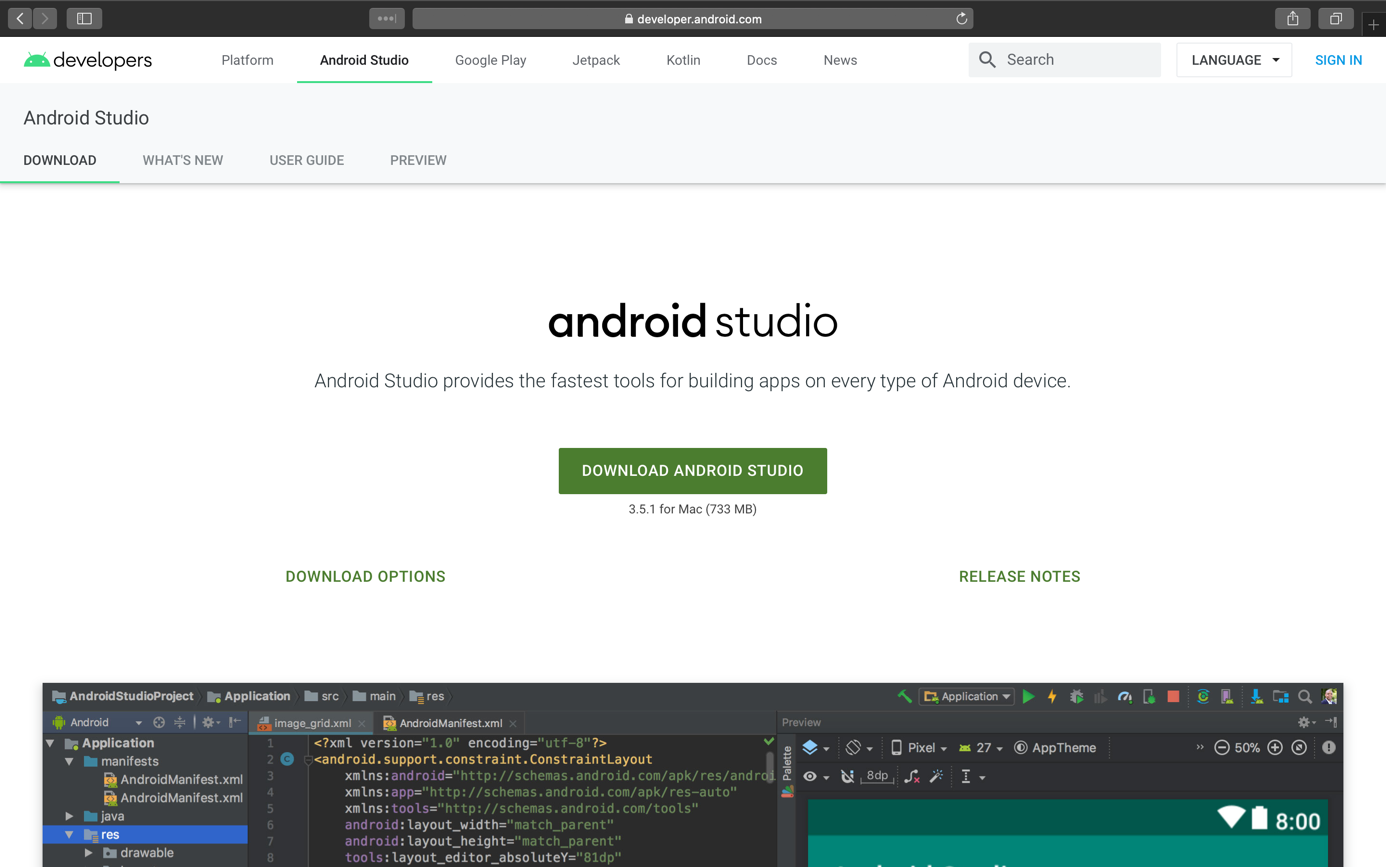
Slika 1. Pronalazak Android Studio razvojnog okruženja

Klikom na prvi link nam se otvara službena developer.android stranica na kojoj klikom na link „download the latest version of Android Studio“ odlazimo na stranicu za preuzimanje zadnje Android Studio verzije.



Slika 2. Pronalazak zadnje verzije Android Studio razvojnog okruženja

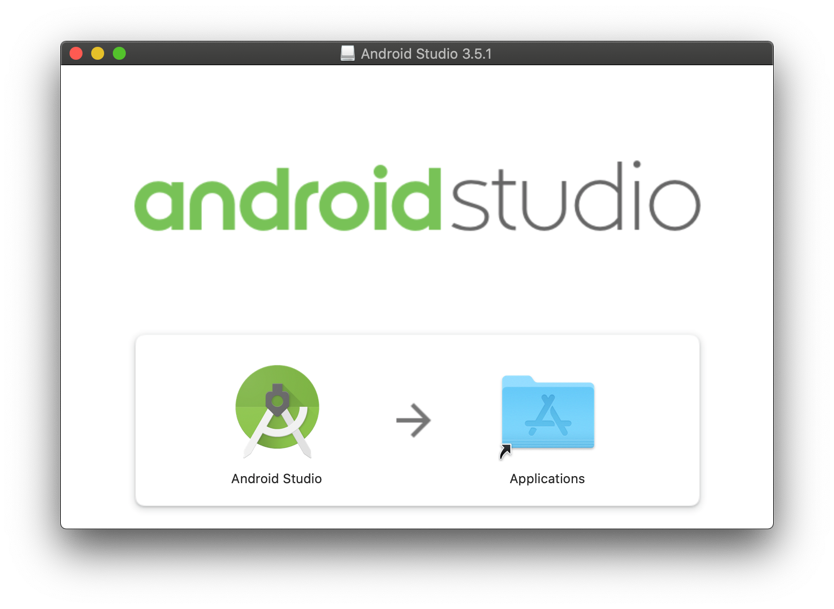
Nakon ovog klika ćemo imati prikazanu zadnju verziju Android Studio razvojnog okruženja za operativni sustav računala na kojemu se nalazimo, u našem slučaju je to verzija 3.5.1 za Mac, kao što je prikazano na sljedećoj slici.



Slika 3. Zadnja verzija Android Studio razvojnog okruženja

Sljedeći korak je preuzimanje Android Studia klikom na zeleni gumb „DOWNLOAD ANDROID STUDIO“, nakon čega ćemo morati potvrditi uvjete korištenja i naše preuzimanje će započeti.

Kada završi preuzimanje instalacijske datoteke u cijelosti potrebno je otvoriti istu. U našem slučaju je potrebno „drag&drop“ akcijom potrebno potvrditi početak instalacije.



Slika 4. Početak instalacijskog postupka

Nakon toga unutra Applications foldera pokrećemo Android Studio i na taj način pokrećemo proces instalacije. S obzirom da se radi o prvoj instalaciji Android Studio razvojnog okruženja, nećemo uvoziti nikakve prethodne postavke. Odabiremo Custom oblik instalacije, po želji se odabire tema razvojnog okruženja. Nakon toga je potrebno označiti Android Virtual Device kako bi instalirali, virtualni Android uređaj, kako mu i samo ime govori. Iduće je potrebno definirati količinu memorije koje ćemo dodijeliti takozvanom „Intel Hardware Accelerated Execution Manageru, skraćenica HAXM. Radi se o programu razvijenom od strane Intela i ubrzava rad x86 emulatora. Ako nemate Intel CPU, nećete imati ovu mogućnost. Ako imate 8 GB RAM-a ili više dovoljno je ostaviti na predloženoj vrijednosti(2 GB), ukoliko imate manje, primjerice 4GB, bilo bi dobro razmisliti o dodijeljivanju manjeg dijela memorije. Nakon toga klikom na gumb „Finish“ završavamo postavljanje instalacije i kreće preuzimanje SDK-a (software development kit), odnosno softwarea koji će nam omogućiti pisanje Android aplikacija i on je odvojen od onoga što smo preuzeli u početku, a to je samo razvojno okruženje. Nakon što se preuzimanje završilo klikom na „Finish“ potvrđujemo završetak instalacijskog postupka i prikazuje nam se prozor dobrodošlice Android Studio razvojnog okruženja.

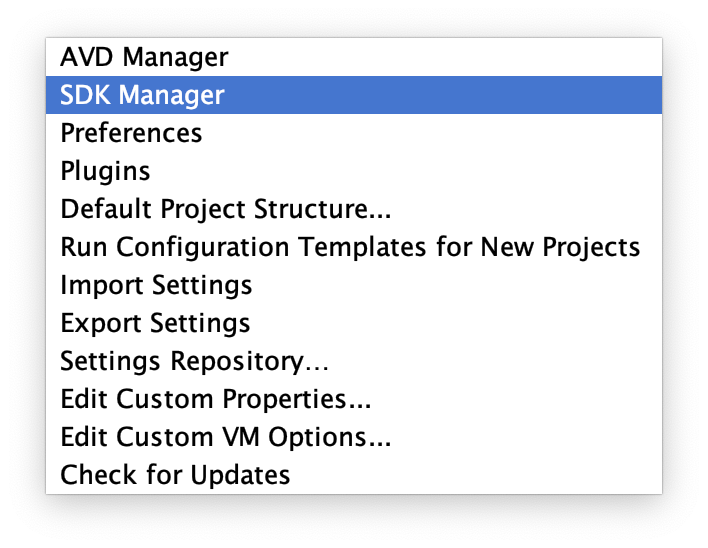
* 1. Postavljanje Android SDK-a

U ovom poglavlju ćemo napraviti određene konfiguracijske promjene i instalirati određene opcije koje mogu biti korisne. Ukoliko već nije pokrenut, potrebno je pokrenuti Android Studio. Na početnom ekranu potrebno je odabrati opciju „Configure“.



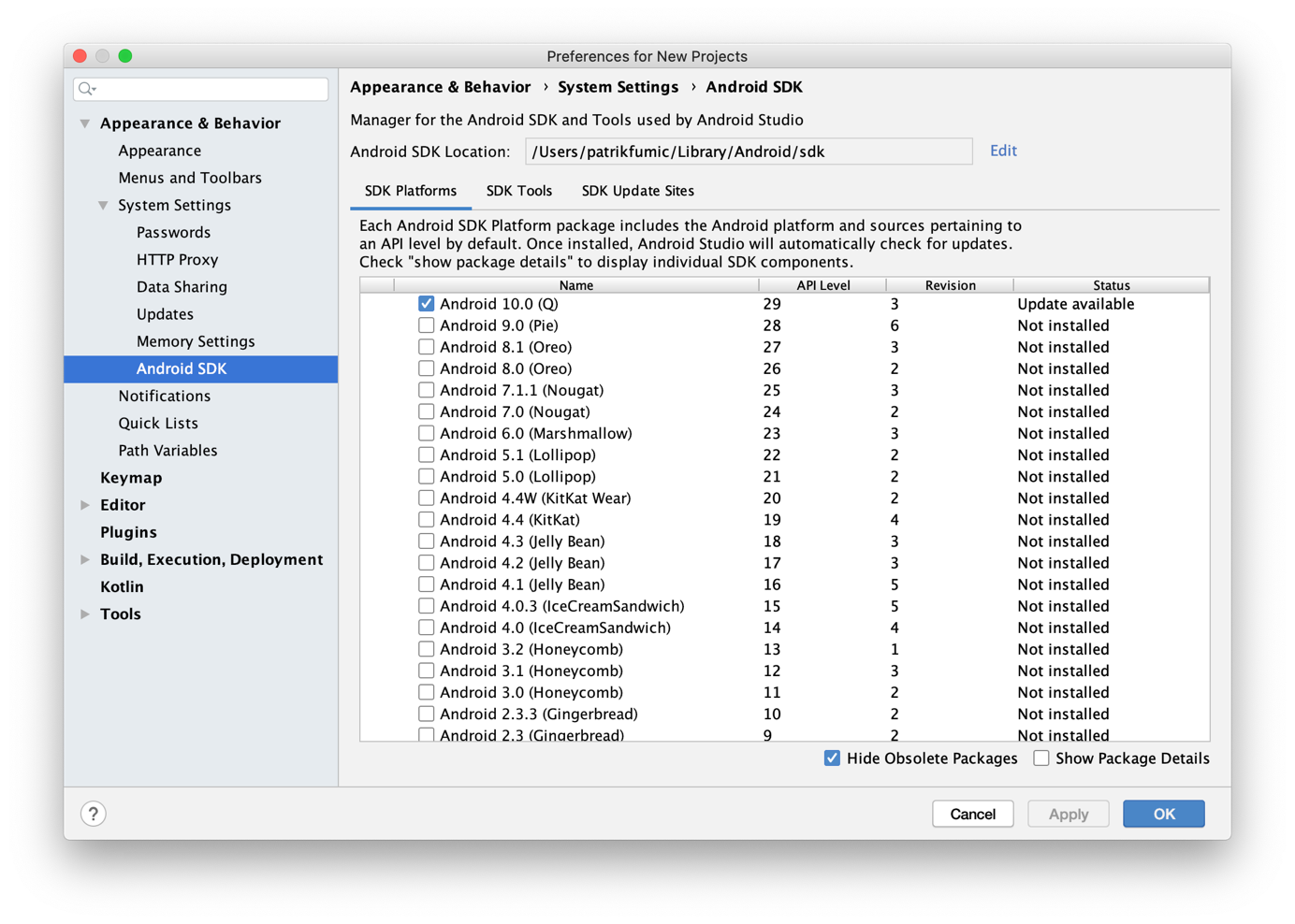
Slika 5. Početni prikaz Android Studio razvojnog okruženja

I nakon toga odabiremo SDK Manager opciju.



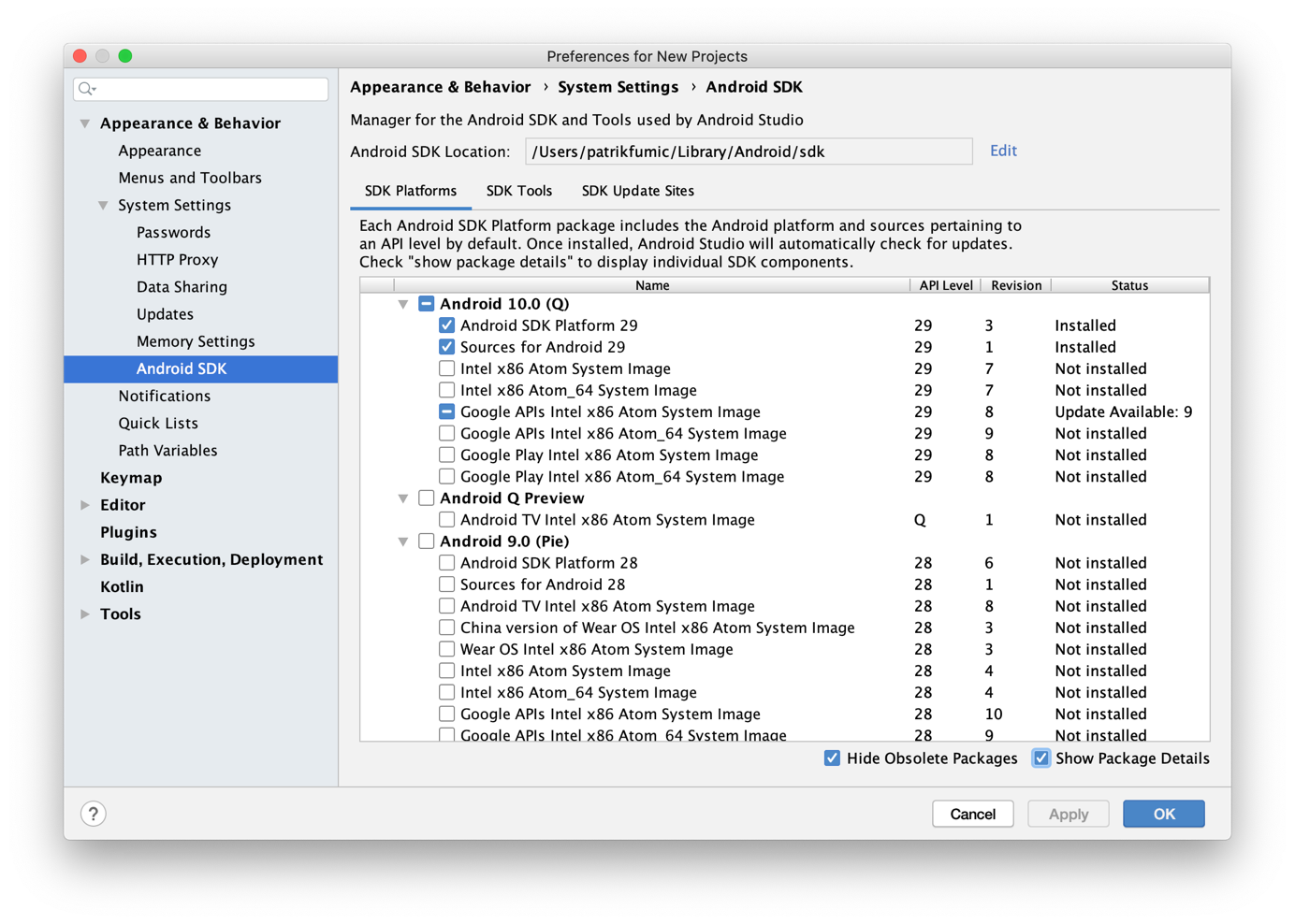
Slika 6. Izbornik mogućih postavki

Nakon što smo izvršili prethodne upute, pojavit će nam se sljedeći prozor na kojemu vidimo Android SDK postavke.



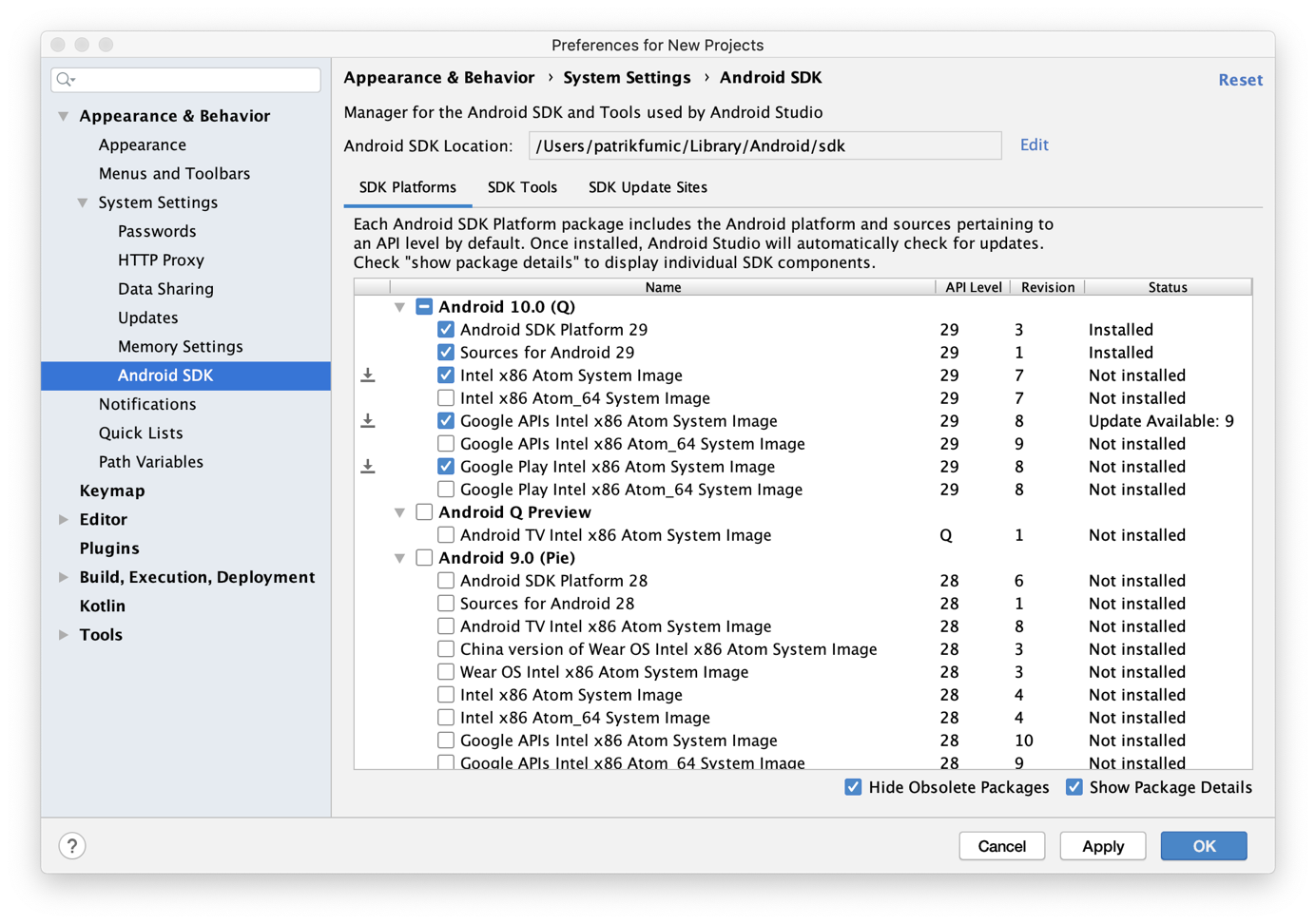
Slika 7. Android SDK postavke

Prvo što je potrebno provjeriti, posebice na Windows operativnim sustavima, je putanju Android SDK-a. Putanju možemo vidjeti pri vrhu prikazanog prozora na prethodnoj slici nakon natpisa „Android SDK Location“. Problem može uzrokovati razmak u samoj putanji koja je navedena. Uvjerite se postoji li negdje razmak u putanji. Ukoliko postoji morat ćete prebaciti lokaciju Android SDK-a. To možete napraviti na način da ručno kopirate prikazanu mapu negdje na istom disku čija putanja ne sadrži razmak. Nakon što smo se osigurali da nam putanja ne sadrži razmake možemo nastaviti dalje. Trenutno se nalazimo u Android SDK upravitelju i ovdje možemo instalirati dodatne značajke ili možemo ažurirati pojedine elemente ukoliko su ažuriranja dostupna. Ukoliko označimo „Show Package Details“ dobit ćemo dodatan prikaz pojedinih Android verzija.



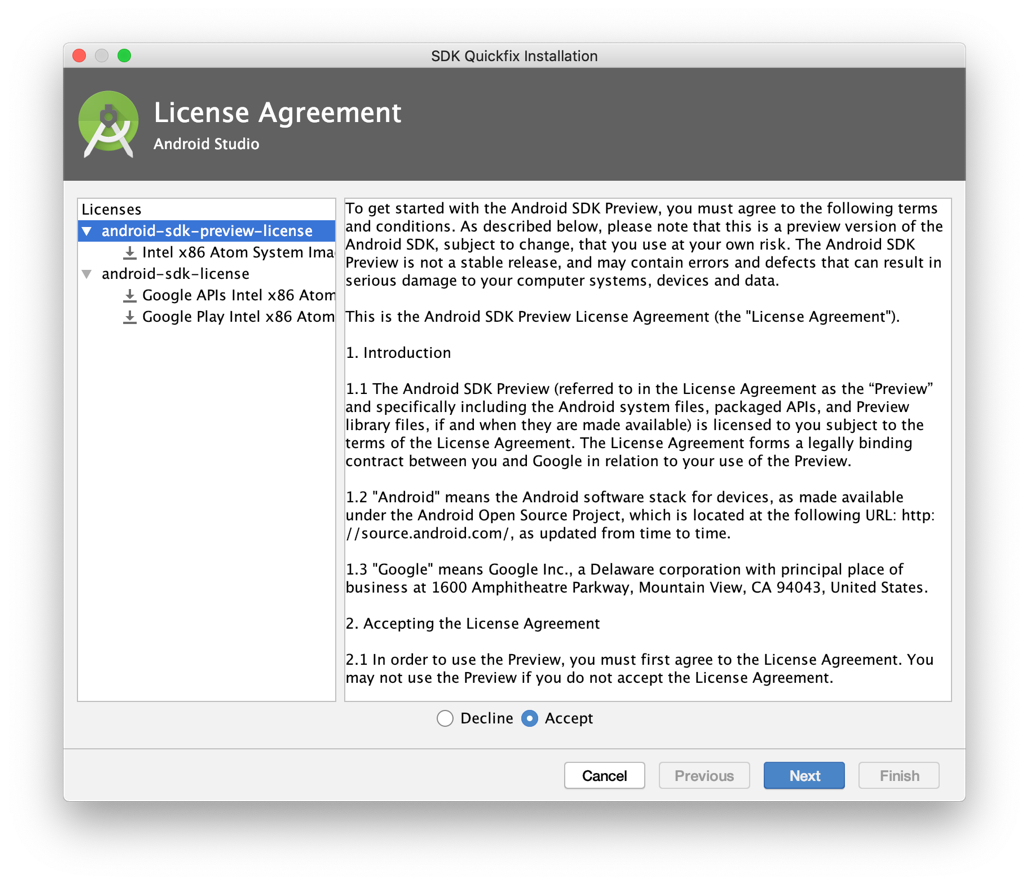
Slika 8. Prikaz detalja paketa u pojedinim verzijama Androida

Važno je naglasiti da nije potrebno instalirati većinu prethodnih Android verzija, zadnja verzija SDK-a će nam omogućiti razvoj aplikacija koje se mogu izvršavati na ranijim verzijama. Druga stvar koju ćemo naglasiti da SDK folder može jako brzo narasti u veličini ukoliko preuzimamo mnogo opcija. Jednostavno je popuniti približno 40 GB čak i bez da smo instalirali sve opcije. Međutim ukoliko želimo prikazati aplikacije razvijene u prethodnim verzijama bit će potrebno preuzeti danu verziju. Ono što se može uočiti da se mogu pojaviti takozvane „Preview“ verzije SDK i tu se radi o verzijama koje su samo za prezentaciju nadolazećih verzija i ne preporuča ih se koristiti za isporuku aplikacija. Mi ćemo svoj fokus usmjeriti Android 10.0 (Q) verziji i želimo imati instalirano sve označene komponente plus Intel x86 Atom System Image i Google Play Intel x86 Atom System Image. Nakon što smo označili sve što je navedeno u prethodnoj rečenici prozor bi trebao izgledati kao što je prikazano na sljedećoj slici i možemo kliknuti na gumb „Apply“.



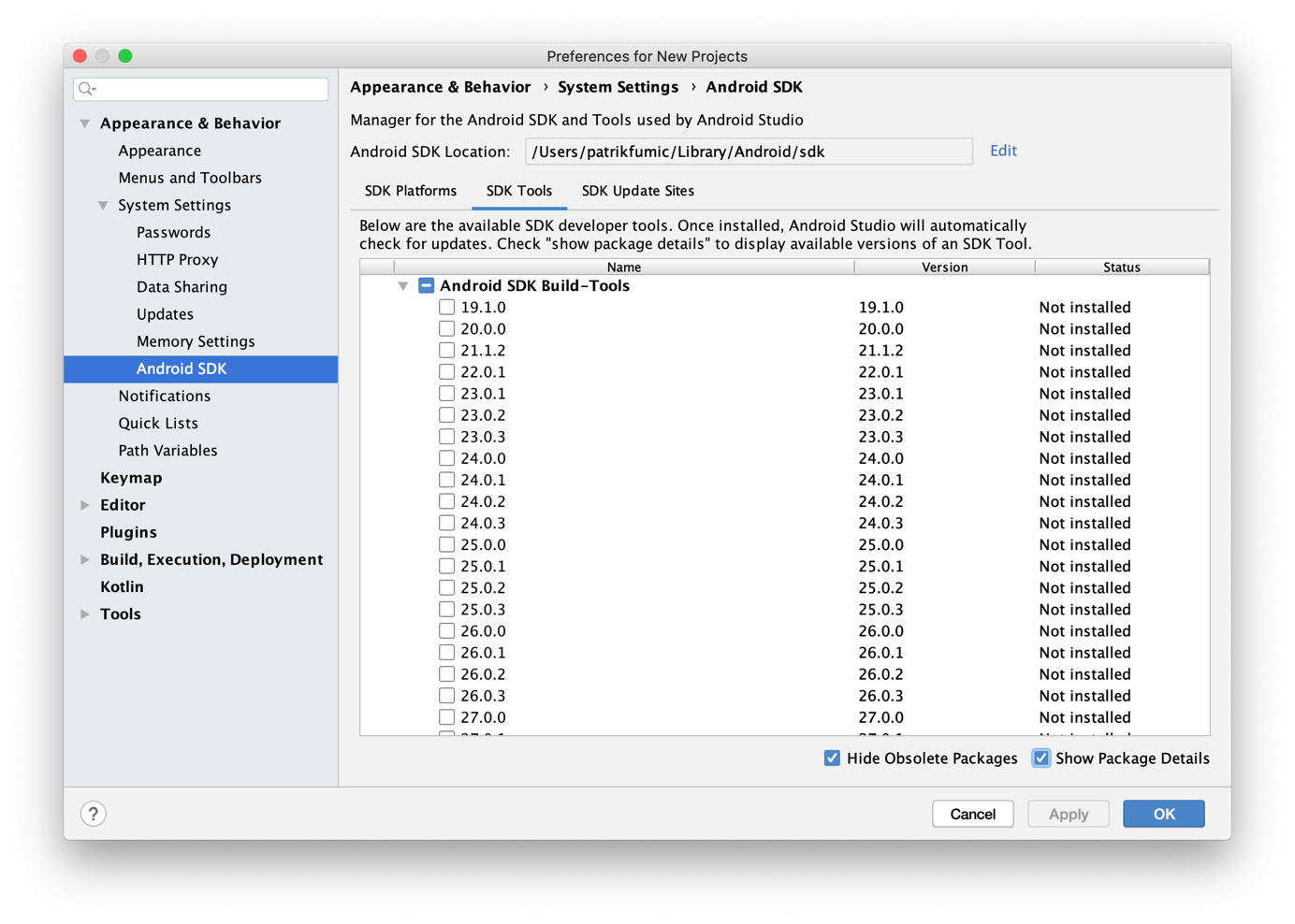
Slika 9. Prikaz označenih potrebnih paketa

Nakon što smo kliknuli na gumb „Apply“ potrebno je prihvatiti uvjete korištenja svih licenca.



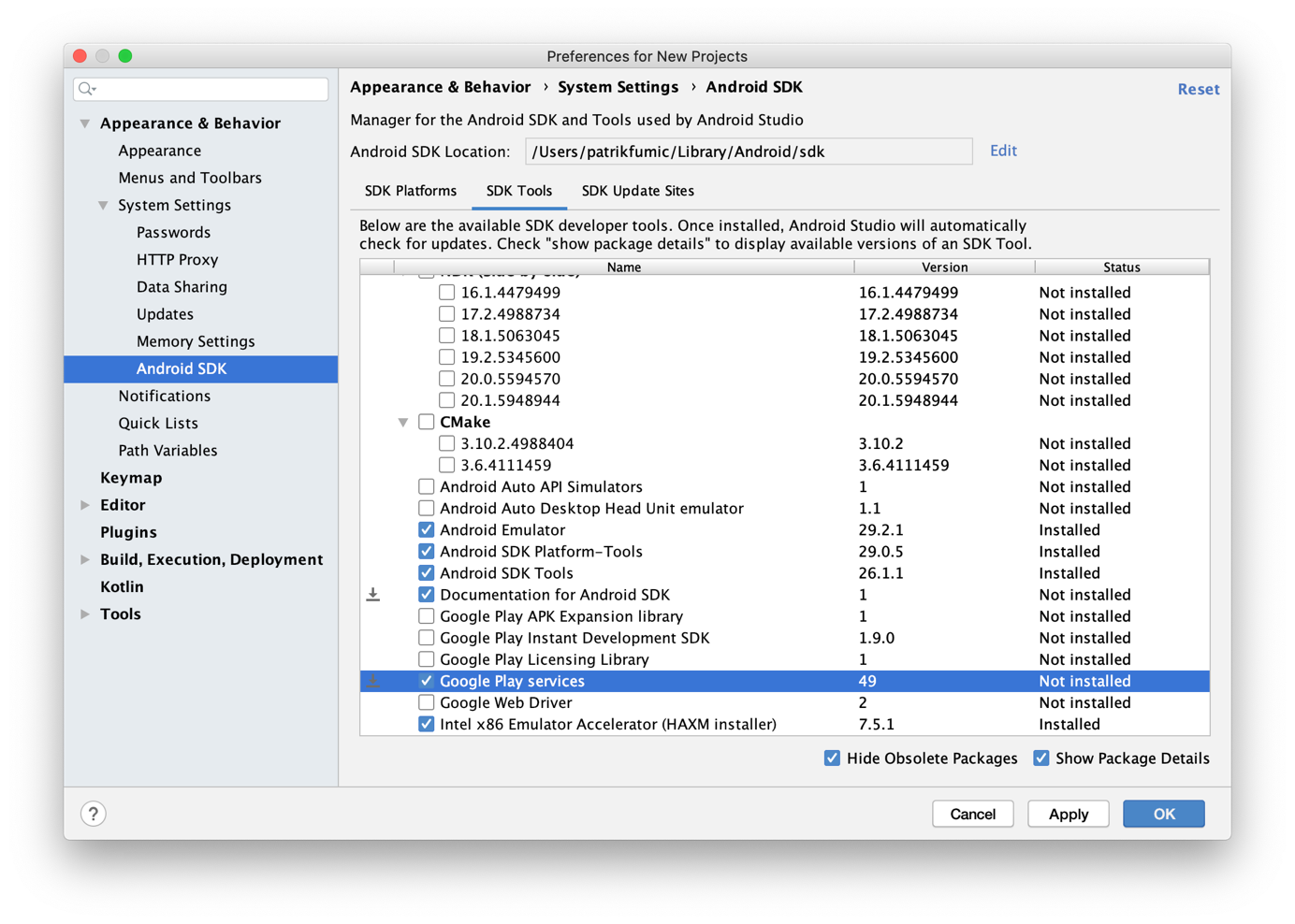
Slika 10. Prihvaćanje uvjeta korištenja paketa

Nakon toga će započeti preuzimanje zatraženih paketa. Na kraju preuzimanja, samo potvrdimo proces klikom na „Finish“ gumb. Nakon toga na prikazu SDK upravitelja odabiremo tab SDK Tools i pri dnu označimo „Show Package Details“.



Slika 11. Prikaz SDK alata

Prvo što je potrebno napraviti je provjeriti postoji li dostupno ažuriranje za neki od instaliranih paketa. Ukoliko postoji potrebno je označiti isti i pustiti ažuriranje. Drugo što je potrebno napraviti je preuzeti zadnju verziju Android SDK Build – Tools. Povlačenjem prikaza dostupnih verzija prema dolje doći ćemo do kraja popisa i želimo imati preuzetu zadnju moguću verziju bez „RC“ kratice, navedena kratica označava pojam „Release Candidate“ što znači da se ne radi o dovršenoj verziji. Sljedeći korak je provjeriti jesu li označeni Android Emulator, Android SDK Platform-Tools, Android SDK Tools, Documentation for Android SDK, Google Play services i Google USB Driver. Svi alati osim zadnja tri bi trebali biti prethodno označeni, dok će zadnja tri biti potrebno ručno preuzeti. Google Play services je iznimno važan zato što nam omogućuje funkcionalnosti kao što su Google Maps, a Google USB Driver nam omogućava stabilnije spajanje s fizičkim Android uređajem.

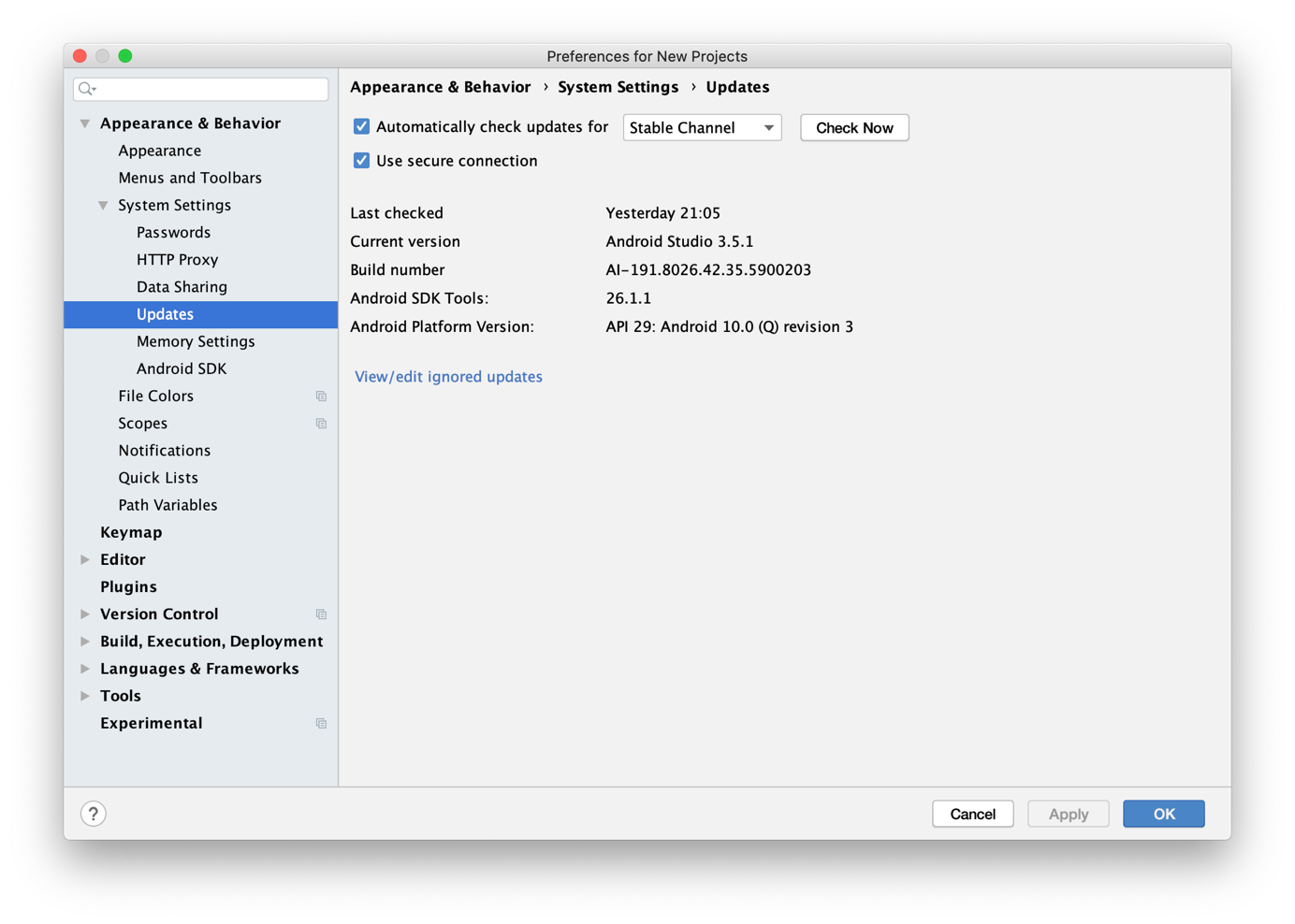


Slika 12. Prikaz potrebnih SDK alata

Važno je naglasiti da se u bilo kojem trenutku možemo vratiti na ovaj prikaz i naknadno instalirati potrebne alate, ukoliko se uvidi potreba za istima.

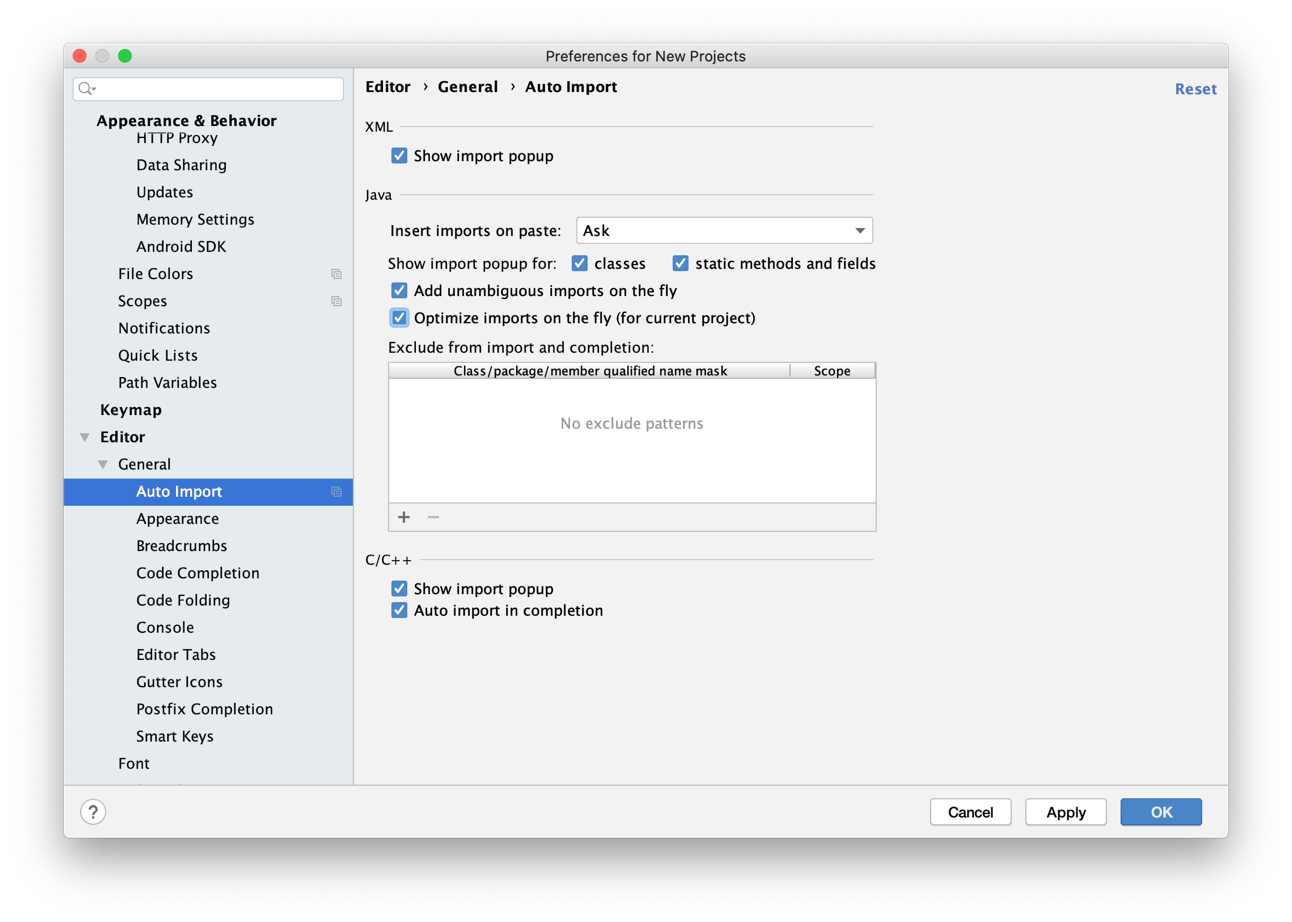
* 1. Preporučene postavke Android Studio razvojnog okruženja

Na početnom prikazu Android Studio razvojnog okruženja odabiremo opciju pri dnu „Configure“ i nakon toga odabiremo „Settings“ na Windows računalima ili „Preferences“ na Mac računalima. Nakon što nam se otvorio traženi prozor potrebno se pozicionirati kao što je prikazano na sljedećoj slici.

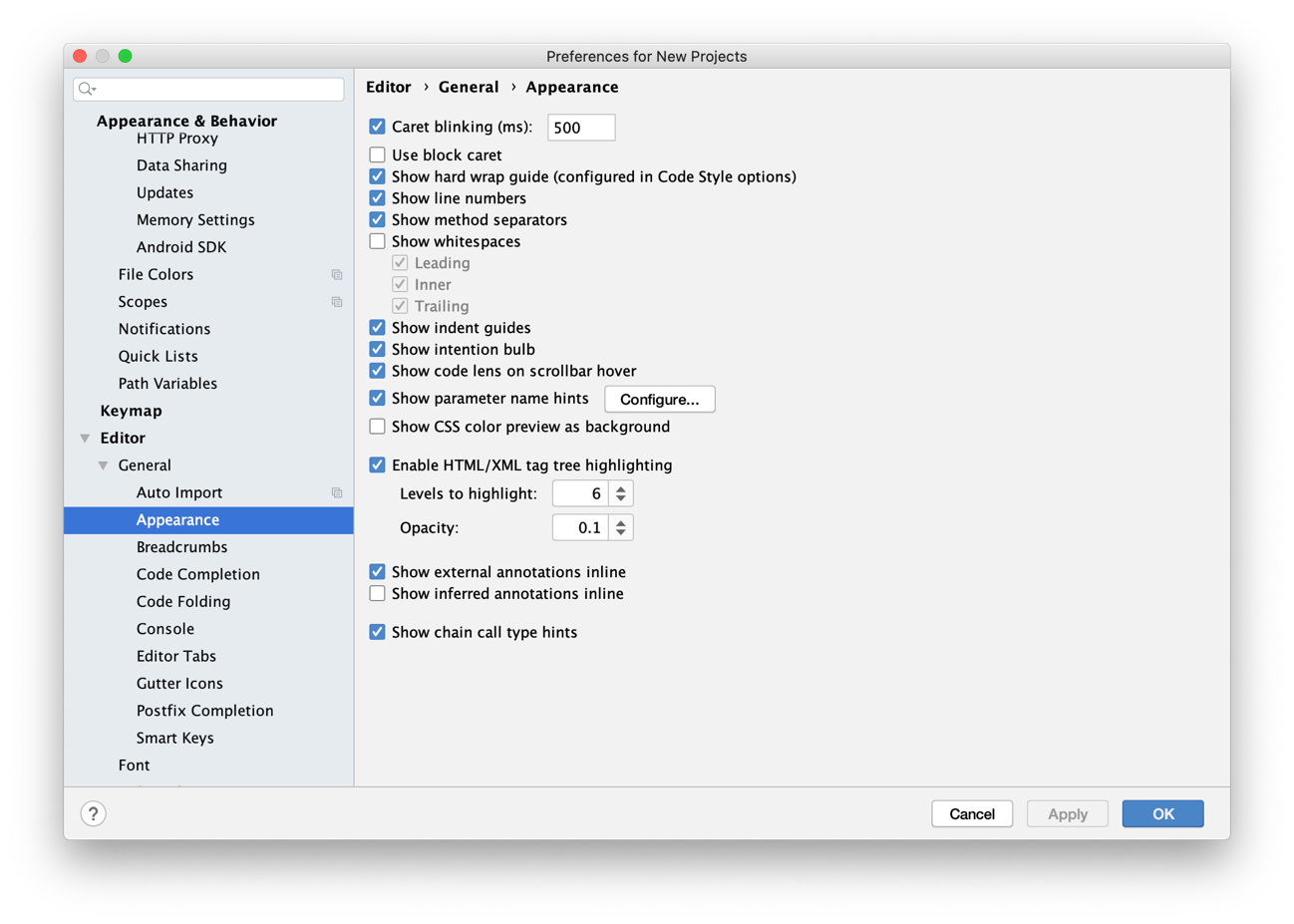


Slika 13. Android Studio Updates

Potrebno je odabrati iz padajućeg izbornika, ukoliko već nije, „Stable channel“ opciju. Nakon toga idemo u Editor odjeljak i potrebno je konfigurirati sve kako je prikazano u nastavku.

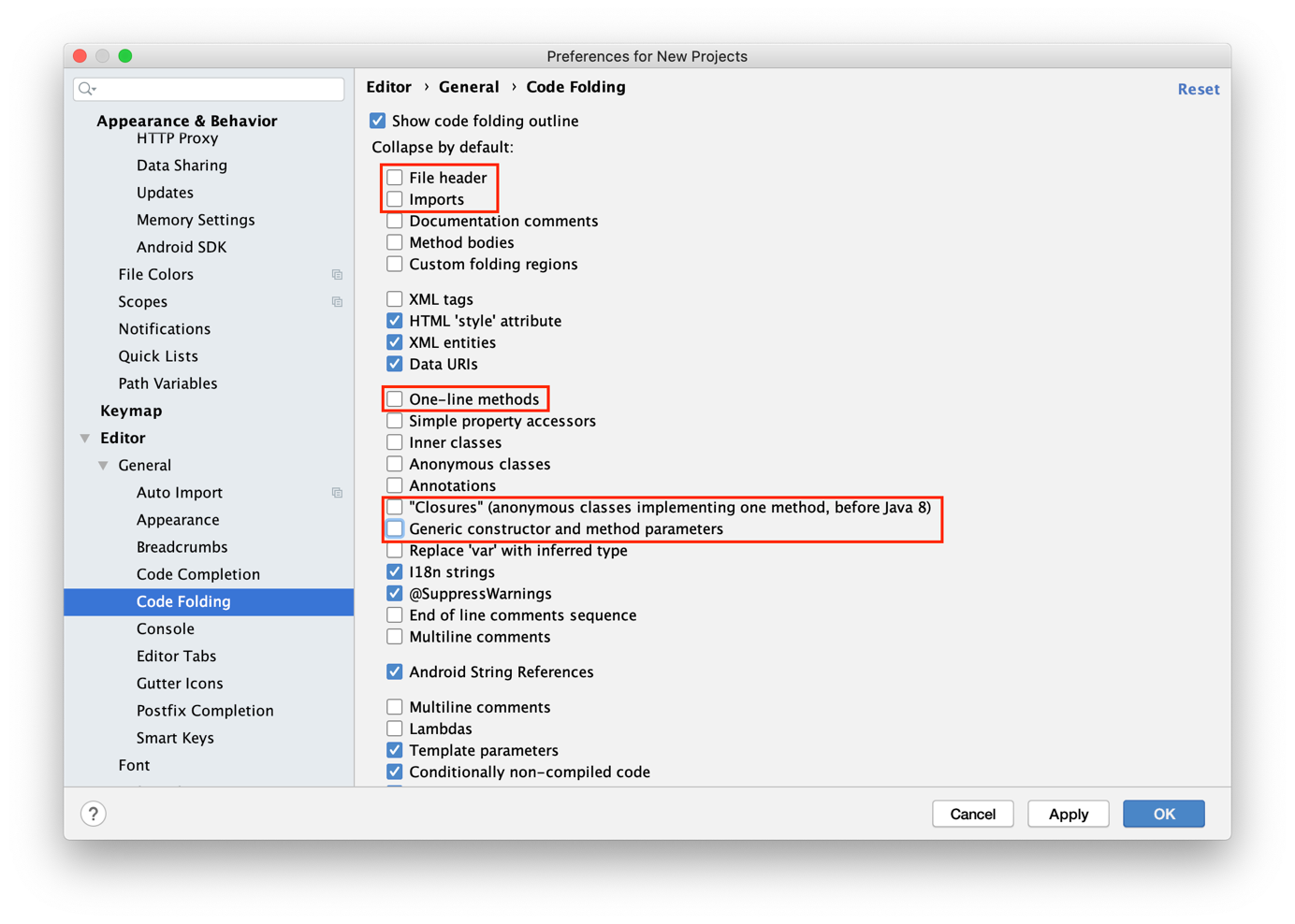


Slika 14. Auto Import



Slika 15. Appearance

„Show method separators“ u prethodnoj slici je u potpunosti opcionalna postavka, ono što nam omogućava je lakše uočavanje različitih metoda i na taj način potencijalno ubrzava rad. U idućem odjeljku Code Folding je potrebno ukloniti neke opcije, za razliku od prethodnih odjeljaka gdje smo ih dodatno označavali.

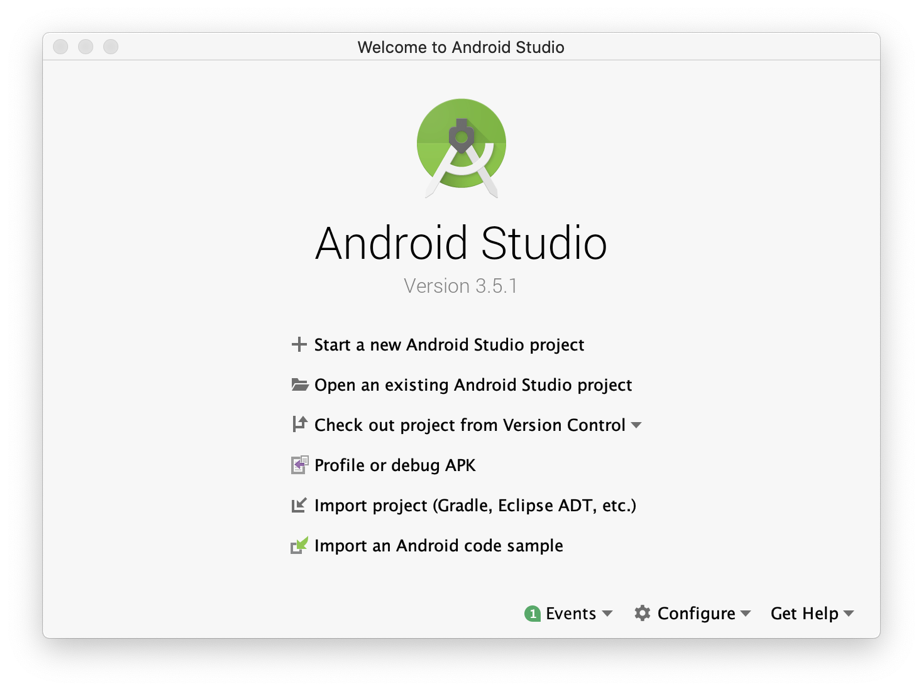


Slika 16. Code Folding

Opcije na prethodnoj slici definiramo na ovaj način zato što se radi o opcijama koje su više usmjerene naprednim Android razvojnim programerima. Za nekoga tko tek počinje s razvojem u Android studio razvojnom okruženju ovo je dobra praksa zato što navedene opcije skrivaju kod i mogu biti zbunjujuće.

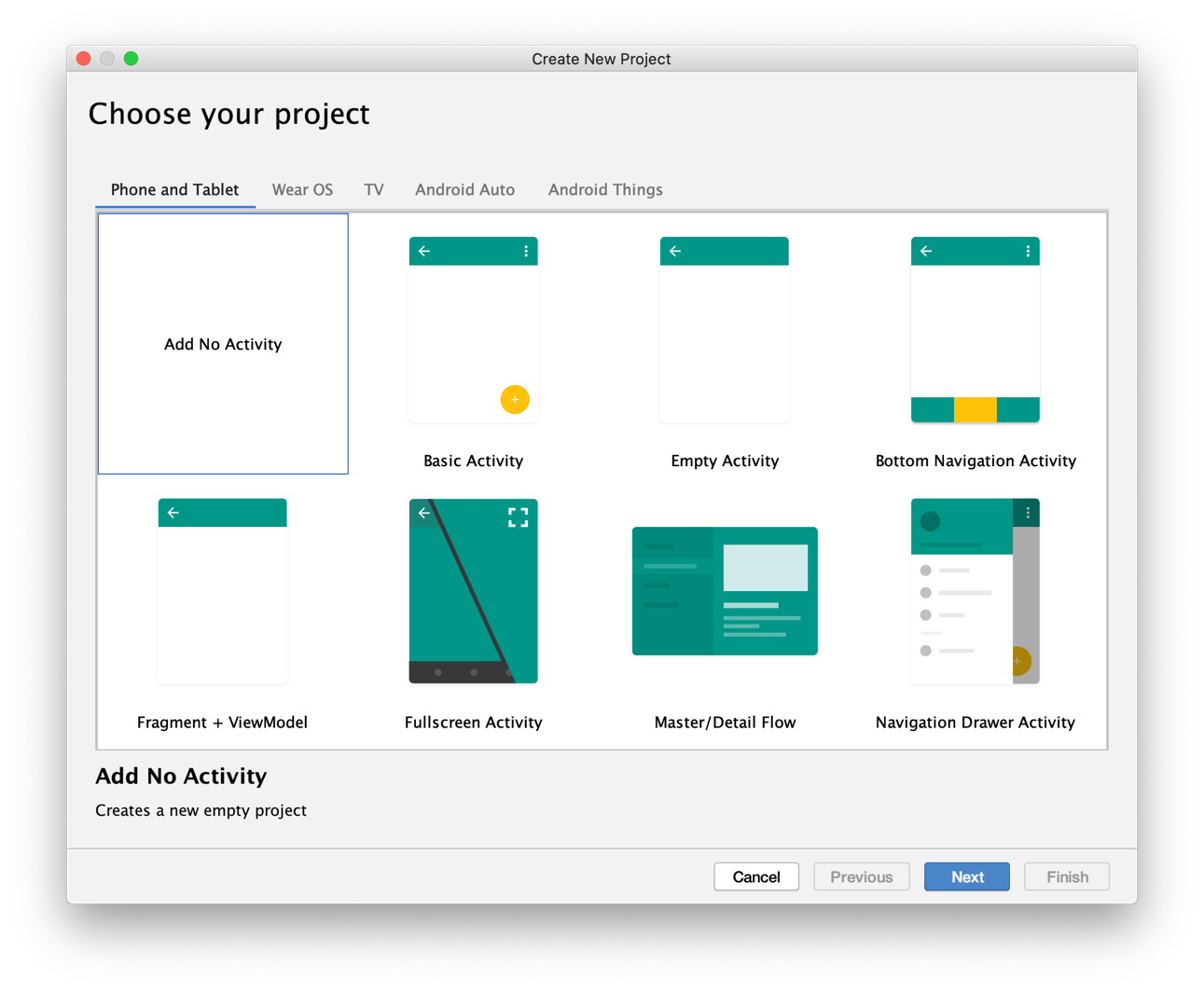
* 1. Kreiranje novog projekta

Prvi korak kod kreiranja novog programa je pokretanje Android Studio razvojnog okruženja i na početnom prozoru odabiranje opcije „Start a new Android Studio project“.



Slika 17. Kreiranje novog projekta

Nakon toga se pojavljuje prozor s prethodno definiranim Android predlošcima. U našem slučaju ćemo odabrati opciju „Empty Activity“.



Slika 18. Odabir Android predloška

U idućem koraku je potrebno navesti ime aplikacije i ime paketa. Package name bi trebao biti jedinstven kako bi referenca na aplikaciju bila jedinstvena u svijetu. Razlog je taj da jednom kad želimo uploadati aplikaciju na Google Play Store, aplikacija mora imati jedinstveni identifikator koji nitko drugi ne koristi. Lokaciju na koju želimo spremiti naš projekt proizvoljno postavljamo, u našem slučaju to je lokacija na koju smo klonirali repozitorij koji smo dobili od strane AIR kolegija. Nakon toga odaberemo jezik u kojemu želimo razvijati projekt, a to je Kotlin. Zadnja postavka koju odabiremo je Minimalna razina API-a. Ovdje dolazimo do trenutka u kojemu moramo odabirati između većeg broja značajki kod novijih verzija ili većeg podržanog broja korisnika koje nam omogućuju starije verzije. Mi smo se odlučili na API 19: Android 4.4 (KitKat). Klikom na gumb „Finish“ potvrđujemo postupak postavljanja projekta.

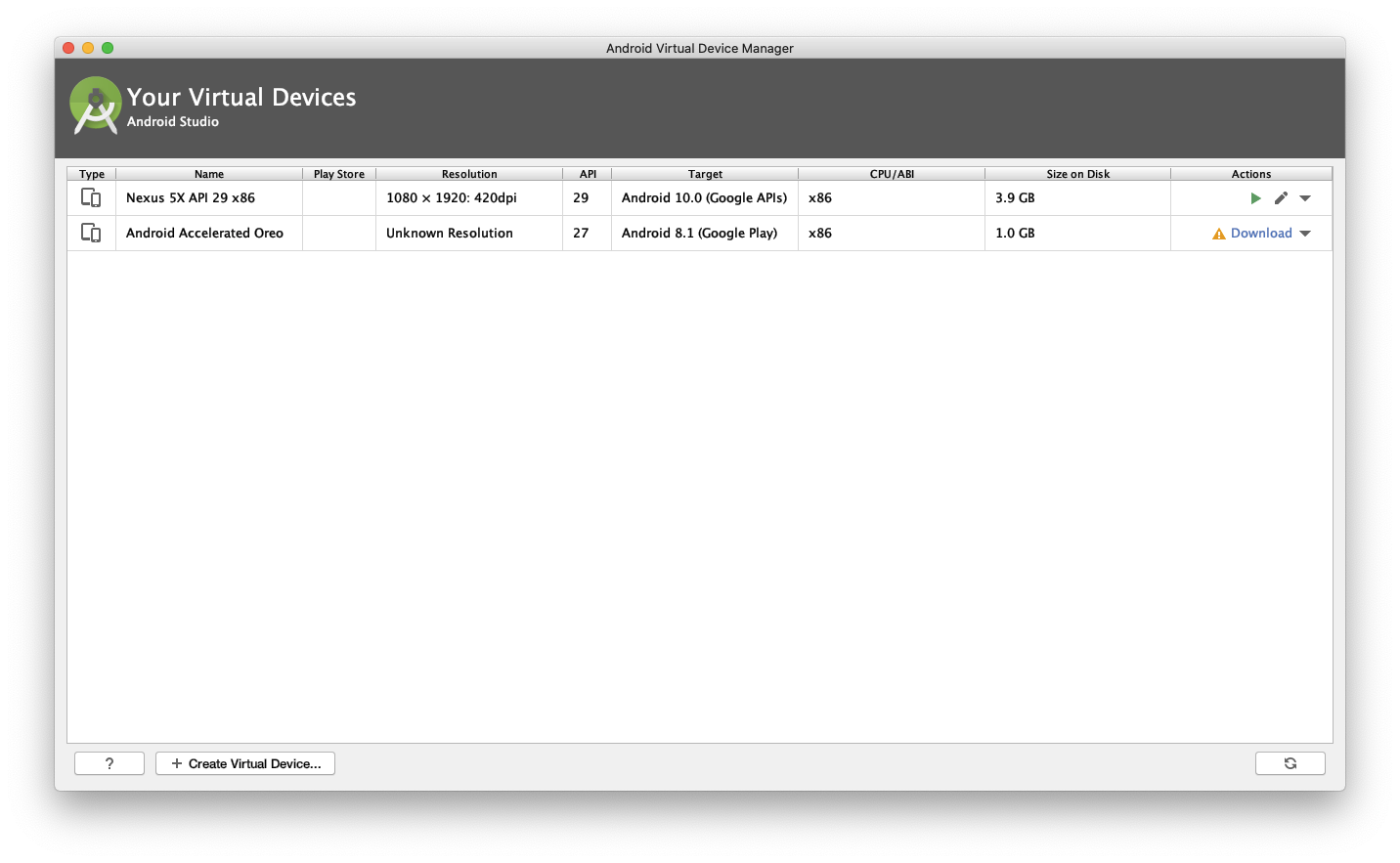


Slika 19. Početno postavljanje projekta

* 1. Postavljanje virtualnog uređaja

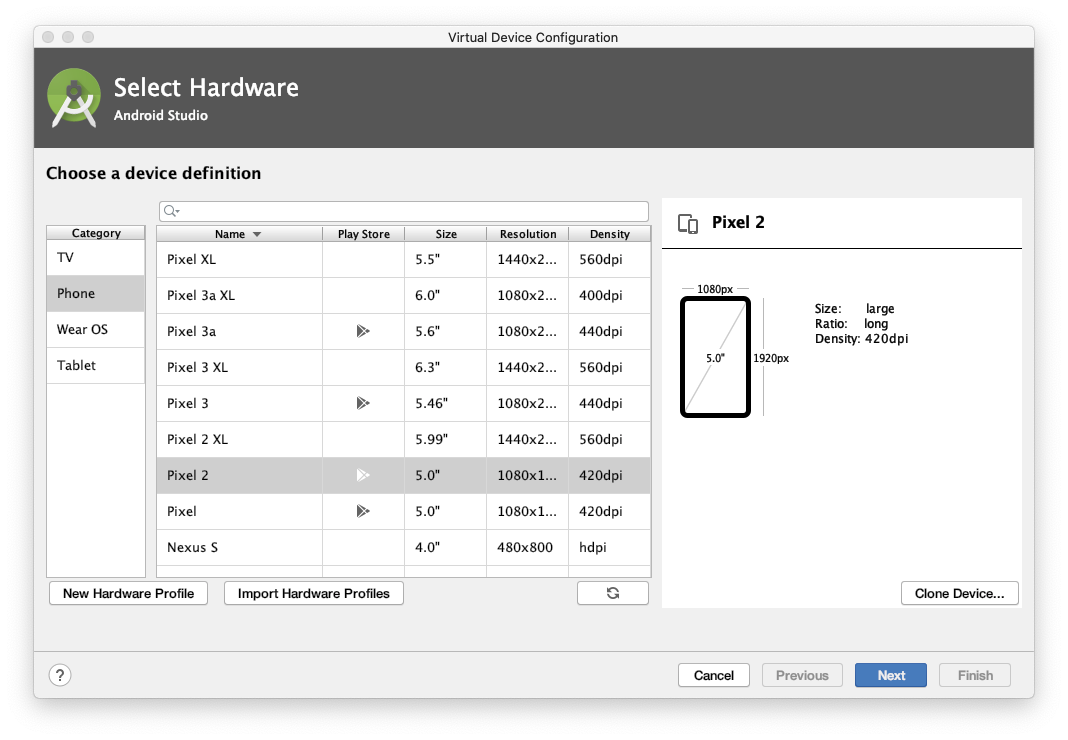
Za pokretanje aplikacija koje razvijamo potrebno je postaviti virtualan uređaj ili koristiti fizički uređaj kao domaćina za pokretanje aplikacije. U ovom poglavlju ćemo opisati postupak kreiranja i postavljanja virtualnog uređaja.

Prvi korak je, unutar Android Studio alata odabrati iz izborne trake Tools i zatim AVD Manager nakon čega dobivamo sljedeći prikaz.



Slika 20. AVD Manager

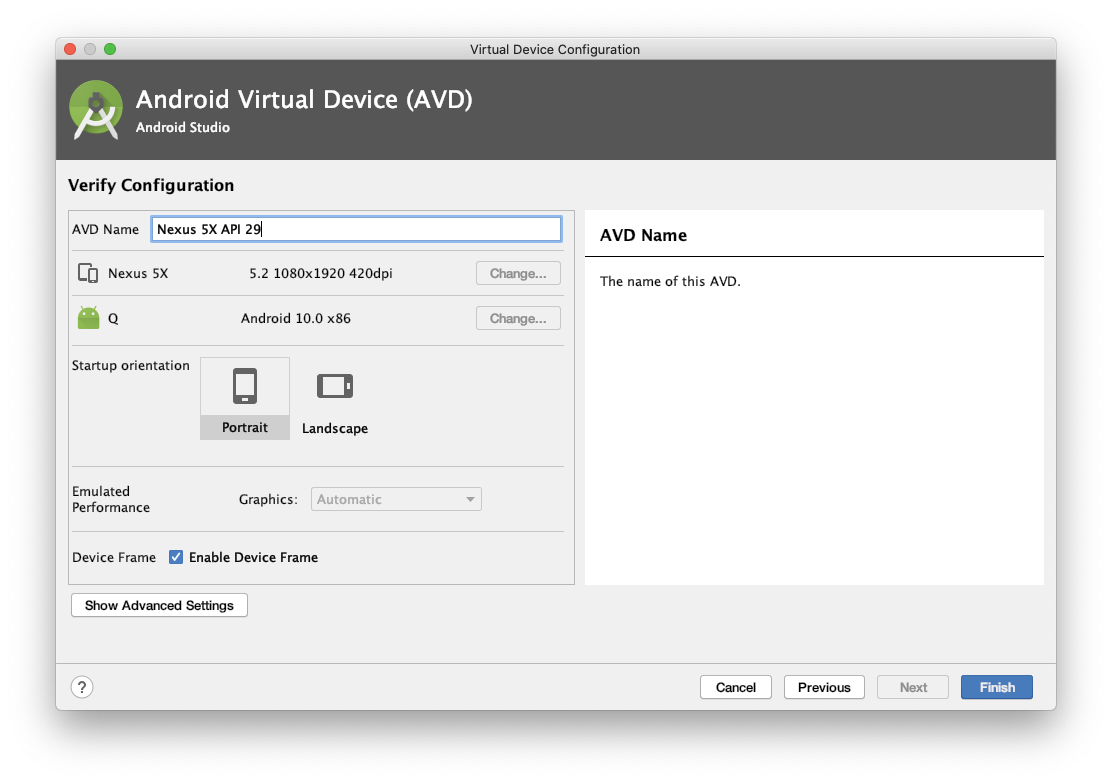
U prikazanom prozoru odabiremo opciju pri dnu „Create Virtual Device“. I nakon toga nam se otvara prozor sa svim mogućim uređajima koje možemo virtualizirati.



Slika 21. Odabir virtualnog uređaja

Mi ćemo u ovom slučaju odabrati Nexus 5X. Također je važno obratiti pozornost koji uređaj podržava Google Play Store, ukoliko nam je potreban. U slučaju da imate nešto slabije računalo dobro je proučiti uređaje i odabrati onaj koji je manje zahtjevan za pokretanje. Nakon što smo odabrali uređaj, prikazuje se prozor u kojem je potrebno odabrati System Image. Ukoliko se nalazite na linuxu najbolji odabir je x86 verzija određenog System Imagea. U slučaju da se nalazite na Windows ili Mac računalu i uspjeli ste aktivirati HAXM kao što je navedeno u prethodnim poglavljima onda je također x86 verzija optimalna. Ako HAXM nije aktivan morat ćete odabrati neku od drugih opcija kao što je armeabi-v7a.

Zadnji korak je dodijeljivanje naziva uređaju i odabiranje početne orijentacije. Ime proizvoljno dodijelite, a za orijentaciju je preporučeno ostaviti u portret prikazu. Postupak završavamo s klikom na gumb „Finish“.

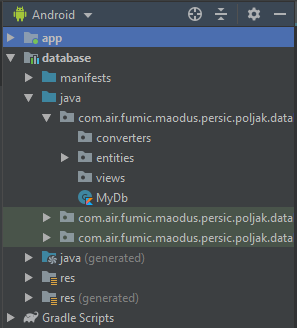


Slika 22. Zaključivanje konfiguracije virtualnog uređaja

* 1. Kreiranje entiteta baze podataka

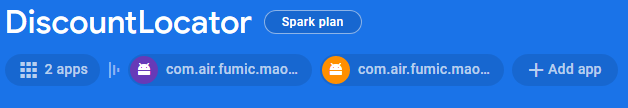
Za kreiranje klasa koje predstavljaju entitete baze podataka koristimo “data class” tip klase koju Kotlin koristi za klase koje služe samo za pohranu podataka. Kotlin za ovakve klase automatski generira equals() / hashCode(), toString(), copy() funkcije za svaki od atributa deklariranih u primarnom konstruktoru klase. Također, automatski generira i tzv. componentN() funkcije za dekonstruktiranje objekta u određeni broj varijabli. U nastavku ćemo objasniti kreiranje novog paketa (eng. package) i kreiranje data class objekata za sve entitete baze podataka.

Prvi korak jest kreiranje modula naziva “*database*”. Desnim klikom odabiremo “*app*”, te odabiremo New -> Modul, odabiremo “*Android library*”. Nakon toga unosimo ime modula kojeg želimo kreirati, u našem slučaju “*database*”, te odabiremo Ok. Nakon toga unutar kreiranog modula, kreiramo pakete “*converters*” “*entities*” i “*views*”. U "*entities*” se nalaze klase entiteta baze, dok se u "*views*” paketu nalaze razni pogledi koje dohvaćamo iz baze podataka. Isto tako, unutar “*coverters*” paketa nalaze se konverteri podataka. Novi paket kreiramo desnim klikom na “*database*” i odabiremo New -> Package. Nakon kreiranja potrebnih paketa struktura projekta treba izgledati kao na slici ispod.



Slika 23. Struktura projekta

Nakon toga, potrebno je kreirati Firebase projekt na kojem ćemo kreirati bazu podataka. Na [službenoj stranici](https://firebase.google.com) Firebase-a prijavljujemo se Google računom, te kreiramo novi projekt. Nakon kreiranja novog projekta odabiremo “*Add app*” kao na slici 12. I dodajemo id aplikacije ili u našem slučaju modula, npr. com.companyname.database. Firebase tada kreira datoteku google-services.json koju je potrebno staviti u root direktorij modula unutar “*Project*” pogleda.



Slika 12. Dodavanje aplikacije u Firebase

Kako Firebase koristi bazu podataka koja se na bazira na tablicama već na strukturi stabla, korijen stabla predstavlja našu bazu podataka. Svako dijete korijena stabla je jedan entitet baze podataka, dok je svako dijete entiteta jedan unos tog entiteta u bazu podataka. Kreirana baza sa nekoliko unosa treba izgledati kao na slici ispod.



Slika 13. Firebase baza podataka

Poslijednji korak dodavanja Firebase-a u naš projekt jest dodavanje tzv. “*dependencies*” u gradle datoteku database modula, te u globalnu (Project: Discount Locator) gradle datoteku kako bi mogli koristiti Google Play Services. Unutar globalne gradle datoteke, pod “*dependencies*” dodajemo liniju koda:

classpath 'com.google.gms:google-services:4.3.3'

Nakon toga u gradle datoteku našeg modula dodajemo:

implementation 'com.google.firebase:firebase-auth:19.1.0'

implementation 'com.google.firebase:firebase-firestore:21.3.0'

implementation 'com.google.firebase:firebase-database:19.2.0'

apply plugin: 'com.google.gms.google-services'

Idući korak je kreiranje klasa entiteta baze unutar naše Android aplikacije. Desnim klikom odabiremo paket “entities”, te odabiremo New -> Kotlin File/Class. Nakon toga unosimo ime klase, odnosno entiteta kojeg kreiramo i odabiremo Ok. U nastavku ćemo prikazati kod svake klase entiteta. Kako smo ranije napomenuli koristimo “*Data Class*” vrstu klase u Kotlinu, stoga će naše klase entiteta sadržavati samo atribute entiteta kojeg prestavljaju.

data **class** User (

var id\_user: Int,

var name: String,

var surname: String,

var username: String,

var email: String,

var password: String,

var adress: String

)

data **class** Store (

var id\_store: Int,

var name: String,

var adress: String,

var email: String,

var phone\_number: String,

var latitude: String,

var longitude: String,

var working\_hours\_from: String,

var working\_hours\_to: String,

var description: String

)

data **class** Product(

var id\_product: Int,

var name: String,

var description: String

)

data **class** Offer (

var id\_offer: Int,

var amount: Int,

var price: Double,

var id\_store: Int,

var id\_product: Int

)

data **class** Discount (

var id\_discount: Int,

var discount\_amount: Double,

var date\_from: String,

var date\_to: String,

var id\_offer: Int

)

data **class** Chart (

var id\_chart: Int,

var amount: Int,

var id\_user: Int,

var id\_discount: Int

)

Nakon što smo kreirali potrebne entitete, potrebno je kreirati klasu koja će izvoditi sve potrebne operacije sa bazom podataka. Unutar direktorija u kojem se nalaze naši paketi kreiramo klasu naziva “*MyDb*”, desni klik na paket com.companyname.database i odabiremo New -> Kotlin File/Class, dodajemo ime i odabiremo Ok. Prva stvar koju moramo napraviti jest konekcija na bazu. Firebase to izvršava pomoću reference (*eng. reference*) na trenutno stanje baze. Referencu na bazu spremamo u varijablu “*database*”, preko koje dohvaćamo sve elemente baze podataka. Referencu dohvaćamo na sljedeći način:

val database = FirebaseDatabase.getInstance().reference

Sada preko reference izvodimo sve operacije nad bazom, insert funkcije izvršavaju se preko ”*insertValue()”* naredbe. U nastavku slijedi kod svih insert funkcija.

fun insertUser(user: User){

database.child("users").child(user.id\_user.toString()).setValue(user)

}

fun insertStore(store: Store){

database.child("store").child(store.id\_store.toString()).setValue(store)

}

fun insertProduct(product: Product){

database.child("product").child(product.id\_product.toString()).setValue(product)

}

fun insertOffer(offer: Offer){

database.child("offer").child(offer.id\_offer.toString()).setValue(offer)

}

fun insertDiscount(discount: Discount){

database.child("discount").child(discount.id\_discount.toString()).setValue(discount)

}

fun insertChart(chart: Chart){

database.child("chart").child(chart.id\_chart.toString()).setValue(chart)

}

Brisanje elemenata izvodi se naredbom “*removeValue()”,* kod svi funkcija brisanja elemenata nalazi se u nastavku.

fun deleteUser(user\_id: Int){

database.child("users").child(user\_id.toString()).removeValue()

}

fun deleteStore(store\_id: Int){

database.child("store").child(store\_id.toString()).removeValue()

}

fun deleteProduct(product\_id: Int){

database.child("product").child(product\_id.toString()).removeValue()

}

fun deleteOffer(offer\_id: Int){

database.child("users").child(offer\_id.toString()).removeValue()

}

fun deleteDiscount(discount\_id: Int){

database.child("discount").child(discount\_id.toString()).removeValue()

}

fun deleteChart(chart\_id: Int){

database.child("chart").child(chart\_id.toString()).removeValue()

}

Ispis svih elemenata entiteta odvija se dodavanjem tzv. “*ValueEventListener*” objekta na entitet ili dijete stabla čije zapise želimo dohvatiti. Svakim dohvaćanjem zapisa potrebno je kreirati ValueEventListener, te ga dodati na željeno dijete stabla (entitet baze).Listener override-a dvije funkcije “*onDataChange()*”, u kojem dohvaća podatke, te “*onCancelled()*” koju poziva u slučaju greške. U nastavku su prikazane funkcije dohvaćanja svih elemenata entiteta.

fun selectUsers(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("users").addValueEventListener(listener)

}

fun selectStore(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("store").addValueEventListener(listener)

}

fun selectProduct(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("product").addValueEventListener(listener)

}

fun selectOffer(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("offer").addValueEventListener(listener)

}

fun selectDiscount(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("discount").addValueEventListener(listener)

}

fun selectChart(){

val listener = object : ValueEventListener {

override fun onDataChange(dataSnapshot: DataSnapshot) {

val post = dataSnapshot.value

Log.i("querry", post.toString())

}

override fun onCancelled(databaseError: DatabaseError) {

Log.e("error", "Getting data failed.")

}

}

database.child("chart").addValueEventListener(listener)

}

Popis literature

Popis slika

[Slika 1. Pronalazak Android Studio razvojnog okruženja 10](#_Toc25184277)

[Slika 2. Pronalazak zadnje verzije Android Studio razvojnog okruženja 11](#_Toc25184278)

[Slika 3. Zadnja verzija Android Studio razvojnog okruženja 11](#_Toc25184279)

[Slika 4. Početak instalacijskog postupka 12](#_Toc25184280)

[Slika 5. Početni prikaz Android Studio razvojnog okruženja 13](#_Toc25184281)

[Slika 6. Izbornik mogućih postavki 13](#_Toc25184282)

[Slika 7. Android SDK postavke 14](#_Toc25184283)

[Slika 8. Prikaz detalja paketa u pojedinim verzijama Androida 15](#_Toc25184284)

[Slika 9. Prikaz označenih potrebnih paketa 16](#_Toc25184285)

[Slika 10. Prihvaćanje uvjeta korištenja paketa 16](#_Toc25184286)

[Slika 11. Prikaz SDK alata 17](#_Toc25184287)

[Slika 12. Prikaz potrebnih SDK alata 18](#_Toc25184288)

[Slika 13. Android Studio Updates 19](#_Toc25184289)

[Slika 14. Auto Import 20](#_Toc25184290)

[Slika 15. Appearance 20](#_Toc25184291)

[Slika 16. Code Folding 21](#_Toc25184292)

[Slika 17. Kreiranje novog projekta 22](#_Toc25184293)

[Slika 18. Odabir Android predloška 22](#_Toc25184294)

[Slika 19. Početno postavljanje projekta 23](#_Toc25184295)

[Slika 20. Struktura projekta 24](#_Toc25184296)

Popis tablica

[Tablica 1. Popis funkcionalnosti 2](#_Toc25169190)

Prilozi (1, 2, …)