Razvoj programskog dodatka za Android Studio

Iličić, Krešimir

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:586021

Rights / Prava: Attribution-NonCommercial-NoDerivs 3.0 Unported/Imenovanje-Nekomercijalno-Bez prerada 3.0

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Krešimir Iličić

RAZVOJ PROGRAMSKOG DODATKA ZA ANDROID STUDIO

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Krešimir Iličić

Matični broj: 45063/16–R

Studij: Poslovni sustavi

RAZVOJ PROGRAMSKOG DODATKA ZA ANDROID STUDIO

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Zlatko Stapić

Varaždin, rujan 2020.

Krešimir Iličić

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Završni rad obrađuje koncept i primjenu programskih dodataka te detaljno pojašnjava proces izrade programskog dodatka i njegove objave u JetBrains repozitoriju. Kroz rad se pomoću nekoliko primjera prikazuju različite uporabe programskih dodataka. U radu se koriste dva razvojna okruženja od kojih je jedno Android Studio, a drugo IntelliJ IDEA. U IntelliJ IDEA se razvija programski dodatak, dok se u Android Studiju taj isti dodatak testira i koristi. Sadržaj je strukturiran na način da je prvo objašnjena svrha i uporaba programskih dodataka, zatim su kratko opisana razvojna okruženja koja se koriste u razvoju i testiranju dodatka, njihove razlike i sam razlog zašto su potrebna. Nakon toga su prikazana dva primjera programskih dodataka od kojih jedan predstavlja dodatak napravljen isključivo za pomoć u razvoju, a drugi predstavlja tzv. kozmetički dodatak koji može, ali ne mora direktno utjecati na razvoj aplikacije. Osnovni koraci u izradi programskog dodatka su prikazani i objašnjeni kroz gotov projekt te je u nastavku radu prikazan i razvoj vlastitog programskog dodatka. Nakon izrade vlastitog programskog dodatka prikazani su i koraci koji su potrebni kako bi se programski dodatak objavio na JetBrains repozitorij.

Ključne riječi: Razvojno okruženje, Android Studio, Programski dodatak, Tehnologije, Razvoj, Aplikacije, Repozitorij

Sadržaj

Sadržaj iii
1. Uvod 1
2. Programski dodatci 2
2.1. Općenito o programskim dodatcima 2
2.2. Razvojno okruženje Android Studio 5
2.3. Razvojno okruženje IntelliJ IDEA 6
2.4. Primjer programskog dodatka 8
2.4.1. Praktičan programski dodatak 8
2.4.2. Kozmetički programski dodatak 9
3. Razvoj programskog dodatka 11
3.1. Alati za razvoj programskog dodatka 11
3.1.1. IntelliJ IDEA 11
3.1.2. Android Studio 12
3.2. Postavljanje razvojnog okruženja IntelliJ IDEA 12
3.3. Osnovni proces kreiranja programskog dodatka 14
3.4. Implementacija funkcionalnosti programskog dodatka 27
4. Repozitorij 41
4.1. Dodavanje programskog dodatka u repozitorij 41
5. Zaključak 46
Popis literature 47
Popis slika 50
Popis programskih kodova 52
Prilozi 53

1. Uvod

Programiranje, odnosno izrada aplikacija često iziskuje puno vremena koje se može uštedjeti uz pomoć programskih dodataka. Proces izrade aplikacije je takav da se često iznova ponavljaju isti koraci i na taj način gubi vrijeme. Ponekad je to izrada specifične klase kad se izrađuje nova aplikacija sličnoga tipa kao prijašnja ili dodavanje programskog koda na specifični dio projekta npr. na početak svake klase. Jedno od rješenja za ubrzavanje procesa i izbjegavanje ponovnog pisanja istog koda je to da se kod kopira iz starih projekata. Takvo rješenje možda i je bolje od ručnog pisanja cijelog koda „od nule“, ali i dalje nije optimalno. Proces se može znatno ubrzati koristeći programske dodatke unutar Android Studija koji se nalaze na internetu kao već gotovo rješenje ili se izrađuje vlastiti programski dodatak kao što je u slučaju ovog rada.

Cilj ovog rada je objasniti i približiti uporabu programskih dodataka (eng. *Plugin*) unutar razvojnog okruženja Android Studija kako bi se prikazale neke od puno mogućnosti programskih dodataka. Prvo će biti objašnjen teoretski dio iza programskih dodatka i što su oni ustvari, u koje svrhe se koriste i kako se koriste. Nakon što se objasni teoretski dio, također će biti prikazan i praktični dio razvijanja jednog jednostavnog programskog dodatka kako bi se prikazao cijeli proces izrade. Nakon praktične izrade dodatka, bit će prikazani potrebni koraci kako bi se vlastiti programski dodatak objavio u JetBrains repozitorij.

Osobni razlozi i motivacija za odabir ove teme su ti da mi je tema zanimljiva zbog male „pokrivenosti“ informacija oko programskih dodatka, a ulogu igra i to što smatram da ih ljudi ne koriste dovoljno. Smatram da korištenjem određenih programskih dodataka možemo uštedjeti vrijeme i energiju te poboljšati fokus na programiranje. Teško je definirati glavnu ideju iza programskih dodataka jer su mogućnosti velike i postoji jako puno dodataka različitih svrha, od funkcionalnih do kozmetičkih.

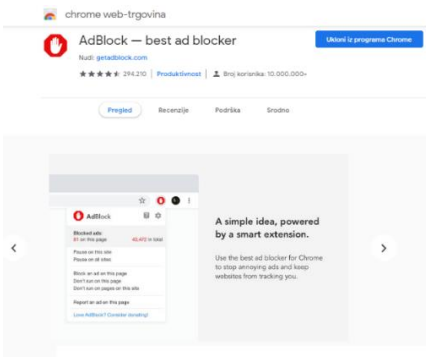
2. Programski dodatci

Kroz ovo poglavlje prikazati će se što je to programski dodatak i kako radi. Razvoj programskog dodatka uvijek započinje korištenjem odabranog razvojnog okruženja. Prvi dio poglavlja posvećen je pojašnjenju svrhe, odnosno uporabe programskih dodataka. U ovom poglavlju će također biti prikazana dva najčešće korištena razvojna okruženja za razvoj Android aplikacija i jedan od njih će biti korišten za razvoj programskog dodatka. Poveznica programskih dodataka i Android aplikacija je ta što programski dodatci mogu ubrzati proces razvoja upravo tih aplikacija kroz odabrano razvojno okruženje.

2.1. Općenito o programskim dodatcima

Programski dodatci su proširenje za razvojno sučelje i omogućavaju lakši rad prilikom razvijanja aplikacija. Konkretan primjer jednog ovakvog programskog dodatka će biti prikazan nešto kasnije u radu, pa će i uporaba dodataka biti jasnija. Kad se govori o programskim dodatcima djeluje kao da su isto što i biblioteka, no programski dodatci su ipak nešto drugo. Biblioteke koriste različite klase, metode i ostale resurse i može ih se upotrijebiti u više različitih projekata, ali uklanjanje jedne biblioteke korištene u pisanju programskog koda, može dovesti do toga da aplikacija više ne funkcionira.

Prema službenoj dokumentaciji Android Studija, biblioteke sadrže sve potrebno za izradu aplikacije, uključujući izvorni kod, razne resurse i sl. [1]. Programski dodatci su namijenjeni za specifične aplikacije koje oni proširuju, odnosno dodaju im nove mogućnosti. Uklanjanjem programskog dodatka samo se gubi to proširenje, ali aplikacija može normalno funkcionirati. Primjer koji bi većini ljudi vrlo lako objasnio koncept programskog dodatka je Adblock za Google Chrome koji sakriva sve oglase od krajnjeg korisnika. Slika 1. prikazuje Adblock na Internet trgovini programskih dodataka za preglednik Chrome.



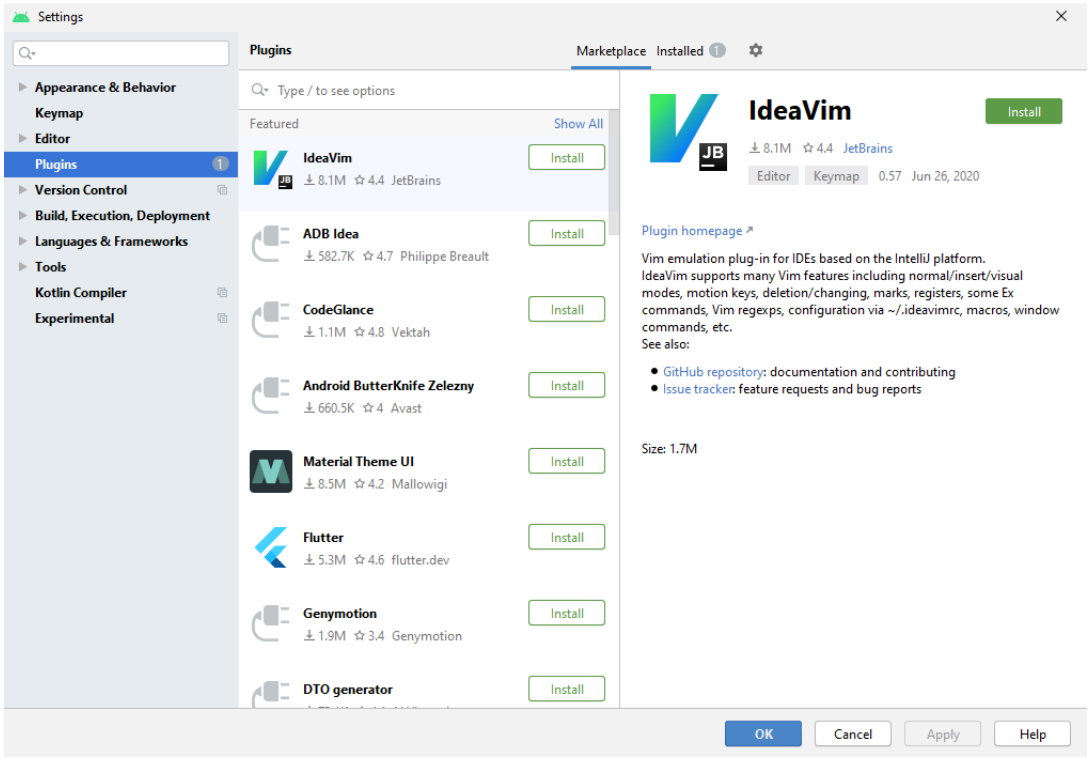
Slika 1. Adblock, Google Chrome internet trgovina

Iako ovaj dodatak nije vezan za Android Studio, dobar je primjer kako bi se lakše shvatio pojam programskog dodatka. Adblock je proširenje za Internet preglednik Google Chrome i uklanjanjem njega, Google Chrome i dalje može normalno funkcionirati. Razlog tome je sam način pisanja koda programskih dodataka koji se izrađuju tako da se ne stvara ovisnost glavne aplikacije o proširenju, već se njegova struktura određuje da se glavni dio umjesto modificiranja proširuje. Biblioteke u drugu ruku, sadrže različite metode koje se koriste prilikom pisanja programskog koda te brisanjem biblioteke korištena metoda više nije valjana i aplikacija neće raditi. Primjer koji najbolje prikazuje što je biblioteka ustvari je upotreba math.h biblioteke u programskom jeziku C. Bez ove biblioteke bi ručno pisali svoje matematičke metode za izračun korijena, potencije i slično, dok je uz pomoć navedene biblioteke dovoljno napisati naredbu `SQRT` i ona sama računa korijen iz danog broja. Ukoliko bi se uklonila spomenuta biblioteka, naredba `SQRT` bi postala nevažeća jer više nije dostupna.

Razvojna okruženja poput Android Studija imaju veoma sličan način dodavanja novih programskih dodataka u razvojno okruženje, a dodaju se tako da se otvore postavke Studija i pronađu postavke za programske dodatke, no nešto više o tome kasnije u radu. Programski dodatci su jako rašireni i postoje razne upotrebe poput automatizacije ponavljajućih procesa, stvaranja prečaca, alarma nakon završenog procesa izrade, ali postoje i kozmetički dodatci koji npr. mijenjaju boje zagrada u dugine boje što će biti prikazano u kasnijem primjeru.

Teško je pokriti što se sve može postići s programskim dodatcima jer broj dostupnih dodataka raste iz dana u dan kao što se vidi na službenom JetBrains repozitoriju gdje je u vrijeme pisanja ovog rada dostupno 4251 programski dodatak za Android Studio. Gradle kojeg Android koristi kako bi olakšao kompiliranje (eng. *Compile*) raznih Android projekata to radi upravo kroz različite programske dodatke jer on sam po sebi nema puno mogućnosti. Jedan od primjera gdje Gradle koristi programske dodatke je i kako službena dokumentacija navodi, sposobnost da Gradle kompilira Java programski kod [2].

Programske dodatke se može pronaći na jednostavan način tako da se ide putanjom „File“, zatim „Settings“ i konačno „Plugins“. Gornja kartica sadrži Marketplace kao što se vidi na slici 2. gdje se nalazi jako puno različitih programskih dodataka što govori koliko su oni dostupni i rašireni u programerskoj zajednici.



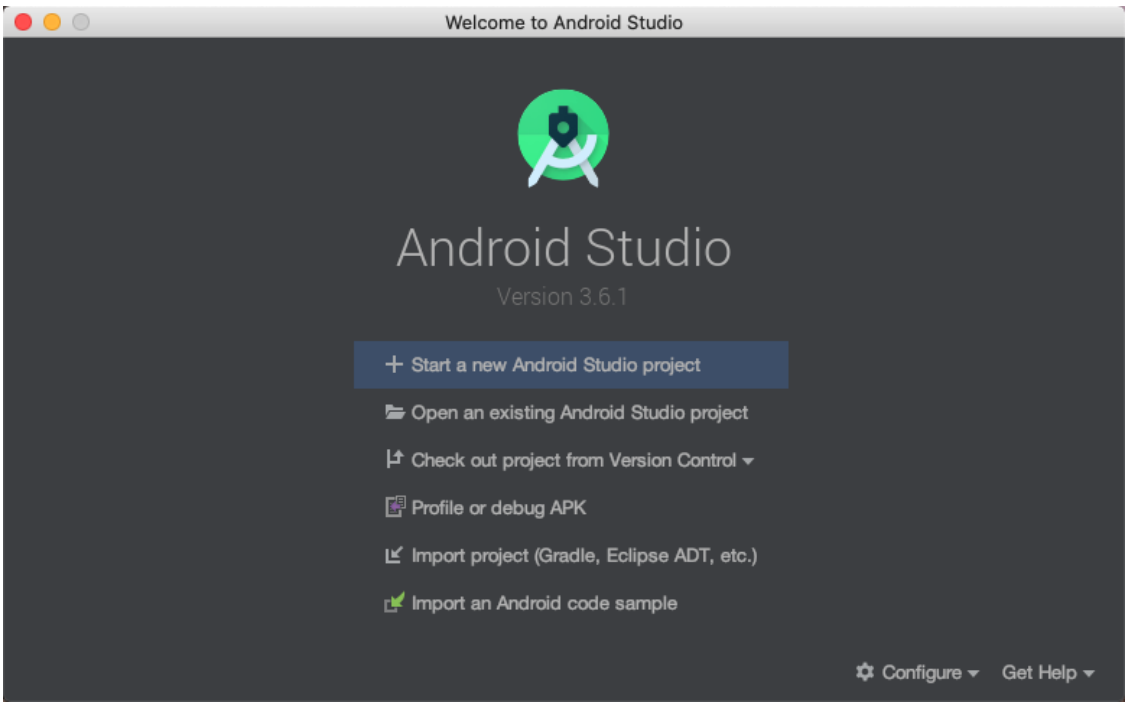
Slika 2. Plugins, Android Studio

Kao i kod drugih alata poput Google Chrome-a, tako i za Android Studio postoje besplatni dodatci i oni koje je potrebno platiti. Primjer je jedan programski dodatak zvan "Auto Java Code Suggestion" dostupan na Internet trgovini tvoraca IntelliJ IDEA razvojnog okruženja [3]. Ovaj programski dodatak olakšava pisanje Java koda unutar Android Studija ili sličnog razvojnog okruženja. Funkcionira tako da kad je napisana ključna riječ, on ponudi konkretan kod kako ga ne bi sami pisali. Autor ovog programskog dodatka je odlučio da cijena iznosi 1.9$ za svakoga tko ga želi koristiti u razvojnom okruženju.

Svi programski dodatci objavljeni na ovoj trgovini su prvo pregledani i mora ih odobriti JetBrains, odnosno tvorci IntelliJ IDEA razvojnog okruženja. Svatko može objaviti svoj programski dodatak i staviti cijenu na njega, ali se licenciranje dogovara sa JetBrainsom. Osim onih koje je potrebno platiti, repozitorij sadrži i puno besplatnih programskih dodataka koji se također mogu pronaći na njihovoj trgovini, a također je moguće preuzeti dodatke i s drugih alternativnih stranica.

2.2. Razvojno okruženje Android Studio

Android Studio je razvio Google uz pomoć JetBrainsa i ono je službeno razvojno okruženje koje se može koristiti kako bi razvili aplikacije za razne Android uređaje. Razvijen je na temelju IntelliJ IDEA razvojnog okruženja, a kasnije su ostvarili i suradnju kako navodi službena dokumentacija Android Studija [4]. Dostupan je za instalaciju na operacijskim sustavima Windows, MacOS i Linux. Svaka izmjena dodana u Android Studio je dostupna i u IntelliJ IDEA jer koriste zajednički format projekata što znači da se projekti napravljeni unutar Android Studija mogu otvoriti i u IntelliJ IDEA. Obrnuta situacija je nešto drugačija, jer ako se unutar Android Studija otvara projekt koji je stvoren u drugom razvojnom okruženju poput IntelliJ IDEA, on mora biti napisan na temelju Gradlea. Projekti koji ne koriste Gradle se moraju pripremiti tako da se neke datoteke ručno izrade te određeni dio koda kopira kako bi se projekt mogao koristiti kao što je navedeno u službenoj dokumentaciji Android Studija [5]. Na slici 3. se vidi kako izgleda početni zaslon nakon što se pokrene Android Studio.



Slika 3. Android Studio početni zaslon [6]

Osim programskog jezika Java, 17.05.2017 Google dodaje potporu u Android Studio za još jedan programski jezik zvan Kotlin što objavljuju i na službenoj stranici JetBrainsa [7]. Nešto poslije toga postati Kotlin je postao i preferirani jezik za razvoj Android aplikacija u Android Studiju. Ova dva jezika se mogu koristiti kako za razvoj Android aplikacija, tako i

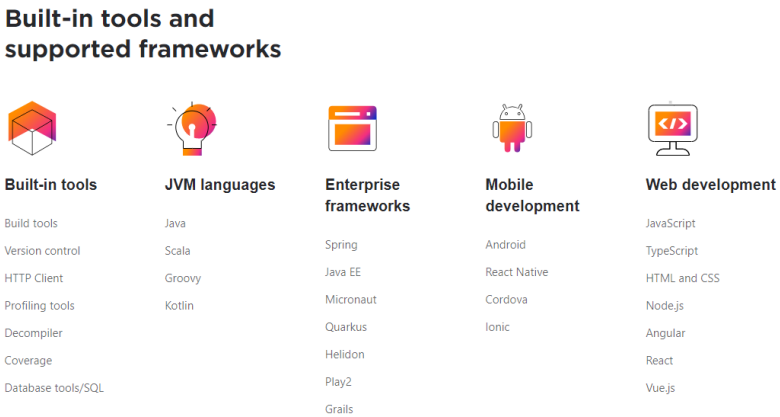
Android programskih dodataka, a čak je moguće koristiti i mix ova dva jezika. Upravo u Kotlinu će biti napisan programski dodatak za koji će biti prikazani koraci nešto kasnije u nastavku rada.

Odabir razvojnog okruženja za razvoj Android aplikacija je preferencija korisnika. Najčešće, radno mjesto na kojem se nalazimo zahtjeva korištenje istog razvojnog okruženja kako bi svi bili usklađeni. Nakon toga, korisnik je naviknut na korištenje jednog razvojnog okruženja npr. Android Studija i stječe naviku korištenja istog. Između Android Studija i IntelliJ IDEA razvojnog okruženja ne postoje bitne razlike u smislu dosega mogućnosti za razvoj Android aplikacija, ali one postoje u razvoju programskih dodataka.

Oba razvojna okruženja mogu izvršavati jednako kompliciranе procese, razlika je što IntelliJ IDEA osim za Android ima mogućnost za razvoj mnogo drugih projekata kao npr. projekata koji koriste Angular ili Scalu. Trenutno ne postoje relevantna istraživanja o brzini pojedinih razvojnih okruženja, ali iz osobnog iskustva mogu reći da ne vidim razliku prilikom razvoja Android aplikacija već su jednako brzi. U ovom radu će biti korišten IntelliJ IDEA za razvoj programskog dodatka i Android Studio za testiranje istog, a kasnije u radu će biti objašnjeno i zašto se koristi IntelliJ IDEA za razvoj programskog dodatka umjesto Android Studija.

2.3. Razvojno okruženje IntelliJ IDEA

IntelliJ IDEA je uz Android Studio još jedno razvojno okruženje u kojem se mogu izrađivati aplikacije za Android. Osim aplikacija, također služi i za pisanje programskih dodataka za Android Studio. No, za razliku od Android Studija, IntelliJ IDEA pokriva puno više alata, programskih jezika i platformi kao što se vidi na slici 4.



Slika 4. Izgled početne stranice IntelliJ IDEA [1]

[1] Slika zaslona na: https://www.jetbrains.com/idea/

S obzirom na to da je tema ovog rada razvoj programskog dodatka za Android Studio, taj dio razvojne mogućnosti je jedino i bitan. Upravo ovo razvojno okruženje je bilo temelj za razvoj Android Studija, te su Android Studio i programski dodatak Android za IntelliJ IDEA nastali iz istog programskog koda [8]. Kao što je prijašnje objašnjeno, IntelliJ IDEA može učitavati projekte iz Android Studija i moguće je normalno nastaviti raditi na njima. Također, prijašnje spomenuti programski jezik Kotlin su razvili upravo ljudi odgovori i za ovo razvojno okruženje, odnosno JetBrains [9], pa je logično zaključiti da je primarni programski jezik za razvoj Android aplikacija Kotlin.

Na slici 5. vidimo da je čak i početni zaslon IntelliJ IDEA jako sličan početnom zaslonu Android Studija. Netko tko se naviknuo na Android Studio ne bi trebao imati problema s ovim razvojnim okruženjem, jer je svaki segment, pa čak i raspored alata veoma sličan Android Studiju.



Slika 5. Početni zaslon IntelliJ IDEA

IntelliJ IDEA dolazi u dvije različite inačice, Community inačica koja je besplatna i Ultimate koja se plaća. Ultimate inačica sadrži dodatne alate koji olakšavaju programiranje u različitim programskim jezicima, ali i pruža više mogućnosti pa tako i pisanje JavaScripta koji nije dostupan u Community inačici [10]. Ultimate je više orijentiran za razvoj mrežnih aplikacija i alata za poduzeća, dok je Community inačica više za Java virtualni stroj (eng. *Java Virtual Machine - JVM)* i Android. Većini studenata, ali i ostalim programerima će ova besplatna inačica biti sasvim dovoljna za razvoj željenih aplikacija ako je cilj Android programiranje. Naknadna nadogradnja na inačicu koja se plaća, odnosno Ultimate je uvijek moguća bez gubitka prethodnih radova.

U ovom radu bit će korištena Community inačica, prvenstveno zato što je besplatna, otvorenog koda, ali i zato što sadrži sve potrebno kako bi se razvio programski dodatak za Android Studio. Većina mogućnosti u IntelliJ IDEA razvojnom okruženju su upravo nastali kroz programske dodatke koji se po potrebi omogućuju ili onemogućuju. S obzirom na to da je Android Studio najviše orijentiran na razvijanje Android aplikacija, ipak postoji više literature za razvoj programskih dodataka u IntelliJ IDEA razvojnom okruženju. Izgled samog sučelja i raspored alatnih traka te izbornika je gotovo identičan pa snalaženje u samom okruženju ne bi trebalo predstavljati problem ukoliko programer prelazi sa Android Studija. Prednost ovog okruženja je u postojanju službenog JetBrains repozitorija u kojem se nalazi puno programskih dodataka koji mogu biti veoma korisni ne samo za Android već i razvoj drugih aplikacija.

2.4. Primjer programskog dodatka

U ovom dijelu poglavlja će se prikazati konkretni primjeri programskih dodataka koji se koriste unutar Android Studija. Programski dodatci osim što olakšavaju rad, mogu poslužiti i u razne svrhe, poput vizualne izmjene nekog dijela Android Studija ili jednostavne pozdravne poruke prilikom učitavanja Android Studija. U nastavku ovog poglavlja će biti prikazan „koristan“ programski dodatak koji olakšava rad unutar Android Studija i jedan „kozmetički“ koji nam zapravo ne mora, ali može povećati učinkovitost.

2.4.1. Praktičan programski dodatak

Primjer programskog dodatka koji može biti iznimno koristan prilikom razvoja raznih aplikacija je Gradle Killer. Gradle Killer je programski dodatak koji može odabirom jednog gumba prekinuti Gradle proces izrade [11]. Većina korisnika koja koristi Android Studio se barem jednom susrela sa situacijom gdje su slučajno pokrenuli kompajler bez mogućnosti da zaustave proces. Ovakva situacija može biti naporna, pogotovo ako je računalo korisnika lošije konfiguracije, jer proces izrade može potrajati i do nekoliko minuta.

Situacija poput navedene se može izbjeći korištenjem Gradle Killera koji može zaustaviti proces jednim klikom. Kad ne bi koristili ovaj programski dodatak, jedini načini da se zaustavi Gradle proces bi bio pričekati da završi proces, što može potrajati i nekoliko minuta ovisno o veličini projekta i konfiguraciji računala, ugasiti cijeli Android Studio i ponovno ga pokrenuti ili ručno ugasiti proces putem upravitelja zadataka. Ukoliko bi projekt bio ručno ugašen putem upravitelja zadataka, postoji opasnost da se projekt ošteti ako je zaustavljen krivi proces i zato je bolje koristiti programski dodatak da zaustavi proces.

Na slici 6. je vidljivo kako izgleda ovaj programski dodatak nakon što se doda u Android Studio. Neki od programskih dodataka poput ovoga, nakon dodavanja u Android Studio

prikažu svoju ikonu u alatnoj traci, ovisno o svrsi programskog dodatka i načinu izrade. Također je moguće i da se programski dodatak prikaže u obliku pozdravne poruke, u bočnoj traci ili na nekom drugom mjestu u Android Studiju, ovisno o definiranoj poziciji unutar vlastitog koda. Kao što je već navedeno, pritiskom na gumb „Kill Gradle“ zaustavlja se Gradle proces izrade i odmah se može nastaviti s daljnjim radom.



Slika 6. Izgled programskog dodatka Gradle Killer [12]

2.4.2.Kozmetički programski dodatak

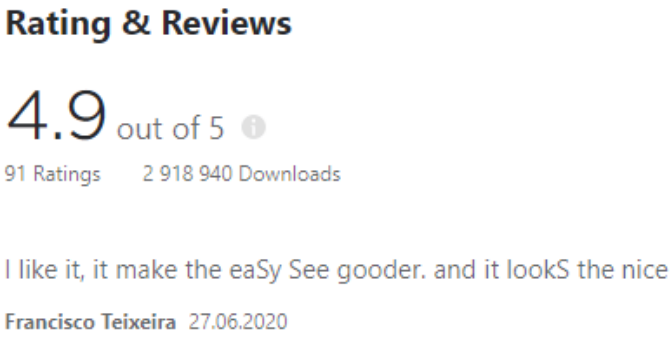
Kao primjer kozmetičkog programskog dodatka odabran je Rainbow Brackets koji je dostupan besplatno na Internet trgovini JetBrainsa [13]. Ovaj jednostavni programski dodatak ima svrhu promjene boje svakoj od zagrada koje se napišu, odnosno svaki programski blok ima istu boju na otvorenoj i zatvorenoj zagradi. Tip zagrada kojima se mijenja boja su oble, uglate i vitičaste kao što se vidi na slici 7. Nije bitno piše li korisnik Javu ili Kotlin, ovaj programski dodatak će raditi u oba slučaja. Ako prva otvorena zagrada ima plavu boju, onda će i njezin par, odnosno zatvorena zagrada koja zatvara programski blok također biti plava. Sljedeći par otvorene i zatvorene zagrade će poprimiti drugu boju npr. žutu i tako je programski kod puno pregledniji.



Slika 7. Primjer dodatka Rainbow Brackets [14]

Iako je naziv ovog poglavlja kozmetički programski dodatak, Rainbow Brackets bi se čak mogao svrstati i u praktične jer pomaže prilikom snalaženja u programskom kodu. Na temelju Rainbow Bracketsa moguće je vidjeti da programski dodatak ne mora imati kompliciranu funkciju da bi olakšao pisanje koda i snalaženje u istom. Na slici 7. vidljivo je kako se zagrade vizualno mijenjaju da bi se olakšalo snalaženje. Za razliku od klasičnih dodataka, ovaj programski dodatak nije dodao potpuno novu funkciju, već samo izmijenio postojeću boju teksta, odnosno postojeću boju teksta zadanu u samom razvojnom okruženju.

Na slici 8. vidljivo je da ovaj programski dodatak ima velikih 2 918 940 preuzimanja što je dobar statistički pokazatelj kako programski dodatak ne mora imati kompleksnu funkcionalnost da bi bio koristan za programere. Većina korisnika voli vizualno prilagoditi razvojno sučelje jer im je tako lakše, zabavnije i preglednije. Može se reći da sve dok programski dodatak olakšava razvoj aplikacije, može se smatrati korisnim.



Slika 8. Broj preuzimanja Rainbow Bracketsa [2]

[2] Slika zaslona na: https://plugins.jetbrains.com/plugin/10080-rainbow-brackets

3. Razvoj programskog dodatka

Kroz ovo poglavlje će biti prikazan razvoj programskog dodatka za Android Studio. Najprije će biti nabrojani i ukratko objašnjeni potrebni alati, zatim predstavljeno razvojno okruženje u kojem će se razvijati programski dodatak te konačno prikazan i sam proces razvoja dodatka i njegova upotreba.

Sam proces razvoja programskog dodatka nije kompliciran, ali težina razvoja je u nedostatku literature za razvoj programskih dodataka za Android Studio. Literatura dostupna za razvoj programskih dodataka je prisutna većinom samo u tehničkoj dokumentaciji razvojnih okruženja [15] te na blogovima raznih programera. Razvoj je također prikazan i u knjizi naziva Expert Android Studio [16], ali nije prilagođen početnicima i prikazan je samo najosnovniji oblik programskog dodatka.

3.1. Alati za razvoj programskog dodatka

Alati potrebni za razvoj programskog dodatka za Android Studio su IntelliJ IDEA razvojno okruženje i sam Android Studio. IntelliJ IDEA se koristi kako bi se razvio programski dodatak, dok se Android Studio koristi kako bi se testirao programski dodatak i koristio. Svaki od ova dva alata dolazi u više inačica i stalno se radi na njihovom poboljšanju i dodavanju novih mogućnosti.

3.1.1.IntelliJ IDEA

IntelliJ IDEA razvojno okruženje dolazi u dvije inačice koje su Community i Ultimate. U slučaju razvoja programskog dodatka u ovom radu korištena je Community inačica zato što je besplatna i pruža sve što je potrebno da se razvije programski dodatak. Bitno je napomenuti da se u većini slučajeva koristi upravo IntelliJ IDEA kao razvojno okruženje za razvoj programskog dodatka za Android Studio. Razlog je taj što Android Studio iako je nastao na temelju IntelliJ IDEA, nije nastao na cijelom, već samo na dijelu vezanom uz Android. Zbog ovih razloga, nedostaju mu određene mogućnosti kako bi korisnici unutar Android Studija mogli razvijati programske dodatke, što ne znači da se u budućnosti neće proširiti mogućnosti razvoja unutar Android Studija.

Jedini način da se programski dodatak razvije u Android Studiju je da se ručno napravi build.gradle i plugin.xml datoteke koje će biti objašnjene nešto kasnije u radu. Osim toga, potrebno bi se bilo ručno povezati s Java klasama IntelliJ IDEA platforme te bi se testiranje

programskog dodatka svejedno odvijalo kroz simulaciju IntelliJ IDEA razvojnog okruženja. Da bi se izbjegle komplikacije, puno je bolje koristiti razvojno okruženje koje sadrži sve potrebno za razvoj programskog dodatka, a također je činjenica da za IntelliJ IDEA postoji i tehnička dokumentacija koja objašnjava cijeli proces razvoja dodatka, dok za Android Studio nema ništa slično.

3.1.2.Android Studio

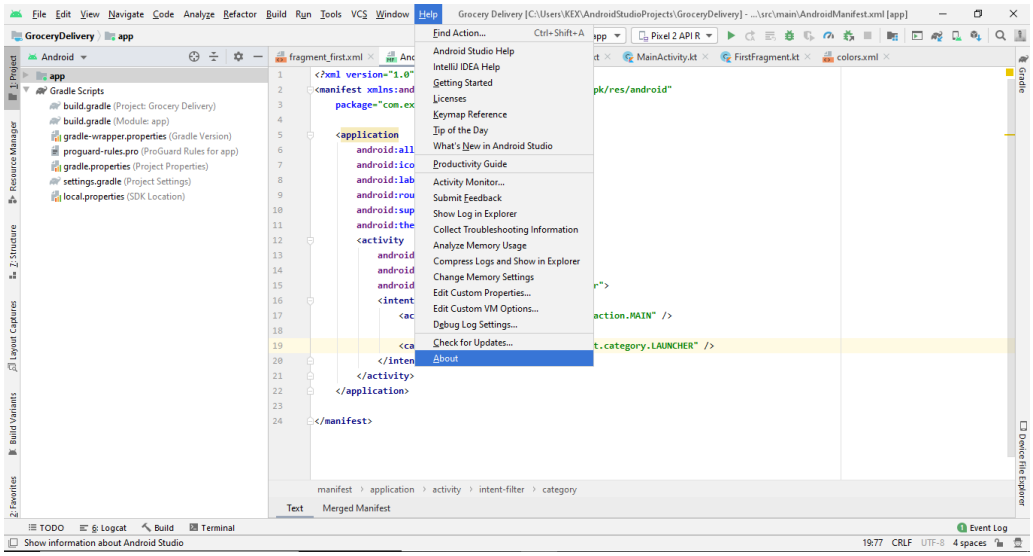
Osim IntelliJ IDEA, potreban je i sam Android Studio za koji će biti razvijen programski dodatak. Iako se testiranje dodatka može izvršiti i unutar IntelliJ IDEA, krajnji cilj je ipak razviti dodatak za Android Studio pa da bi se izbjegli mogući problemi preporučljivo je povremeno ili kompletno testiranje napraviti u Android Studiju.

Obzirom da inačice Android Studija izlaze nakon IntelliJ IDEA, postoji mogućnost nekompatibilnosti. Klasičan primjer ovakvog problema bi bila upotreba novih klasa ili bilo kojih mogućnosti koje su došle u novoj inačici IntelliJ IDEA okruženja, a da Android Studio još nije uvrstio te promjene u svoje razvojno okruženje. Kad bi razvijali programski dodatak sa tim novim mogućnostima unutar IntelliJ IDEA okruženja, nakon dodavanja u Android Studio prikazala bi se greška, jer Android Studio ne bi prepoznao npr. novu korištenu klasu unutar koda programskog dodatka. Iz ovih razloga preporučljivo je koristiti nešto stariju inačicu IntelliJ IDEA razvojnog okruženja za koju postoji ravnopravna inačica Android Studija.

3.2. Postavljanje razvojnog okruženja IntelliJ IDEA

Prijašnje spomenuti problem nekompatibilnosti se može izbjeći na način da se koriste jednake inačice Android Studija i IntelliJ IDEA razvojnog okruženja. Postoje 2 načina da se inačice lako usklade, ovisno koje razvojno okruženje već imamo instalirano. U ovom slučaju, Android Studio je već bio instaliran na računalo pa će se krenuti od njegove inačice.

Prvi korak se sastoji od toga da se pokrene Android Studio te se zatim u izborničkoj traci odabere „Help“ i u padajućem izborniku se odabere „About“ kao što se vidi na slici 9.



Slika 9. Provjera inačice Android Studija

Nakon što se odabere „About“, pojaviti će se prikaz kao na slici 10. Prikaz sadrži najosnovnije informacije vezane uz trenutnu inačicu Android Studija.



Slika 10. Prikaz detalja o inačici Android Studija

Ono što je bitno iz ovog prikaza je inačica trenutno instaliranog Android Studija koja je 192.7142.36.36.6308749. Sam broj ne govori ništa što bi pomoglo u određivanju ravnopravne inačice s obzirom na to da su nazivi inačica IntelliJ IDEA npr. 2018.2.8, ali zapravo je 192 naziv grane na kojoj je pojedina inačica IntelliJ IDEA objavljena, u ovom slučaju je to 2019.2 kao što vidimo na službenim stranicama inačica IntelliJ IDEA [17].

S obzirom na to da IntelliJ IDEA 2019.2 ima 2 različite inačice, odnosno 2019.2.3 i 2019.2.4, potrebno ih je usporediti sa prijašnje navedenom punom inačicom Android Studija i

na temelju toga dodatno usporediti sa oba dvije inačice na stranici IntelliJ IDEA okruženja [18]. Na slici 11. vidimo kako je tražena inačica IntelliJ IDEA 2019.2.4 jer odgovara inačici Android Studija 192.7142.36, dok 2019.2.3 odgovara inačici Android Studija 192.6817.14. U pitanju su jako male razlike između ove dvije inačice, no kako bi bili sigurni da će sve biti usklađeno i raditi u oba razvojna okruženja, sigurnije je staviti potpuno ravnopravne.



Slika 11. Odgovarajuća inačica IntelliJ IDEA[3]

Kao što je navedeno na početku, postoji i još jedan način za uskladiti inačice a to je slučaj gdje je prvo instaliran IntelliJ IDEA. Zapravo je način usklađivanja isti, samo je obrnuti postupak. Pomoću konkretne inačice IntelliJ IDEA se potraži koja je to inačica Android Studija na istoj stranici koju slika 11. prikazuje, odnosno na dijelu internet stranice od IntelliJ IDEA gdje se vidi popis inačica. Kako bi se provjerila instalirana inačica IntelliJ IDEA na vlastitom računalu, potrebno je pokrenuti IntelliJ IDEA i pogledati natpis na početnom zaslonu razvojnog okruženja kao što se vidi na slici 5. gdje je inačica 2020.1.2. ili pokrenuti razvojno okruženje i na isti način kao kod Android Studija, pronaći inačicu u postavkama.

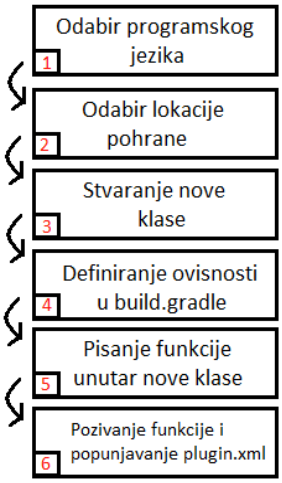
Nakon što su inačice usklađene, sljedeći korak je sam razvoj programskog dodatka. Programski dodatak se razvija unutar razvojnog okruženja IntelliJ IDEA kao što je to ranije spomenuto u radu, najviše iz razloga što ovo razvojno okruženje sadrži već gotove alate i predloške poput plugin.xml i build.gradle datoteka koje olakšavaju pisanje i dokumentiranje samog programskog dodatka. U sljedećem poglavlju rada na gotovom projektu će biti prikazani svi potrebni koraci da se izradi programski dodatak.

3.3. Osnovni proces kreiranja programskog dodatka

U nastavku ovog poglavlja bit će prikazan primjer osnovnog procesa kreiranja programskog dodatka, dok će detalji i objašnjenje biti prikazani prilikom razvoja vlastitog

[3] Slika zaslona na: https://www.jetbrains.com/idea/download/other.html

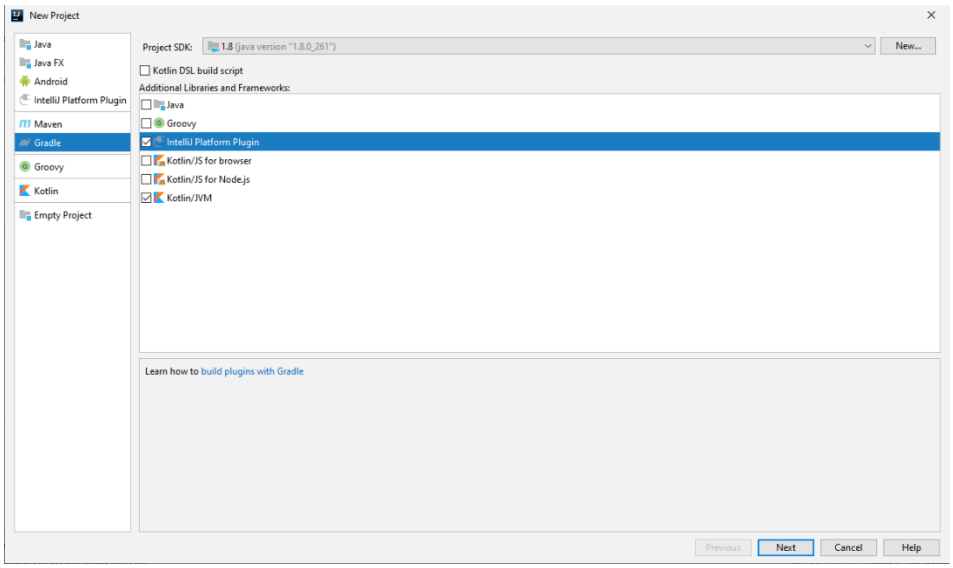
programskog dodatka u poglavlju 3.4. Proces započinje odabirom razvojnog sučelja koje će se koristiti za razvoj programskog dodatka, u ovom slučaju IntelliJ IDEA. Zatim, kao što dijagram na slici 12. prikazuje, odabiru se željeni programski jezici u kojima će se programski dodatak razvijati. Dijagram je izrađen u alatu Paint dostupnom besplatno uz operacijski sustav Windows 10. Nakon odabranih jezika zadaje se lokacija pohrane i pokreće razvojno okruženje.



Slika 12. Dijagram nastajanja programskog dodatka u IntelliJ IDEA

Razvoj se zatim nastavlja stvaranjem nove klase u kojoj će se nalaziti funkcionalnost programskog dodatka. Zadaju se sve ovisnosti (eng. *Dependency*) unutar build.gradle datoteke te se piše funkcija unutar nove klase koju konfiguracijska datoteka plugin.xml poziva pomoću akcija (eng. *Actions*). Nakon što su svi koraci završeni, programski dodatak se može testirati i ako je sve uredu, izraditi. U nastavku rada će se na primjeru gotovog projekta prikazati svi ovi koraci i spomenute datoteke.

Razvoj programskog dodatka započinje tako da se pokrene razvojno sučelje IntelliJ IDEA i odabere opcija novi projekt (eng. *New Project*). Ova opcija otvara novi prozor u kojem je potrebno odabrati tip projekta i programske jezike koji se koriste prilikom rada kao što se vidi na slici 13.

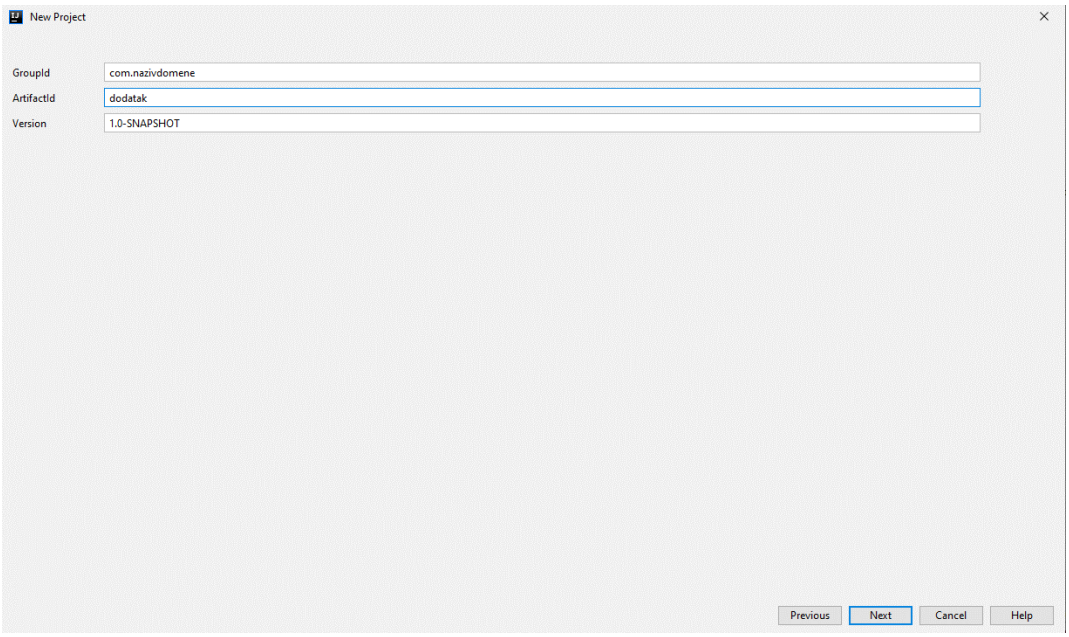


Slika 13. Novi projekt

Za potrebe pisanja ovog programskog dodatka odabire se Gradle koji služi kako bi se automatizirao proces izrade aplikacije ili programskog dodatka kao u ovom slučaju. S obzirom na to da se Gradle koristi i u Android Studiju, odabire se rad uz njegovu podršku i odabire se programski dodatak za platformu IntelliJ (eng. *IntelliJ Platform Plugin*) koji sadrži dodatne biblioteke i okvire (eng. *Framework*) kao što i piše pri vrhu. Programski jezik u kojem će dodatak biti napisan je prijašnje spomenuti Kotlin.

Jedan od razloga odabira ovog programskog jezika je što ga je razvio JetBrains, a također je dostupno i više literature u kojoj se koristi upravo ovaj jezik. Ukoliko se netko odluči za razvoj u Javi umjesto Kotlina, to nije problem jer ova dva jezika se mogu međusobno konvertirati uz pomoć samog razvojnog okruženja. Na službenim stranicama JetBrainsa je objašnjen cijeli proces pretvorbe pa čak i korištenje oba jezika u jednom projektu [19].

Sljedeći korak u razvoju programskog dodatka je definiranje tri stavke projekta kao što je prikazano na slici 14. Prva stavka je *GroupId* koja je kao i ostale dvije stavke nastala na zajedničkoj konvenciji o imenovanju kojih se pridržavaju i Gradle i Maven. Maven je veoma sličan Gradleu, ali više korisnika ipak koristi Gradle za razvoj Android aplikacija, vjerojatno zato što je puno brži kako je i studija slučaja koju je Gradle proveo pokazala [20]. Obzirom da Gradle ne sadrži opis značenja ovih stavki, koristiti će se službena dokumentacija Mavena.

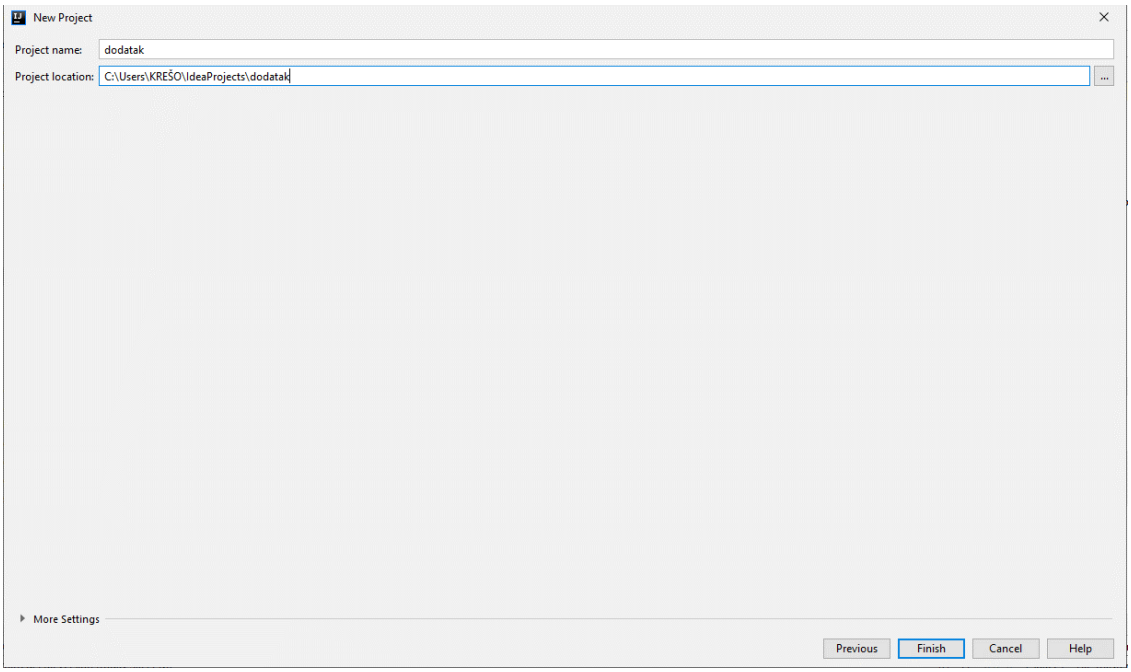


Slika 14. Opisne stavke projekta

Groupid jedinstveno označava tko stoji iza projekta, ukoliko projekt nije samo za lokalnu upotrebu onda se najčešće stavlja obrnuti naziv vlastite domene npr. com.nazivdomene, često grupacije i poduzeća koja stoje iza programskog dodatka koriste vlastitu domenu poput com.apple kako bi lakše opisali vlasnike iza projekta. Može sadržavati beskonačno podgrupa koje možemo koristiti ukoliko planiramo imati više modula za taj projekt npr. com.nazivdomene.auto.navigacija. Ovo nije definirano kao pravilo, može se koristiti i jedna riječ, ali je puno teže dobiti dozvolu za objavu u repozitorij ako opisne stavke nisu definirane po pravilima kao što stoji u dokumentaciji [21].

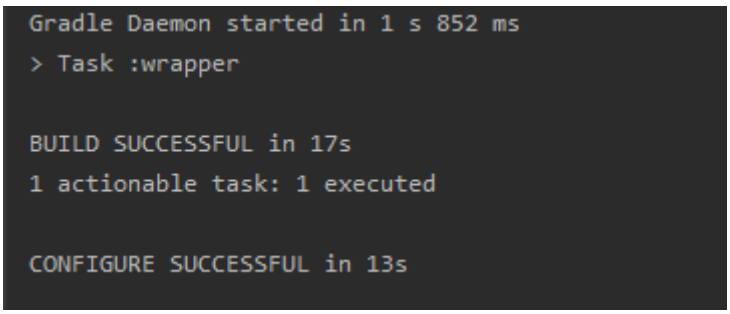
Preostale dvije stavke *artifactid* te inačicu (eng. *Version*) također možemo pronaći objašnjenu u toj istoj službenoj dokumentaciji. *Artifactid* označava jedinstveni naziv projekta, pa bi tako npr. odabrani naziv projekta „dodatak“ u konačnici bio com.nazivdomene.dodatak a kad bi postojao još neki zaseban projekt npr. naziva auto, njegov id bi bio com.nazivdomene.auto. Također je moguće ići i još razina ispod ukoliko se radi nadogradnja na već postojeći projekt, ali u svrhe rada bit će korištena samo osnovna razina, odnosno konačni id će biti com.nazivdomene.dodatak. Version označava inačicu projekta koji se kreira, u ovom slučaju se može staviti 1.0 jer je ovo prva inačica programskog dodatka.

Nakon što su sve stavke ispunjene, sljedeći korak koji je vidljiv na slici 15. zahtjeva definiranje lokalnog naziva projekta i lokaciju gdje će se pohraniti datoteke vezane uz projekt. Za razliku od Android Studija, IntelliJ IDEA nema problem sa nazivom mape koji sadrži slova nedostupna u engleskom jeziku poput „š“ u mapi „KREŠO“ gdje će biti pohranjen ovaj programski dodatak kao što je vidljivo na slici 15.



Slika 15. Odabir lokacije pohrane projekta

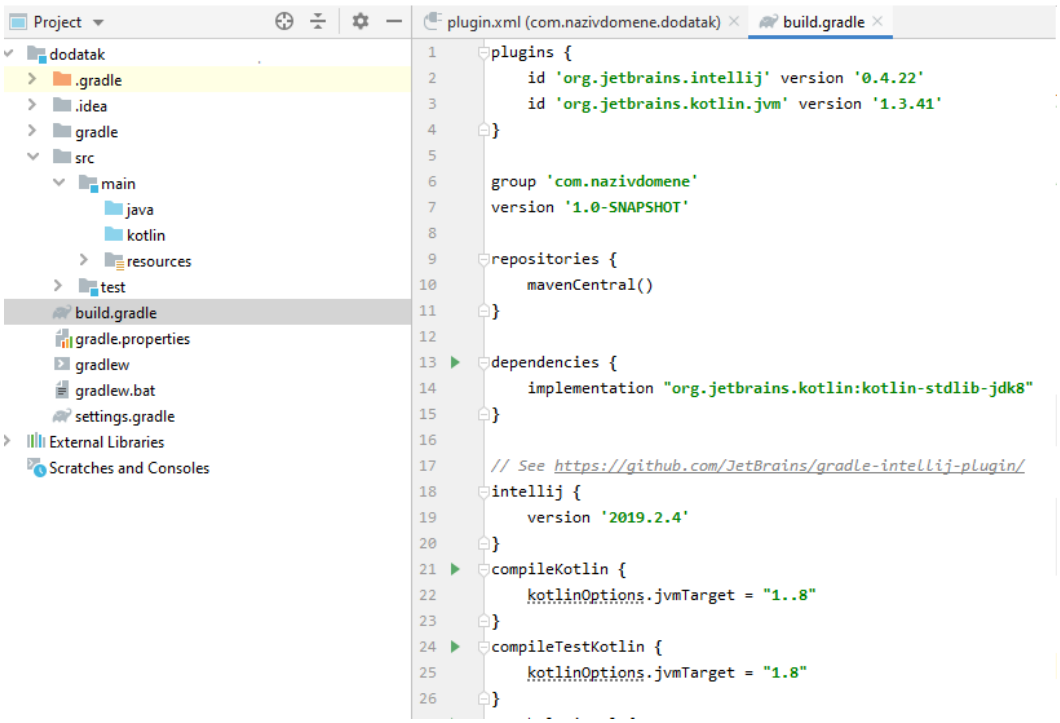
Kad je ovaj korak završen, pritiskom na gumb završi (eng. *Finish*) se kreira projekt i započinje Gradle proces izrade projekta. S obzirom na to da je u prijašnjim koracima odabran IntelliJ Platform Plugin i Kotlin kao jezik, struktura projekta će automatski biti prilagođena tim odabirima pa tako se stvara datoteka plugin.xml i build.gradle koje su jako bitne za daljnji razvoj programskog dodatka. Nakon pritisak na gumb završi, potrebno je sačekati da Gradle završi svoj proces izrade svih datoteka kako bi se mogao nastaviti daljnji rad u projektu. Na slici 16. je vidljivo kako izgleda kad se ovaj proces uspješno završi.



Slika 16. Gradle proces izrade

Nakon što je proces izrade završio pojavljuju se potrebne datoteke i mape u strukturi projekta kao što je vidljivo na slici 17. Struktura se nalazi sa lijeve strane i sadrži sve datoteke i mape koje sadrži ovaj projekt, moguće je dodavati i brisati nove datoteke, mape, klase i sl.

jednostavnim pritiskom desnog klika miša na mjesto gdje ih se želi dodati i zatim odabrati željenu funkciju. Početne datoteke koje se generiraju su već spomenuti build.gradle i plugin.xml te ih na slici 17. možemo vidjeti otvorene u uređivaču koda. Prema službenoj dokumentaciji Android Studija, build.gradle datoteka definira konfiguraciju procesa izrade koja se primjenjuje na sve module unutar projekta na kojem se radi [22].



Slika 17. Početna struktura projekta

Unutar build.gradle datoteke se definira koje repozitorije projekt, odnosno Gradle koristi te se definiraju ovisnosti koje biblioteke i programski dodatci u projektu potražuju. Kao primjer se na slici 17. vidi ovisnost `org.jetbrains.kotlin:kotlin-stdlib-jdk8` koja je ustvari proširenje na klasičnu Kotlin biblioteku i sadrži dodatne mogućnosti razvojnog kompleta jave (eng. *Java Development Kit*). Ovu ovisnost Gradle automatski dodaje u build.datoteku jer je potrebna za razvoj programskog dodatka.

Kroz build.gradle se može upravljati i inačicama Jave i Kotlina koji se koriste prilikom razvoja samog projekta, također se u polje ovisnosti i repozitorija mogu dodavati dodatne biblioteke i sl. koje se planiraju koristiti u projektu. Potrebno je pripaziti da korištena inačica IntelliJ IDEA razvojnog okruženja podržava dodatnu biblioteku odnosno repozitorij koji se dodaje u build.gradle. Sljedeća datoteka koja je bitna za razvoj programskog dodatka je plugin.xml koji je više puta spomenut u radu. Slika 18. prikazuje kako izgleda sadržaj ove datoteke i zašto je on bitan za razvoj programskog dodatka.



Slika 18. Plugin.xml

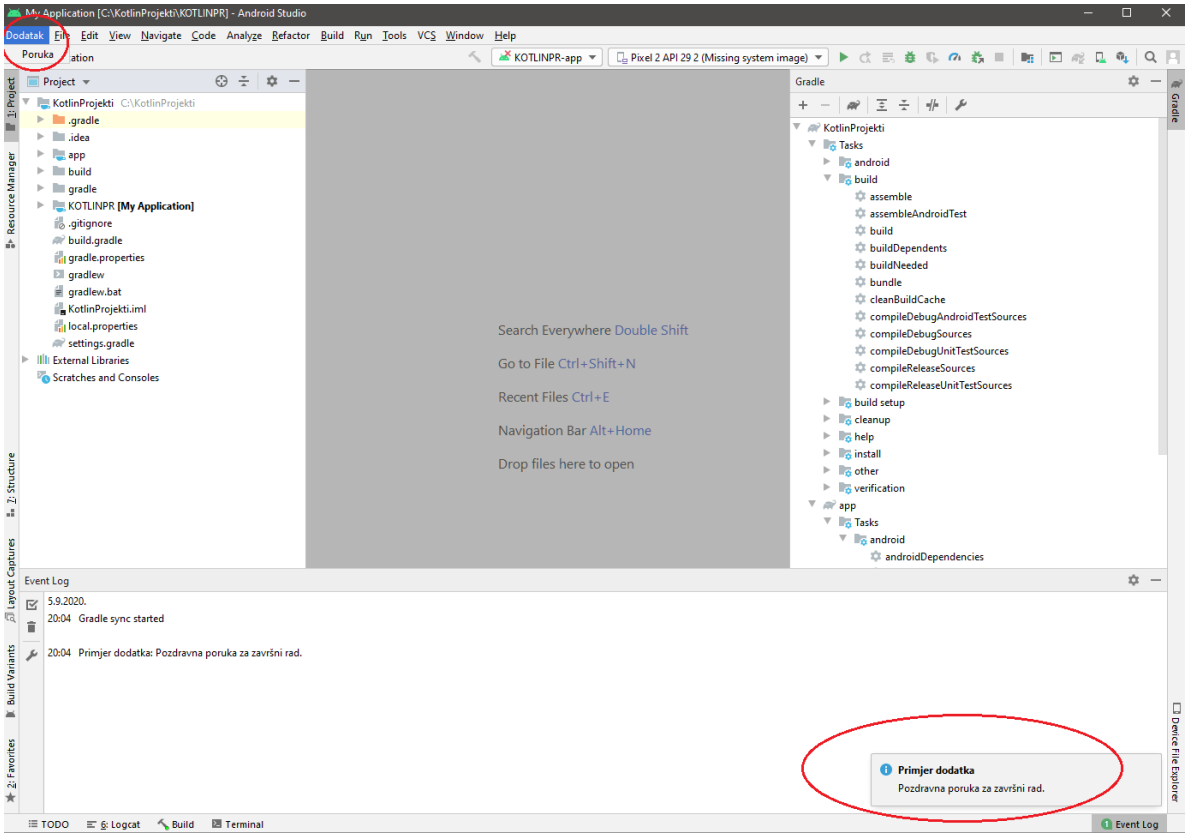
Plugin.xml je konfiguracijska datoteka programskog dodatka koja sadrži sve osnovne opisne informacije i akcije koje programski dodatak sadrži. Osim osnovnih elemenata korištenih u automatski generiranoj datoteci, postoji mogućnost dodavanja još različitih elemenata kao što je vidljivo u službenoj IntelliJ dokumentaciji o konfiguraciji Plugin.xml datoteke [23]. Unutar ove konfiguracijske datoteke vidljive su osnovne informacije projekta, odnosno programskog dodatka pa tako vidimo prijašnje spomenuti id koji je u slučaju ovog projekta `com.nazivdomene.dodatak` što se vidi i na prijašnjoj slici. Osim toga, sadrži i podatke poput naziva programskog dodatka, adrese elektroničke pošte vlasnika dodatka te kratkog opisa svrhe programskog dodatka.

Osim opisnih elemenata, prisutna su i dva jako bitna elementa, odnosno ekstenzije (eng. *Extensions*) i akcije. Ekstenzije se prema službenoj dokumentaciji razvoja programskih dodataka za IntelliJ sučelje koriste kako bi se proširila funkcionalnost IntelliJ platforme na način koji nije jednostavan poput dodavanja akcije u alatnu traku ili izbornik [24]. Ekstenzije omogućavaju programskog dodatku koji se razvija da komunicira s drugim dodatcima ili samim razvojnim okruženjem. Ukoliko se žele proširiti funkcionalnosti drugih programskih dodataka, potrebno ih je navesti pod `<extensions>` blok koda unutar plugin.xml datoteke. Također, ukoliko je cilj da drugi programski dodatci mogu koristiti ovaj programski dodatak, potrebno je navesti tzv. ekstenzijske točke (eng. *Extension points*). Ekstenzije dolaze do izražaja kad se npr. želi omogućiti zaslon s postavkama unutar samog razvojnog okruženja, s obzirom na to da se ovaj programski dodatak razvija za Android Studio, u tom slučaju bi trebalo napraviti ekstenziju funkcionalnosti Android Studija.

Programski dodatak koji će biti razvijen kroz ovaj rad je jednostavnijeg tipa pa neće biti potrebe za ovakvim ekstenzijama. Složeniji programski dodatci kojima se želi dodati

mogućnost mijenjanja Android Studija na način koji nije da se samo pojavi unutar alatne trake ili izbornika imaju barem jednu točku ekstenzije.

Prema službenoj dokumentaciji akcije su klasa izvedena iz *AnAction* klase koja sadrži metodu actionPerformed() i koja se poziva nakon što je odabrana stavka u izborniku, alatnoj traci ili drugoj definiranoj poziciji unutar razvojnog okruženja [25]. Pojedini programski dodatak može sadržavati više različitih akcija, ali sve moraju biti napisane unutar plugin.xml datoteke. U nastavku će biti prikazan jedan jednostavan programski dodatak koji nakon pritiska na gumb „Poruka“ izbacuje pozdravnu poruku koja je prikazana na slici 19.



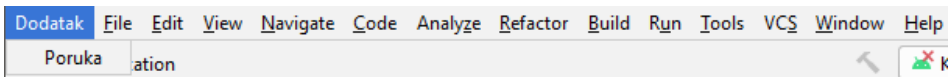
Slika 19. Pozdravna poruka

Primjer isječka koda akcije (Programski kod 1. Akcija primjera programskog dodatka) unutar konfiguracijske datoteke plugin.xml prikazuje koje elemente se može definirati prilikom konfiguracije programskog dodatka. Moguće je dodati još dodatnih elemenata, a isto tako je moguće i ukloniti neke poput „`text`“ što će biti prikazano u razvoju vlastitog dodatka. „`Add to group`“ element dodaje ovu definiranu akciju zadanoj grupi, koja je u ovom slučaju „`MainMenu`“ na prvo mjesto izborne trake pomoću `anchor="first"`. Grupe su dobra praksa i dosta aplikacija ih se pridržava poput Worda koji grupira elemente poput izreži, kopiraj i zalijepi.

```
<actions>
  <group id="Dodatak.TopMenu"
         text="Dodatak"
         description="Dodatak gumb izbornik">
    <add-to-group group-id="MainMenu" anchor="first"/>
    <action id="MyAction"
            class="action.MyAction"
            text="Poruka"
            description="MyAction"/>
  </group>
</actions>
```

Programski kod 1. Akcija primjera programskog dodatka

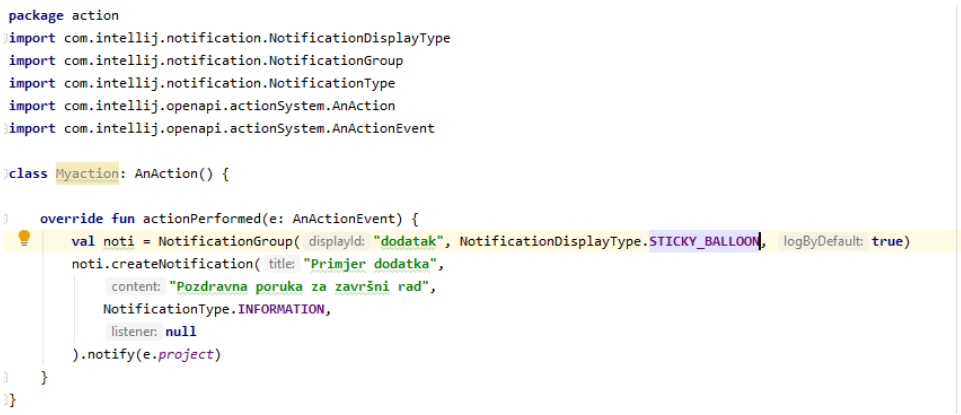
Na slici 20. je prikazano kako u praksi izgleda programski dodatak s ovom s ovom konfiguracijom. U slučaju da je vrijednost sidra (eng. *Anchor*) bila „last“ dodatak bi bio na zadnjem mjestu izborne trake. Svaka akcija ima jedinstveni *action id* koji je u ovom slučaju istog naziva kao i klasa u kojoj se nalazi funkcionalnost dodatka, odnosno MyAction.



Slika 20. Dodatak u izbornoj traci

Tekst element (eng. *Text*) unutar Action id oznake označava naziv gumba nakon što je odabrana njegova grupa, odnosno to je tekst gumba koji će se pokazati nakon što se u izborniku odabere gumb „Dodatak“ i taj tekst je u ovom slučaju „Poruka“.

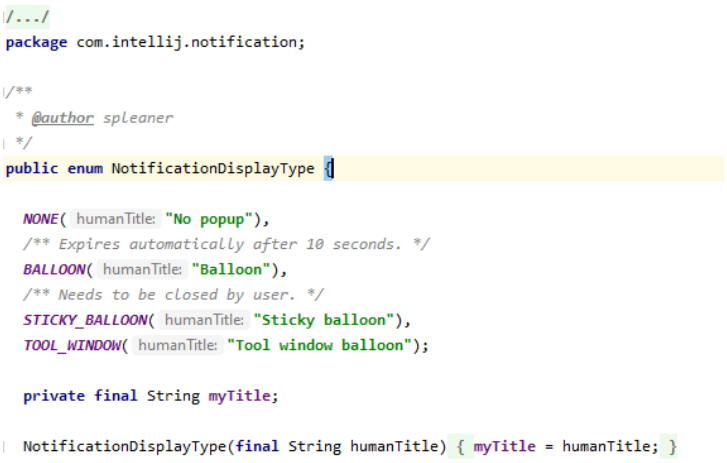
Ovo je među najjednostavnijim formama programskog dodatka, konkretna usporedba bi bila sa „Hello world“ aplikacijom prilikom prvog doticaja sa programiranjem, ali dovoljna forma da se shvati osnovni koncept programskog dodatka. Glavna funkcionalnost odnosno kod programskog dodatka se nalazi ustvari u klasi koju pozivamo unutar prijašnje spomenute akcije koja se nalazi u konfiguracijskoj datoteci plugin.xml. Na slici 21. je prikazana klasa koja se poziva u akcijama, odnosno klasa MyAction.kt koja sadrži funkcionalnost programskog dodatka. Nastavak .kt u nazivu klase označava jezik pisanja koji je u ovom slučaju Kotlin.



Slika 21. Sadržaj klase programskog dodatka

Pri samom vrhu klase je vidljiv naziv paketa, odnosno `actions` u kojem se nalazi ova klasa te bi bila dobra praksa sve ostale klase koje su povezane stavljati u isti paket zbog lakšeg snalaženja i iskoristivosti u drugim projektima. Prema službenoj dokumentaciji Kotlina, paketi mogu sadržavati klase, sučelja ili metode [26]. Navedeni paket nije bio unutar postojećih paketa već je ručno kreiran i unutar njega je napisana navedena klasa, dok su neki paketi već gotovi pa tako npr. u programskom jeziku Java imamo paket java.util koji sadrži kolekcije okvira, klasa, i sl. [27]. Zbog sigurnosnih razloga je preporučljivo uvijek koristiti dostupne pakete ako je to moguće, jer su ih najčešće razvili ljudi koji stoje iza razvojnog sučelja ili programskog jezika što pridodaje na sigurnosti samog paketa.

Ispod naziva paketa nalaze se sve uvezene klase koje se po potrebi dohvaćaju, odnosno uključuju u projekt. Osim klasa, također je moguće uvesti (eng. *Import*) metode, sučelja i objekte. U ovom slučaju, jedna od uvezenih klasa je `NotificationDisplayType` koja sadrži mogućnost odabira izgleda skočnog prozora obavijesti. Na slici 22. se vidi kako su to tri vrste prozora, odnosno `BALLOON`, `STICKY_BALLOON`, `TOOL_WINDOW` te četvrta odnosno mogućnost bez skočnog prozora.



Slika 22. Klasa NotificationDisplayType

Ova klasa je već dostupna unutar samog razvojnog sučelja IntelliJ IDEA te ju nije bilo potrebno preuzimati iz drugih izvora, ali ju je potrebno uključiti u samu klasu naredbom `Import`. Ona se nalazi unutar `com.intellij.notification` paketa i dodana je kao pojedinačna klasa, ali također je moguće uključiti sve metode i klase unutar tog paketa bez da se ručno dodaju jedna po jedna. Ipak, poželjno je uvesti samo one elemente iz paketa koje ćemo koristiti kako bi se izbjegli mogući konflikti u imenovanju klasa. Način na koji se uključi sav sadržaj paketa je da se napiše import i naziv paketa te se zatim na kraj doda simbol točke i zvjezdice što bi u ovom slučaju bilo `import com.intellij.notification.*` i na taj su način sve pripadajuće klase, metode i ostali elementi je uključeno u klasu i može se koristiti.

Nakon što su uvezeni svi željeni paketi i pripadajući elementi, potrebno je napisati samu funkcionalnost programskog dodatka. U ovom slučaju ta funkcionalnost je skočni prozor sa pozdravnom porukom koji se pokrene nakon pritiska na gumb „Poruka“. Prvi korak je premostiti klasu AnActionEvent koja se pokreće nakon akcije spomenute u konfiguracijskoj datoteci plugin.xml. Progamski kod 2. prikazuje funkciju koja se izvršava nakon pritiska na gumb „Poruka“.

```
override fun actionPerformed(e: AnActionEvent) {
    val noti = NotificationGroup("dodatak",
NotificationDisplayType.BALLOON, true)
    noti.createNotification("Primjer dodatka",
            "Pozdravna poruka za završni rad",
            NotificationType.INFORMATION,
            null
    ).notify(e.project)
}
```

Programski kod 2. Funkcija primjera programskog dodatka

Ovaj programski kod kreira i definira izgled skočnog prozora obavijesti koji će se prikazati nakon pritiska na gumb „Poruka“. U imenovanju funkcije vidimo prijašnje spomenuto premoštenje klase AnActionEvent te se funkciji daje naziv `actionPerformed`. Ova novonastala funkcija koristi funkciju prijašnje uvezenih paketa obavijesti (eng. *Notification*) u kojoj se definira id, naslov, poruka i tip obavijesti koja se pojavi nakon pritiska na gumb „Poruka“ u donjem desnom uglu kao na slici 19. Nakon što se definira funkcionalnost, a može i obrnuto, popunjavaju se tipične informacije o vlasniku programskog dodatka, njegovom opisu i ova klasa se poziva unutar akcija. Na slici 24. je prikazan izgled upotpunjene konfiguracijske datoteke Plugin.xml

```
<idea-plugin>
    <id>myplugin2.dodatak</id>
    <name>Skočni prozor obavijesti</name> //Ovdje se definira naziv programskog dodatka
    <vendor email="ilicic.kresimir@gmail.com" url="http://www.našadomena.com">Naziv poduzeća</vendor> //Kontakt podatci vlasnika

    <description><![CDATA[
    Ovaj programski dodatak je jednostavan prikaz skočnog prozora obavijesti nakon što korisnik pritisne na gumb "Poruka"</em>
    ]]></description>


    <extensions defaultExtensionNs="com.intellij">
        <!-- Add your extensions here -->
    </extensions>

    <actions>
        <group id="Dodatak.Topmenu"
               text="Dodatak"
               description="Dodatak gumb izbornik">
            <add-to-group group-id="MainMenu" anchor="first"/>
            <action id="MyAction"
                    class="action.Myaction"
                    text="Poruka"
                    description="MyAction"/>
        </group>
    </actions>
</idea-plugin>
}
```

Slika 24. Potpuna Plugin.xml datoteka

Donji dio konfiguracijske datoteke, odnosno akcije, je već prijašnje objašnjen, no gornji dio sadrži neke od dodatnih informacija. Upotpunjavaju se naziv programskog dodatka, kontakt podatci vlasnika ovog programskog dodatka te opis same svrhe programskog dodatka. Funkcionalnost ovog programskog dodatka rađena je prema gotovom projektu autora Marcosa Holgada na Internet stranici ProAndroidDev [28] i korištena je kako bi se objasnili osnovni principi izrade programskog dodatka, dok će vlastiti programski dodatak bit izrađen i objašnjen u poglavlju 3.4.

Sljedeći korak je veoma bitan da bi provjerili kako radi izrađeni programski dodatak unutar Android Studija. Potrebno je otvoriti prijašnje spomenutu build.gradle datoteku i promijeniti nekoliko elemenata kao što je vidljivo na slici 25.

```
plugins {
    id 'org.jetbrains.intellij' version '0.4.21'
    id 'org.jetbrains.kotlin.jvm' version '1.3.41'
}

group 'myplugin2'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
}

// See https://github.com/JetBrains/gradle-intellij-plugin/
intellij {
    version '2019.2.4'
    alternativeIdePath 'C:\\Program Files\\Android\\Android Studio'
}
compileKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
compileTestKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
patchPluginXml {
    changeNotes """
      Add change notes here.<br>
<em>most HTML tags may be used</em>"""
}
```

Slika 25. Uređena build.gradle datoteka

U ovom slučaju je bilo potrebno izmijeniti `implementation` koji je vidljiv na slici 17. sa `compile` unutar stavke `dependencies`. Razlog je što programski dodatak Gradle unutar IntelliJ IDEA razvojnog okruženja još uvijek ne podržava implementaciju ili korištenje api pa se umjesto toga koristi `compile`. S obzirom na to da je poželjno programski dodatak testirati u Android Studiju, potrebno je dodati alternativni pristup mapi gdje je razvojno okruženje instalirano, odnosno `alternativeIdePath` koji je u ovom slučaju `'C:\\Program Files\\Android\\Android Studio'`. Ova putanja se upisuje unutar oznake `Intellij` gdje se nalazi zadana inačica razvojnog okruženja IntelliJ.

Nakon što su sve stavke ispunjene, programski dodatak se može testirati tako što se iz padajućeg izbornika unutar glavne izborne trake odabere „Plugin“ i pritisne na tipku pokreni kao što je vidljivo na slici 26. ili se jednostavno pokrene kombinacijom tipki Shift i F10.



Slika 26. Testiranje programskog dodatka

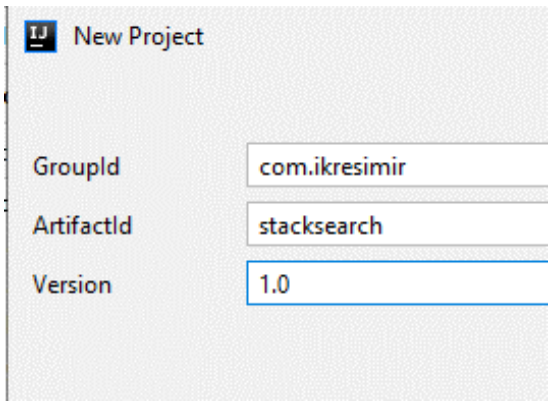
Nakon što se pokrene testiranje, potrebno je sačekati da se izvrši programski kod, pokrene Android Studio i nakon povratne poruke unutar studija o izvršenju Gradle procesa može se pokrenuti programski dodatak na način da se iz izbornika odabere „Dodatak“ i zatim unutar njega „Poruka“. Nakon ove demonstracije jednostavnog programskog dodatka, vrijeme je za razvijanje vlastitog programskog dodatka gdje će proces biti objašnjen u sljedećem poglavlju zajedno sa bitnim detaljima.

3.4. Implementacija funkcionalnosti programskog dodatka

Funkcionalnost vlastitog programskog dodatka će biti mogućnost pretraživanja programskog koda ili bilo kojeg označenog teksta unutar uređivača koda preko Internet stranice Stackoverflow. Razlog je taj što kako tokom studiranja tako i na poslu s vremena na vrijeme programeri traže pomoć na ovoj internet stranici koja je jako veliki izvor programerskih pitanja i odgovora, točnije izvor 20,097,101 pitanja u ovom trenutku pisanja rada [29].

Kako bi se olakšao proces pretrage pitanja na Stackoverflowu, ovaj programski dodatak će zaobići klasični postupak gdje programer mora kopirati dio koda koji ne razumije, otvoriti preglednik, otići na Stackoverflow.com, upisati pitanje i pretražiti ga. Način na koji će ovaj programski dodatak zaobići sve ove korake je takav da će nakon označenog teksta, odnosno programskog koda koji je nejasan, pritiskom na ikonu razvijenog programskog dodatka u alatnoj traci, programski dodatak sam otvoriti preglednik i pretražiti pitanje.

Početni koraci u izradi novog projekta su jednaki kao i kod primjer iz prošlog poglavlja, dok se opisne stavke projekta mogu vidjeti na slici 27.



Slika 27. Opisne stavke vlastitog dodatka

Stavka unutar Groupid je ustvari domena ikresimir.com koja će biti korištena kao identifikator vlasništva nad programskim dodatkom. Artifactid je naziva stacksearch izvedenog iz same svrhe programskog dodatka, odnosno pretraživanja na Stackoverflowu. Inačica je 1.0 bez naziva SNAPSHOT kao na početnim postavkama koje se vide na slici 14. SNAPSHOT prema službenoj dokumentaciji Mavena označava zadnji programski kod u razvoju koji ne garantira da je stabilan ili da se neće mijenjati [30]. Njega nije nužno imati, a s obzirom na to da će ova inačica programskog dodatka biti i konačna, može se ostaviti 1.0 u polju inačice.

Nakon što su unesene ove vrijednosti u polja, nastavak je kao i u prijašnjem primjeru, odnosno potrebno je definirati naziv projekta i mapu gdje će se spremati na računalu. Kad je taj korak završen, pritiskom na gumb završi otvara se projekt i automatski se generira datoteka plugin.xml i build.gradle. S obzirom na to da će se ovaj programski dodatak nakon izrade objaviti u repozitorij, potrebno je definirati podatke o vlasništvu unutar plugin.xml datoteke kao što se vidi na slici 28. Tekst unutar opisa (eng. *Description*) će se nalaziti i u opisu programskog dodatka prilikom dodavanja u Android Studio, odnosno instalacije i na stranici programskog dodatka unutar JetBrains repozitorija.

```
<idea-plugin>
    <id>com.ikresimir.stacksearch</id>
    <name>Stack Search</name>
    <vendor email="ilicic.kresimir@gmail.com" url="http://www.ikresimir.com">Ikresimir</vendor>

    <description><![CDATA[
    This plugin will enable the user to search the highlighted text on Stack Overflow without the need of manually copying and pastin
    The main reason this plugin exists is to save some time for the programmer.
    ]]></description>
```

Slika 28. Opisni podatci u plugin.xml

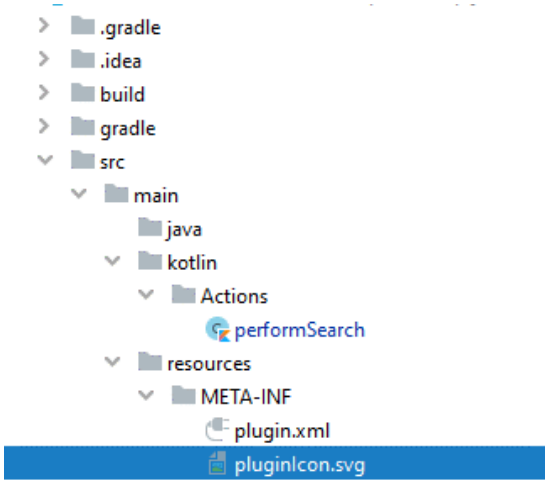
Vidljivo je kako je dodan naziv programskog dodatka, odnosno naziv je Stack Search te su postavljeni ostali podatci poput adrese elektroničke pošte i domena vlasnika. Osim ovih opisnih podataka dodana je i ikona za programski dodatak kako bi se pomoću klika na ikonu mogao pokrenuti programski dodatak, odnosno izvršiti njegova funkcionalnost. Izgled ikone je vidljiv na slika 29. te je izrađena uz pomoć mrežnog alata Boxy SVG za stvaranje vektorskih slika [31].



Slika 29. Ikona vlastitog dodatka

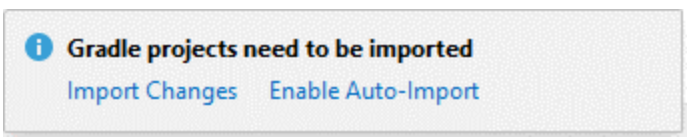
S obzirom na to da službena dokumentacija JetBrainsa navodi kako naziv ikone prema konvenciji treba biti pluginIcon.svg, tako glasi i naziv ikone u ovom vlastitom programskom dodatku te mora biti postavljena u mapu META-INF koja također sadrži i plugin.xml datoteku

[32]. Bitno je napomenuti kako format ikone mora biti svg, odnosno vektorski format kako prilikom skaliranja ikone ne bi došlo do gubitka kvalitete. Na slici 30. je vidljiva struktura projekta i lokacija datoteke ikone koja je odabrana prema prijašnje spomenutim pravilima. Prema konvenciji je također moguće i dodati tamnu verziju ikone ako se koristi tamna tema razvojnog okruženja i u tom slučaju se dodaje još jedna inačica ikone naziva pluginIcon_dark.svg. S obzirom na to da Android Studio koji se koristi za testiranje vlastitog programskog dodatka ima light, odnosno svijetlu temu, nema potrebe za takvom ikonom.



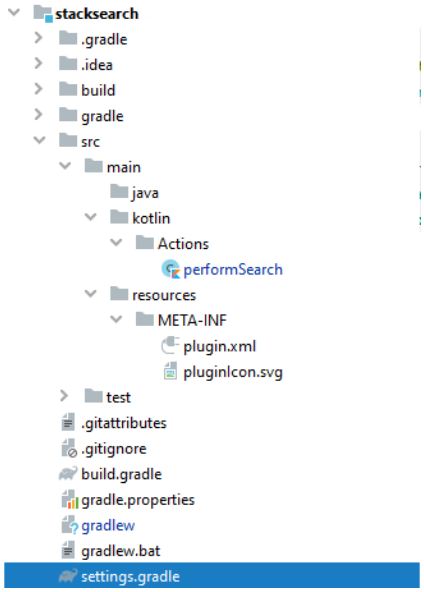
Slika 30. Lokacija ikone programskog dodatka

Osim definiranja podataka o vlasniku, također je potrebno izmijeniti build.gradle datoteku gdje će se definirati razvojno okruženje za testiranje, odnosno Android Studio i promijeniti `implementation` u `compile` iz prijašnje spomenutih razloga. Definiranje razvojnog okruženja za testiranje se radi uz pomoć naredbe `alternativeIdePath 'C:\Program Files\Android\Android Studio'` gdje `'C:\Program Files\Android\Android Studio'` predstavlja lokaciju gdje je razvojno okruženje instalirano. Ostali elementi u build.gradle datoteci mogu ostati kako su i automatski generirani, no bitno je naglasiti kako je nakon svake izmjene unutar build.gradle datoteke potrebno uvesti promjene osim ako nije uključena opcija automatskog uvoza. IntelliJ IDEA razvojno okruženje će korisnika upozoriti ako nije uključen automatski uvoz putem skočnog prozora kao što se vidi na slici 31.



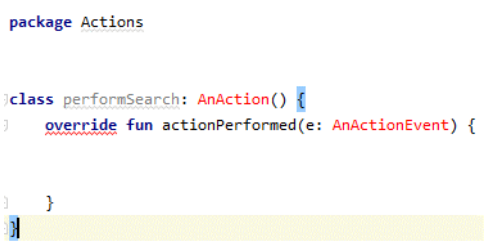
Slika 31. Obavijest o promjenama koje nisu uvezene

Nakon što su postavljene osnovne konfiguracijske datoteke, sljedeći korak je izrada same klase koja će sadržavati funkcionalnost vlastitog programskog dodatka. S obzirom na to da je odabrani programski jezik za razvoj vlastitog dodatka Kotlin, unutar mape naziva Kotlin se dodaje novi paket te zatim u njemu nova Kotlin klasa. Kao što se vidi na slici 32. u projektnoj strukturi je dodan paket naziva Actions te unutar njega klasa naziva `performSearch`. Razlog imenovanja varijabli, pisanja opisa i imenovanja ostalih elemenata na engleskom jeziku je taj što će se ovaj programski dodatak dodati u repozitorij JetBrainsa gdje je velika većina programskih dodataka na engleskom jeziku.



Slika 32. Projektna struktura vlastitog dodatka

Osim paketa i klase, u projektnoj strukturi se također vidi i lokacija prijašnje spomenute ikone programskog dodatka. Prvi korak pisanja klase koja će sadržavati funkcionalnost programskog dodatka je nasljeđivanje AnAction() konstruktora i premoštenje AnActionEvent događaja. Nakon što su ovi koraci napravljeni, klasa će imati podvučene elemente crvenom bojom kao što se vidi na slika 33.



Slika 33. Greška u klasi

Razlog je u tome što nisu dodane potrebne biblioteke i paketi kako bi se mogle koristiti naslijeđene funkcije. Prelaskom miša preko crveno podvučenih riječi nudi se mogućnost uvoza potrebnih biblioteka nakon kojih su dobivene potrebne metode i greške više nisu prisutne. Nakon što su ove dvije biblioteke dodane, sljedeći korak je ručno dodati jednu od biblioteka koja sadrži potrebnu mogućnost pokretanja internet preglednika nakon pritiska na gumb, odnosno ikonu programskog dodatka. Biblioteka koja sadrži potrebnu funkciju za pokretanje Internet preglednika se naziva BrowserUtil te ju se mora uvesti naredbom `import com.intellij.ide.BrowserUtil`.

Osim ove biblioteke, potrebne su i dvije biblioteke koje služe za upravljanje tekstom, a one se nazivaju LangDataKeys i Editor te služe kako bi se dohvatio označeni tekst i spremio u varijablu tipa string koja će se koristiti za pretragu pitanja na Stackoverflowu. Potrebna je još i varijabla sa fiksnim dijelom poveznice Stackoverflowa na koju će se nadodati pitanje, odnosno prijašnje spomenuta varijabla s dohvaćenim tekstom. Fiksni dio je `https://stackoverflow.com/search?q=`, te sve što dolazi nakon znaka jednakosti se pretražuje u bazi pitanja.

Kad se spoji varijabla fiksnog dijela i varijabla gdje se sprema označeni tekst, rezultat je kompletna poveznica koja bi bila jednaka kao da se ručno pretraživalo pitanje na Stackoverflow internet stranici. Takva varijabla je vidljiva na sljedećem isječku programskog koda (Programski kod 3. Funkcija vlastitog programskog dodatka) pod nazivom `predefinedSearchText`. Ovu „konačnu“ varijablu koristimo unutar `BrowserUtil` funkcije koja će otvoriti Internet preglednik na poveznici koja je ustvari ta varijabla. Sljedeći programski isječak prikazuje kako izgleda kompletan kod, odnosno funkcija klase `performSearch` vlastitog programskog dodatka.

```
package Actions
import com.intellij.openapi.actionSystem.AnAction
import com.intellij.openapi.actionSystem.AnActionEvent
import com.intellij.ide.BrowserUtil;
import com.intellij.openapi.actionSystem.LangDataKeys
import com.intellij.openapi.editor.Editor
class performSearch: AnAction() {
    override fun actionPerformed(e: AnActionEvent) {
        val caretModel = (e.getData(LangDataKeys.EDITOR) as
Editor).caretModel
        val currentCaret = caretModel.currentCaret
        val selectedText = currentCaret.selectedText
        val predefinedSearchText = "https://stackoverflow.com/search?q=" +
selectedText;
        BrowserUtil.browse(predefinedSearchText )
    }
}
```

Programski kod 3. Funkcija vlastitog programskog dodatka

Kao što je vidljivo, kod je kratak i jednostavan jer koristi funkcije iz već postojećih biblioteka koje su dostupne uz osnovnu instalaciju IntelliJ IDEA razvojnog okruženja, nije bilo potrebe za dodatnim bibliotekama. IntelliJ IDEA okruženje ima velik izbor biblioteka te je zbog toga dobar odabir za razvoj kako Android Studio programskih dodataka tako i za ostale programske projekte.

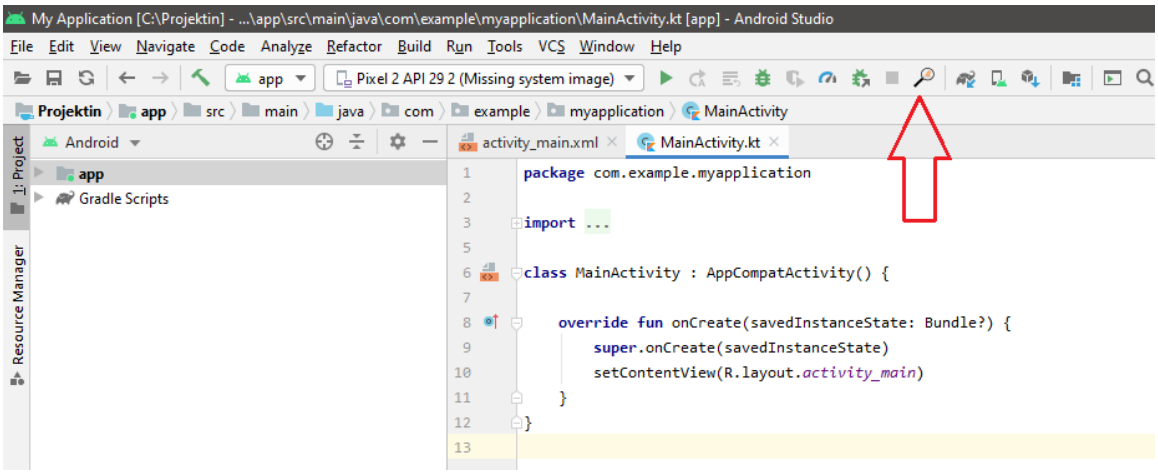
Nakon što je funkcionalnost programskog dodatka napisana, potrebno je registrirati akciju unutar plugin.xml konfiguracijske datoteke. Razlika ovog i primjera ranije prikazanog u radu je što za vlastiti dodatak neće biti ista pozicija unutar Android Studija. Umjesto pozicije unutar izbornika, bit će unutar alatne trake i imati ikonu za razliku od tekstualnog izgleda u prijašnjem primjeru. Sljedeći programski isječak (Programski kod 4. Akcija vlastitog programskog dodatka). prikazuje definirane akcije unutar plugin.xml datoteke.

```
<actions>
  <group id="stacksearch.Topmenu" description="Search code on Stack
Overflow">
    <add-to-group group-id="ToolbarRunGroup" anchor="last"/>
    <action id="performSearch"
            class="Actions.performSearch"
            description="Perform search"
            icon="/META-INF/pluginIcon.svg"/>
  </group>
</actions>
```

Programski kod 4. Akcija vlastitog programskog dodatka

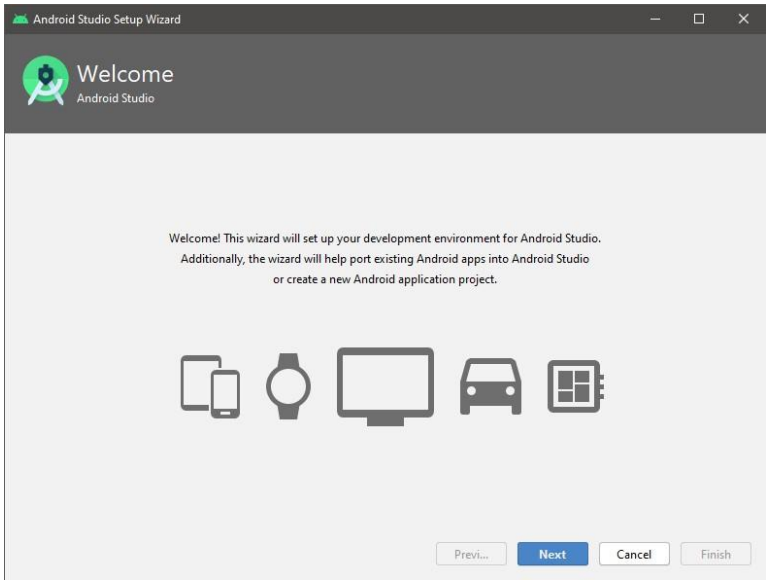
Poprilično slična definicija akcije je i za vlastiti programski dodatak kao što je bila u prvom primjeru dodatka što se vidi na programskom kodu 1. Razlika je u tome što nema potrebe za atributom `text` unutar grupe i akcije s obzirom na to da se ovom programskom dodatku pristupa putem ikone a ne teksta. Potrebno je dodati ovisnost unutar konfiguracijske datoteke i definirati na kojim platformama će vlastiti programski dodatak raditi. S obzirom na to da su korištene samo osnovne biblioteke, dovoljno je napisati liniju iznad akcija `<depends>com.intellij.modules.lang</depends>` koja prema službenoj dokumentaciji osim drugih funkcionalnosti omogućava i označavanje teksta (eng. *Highlight*) koji je bitan dio ovog programskog dodatka. [33].

Također, vlastiti dodatak pripada drugoj grupi, odnosno `ToolbarRunGroup` kao što se vidi na prijašnjoj slici, to je kako bi ikona dodatka bila na vidljivom mjestu. Osim toga vidljiv je i atribut za lokaciju ikone dodatka kako bi se prikazala autorska ikona i omogućilo raspoznavanje vlastitog dodatka od drugih gumbova u alatnoj traci. Na slici 36. vidljivo je kako ikona vlastitog programskog dodatka izgleda unutar Android Studija.



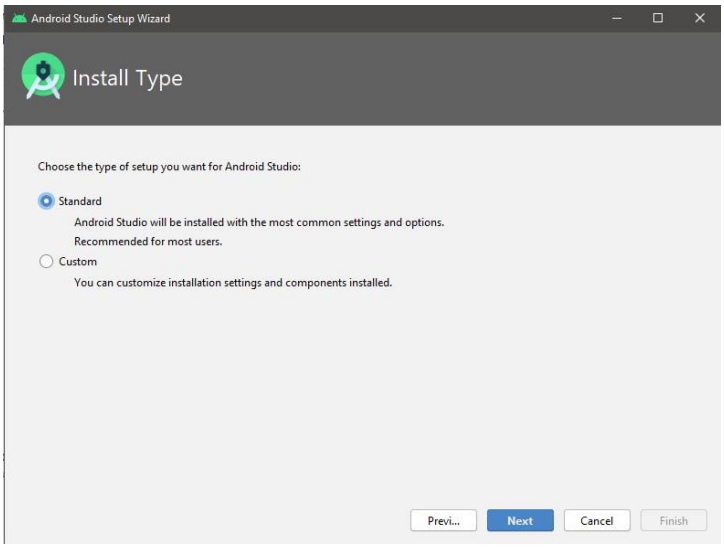
Slika 36. Ikona unutar Android Studija

S obzirom na to da se funkcionalnost vlastitog programskog dodatka nalazi u paketu Actions i klasi performSearch, tako se navodi i u class atributu unutar akcija kako bi se povezao gumb sa funkcionalnošću. Nakon što je ispunjena i konfiguracijska datoteka plugin.xml, potrebno je testirati funkcionalnost programskog dodatka. U prijašnjim koracima navedeno je kako je za razvojno sučelje za testiranje postavljen Android Studio. Nakon pokretanja testiranja dodatka kombinacijom tipki Shift i F10 ili klikom na gumb pokreni u alatnoj traci, pokreće se Android Studio i potrebno ga je postaviti za prvo testiranje projekta. Nakon postavljanja, sljedeći put će se moći samo pokrenuti i odmah testirati dodatak bez dodatnih koraka. Na slici 37. prikazan je izgled početnog zaslona Android Studija nakon što se pokrene.



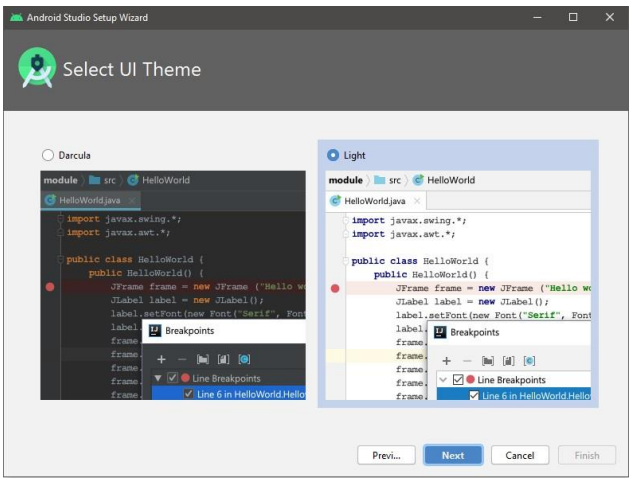
Slika 37. Postavljanje Android Studija za testiranje

Nakon pritiska na gumb dalje (Eng. *Next*), otvara se zaslon na kojem je potrebno odabrati tip instalacije Android Studija, odnosno standardni ili prilagođeni kao je prikazano na slici 38. S obzirom na to da Android Studio trenutno služi samo za testiranje, standardna instalacija je dovoljna jer sadrži sve potrebne elemente kako bi se testirao programski dodatak.



Slika 38. Tip instalacije Android Studija

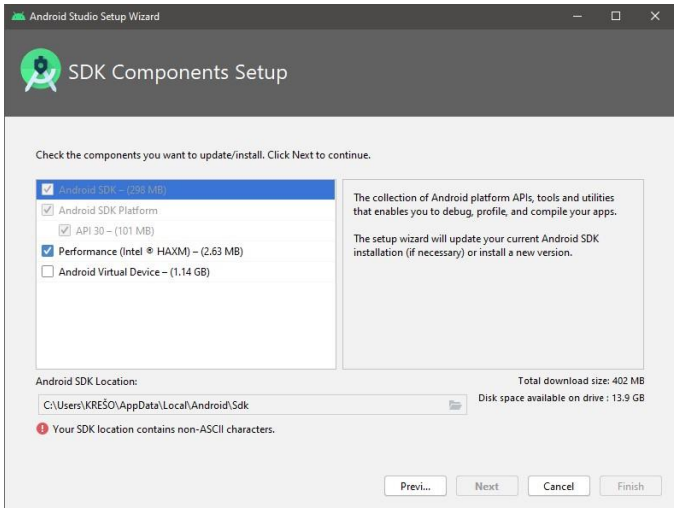
Sljedeći korak je odabir teme koja je stvar osobnog ukusa te je u ovom slučaju odabrana light tema, odnosno tema svjetlijih tonova što se vidi i na slici 39.



Slika 39. Odabir teme Android Studija

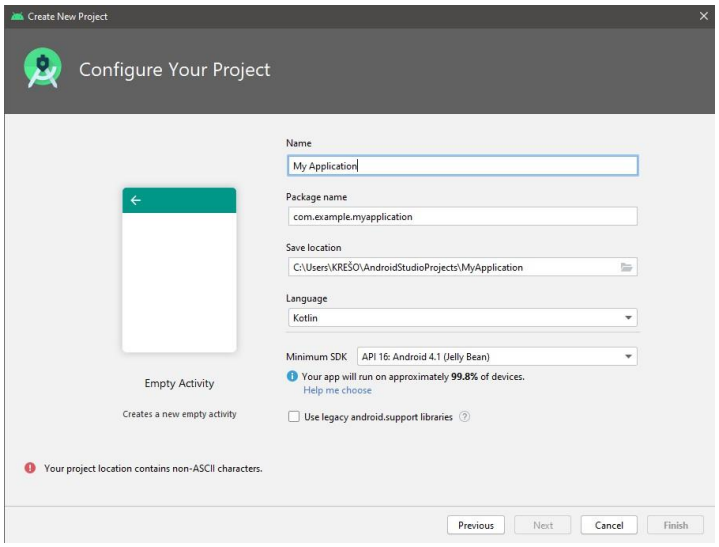
Nakon odabrane teme, sljedeći korak je odabir komponenata koje će se instalirati i bez kojih je nemoguće testirati vlastiti programski dodatak unutar Android Studija. S obzirom na to

da se programski dodatak ne testira na mobilnim uređajima već unutar razvojnog okruženja, opcija Android Virtual Device nije potrebna. Na slici 40. se vidi prijašnje spomenuti problem koji se ne pojavljuje u IntelliJ IDEA razvojnom okruženju, a problem su posebni znakovi u nazivu mape gdje će se izvršiti instalacija Android Studija. Potrebno je pogledati koji dio putanje stvara problem, u ovom slučaju „KREŠO“ te umjesto te mape napraviti novu koja sadrži samo ASCII znakove u nazivu.



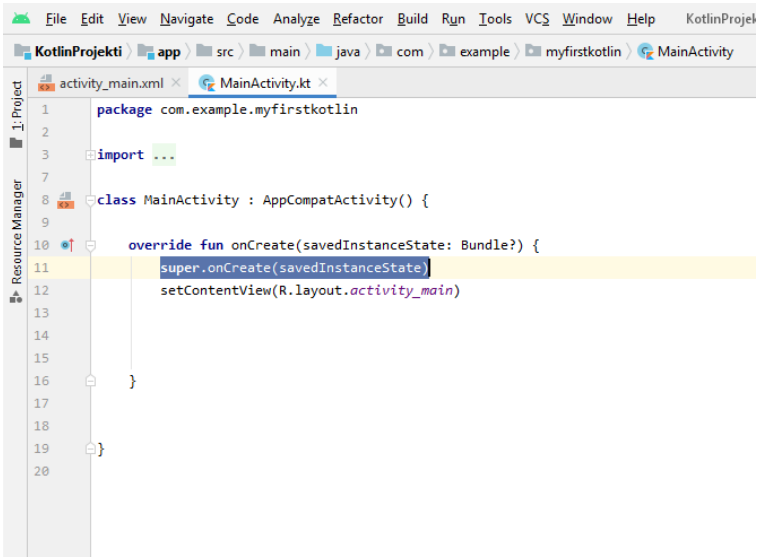
Slika 40. Instalacija potrebnih komponenti za testiranje

Nakon što su tražene komponente preuzete s interneta i instalirane, otvara se početni zaslon Android Studija već prikazan na slici 3. ali ovaj put u svjetlom izdanju. Na početnom zaslonu je potrebno odabrati novi Android Studio projekt te zatim iz ponuđenih predložaka odabrati onaj koji odgovara testiranju programskog dodatka, u ovom slučaju je prazna aktivnost (eng. *Empty activity*) dovoljna. Nakon koraka dalje, potrebno je definirati osnovne informacije o projektu koje ne utječu na testiranje iz razloga što se neće testirati mobilne aplikacije već samo programski dodatak unutar razvojnog okruženja. Dovoljno je ispuniti osnovne podatke i lokaciju spremanja projekta kao što je vidljivo na slici 41. i može se preći na sljedeći korak.



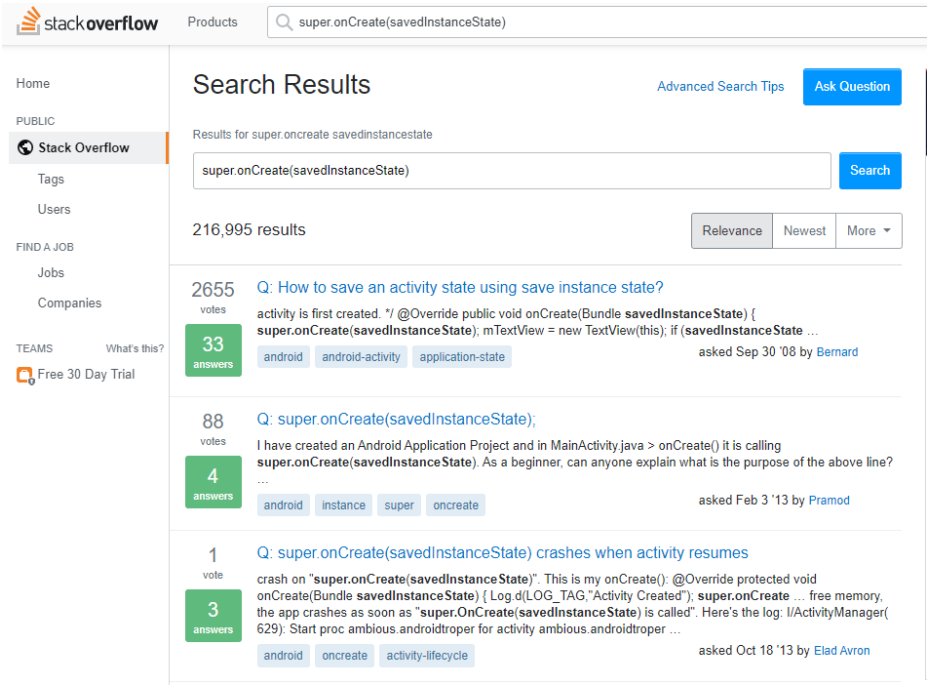
Slika 41. Postavljanje projekta za testiranje

Ovim korakom postavljanje razvojnog okruženja je završilo i testiranje može započeti. S obzirom na to da ovaj programski dodatak radi na način da označeni kod pretražuje na Stackoverflow internet stranici, za potrebe testiranja se označava bilo koji kod što je u ovom slučaju kao što se vidi na slici 42. `super.onCreate(savedInstanceState).`



Slika 42. Označeni tekst prilikom testiranja

Nakon što je označen željeni tekst potrebno je pritisnuti na gumb vlastitog programskog dodatka koji se nalazi u alatnoj traci kao što je prikazano na slici 36. Pritiskom na gumb otvara se zadani Internet preglednik i Internet stranica StackOverflow sa traženim pitanjem u kojem se nalazi označeni tekst iz Android Studija kao što se vidi na slici 43.



Slika 43. Rezultat testiranja dodatka

Na temelju slike 43. se može zaključiti kako je testiranje vlastitog programskog dodatka bilo uspješno te je moguće preći na završni korak, odnosno na izvoz programskog dodatka iz IntelliJ IDEA razvojnog okruženja. Sljedeći korak sastoji se od toga da se unutar build.gradle datoteke obriše ili komentira linija koja označava razvojno okruženje u kojem će se testirati, odnosno „`alternativeIdePath 'C:\\Program Files\\Android\\Android Studio'`" kao što je prikazano na slici 44. Razlog zašto nije izbrisan već komentiran je što bi se ovaj programski dodatak mogao u budućnosti unaprijediti pa bi se testiranje ponovno vršilo na Android Studiju i u tom slučaju bi se maknuo znak komentara. Kada bi se pokušao izvesti programski dodatak bez brisanja ili komentiranja ove linije, Gradle bi u konzoli prikazao povratnu informaciju kako nije u mogućnosti kompilirati programski dodatak. Također je potrebno unutar `changeNotes` obrisati automatski generiran tekst i dodati vlastiti jer bi inače JetBrains odbio objavu programskog dodatka u repozitorij.

```
group 'com.ikresimir'
version '1.0'

repositories {
    mavenCentral()
}

dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
}

// See https://github.com/JetBrains/gradle-intellij-plugin/
intellij {
    version '2019.2.4'
    //alternativeIdePath 'C:\\Program Files\\Android\\Android Studio'
}
compileKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
compileTestKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
patchPluginXml {
    changeNotes """
      1.0 Initial Version"""
}
```

Slika 44. Uklanjanje linije koda iz build.gradle datoteke

Nakon što je prijašnji korak završen, potrebno je izvesti odnosno kompilirati programski dodatak na način da se u terminal napiše naredba `gradlew buildPlugin` i pritisne enter, a također je moguće i putem izbornika kompilirati tako što se odabere build element u izborniku i zatim izgradi projekt (eng. *Build project*). Nakon što je unesena naredba u terminal, pojavljuje se povratna informacija u (ne)uspješnosti kao što je vidljivo na slika 45. Ovaj proces može trajati i nekoliko minuta ovisno o kompleksnosti projekta i jačini računala na kojemu se izvodi ovaj proces.

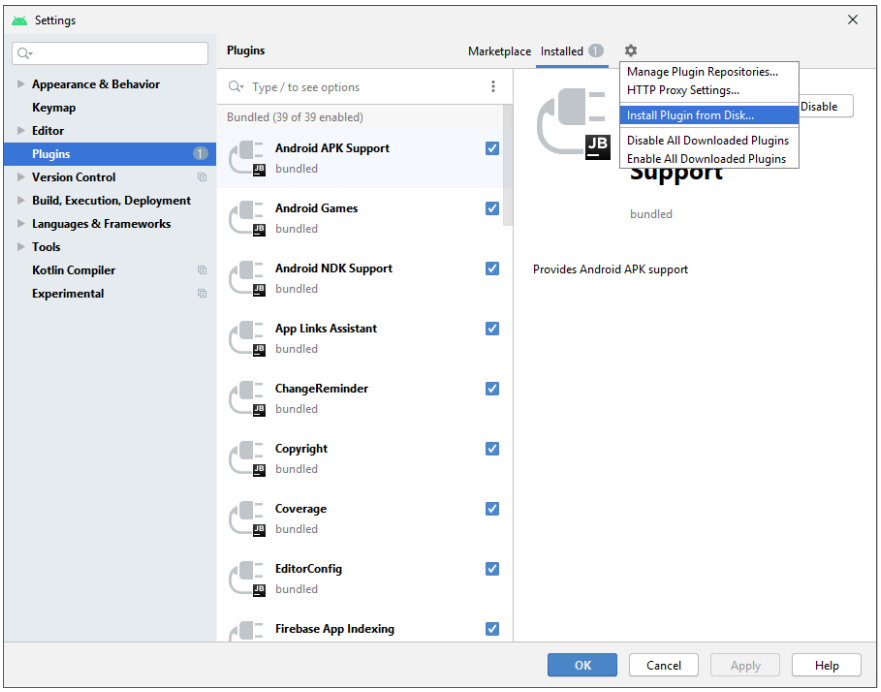
```
C:\Users\KREŠO\IdeaProjects\stacksearch>gradlew buildPlugin
Starting a Gradle Daemon (subsequent builds will be faster)

BUILD SUCCESSFUL in 10s
11 actionable tasks: 1 executed, 10 up-to-date
C:\Users\KREŠO\IdeaProjects\stacksearch>
```

Slika 45. Uspješno kompiliranje programskog dodatka

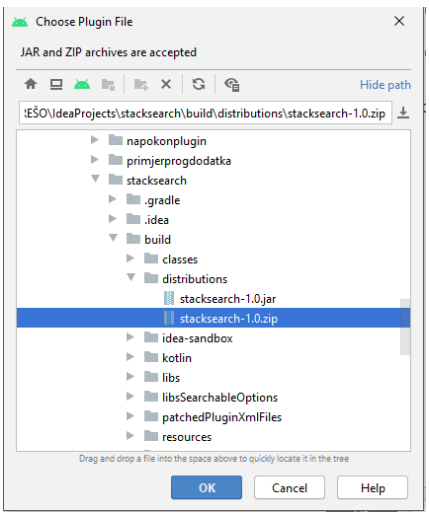
Nakon uspješnog kompiliranja, sljedeći korak je dodavanje programskog dodatka u razvojno okruženje Android Studio. Prvi korak je pokrenuti Android Studio te otvoriti postojeći projekt ili započeti novi. Zatim, u izborniku odabrati File te Settings ili koristiti kombinaciju tipki Ctrl, Alt i S kako bi se otvorio prozor s postavkama. Nakon toga, potrebno je odabrati „Plugins“

s lijeve strane prozora i pritisnuti na ikonu kotačića pored stavke „Installed“ u gornjem dijelu prozora kao što se vidi na slici 46.



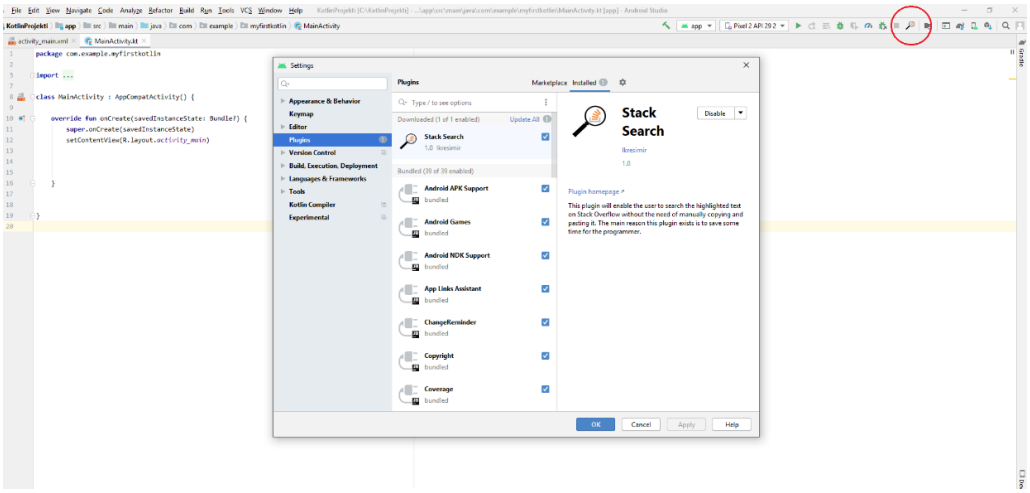
Slika 46. Dodavanje programskog dodatka u Android Studio

Sljedeći korak je odabrati opciju „Install Plugin from Disk…“ i otvoriti mapu gdje je programski dodatak, zatim mapu „Build“ i konačno „Distributions“ gdje će se nalaziti programski dodatak koji je izrađen kao što se vidi na slika 47.



Slika 47. Lokacija programskog dodatka

Pritiskom na tipku „OK“, programski dodatak se učitava u Android Studio te je potrebno ponovno pokrenuti ovo razvojno okruženje. Nakon što se Android Studio ponovno učita, ikona programskog dodatka je vidljiva u alatnoj traci, dok se odlaskom u postavke programskih dodataka kao u prijašnjem koraku, pojavljuje vlastiti programski dodatak s opisom iz konfiguracijske datoteke plugin.xml i zadanom ikonom što se vidi na slici 48.



Slika 48. Uspješno učitan programski dodatak

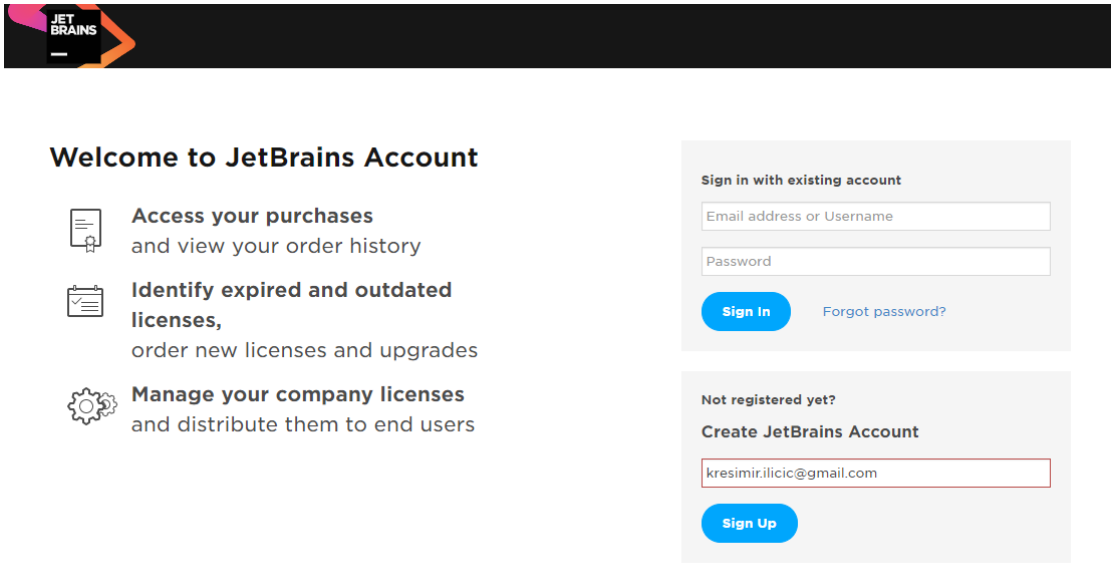
Ovo je ujedno bio i zadnji korak razvoja i implementacije programskog dodatka u razvojno okruženje Android Studio. Nakon što je programski dodatak uspješno testiran, moguće ga je objaviti u JetBrains repozitorij kako bi drugi korisnici također mogli koristiti ovaj razvijeni programski dodatak. U sljedećem poglavlju će biti prikazani potrebni koraci za objavu u repozitorij.

4. Repozitorij

Repozitorij programskih dodataka za Android Studio se nalazi na službenim stranicama JetBrainsa gdje su sadržani i programski dodatci za druga razvojna okruženja. Korisnici mogu slobodno dodavati svoje programske dodatke na repozitorij te će ih JetBrains pregledati i (ne)odobriti. Ovaj proces inače traje do 2 radna dana kao što će se i vidjeti u povratnoj poruci nakon što se vlastiti programski dodatak prenese na repozitorij u nastavku rada. Također, moguće je i postaviti cijenu za vlastiti programski dodatak koju će morati platiti svatko tko ga želi koristiti, ali u tom slučaju prvo je potrebno predati zahtjev i ako je odobren, dogovoriti način licenciranja sa samim JetBrainsom kako piše u službenoj dokumentaciji [34].

4.1. Dodavanje programskog dodatka u repozitorij

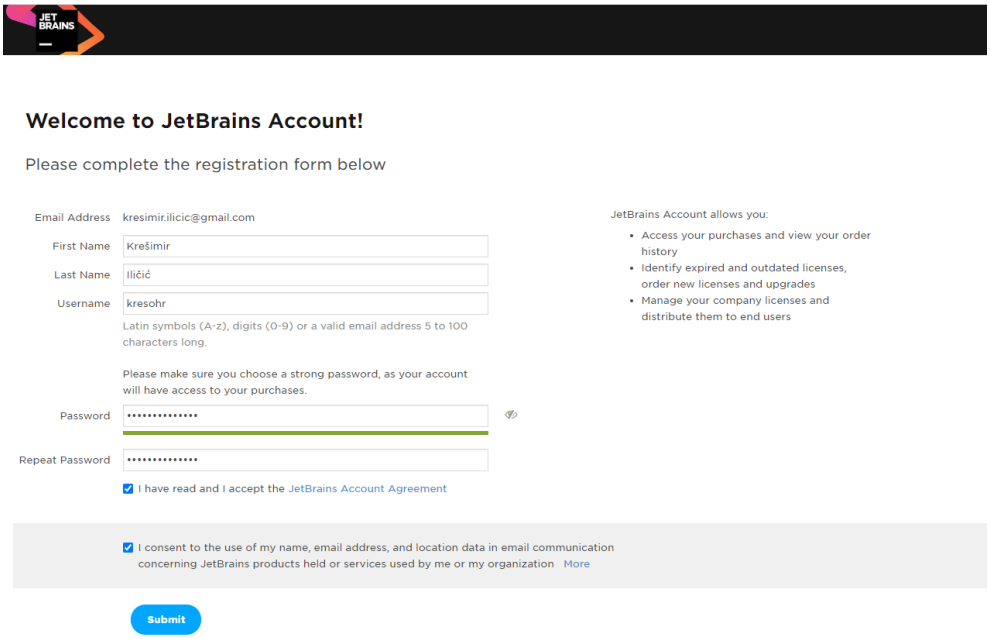
Dodavanje u JetBrains repozitorij se sastoji od nekoliko jednostavnih koraka od kojih je prvi napraviti račun na stranicama JetBrainsa. Nakon što je otvorena registracija, potrebno je popuniti adresu e-pošte na predviđeno mjesto kao što je prikazano na slici 49.



Slika 49. Registracija na JetBrains online trgovinu[4]

[4] Slika zaslona na: https://account.jetbrains.com/login

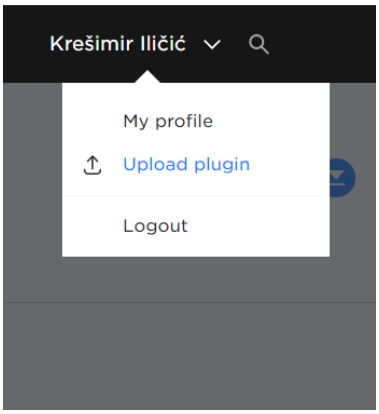
Nakon pritiska na gumb „Sign Up“, pojavljuje se povratna poruka koja govori kako će instrukcije pristići na adresu e-pošte koju smo naveli u prostoru predviđenom za to, u ovom slučaju „kresimir.ilicic@gmail.com“. Na e-poštu će pristići poruka sa aktivacijskim linkom te se pritiskom na njega otvara nastavak registracije na stranici JetBrainsa kao što je vidljivo na slici 50.



Slika 50. Nastavak registracije na JetBrains[5]

Nakon što su popunjeni svi podatci, pritiskom na gumb „Submit“, registracija je završena i može se preći na sljedeći korak. Sljedeći korak je otvoriti vlastiti profil na repozitoriju JetBrainsa gdje se vidi puni popis vlastitih programskih dodataka, komentara, organizacija i tokena. Pritiskom u gornji desni ugao na vlastito ime i prezime otvara se padajući izbornik i potrebno je odabrati stavku „Upload plugin“ kao što se vidi na slici 51.

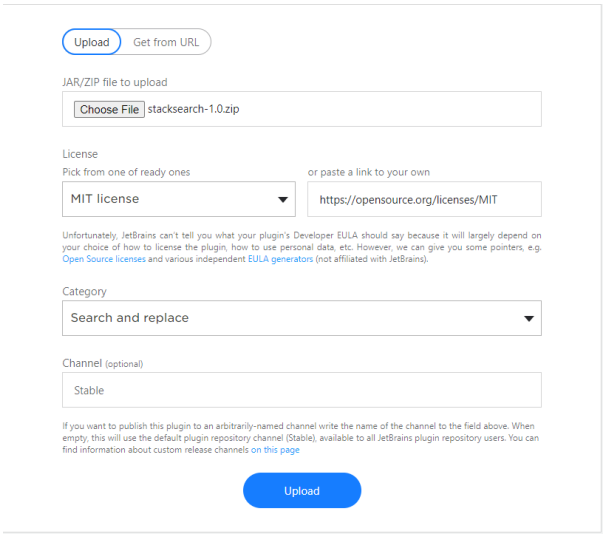
[5] Slika zaslona na: https://account.jetbrains.com/login



Slika 51. Prijenos programskog dodatka[6]

Pritiskom na prijenos programskog dodatka (eng. *Upload plugin*), otvara se stranica gdje je potrebno popuniti osnovne informacije poput licence koju će programski dodatak sadržavati s kojom će se svaki korisnik morati složiti ako želi koristiti taj dodatak te kategoriju u koju programski dodatak pripada. Odabrana licenca za ovaj vlastiti dodatak je MIT licenca jer je ovaj vlastiti programski dodatak otvorenog koda i svima je dopušteno koristiti, uređivati, kopirati i sl. što je upravo ono što ova licenca dozvoljava prema njihovoj službenoj dokumentaciji [35].

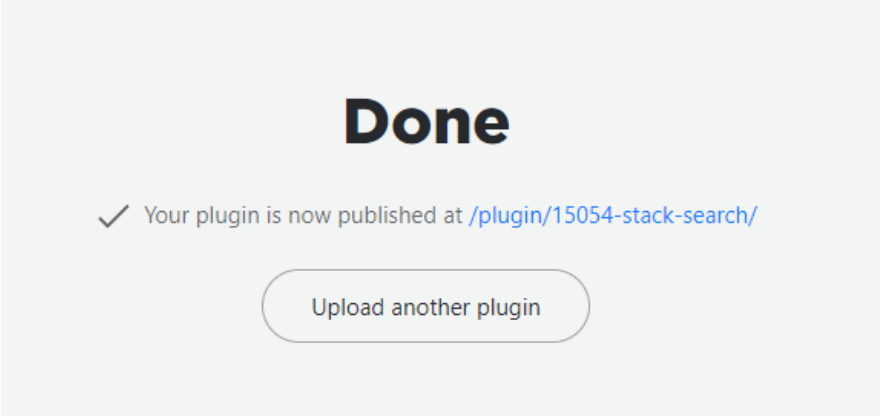
Nakon popunjavanja svih traženih podataka kao na slici 52. potrebno je pritisnuti gumb prenesi (eng. *Upload*) i sačekati da se završi prijenos na stranice JetBrainsa.



Slika 52. Konačni korak objave programskog dodatka[7]

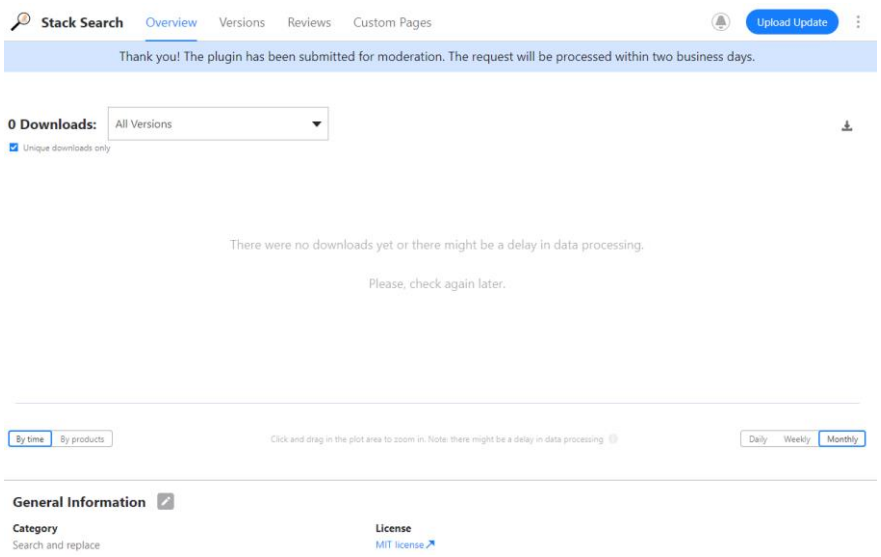
[6] Slika zaslona na: https://plugins.jetbrains.com/author/me
[7] Slika zaslona na: https://plugins.jetbrains.com/plugin/add

Nakon što je završio prijenos programskog dodatka, prikazana je povratna poruka o (ne)uspješnosti objave dodatka u repozitorij. Ukoliko je proces bio uspješan, vidljiva je i poveznica koja je dodijeljena vlastitom programskom dodatku kao što se vidi na slici 53.



Slika 53. Povratna poruka uspješno prenesenog programskog dodatka

Otvaranjem dodijeljene poveznice prikazan je vlastiti programski dodatak, njegove osnovne informacije i povratna poruka koja govori kako će programski dodatak pregledati moderatori i ukoliko ga odobre, korisnici će ga moći skinuti upravo sa prijašnje spomenute poveznice kao što je vidljivo na slici 54.



Slika 54. Povratna poruka nakon objave programskog dodatka u repozitoriju

Proces potvrde programskog dodatka traje oko 2 radna dana nakon čega je programski dodatak objavljen ukoliko zadovoljava kriterije i ne sadrži ništa maliciozno u programskom kodu. Ovim korakom je završeni cijeli proces izrade i objave programskog dodatka, naknadni postupci nakon što je odobren programski dodatak mogu biti dodavanje nove inačice programskog dodatka, mijenjanje osnovnih informacija ili dodavanje dodatne stranice kao što se vidi na slici 54. pod stavkom „Custom Pages“. Programski dodatak se nakon odobrenja može pronaći kroz tražilicu repozitorija koristeći ključne riječi Stack, Search, Stackoverflow jer te riječi su sadržane u naslovu i opisu programskog dodatka. Naslov i opis dodatka se mogu naknadno promijeniti tako da se željeni podatci unutar konfiguracijske datoteke plugin.xml izmjene i dodaju na JetBrains repozitorij kao ažuriranje (eng. *Update*) programskog dodatka.

5. Zaključak

Primjena programskih dodataka je široka i postoje različiti razlozi zašto bi se netko odlučio koristiti ili čak razviti dodatak, jedan od čestih razloga je povećanje produktivnosti. Određeni procesi uzimaju vrijeme i energiju što je moguće izbjeći baš kao što je i prikazano kroz razvijeni programski dodatak. Prednost razvoja programskih dodataka u IntelliJ IDEA razvojnom okruženju je što se mogu koristiti u više različitih programskih okruženja poput Android Studija koji je ciljano okruženje ovog rada. Programski dodatci se pojavljuju u besplatnoj i inačici koju je potrebno platiti.

Razvoj programskih dodataka generalno nije kompliciran proces što se i vidjelo prilikom prikazanih koraka u razvoju vlastitog programskog dodatka. Osnovni razvoj se svodi na definiranje ovisnosti unutar build.gradle datoteke, akcija i ostalih opisnih stavki unutar plugin.xml datoteke te pisanja same funkcionalnosti kroz nasljeđivanje klase AnAction. Ono što predstavlja problem u razvoju je mala količina dokumentacije za razvoj programskih dodataka za Android Studio, odnosno potrebno je izdvojiti dosta vremena kako bi se shvatili svi potrebni koraci. Osobno mi je bilo puno lakše shvatiti razvoj mobilnih aplikacija jer postoji mogućnost početka od skroz osnovnih stvari, dok je razvoj programskih dodataka dokumentiran za osobe sa predznanjem programiranja.

Objava programskog dodatka na repozitorij je veoma jednostavan proces bez nepotrebnih komplikacija. Sve što je potrebno za objaviti dodatak na repozitorij je prenijeti programski dodatak, definirati licencu i kategoriju te opcionalno kanal distribucije. Nakon što se ispune ovi koraci potrebno je do 2 radna dana kako bi JetBrains pregledao predani programski dodatak i odobrio ga ukoliko zadovoljava i nije maliciozan. Smatram da su programski dodatci nešto jako pozitivno s obzirom na to da imaju potencijal olakšati i ubrzati rad programera, a također i napraviti okruženje ugodnije ukoliko je dodatak kozmetičke naravi.

Popis literature

[1] Developers Android (2020) „Create an Android library“ [na internetu]. Dostupno: https://developer.android.com/studio/projects/android-library [pristupano 23.06.2020.]

[2] Gradle (2020) „Using Gradle Plugins“ [na internetu]. Dostupno: https://docs.gradle.org/current/userguide/plugins.html [pristupano 26.06.2020.]

[3] Jet Brains (2020) „Auto Java Code Suggestions“ [na internetu]. Dostupno: https://plugins.jetbrains.com/plugin/14070-auto-java-code-suggestions/versions [pristupano 26.06.2020.]

[4] Developers Android (2020) „Meet Android Studio“ [na internetu]. Dostupno: https://developer.android.com/studio/intro [pristupano 26.06.2020.]

[5] Developers Android (2020) „Migrating from IntelliJ“ [na internetu]. Dostupno: https://developer.android.com/studio/intro/migrate#migrating_from_intellij [pristupano 27.06.2020.]

[6] Android Studio početni zaslon [slika] (2020) Dostupno: https://developer.android.com/training/basics/firstapp/images/studio-welcome_2x.png?hl=zh-tw [pristupano 27.06.2020.]

[7] „Kotlin on Android. Now official“ (17.05.2020.). Jet Brains Blog [Na internetu]. Dostupno: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/ [pristupano 27.06.2020.]

[8] D. Jemerov, „IntelliJ IDEA and Android Studio FAQ“, 2013. [Na internetu]. Dostupno: https://blog.jetbrains.com/idea/2013/05/intellij-idea-and-android-studio-faq/ [pristupano 28.06.2020.]

[9] Kotlin (2020) „FAQ“ [na internetu]. Dostupno: https://kotlinlang.org/docs/reference/faq.html [pristupano 28.06.2020.]

[10] IntelliJ (2020) „Choose your edition“ [na internetu]. Dostupno: https://www.jetbrains.com/idea/features/editions_comparison_matrix.html [pristupano 28.06.2020.]

[11] Jet Brains (2020) „Gradle Killer“ [na internetu]. Dostupno: https://plugins.jetbrains.com/plugin/7794-gradle-killer [pristupano 30.06.2020.]

[12] Izgled programskog dodatka Gradle Killer [slika] (2020) Dostupno: https://miro.medium.com/max/441/1*3Y4QdpUKbnb6fbetQqwG2A.png [pristupano 30.06.2020.]

[13] Jet Brains (2020) „Rainbow Brackets“ [na internetu]. Dostupno: https://plugins.jetbrains.com/plugin/10080-rainbow-brackets [pristupano 30.06.2020.]

[14] Primjer dodatka Rainbow Brackets [slika] (2020) Dostupno: https://plugins.jetbrains.com/files/10080/screenshot_17372.png [pristupano 01.07.2020.]

[15] Jet Brains (2020) „Android Studio Plugin Development“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/products/android_studio.html [pristupano 03.07.2020.]

[16] Yener, M. i Dundar, O., 2016. *Expert Android Studio*. 1st ed. Indianapolis, IN: Wrox a Wiley brand.

[17] Jet Brains (2020) „Build Number Ranges“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/getting_started/build_number_ranges.html [pristupano 06.07.2020.]

[18] Jet Brains (2020) „Other Versions“ [na internetu]. Dostupno: https://www.jetbrains.com/idea/download/other.html [pristupano 06.07.2020.]

[19] „Mix Java and Kotlin in one project“ (08.05.2020) Jet Brains [na internetu]. Dostupno: https://www.jetbrains.com/help/idea/mixing-java-and-kotlin-in-one-project.html [pristupano 03.08.2020.]

[20] Gradle (2020) „Gradle vs Maven: Performance Comparison“ [na internetu]. Dostupno: https://gradle.org/gradle-vs-maven-performance/ [pristupano 03.08.2020.]

[21] Apache Maven Project (2020) „Guide to naming conventions on groupId, artifactId, and version“ [na internetu]. Dostupno: https://maven.apache.org/guides/mini/guide-naming-conventions.html#guide-to-naming-conventions-on-groupid-artifactid-and-version [pristupano 03.08.2020.]

[22] Developers Android (2020) „Configure your build“ [na internetu]. Dostupno: https://developer.android.com/studio/build#top-level [pristupano 04.08.2020.]

[23] IntelliJ Platform SDK DevGuide (2020) „Plugin Configuration File - plugin.xml“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/plugin_structure/plugin_configuration_file.html [pristupano 04.08.2020.]

[24] IntelliJ Platform SDK DevGuide (2020) „Plugin Extensions“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/plugin_structure/plugin_extensions.html [pristupano 05.08.2020.]

[25] IntelliJ Platform SDK DevGuide (2020) „Plugin Actions“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/plugin_structure/plugin_actions.html [pristupano 05.08.2020.]

[26] Kotlin (2020) „Packages and Imports“ [na internetu]. Dostupno: https://kotlinlang.org/docs/reference/packages.html [pristupano 06.08.2020.]

[27] Java Platform SE 8 (2020) „Package java.util“ [na internetu]. Dostupno: https://docs.oracle.com/javase/8/docs/api/java/util/package-summary.html [pristupano 06.08.2020.]

[28] „Write an Android Studio Plugin Part 1: Creating a basic plugin“ (16.11.2018) ProAndroidDev [na internetu]. Dostupno: https://proandroiddev.com/write-an-android-studio-plugin-part-1-creating-a-basic-plugin-af956c4f8b50 [pristupano 06.08.2020.]

[29] „All Questions“ (07.08.2020) Stack Overflow [na internetu]. Dostupno: https://stackoverflow.com/questions/ [pristupano 07.08.2020.]

[30] Maven (2020) „What is a SNAPSHOT version?“ [na internetu]. Dostupno: https://maven.apache.org/guides/getting-started/index.html#What_is_a_SNAPSHOT_version [pristupano 07.08.2020.]

[31] Boxy SVG. *Boxy SVG Editor* (inačica 3.43.2) (2020) [na internetu]. Dostupno: https://boxy-svg.com/app [pristupano 15.08.2020.]

[32] IntelliJ Platform SDK DevGuide (2020) „Adding Plugin Logo Files to a Plugin Project“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/plugin_structure/plugin_icon_file.html#adding-plugin-logo-files-to-a-plugin-project [pristupano 15.08.2020.]

[33] IntelliJ Platform SDK DevGuide (2020) „Modules Available in All Products“ [na internetu]. Dostupno: https://jetbrains.org/intellij/sdk/docs/basics/getting_started/plugin_compatibility.html#modules-available-in-all-products [pristupano 17.08.2020]

[34] „1. Submit a request to sell plugins at the Marketplace“ (29.03.2020) JetBrains Marketplace [na internetu]. Dostupno: https://plugins.jetbrains.com/docs/marketplace/submit-a-request-to-sell-plugins-at-the-marketplace.html [pristupano 20.08.2020.]

[35] Open Source Initiative (2020) „The MIT License“ [na internetu]. Dostupno: https://opensource.org/licenses/MIT [pristupano 28.08.2020.]

Popis slika

Slika 1. Adblock, Google Chrome internet trgovina 2
Slika 2. Plugins, Android Studio..... 4
Slika 3. Android Studio početni zaslon [6]..... 5
Slika 4. Izgled početne stranice IntelliJ IDEA 6
Slika 5. Početni zaslon IntelliJ IDEA..... 7
Slika 6. Izgled programskog dodatka Gradle Killer [12]..... 9
Slika 7. Primjer dodatka Rainbow Brackets [14]..... 9
Slika 8. Broj preuzimanja Rainbow Bracketsa 10
Slika 9. Provjera inačice Android Studija..... 13
Slika 10. Prikaz detalja o inačici Android Studija..... 13
Slika 11. Odgovarajuća inačica IntelliJ IDEA 14
Slika 12. Dijagram nastajanja programskog dodatka u IntelliJ IDEA 15
Slika 13. Novi projekt..... 16
Slika 14. Opisne stavke projekta..... 17
Slika 15. Odabir lokacije pohrane projekta..... 18
Slika 16. Gradle proces izrade..... 18
Slika 17. Početna struktura projekta..... 19
Slika 18. Plugin.xml..... 20
Slika 19. Pozdravna poruka..... 21
Slika 20. Dodatak u izbornoj traci..... 22
Slika 21. Sadržaj klase programskog dodatka..... 23
Slika 22. Klasa NotificationDisplayType 24
Slika 24. Potpuna Plugin.xml datoteka..... 25
Slika 25. Uređena build.gradle datoteka..... 26
Slika 26. Testiranje programskog dodatka..... 26
Slika 27. Opisne stavke vlastitog dodatka 27
Slika 28. Opisni podatci u plugin.xml..... 28
Slika 29. Ikona vlastitog dodatka..... 28
Slika 30. Lokacija ikone programskog dodatka 29
Slika 31. Obavijest o promjenama koje nisu uvezene 29
Slika 32. Projektna struktura vlastitog dodatka..... 30
Slika 33. Greška u klasi..... 30
Slika 36. Ikona unutar Android Studija 33
Slika 37. Postavljanje Android Studija za testiranje..... 33

Slika 38. Tip instalacije Android Studija 34
Slika 39. Odabir teme Android Studija 34
Slika 40. Instalacija potrebnih komponenti za testiranje 35
Slika 41. Postavljanje projekta za testiranje 36
Slika 42. Označeni tekst prilikom testiranja 36
Slika 43. Rezultat testiranja dodatka 37
Slika 44. Uklanjanje linije koda iz build.gradle datoteke 38
Slika 45. Uspješno kompiliranje programskog dodatka 38
Slika 46. Dodavanje programskog dodatka u Android Studio 39
Slika 47. Lokacija programskog dodatka 39
Slika 48. Uspješno učitan programski dodatak 40
Slika 49. Registracija na JetBrains online trgovinu 41
Slika 50. Nastavak registracije na JetBrains 42
Slika 51. Prijenos programskog dodatka 43
Slika 52. Konačni korak objave programskog dodatka 43
Slika 53. Povratna poruka uspješno prenesenog programskog dodatka 44
Slika 54. Povratna poruka nakon objave programskog dodatka u repozitoriju 44

Popis programskih kodova

Programski kod 1. Akcija primjera programskog dodatka..... 22
Programski kod 2. Funkcija primjera programskog dodatka..... 24
Programski kod 3. Funkcija vlastitog programskog dodatka 31
Programski kod 4. Akcija vlastitog programskog dodatka 32

Prilozi

[1] Izvorni kod programskog dodatka dostupan je na GitHub servisu: https://github.com/kresohr/stacksearch

[2] Objavljeni programski dodatak je na JetBrains repozitoriju: https://plugins.jetbrains.com/plugin/15061-stack-search