SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
SMJER ISTRAŽIVAČKA INFORMATIKA
Ak. god. 2018./2019.

Srđan Vučković

Proširenja funkcionalnosti Android platforme

Diplomski rad

Mentor: dr.sc. Vedran Juričić, docent

Zagreb, siječanj 2019.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Sadržaj

1. Uvod

Rast popularnosti Android platforme rezultirala je širenjem zajednice razvojnih inženjera koji razvijaju za Android OS(operacijski sustav). Kako bi se olakšao razvoj za Android platformu i sačuvalo razvojne inženjere od pisanja velikih količina generičkog koda pojavila su se proširenja funkcionalnosti za Android platformu.

Rad govori o Android platformi i proširenjima funkcionalnosti, njihovoj upotrebi i uključivanju u aplikacije te njihovim prednostima i nedostatcima, a rad se posebno fokusira na ORM proširenja funkcionalnosti.

U radu se također pregledavaju osnovne značajke Android OS-a, razmatra zajednica otvorenog koda i implikacije koje zajednica ima na Android platformu. Nadalje, sagledavaju se prednosti i nedostatci programskih jezika Java i Kotlin, ukratko predstavljaju osnove korištenja okruženja Android Studio, te se uspoređuju Gradle i Maven alati za izgradnju i opisuje način korištenja emulatora u razvoju aplikacija.

U posljednjih nekoliko godina popularna proširenja funkcionalnosti opisana u radu koriste se na velikom broju projekata, većina grešaka povezanih s istim uklonjene su, a zajednica uključena u njihov razvoj narasla je i zbog toga su odabrana za prikaz u radu.

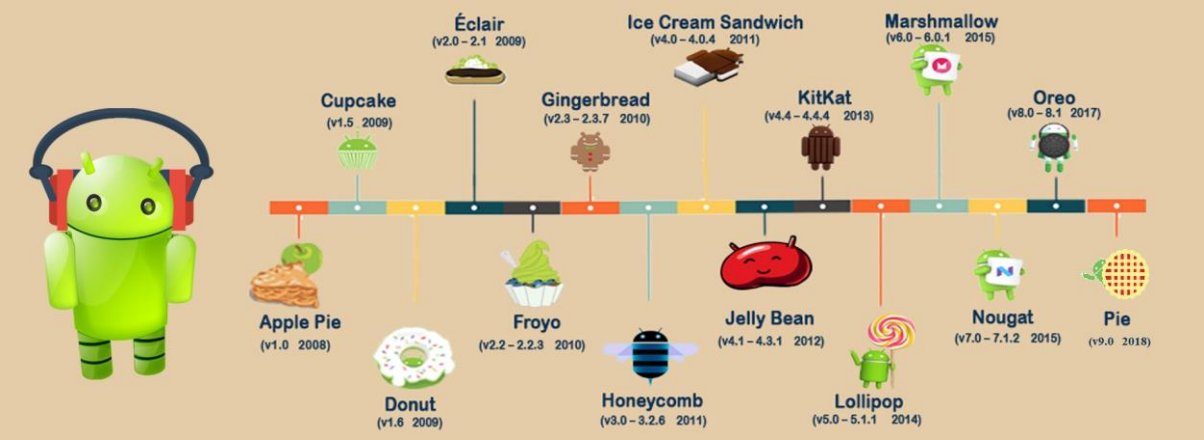
Prikazano je nekoliko najpopularnijih ORM rješenja, te dva pristupa koja ne koriste ORM. U radu je provedena analiza brzine CRUD (Create, Read, Update, Delete) operacija u tim proširenjima za Android platformu i objašnjene su prednosti i nedostatci pojedinih proširenja.

Cilj je ovog rada pregledno prikazati proširenja funkcionalnosti za Android platformu, s opširnijim prikazom ORM proširenja funkcionalnosti te prednosti i nedostatke istih u nadi da će rad biti koristan razvojnim inženjerima pri definiranju rješenja s obzirom na specifične potrebe projekta.

2. Android platforma

Android je mobilni operacijski sustav koji je razvio Google. Bazira se na modificiranoj verziji Linux jezgre operacijskog sustava i primarno je dizajniran za mobilne uređaje koji imaju ekran na dodir kao što su pametni telefoni i tableti. No, Google je razvio i Android TV za televizore, Android Auto za automobile, te Wear OS za ručne satove, svaki od njih sa specijaliziranim korisničkim sučeljem. Varijante Androida se koriste na igraćim konzolama, digitalnim kamerama i osobnim računalima.

Inicijalno razvijen od strane Android inc. koji je Google kupio 2005. godine, Android je otkriven 2007. godine, a prvi komercijalni uređaj lansiran je u rujnu 2008. godine. Operacijski sustav je od tada prošao kroz brojna izdanja, kao što je vidljivo na slici 1, a najnovija aktualna verzija je 9 „Pie“, objavljena u kolovozu 2018. godine. Izvorni kod Androida je poznat kao Android Open Source Project (AOSP) i primarno je licenciran pod Apache licencom.



Slika 1. Verzije Android OS-a[1]

Android je povezan s paketom vlasničkog softvera koji je razvio Google, a zove se Google Mobile Services[2] (GMS) i često dolazi instaliran na uređajima. Uključuje preglednik Google Chrome i Google Search, te uvijek uključuje temeljne aplikacije za servise kao što su Gmail i Google Play trgovina. Google Play je uključen kao platforma za distribuciju i razvoj. Aplikacije koje se mogu naći na Play trgovini su licencirane od strane proizvođača i certificirane od strane standarda koje je postavio Google, ali AOSP se koristi kao osnova Android ekosustava. Android je najprodavaniji operacijski sustav za pametne telefone u svijetu od 2011. godine, za

[1] URL: http://magivatech.com/blog/info/android-versions (23.01.2019.)
[2] URL: https://www.android.com/gms/ (13.12.2018.)

tablete od 2013. godine. Ima preko dvije milijarde aktivnih korisnika mjesečno, najveću bazu korisnika od svih operacijskih sustava i od lipnja 2018. godine Google Play trgovina ima preko 3.3 milijuna aplikacija.[3]

2.1. Osnovne značajke

- otvorenost – razvojnom inženjeru omogućava potpunu slobodu u razvoju novih i već postojećih aplikacija, a proizvođaču uređaja slobodno korištenje i prilagodbu platforme bez plaćanja autorskih prava
- sve aplikacije su ravnopravne – ne postoji razlika između osnovnih jezgrenih aplikacija uređaja i dodatnih aplikacija
- automatsko upravljanje životnim ciklusom aplikacije – omogućava nadzor pokretanja i izvršavanja aplikacija na sistemskoj razini optimiziranim korištenjem memorije i procesorske snage. Krajnji korisnik više ne brine o gašenju određenih aplikacija prije pokretanja drugih
- brisanje granica između "klasičnih" aplikacija – mogućnost razvoja novih i inovativnih aplikacija temeljenih na međusobnoj suradnji različitih tehnologija
- brz i jednostavan razvoj aplikacija – omogućen bogatom bazom korisnih programskih biblioteka (engl. *libraries*), proširenjima funkcionalnosti i alata za izradu aplikacija
- visokokvalitetni grafički prikaz i zvuk – podržana 2D vektorska i 3D OpenGL (engl. *Open Graphics Library*) grafika, te ugrađeni kodeci svih često korištenih audio i video formata.

2.2. Zajednica otvorenog koda

Kod Android operacijskog sustava objavio je Google pod licencom otvorenog koda, a njegova otvorena priroda privukla je veliku zajednicu razvojnih inženjera i entuzijasta koji koriste otvoreni kod kao osnovu za projekte koji su pokretani od strane zajednice. Projekti ove vrste pružaju ažuriranja za starije uređaje i dodaju nove značajke za napredne korisnike ili donose Android na uređaje koji su originalno bili lansirani s drugim operacijskim sustavima. Izdanja razvijena od strane zajednice često donose nove značajke i ažuriranja za uređaje mnogo brže nego što ih izdaju originalni proizvođači uređaja, s usporednom razinom kvalitete. Izdanja razvijena od strane zajednice dolaze korijenski modificirani i omogućavaju korisnicima

[3] URL: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (13.12.2018.)

napredne mogućnosti za podešavanje frekvencije procesora uređaja i slično. Najkorišteniji ugrađeni softver bio je CyanogenMod[4], koji je sada ukinut, a naslijedio ga je LineageOS.

U prošlosti proizvođači uređaja nisu podržavali razvoj nezavisnog ugrađenog softvera. Proizvođači su izrazili zabrinutost oko nepropisnog funkcioniranja uređaja koji pokreću neslužben softver i komplikacija do kojih može doći korištenjem istog. S vremenom su proizvođači kao što su HTC, Samsung, Motorola i Sony omekšali stavove oko nezavisnog razvijanja softvera i čak pružaju podršku kod nezavisnog razvoja. Kao rezultat, tokom vremena smanjila su se ograničenja na hardveru koja onemogućuju korištenje neslužbenog ugrađenog softvera, korisnici koji žele koristiti ugrađeni softver često se moraju odreći garancije uređaja.

2.3. Aplikacije

Aplikacije koje proširuju funkcionalnost uređaja napisane su koristeći Android Software development kit (SDK) i često s Java programskim jezikom. Java se može kombinirati s C/C++ programskim jezicima, a u svibnju 2017. godine Google je objavio službenu podršku za razvoj Android aplikacija u Kotlin programskom jeziku.[5]

SDK uključuje opsežan set alata za razvoj, uključujući ispravljač grešaka, softverske biblioteke, emulator, dokumentaciju, uzorak koda i vodič. Inicijalno Google je podržavao Eclipse IDE (integrirano razvojno okruženje, engl. *Integrated development environment*) koji je koristio proširenje funkcionalnosti Android Development Tools (ADT). U prosincu 2014. godine Google je objavio Android Studio, koji je baziran na IntelliJ IDEA, kao primarno okruženje za razvoj Android aplikacija.

Android ima rastući izbor nezavisnih aplikacija koje se mogu preuzeti i instalirati koristeći APK (Android application package) datoteku ili preuzimanjem s trgovine aplikacija koja omogućava korisnicima instalaciju, ažuriranje i uklanjanje aplikacija s njihovih uređaja. Google Play trgovina je primarna trgovina instalirana na Android uređaje koji se slažu s Googleovim zahtjevima za kompatibilnost i licencama. Google Play trgovina omogućava korisnicima preuzimanje aplikacija razvijenih od strane nezavisnih razvojnih inženjera, a od srpnja 2013. godine instalirano je 50 milijardi aplikacija.[6]

[4] URL: https://www.androidpolice.com/2012/05/28/cyanogenmod-has-been-installed-over-2-million-times-doubles-install-numbers-since-january/ (14.12.2018.)

[5] URL: https://techcrunch.com/2017/05/17/google-makes-kotlin-a-first-class-language-for-writing-android-apps/ (14.12.2018.)

[6] URL: https://mashable.com/2013/07/24/google-play-1-million/?europe=true#gwIL7YDq_iqs (14.12.2018.)

Zbog otvorene prirode Androida postoji nekoliko nezavisnih trgovina za aplikacije koje pružaju zamjene za uređaje kojima nije dozvoljeno da budu prodani s Google Play trgovinom. Pružaju i aplikacije koje nisu ponuđene na Google Play trgovini zbog kršenja Googleovih pravila ili drugih razloga. Primjeri tih trgovina su Amazon App trgovina, GetJar i SlideMe.

2.4. Java[7]

Java je programski jezik za razvoj softvera na raznim platformama. Kad razvojni inženjer piše Java aplikaciju, kompilirani kod (engl. *bytecode*) može se pokrenuti na većini operacijskih sustava, uključujući Windows, Linux i Mac OS. Na Android operacijskom sustavu kompiliranjem Java bytecodea dobiva se Dalvik bytecode kojeg Android pokreće. Java dosta svoje sintakse dijeli s C i C++ programskim jezicima. Java se može pokrenuti na većini preglednika te postoji jako velik broj proširenja funkcionalnosti za Java programski jezik u Android SDK-u. Postao je jedan od standardnih programskih jezika u poslovanju, zahvaljujući multifunkcionalnosti, nazadnoj kompatibilnosti verzija i više platformskom softveru koji je pogodan za vrlo velik raspon zadataka. Neki od zadataka za koji se koristi su razvoj velikih portala i online dućana, stvaranje mobilnih aplikacija i primjena u korporativnim programima. Negativna strana je enterprise orijentacija Jave te potreba za podrškom svih postojećih projekata u vrijeme izlaska novih verzija što usporava njezin razvoj.

Prednosti i nedostatci Java programskog jezika[8]

Prednosti

- Lagan za učenje i razumijevanje
- Fleksibilan
- Dobar izbor za više platformske aplikacije
- Android se oslanja na Javu, Android SDK sadrži mnogo standardnih Java biblioteka
- Java ima velik ekosustav otvorenog koda, dijelom za to zaslužno je Googleovo prihvaćanje JVM-a(Java Virtual Machine) za Android
- Java aplikacije su kompaktnije u usporedbi s Kotlinom, osiguran je brži proces izgradnje od Kotlina
- Veliki broj razvojnih inženjera

[7] URL: https://www.techopedia.com/definition/3927/java (27.09.2018.)
[8] URL: https://www.netguru.co/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app (30.09.2018.)

- Jedan od najraširenijih programskih jezika[9]

Nedostatci

- Java ima ograničenja koja uzrokuju probleme u Android API(Application Programming Interface) dizajnu
- Kao opširan jezik Java zahtijeva pisanje mnogo više koda, što donosi veći rizik od grešaka
- Sporiji je u usporedbi s ostalim jezicima i zahtijeva puno memorije

2.5. Kotlin[10]

Kotlin se odlično slaže s razvojem Android aplikacija zato što donosi sve prednosti modernog jezika na Android platformu bez donošenja novih ograničenja. Potpuno je kompatibilan s JDK-om (Java Development Kit) što osigurava da se aplikacije napisane u Kotlinu mogu pokretati na starijim Android uređajima. Kotlin aplikacija jednako je brza kao i Java aplikacija zbog slične strukture kompiliranog koda, a kod koji koristi lambde često je brži nego isti kod napisan u Java programskom jeziku. 100% je interoperabilan s Javom, što omogućava upotrebu svih postojećih Android proširenja funkcionalnosti u Kotlin aplikaciji. Za Java razvojnog inženjera započeti razvijati s Kotlinom nije komplicirano. Automatski konverter Jave u Kotlin, uključen u proširenje funkcionalnosti Kotlin, pomaže s prvim koracima. Jedna od najvažnijih kvaliteta Kotlina je ekspresivnost što znači da se može napisati puno više s manje koda. Kotlin je zapravo objektno orijentiran, a ne potpuno funkcionalan jezik. Kao i mnogi moderni jezici, koristi razne koncepte iz funkcionalnog programiranja, kao što su lambda izrazi da bi riješio probleme na lakši način. Koristi proširene funkcije što znači da se bilo koja klasa može proširiti s novim značajkama čak i ako nemamo pristup izvornom kodu.

Prednosti i nedostatci Kotlin programskog jezika[11]

Prednosti

- Postao je popularan u razvoju za Android, ali se također koristi u projektima pozadinskih servisa (engl. *back-end*) kao što su Spring 5

[9] URL: https://plumbr.io/blog/java/how-many-java-developers-in-the-world (30.09.2018.)

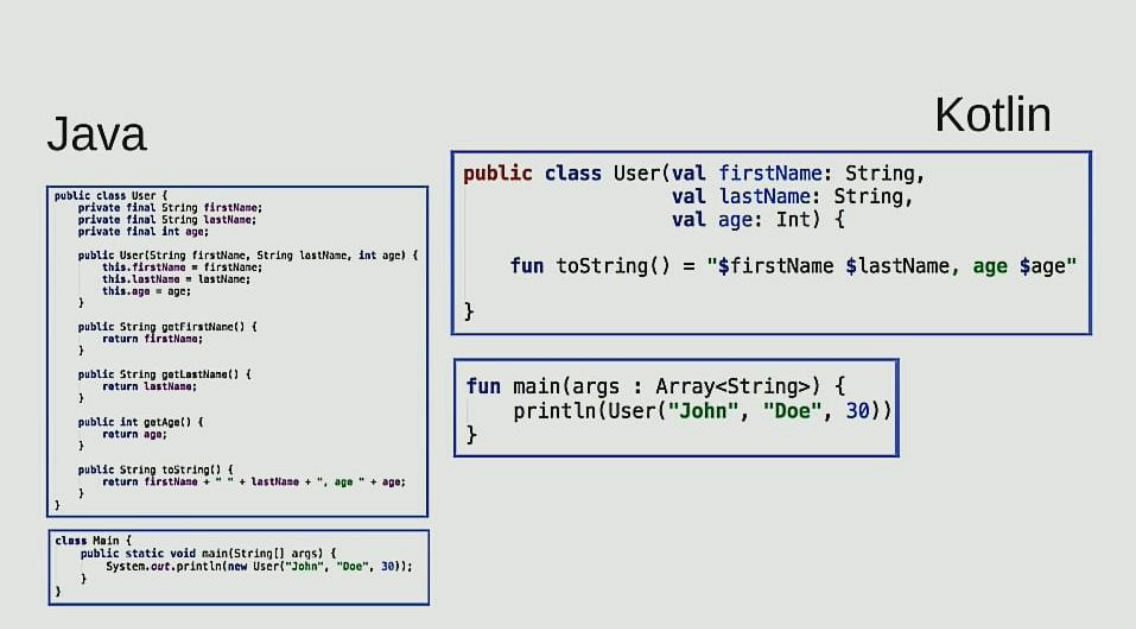
[10] URL: https://kotlinlang.org/docs/reference/android-overview.html (27.09.2018.)

[11] URL: https://www.netguru.co/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app (30.09.2018.)

- Prelazak s Jave na Kotlin je jednostavan, instalira se Kotlin proširenje funkcionalnosti, uključi se u Gradle datoteku izgradnje i stisne 'Convert'
- Uključuje pametne funkcije ekstenzija što pomaže razvojnim inženjerima da izgrađuju dobre API-eve
- Puno je koncizniji od Jave, kao što je vidljivo na slici 2, što znači manju mogućnost pogreške
- Interoperabilan s Javom
- Besplatan je. (osim učenja i treninga)

Nedostatci

- Poprilično visoka krivulja učenja kad se cijeli tim prebacuje na Kotlin



Slika 2. Prikaz količine Java i Kotlin koda za dvije jednake funkcionalnosti[12]

2.6. Android Studio

Android Studio[13] je službeni IDE za razvoj Android aplikacija, baziran na IntelliJ IDEA. Povrh IntelliJevog moćnog urednika koda i razvojnih alata,

Android studio pruža značajke koje unapređuju produktivnost razvoja Android aplikacija.

[12] URL: https://medium.com/@snrawoof93/java-vs-kotlin-android-development-7550660dc2b0 (30.09.2018.)
[13] URL: https://developer.android.com/studio/intro/ (10.01.2019.)

Njegov fleksibilan sustav izgradnje baziran je na Gradleu. Ima brz emulator s velikim brojem značajki i ujedinjeno okruženje u kojemu je moguće razvijati za sve Android uređaje. Značajno je i trenutačno pokretanje promjena za pokrenutu aplikaciju bez izgradnje novog APK-a.

Android studio ima predloške koda i integraciju s Githubom kao pomoć u izgradnji uobičajenih značajki aplikacija te za uvoz primjera koda. Od alata, tu su opsežni alati i okviri za testiranje te alati za pregled performansi, iskoristivosti, kompatibilnosti verzija i drugih problema.

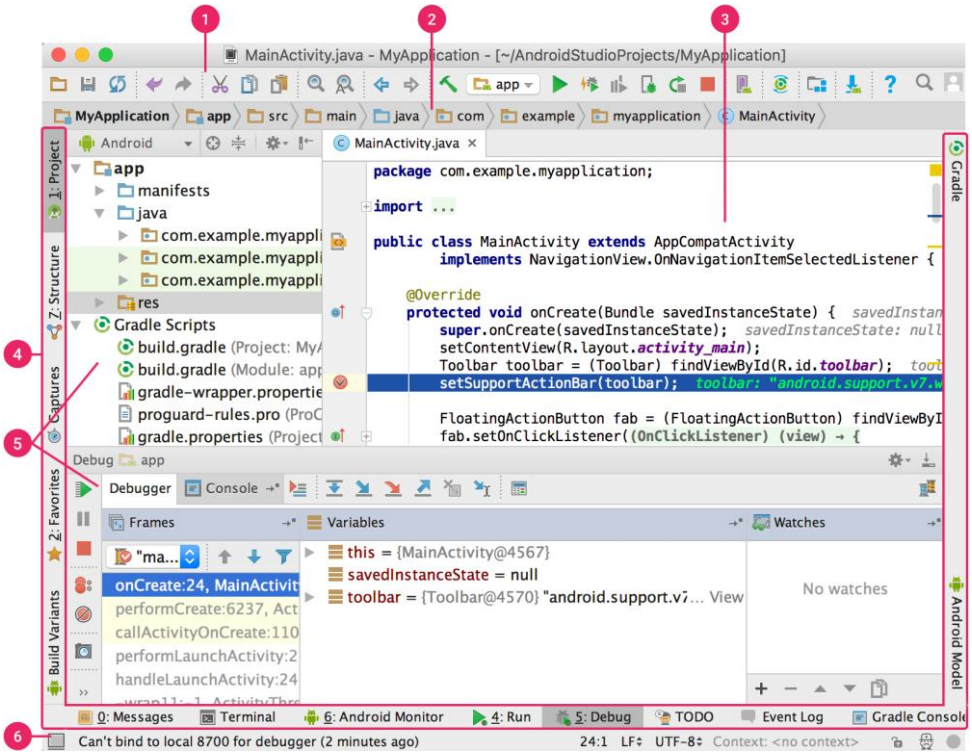
Valja istaknuti i podršku za C++ i NDK te ugrađenu podršku za Google Cloud platformu.

Svaki projekt u Android Studiju sadrži jedan ili više modula s datotekama izvornog koda i datotekama resursa. Tipovi modula uključuju:

- Modul Android aplikacije
- Modul biblioteke
- Modul Google aplikacijskog mehanizma

Android Studio prikazuje projektne datoteke u pogledu Android projekt. Taj pogled je organiziran prema modulima da pruži brzi pristup izvornim datotekama projekta.

2.6.1. Korisničko sučelje



Slika 3. Android Studio korisničko sučelje[14]

[14] URL: https://developer.android.com/studio/intro/ (10.01.2019.)

1. Alatna traka omogućuje veliki raspon radnji, uključujući pokretanje aplikacije i Android alata.
2. Navigacijska traka pomaže s navigacijom kroz projekt i s njom se otvaraju datoteke za uređivanje. Pruža kompaktan pogled strukture vidljive u projektnom prozoru.
3. Prozor urednik koristi se za kreiranje i modificiranje koda. Ovisno o trenutnom tipu datoteke, urednik se mijenja.
4. Prozor alatne trake prožima se po vanjskom rubu IDE prozora i sadrži gumbe koji omogućavaju proširenje ili smanjenje individualnih prozora alata.
5. Alatni prozor daje pristup specifičnim zadatcima kao što su upravljanje projektom, pretraživanje, kontrola verzije, itd.
6. Statusna traka prikazuje status projekta i IDE-a, kao i bilo kakva upozorenja i poruke.

2.6.2. Značajke i funkcionalnost

Android Studio ima tri vrste dovršavanja koda kojima se može pristupiti iz alatne trake ili prečacima na tipkovnici.

Osnovno dovršavanje

Prikazuje osnovne prijedloge za varijable, tipove, metode, izraze, itd. Ako se dvaput pozove osnovno dovršavanje prikazuje se više rezultata, uključujući privatne članove i neuvezene statične članove.

Pametno dovršavanje

Prikazuje relevantne opcije bazirane na kontekstu. Pametno dovršavanje razumije očekivani tip podataka i njihov tok. Ako se dvaput pozove pametno dovršavanje prikazuje se više rezultata, uključujući i nizove.

Dovršavanje izraza

Dovršava trenutni izraz umjesto razvojnog inženjera, dodaje zagrade i formatiranje.

Značajke najnovijeg stabilnog izdanja Android Studija 3.3[15]

Razvoj

- Navigacijski urednik
- IntelliJ 2018.2.2 ažuriranje platforme

[15] URL: https://medium.com/mindorks/project-marble-android-studio-stable-release-3-3-d7b177ccbd2c (15.01.2018.)

- Kotlin 1.3.11 Ažuriranje
- Bolja podrška za C++
- Novo ažuriranje projektnog čarobnjaka
- Brisanje nekorištenih IDE datoteka
- IDE povratne informacije korisnika

Izgradnja

- Unaprijeđeno inkrementalno kompiliranje Jave kod korištenja anotacijskih procesora
- Konfiguracija lijenih zadataka
- Sinkronizacija projektnih varijanti
- Android snop aplikacija podržava instantne aplikacije

Testiranje

- Pokretanje više instanci emulatora
- Android 9 Pie – podrška Emulatora
- Unapređenje brzine spremanja ekrana na emulatoru
- Optimizacija
- Unapređenje alata za profiliranje
- Unapređenje alata za profiliranje memorije, mreže i procesora

2.7. Gradle[16]

Gradle je alat za automatizaciju otvorenog koda koji se bazira na fleksibilnosti i performansama.

Gradle služi kako bi uzeo izvorne programske datoteke (.java ili .xml) te na njih primijenio alate kompiliranja i povezivanja. Grupira sve te datoteke u jednu komprimiranu .apk datoteku s kojom Android sustav zna raditi. Gradle skripte izgradnje napisane su koristeći Groovy ili Kotlin DSL(engl. *Domain Specific Language*).

Gradle je izrazito podesiv, brz i moćan. Podržava veliki broj glavnih IDE-a kao što su Android Studio, Eclipse, IntelliJ IDEA, Visual Studio 2017 i XCode. Također se može pozvati preko sučelja komandne linije u terminalu ili preko servera za kontinuiranu integraciju. Dizajniran je za više projektne izgradnje, koje mogu jako narasti. Podržava inkrementalne izgradnje tako što inteligentno određuje koji dijelovi drveta razvoja (engl. *build tree*) su najnoviji.

[16] URL: https://docs.gradle.org/current/userguide/userguide.html (26.09.2018.)

2.8. Maven[17]

Maven je alat koji se koristi za izgradnju i upravljanje bilo kakvim projektom baziranom na Java programskom jeziku. Glavni cilj Mavena je da dopusti razvojnom inženjeru da shvati kompletnu sliku procesa razvoja u najkraćem mogućem vremenu.

Olakšava proces izgradnje – Maven ne eliminira potrebu za poznavanjem inherentnih mehanizama, ali štiti korisnika od detalja.

Daje uniformni sustav izgradnje – Dopušta projektu da se izgradi sa setom proširenja funkcionalnosti koje dijele svi projekti koji koriste Maven.

Pruža kvalitetnu informaciju o projektu – Pruža dokument sa zapisom promjena koji se kreira direktno iz kontrole izvora, listu zavisnosti i testove.

Pruža osnovna pravila i najbolje prakse razvoja softvera – Zadržava testni kod u odijeljenom, ali paralelnom stablu izvora (engl. *source tree*), koristi konvencije imenovanja testova da bi locirao i izvršio testiranje i priprema okolinu za testiranje.

Omogućuje transparentnu migraciju novih značajki – Dozvoljava lagan način klijentima da ažuriraju instalacije i iskoriste prednosti promjena koje nisu napravljene od strane Mavena.

Postoje temeljne razlike u pristupu ova dva alata izgradnji[18]. Gradle se bazira na grafovima zavisnosti zadataka, u kojem su zadaci oni koji „obavljaju posao“. Maven se bazira na fiksnom i linearnom modelu u fazama. Kod Mavena ciljevi su pridruženi s projektnim fazama i služe sličnoj funkciji kao Gradleovi zadaci, time što su kod Mavena ciljevi ti koji „obavljaju posao“.

Oba sustava dopuštaju modularnu izgradnju koja se događa u paraleli. Gradle dozvoljava inkrementalnu izgradnju jer provjerava koji su zadaci ažurirani, a koji nisu.

Oba sustava također preuzimaju zavisnosti iz repozitorija. Maven koristi Maven Central, a Gradle koristi JCenter. Google je prepoznao da je Gradle jedan od najnaprednijih sustava za izgradnju na tržištu i službeno se koristi za Android razvoj. Također Gradle je sustav baziran na proširenjima funkcionalnosti.

[17] URL: https://maven.apache.org/what-is-maven.html (26.9.2018.)

[18] URL: https://dzone.com/articles/gradle-vs-maven (27.09.2018.)

2.9. Emulatori

Emulator[19] je softver koji omogućava jednom računalnom sustavu (domaćin) da imitira funkcije drugog računalnog sustava (gost). Omogućava domaćinu da pokreće aplikacije, alate, periferne uređaje i ostale komponente namijenjene za gost sustav. Postoje različite vrste emulatora koji repliciraju stvari poput hardvera, softvera, operacijskog sustava ili CPU-a (procesora). U većini slučajeva emulira se arhitektura hardvera da bi se postigla okolina slična sustavu gosta.

Emulator rekreira originalnu računalnu okolinu uz pomoć softvera i hardvera. Proces kreacije autentičnog emulatora je kompleksan i zahtjeva puno vremena. Jednom kad se napravi pruža autentični doživljaj originalnog računalnog sustava bez potrebe za istim.

Upotreba emulatora za mobilne sustave je nešto što većina stručnjaka smatra kritičnim za razvoj. Pomaže razvojnim inženjerima da dobiju osjećaj za plan i shemu mobilnih proizvoda u ranoj fazi razvoja, prije nego zapravo počnu pisati kod za projekt.

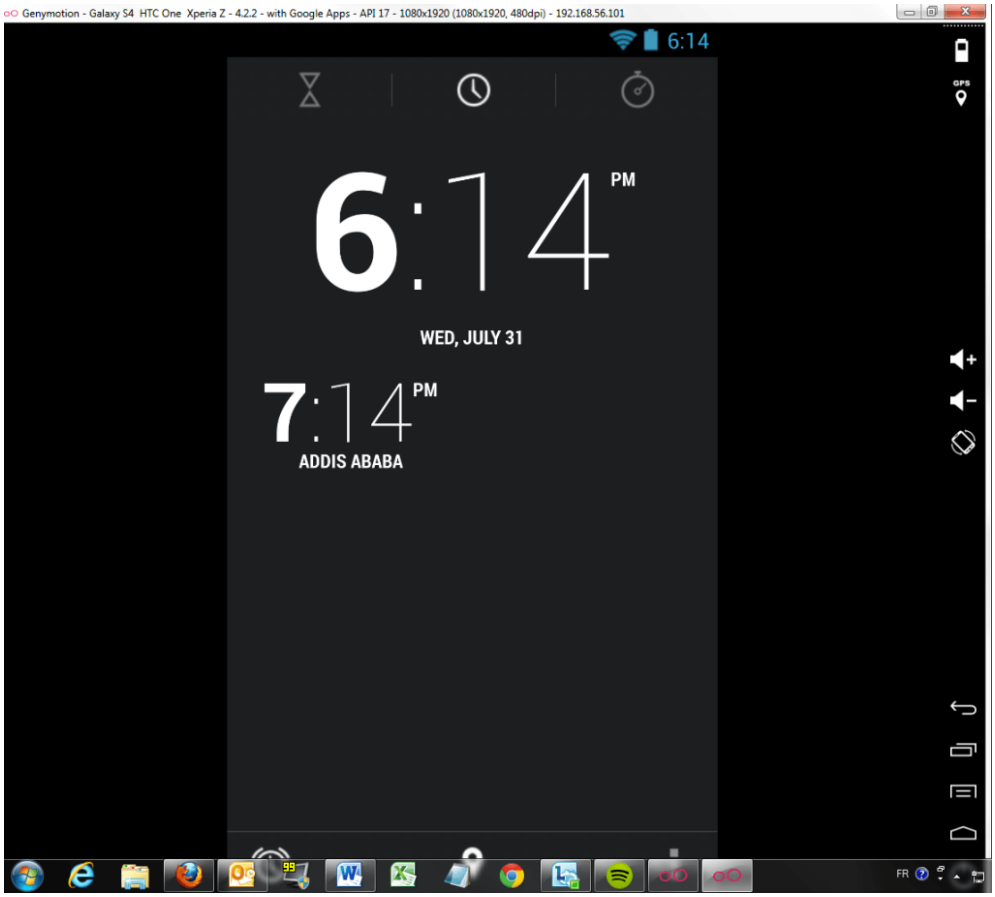
Najpopularniji emulatori za Android su Bluestacks, GenyMotion, Nox, KOPLAYER, Andy i MeMU. Ukratko će biti predstavljena dva, Genymotion kao emulator najpogodniji za razvoj i Bluestacks kao emulator koji se koristi za zabavu.

GenyMotion

GenyMotion[20] je Android emulator koji se sastoji od kompletnog seta senzora i značajki s kojima se vrši interakcija s Android virtualnom okolinom. Koristi se za testiranje Android aplikacija na velikom rasponu virtualnih uređaja s kojima se može razvijati, testirati ili raditi demonstracije. Lagano se može mijenjati virtualni uređaj i inačica operacijskog sustava Android. Genymotion je brz, jednostavan za instalirati i lagan za korištenje. Dostupan je za Windows, Mac OS i Linux operacijske sustave. Genymotion se može smatrati ozbiljnijim rješenjem od Bluestacksa koji će biti predstavljen u sljedećem poglavlju, također ima profesionalnije sučelje kao što je vidljivo na slici 3.

[19] URL: https://www.techopedia.com/definition/4788/emulator (27.09.2018.)

[20] URL: https://docs.genymotion.com/latest/pdf/PDF_User_Guide/Genymotion-2.12-User-Guide.pdf (27.09.2018.)



Slika 4. Genymotion ekran[21]

Bluestacks

Bluestacks[22] je jedan od najpoznatijih Android emulatora, vrlo je kvalitetan i pouzdan. Napravljen je da bude što lakši za koristiti i izgleda i ponaša se baš kao Android na tabletu ili pametnom telefonu. Postoji plaćena i besplatna verzija. Besplatna verzija dolazi s nekoliko reklama i sponzoriranih aplikacija. Bluestacks se primarno koristi za mobilne igre i zapravo je sučelje za preuzimanje, instaliranje i korištenje igara i ostalih aplikacija. Bluestacks se više koristi kao emulator za igre i aplikacije, a manje kao emulator za razvoj. Kao što je vidljivo na slici 4, Bluestacks sučelje izgleda više kao sučelje za igranje igara nego emulator za razvijanje Android aplikacija.

[21] URL: https://genymotion.en.softonic.com/ (23.01.2019.)

[22] URL: https://www.bluestacks.com/ (27.09.2018.)

Slika 5. Bluestacks ekran[23]

[23] URL: https://support.bluestacks.com/hc/en-us/articles/360013732972-How-to-install-an-app-on-BlueStacks-4- (27.09.2018.)

3. Proširenja funkcionalnosti

Proširenje funkcionalnosti[24] (engl. *plugin, add-in, add-on, addon* ili *extension*) je softverska komponenta sa specifičnim značajkama koja se dodaje u okruženje kao što je IDE (Android Studio). Kad okruženje ili aplikacija podržavaju proširenja funkcionalnosti, omogućuje se prilagođavanje. Najčešće korišteni primjeri proširenja funkcionalnosti su u internet preglednicima, a dodaju značajke kao što su skeneri virusa, tražilice ili mogućnost korištenja novog oblika datoteka kao što je novi video format. Proširenja funkcionalnosti razvijena su s ciljem da poboljšaju određene značajke programa.

U nastavku se navode neki od razloga zašto se koriste proširenja funkcionalnosti. Primjerice, ona omogućuju neovisnim (engl. *third-party*) razvojnim inženjerima da razvijaju značajke koje unapređuju aplikaciju. Također, koriste se kao podrška za dodavanje novih značajki te za smanjenje veličine aplikacije. Napokon, korisna su i za odvajanje izvornog koda aplikacije zbog nekompatibilnih softverskih licenci.

Tipovi aplikacija koje koriste proširenja funkcionalnosti:

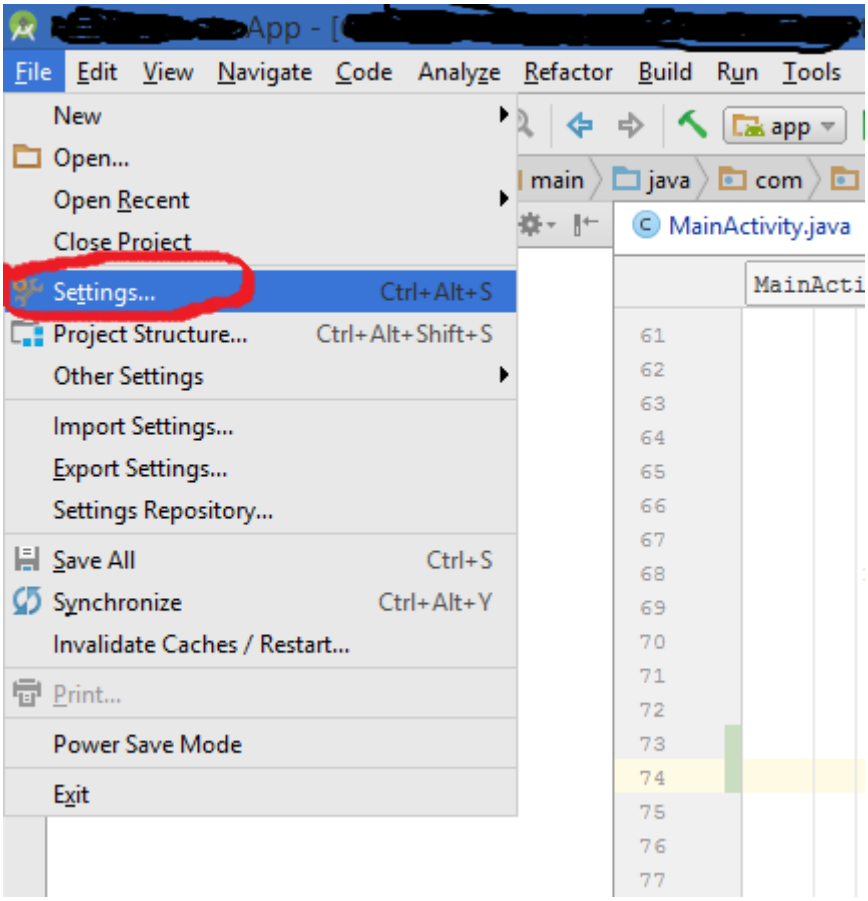
- uređivači audio zapisa koriste proširenja funkcionalnosti da generiraju, procesuiraju ili analiziraju zvuk. Ardour i Audacity su primjeri takvih urednika
- DAWs (engl. *Digital audio workstations*) koriste proširenja funkcionalnosti za generiranje ili procesuiranje zvuka. Primjeri su Logic Pro X i ProTools
- Email klijenti koriste proširenja funkcionalnosti za enkripciju i dekripciju elektroničke pošte. Primjer je Pretty Good Privacy
- Emulatori konzola za video igre često koriste proširenja funkcionalnosti da bi modularizirali i odvojili podsisteme uređaja koje pokušavaju emulirati. Primjer je PCSX2 emulator koji koristi video, audio i optička proširenja funkcionalnosti za određene komponente Playstation 2 konzole
- Softver za grafiku koristi proširenja funkcionalnosti za podršku formata datoteka i procesuiranje slika. Primjer je Photoshop plugin
- Media playeri koriste proširenja funkcionalnosti da podrže tipove datoteka i primjene filtere. Primjeri su foobar2000, GStreamer, Winamp

[24] URL: https://en.wikipedia.org/wiki/Plug-in_(computing) (11.11.2018.)

- Urednici teksta i IDE-ovi koriste proširenja funkcionalnosti za podršku programskih jezika ili za unapređenje procesa razvoja (Android Studio, Visual Studio, Eclipse IntelliJIDEA).

3.1. Proširenja funkcionalnosti u Android Studiju

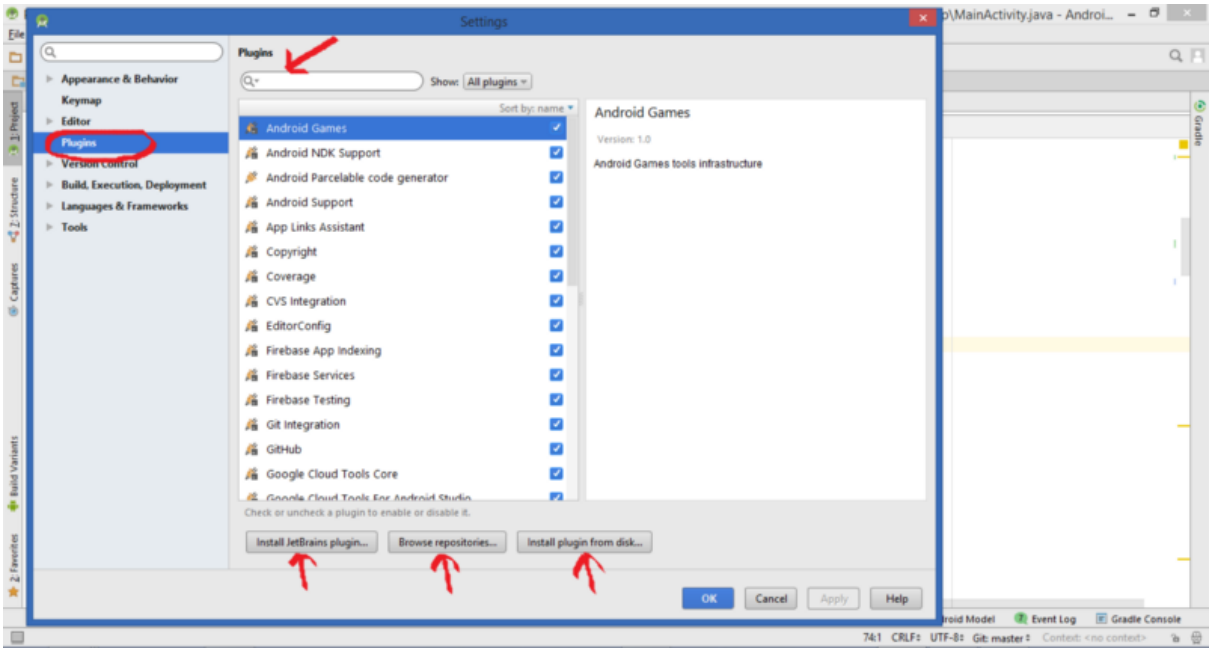
1. Pokrenuti Android studio, započeti novi projekt ili otvoriti postojeći projekt.
2. Kliknuti na File izbornik i kliknuti na opciju Settings prikazano na slici 6.



Slika 6. Instalacija proširenja funkcionalnosti[25]

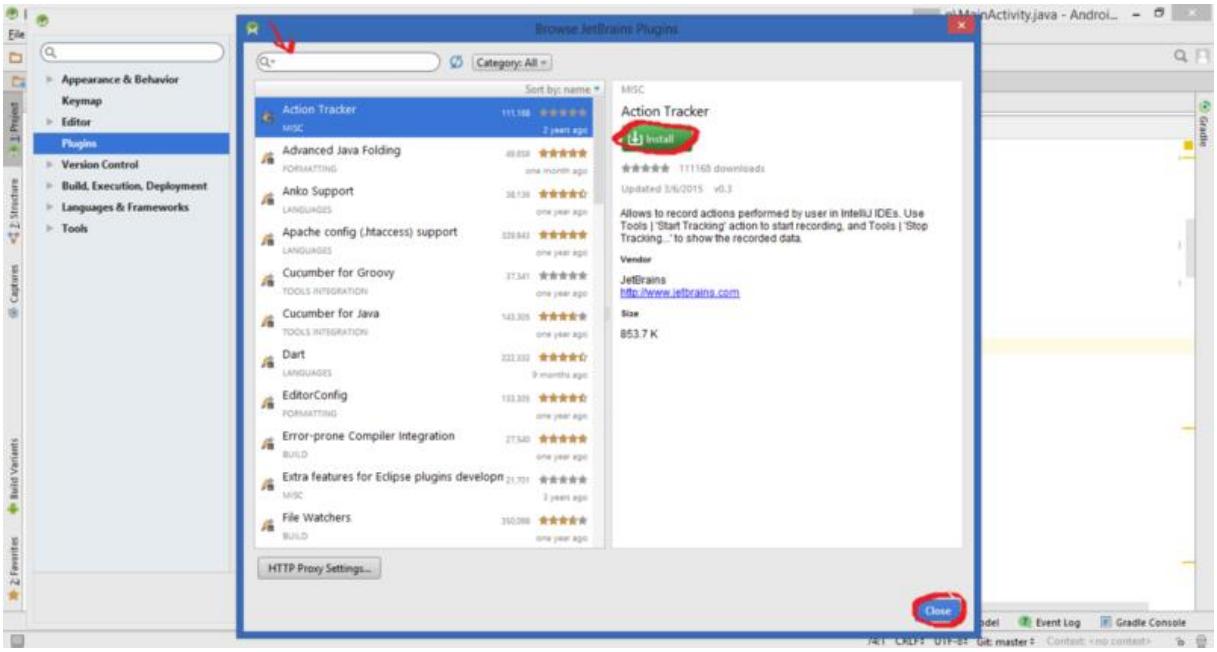
3. Na prozoru Settings kliknuti na Plugins opciju. Ovdje postoje tri gumba ispod popisa instaliranih proširenja. Prvi je namijenjen za instalaciju JetBrains proširenja, drugi za instalaciju repozitorija proširenja izvan JetBrainsa, a posljednji je za instalaciju proširenja s tvrdog diska. Odabire se gumb koji odgovara potrebnoj kategoriji kao što je prikazano na slici 7.

[25] URL: https://medium.com/@adetayo_james/how-to-install-plugins-in-android-studio-f64f1d584438 (3.11.2018.)



Slika 7. Instalacija proširenja funkcionalnosti korak 2[26]

4. Na odabranom prozoru potrebno je upisati ime proširenja koje se instalira u polje za pretraživanje i odabire se zeleni Install gumb na desnoj strani prozora. Nakon instalacije stisne se gumb Close, kao što je vidljivo na slici 8.



Slika 8. Instalacija proširenja fukcionalnosti korak 3[27]

[26] URL: https://medium.com/@adetayo_james/how-to-install-plugins-in-android-studio-f64f1d584438 (3.11.2018.)
[27] isto

3.2. Prednosti i nedostatci[28]

Glavna prednost korištenja proširenja funkcionalnosti je ta što se ne treba razvijati funkcionalnost koju pruža proširenje. Umjesto toga razvojni inženjer se može fokusirati na poslovnu logiku aplikacije, funkcionalnosti koje su najbitnije. Potrebno je vrijeme da se pročita dokumentacija proširenja i shvati kako implementirati proširenje u aplikaciju, ali to je samo dio vremena koje je potrebno da se ta funkcionalnost razvije od početka čime se štedi velika količina vremena.

Popularna proširenja upotrebljavana su u mnogo razvojnih okruženja što stvara petlju povratnih informacija koja rezultira u greškama koda koji su prijavljeni i otklonjeni. Korištenje dokazanog i popularnog proširenja funkcionalnosti ne bi trebala ugroziti kvalitetu i stabilnost aplikacije zato što je kod unaprijed testiran.

Prednost korištenja proširenja funkcionalnosti je što potiče rad i pisanje modularnog koda. Kod proširenja je prirodno odvojen od ostatka koda aplikacije i komunikacija se odvija unutar dobro postavljenih granica. Ako razvojni inženjer odabira sam razviti funkcionalnost, postaje teško odvojiti funkcionalnost od koda aplikacije. Autori proširenja koja su dobro izvedena rade na drugoj razni apstrakcije, onoj koja vodi do čišćeg i boljeg generičkog koda.

Oslanjanje na proširenja funkcionalnosti znači da je kod vezan za proširenje i ako je u nekom trenutku potrebno mijenjati proširenje funkcionalnosti koje se koristi, promjene u kodu su prevelike da bi se mogle prilagoditi novom proširenju. Postoje načini smanjenja štete, to jest korištenje omotača (engl. *wrapper*). Na ovaj način kod direktno ne poziva proširenje, nego apstrakciju.

Jedan od rizika proširenja funkcionalnosti je da može biti napušteno od strane autora. To je posebno problematično na platformama kao što je Android gdje često izlaze nove verzije. Proširenje funkcionalnosti zahtjeva konstantno održavanje da bi se osigurala kompatibilnost s najnovijom verzijom SDK-a. Ako se toga ne pridržava, proširenje ima mogućnost neoptimalnog rada na novijim uređajima.

Korištenje previše proširenja dovodi do problema kao što su konflikti ovisnosti, koje je teško otkriti i ukloniti. Velik broj proširenja će nepotrebno povećati aplikaciju tako što joj povećava veličinu i upotrebu memorije. Zbog toga će usporiti rad aplikacije, a u slučaju Android

[28] URL: https://www.scalablepath.com/blog/third-party-libraries/ (3.11.2018.)

platforme može dovesti do limita od 64K metode[29]. Datoteke Android aplikacija sadrže bytecode datoteke u obliku Dalvik Executable(DEX) datoteka koje sadrže kompilirani kod koji se koristi za pokretanje aplikacija. DEX specifikacije ograničavaju ukupan broj metoda koje se mogu referencirati u jednoj DEX datoteci na 65,536. To uključuje metode Android sustava, metode proširenja funkcionalnosti i metode koje je napisao razvojni inženjer.

Korištenje proširenja funkcionalnosti može dovesti do slabosti u sigurnosti aplikacije te su zbog toga proširenja otvorenog koda postala meta hakera.

3.3. Pravilan odabir[30]

Vrlo je bitno razumijevanje prednosti i nedostataka ekosustava za odabir proširenja funkcionalnosti. Svako proširenje je unikatno i treba se ocjenjivati na osnovu svojih mogućnosti ili nedostataka.

Popularnost proširenja – Razne platforme za razvoj kao Github imaju ugrađenu mogućnost procjene od strane korisnika. Provjera ocjena proširenja je dobar početak kod donošenja odluke o odabiru proširenja. Provjera broja i ozbiljnost problema koji su korisnici prijavili za proširenje još je jedan dobar način odabira. Kada veliki broj razvojnih inženjera koristi i vjeruje proširenju, to je dobar indikator kvalitete. Neka od većih proširenja imaju aktivnu zajednicu koja je uključena u održavanje i rast proširenja. Stackoverflow je dobar indikator toga, ali nije uvijek direktno povezan s kvalitetom. Aktivna zajednica nije uvijek znak kvalitete jer postoje neka proširenja koja su odlična, a nemaju skoro nikakvu zajednicu.

Kredibilitet autora, kvaliteta koda i dokumentacija – Provjerom povijesti autora umanjuje se rizik zapinjanja na napuštenom proširenju ili proširenju prepunom grešaka u kodu. Da li je autor aktivan na Githubu, da li je objavio više proširenja, odgovara li na probleme i rješava li ih. Dobar znak stabilnosti i dugotrajnosti su autorovo podržavanje proširenja kroz promjene platforme ili okvira. Potrebno je ukratko pregledati kod i pokušati procijeniti koliko je dobro strukturiran. Dobri autori dokumentiraju kvalitetno i postoji aktualna README datoteka. Dobro dokumentirano proširenje biti će implementirano lako i bez velikog truda. Dobra kvaliteta koda znači da je proširenje stabilno i ima dobre performanse s malo pogrešaka.

Pravilna licenca, otvoren kod, preporuka zajednice – Licenca je jedan od faktora koji mogu odlučiti o uporabi proširenja. Ako postoje ograničenja za komercijalno lansiranje proizvoda, onda se proširenje ne bi trebalo koristiti. Potrebno je osigurati da se prate pravila i uvjeti licence

[29] URL: https://developer.android.com/studio/build/multidex (23.01.2019.)
[30] URL: https://www.scalablepath.com/blog/third-party-libraries/ (3.11.2018.)

proširenja koje je uključeno u aplikaciju. Pristup kodu je vrlo važan faktor jer bez pristupa se ne može ocijeniti kvaliteta i performanse koda, a nedostatak transparentnosti donosi rizik za sigurnost. Većina platforma ima proširenja koja se uobičajeno koriste u zajednici razvojnih inženjera. Za Android platformu to su Retrofit, Butterknife, Picasso/Glide i RxJava, Dagger. Ova proširenja su visoke kvalitete, dobro održavana i dokumentirana.

3.4. Popularna proširenja

U ovom poglavlju biti će predstavljena najčešće korištena proširenja funkcionalnosti s najvećom zajednicom razvojnih inženjera koji ih koriste. Ova proširenja funkcionalnosti su najviše testirana i većinu problema na koje se može naići kod korištenja istih, lako je ukloniti jer u velikom broju slučajeva netko je naišao na isti problem i za njega postoji rješenje na nekoj od mrežnih stranica kao što su Github i Stackoverflow. U tablici 1 prikazana su proširenja funkcionalnosti razvrstana po njihovoj namjeni.

Tablica 1. Raspodjela proširenja funkcionalnosti po njihovoj namjeni

Namjena proširenja funkcionalnosti	**Ime proširenja funkcionalnosti**
Redukcijsko proširenje funkcionalnosti	Butterknife, Butterknife Zelezny
Mreža i povezivanje s API	Retrofit, OkHttp
Učitavanje slika s interneta	Picasso
Testiranje	Espresso
Reaktivno razvijanje koda	RxJava
Injekcija zavisnosti	Dagger

3.4.1. Butterknife i Butterknife Zelezny

Butterknife[31] je alat za povezivanje pogleda (engl. *view*) koji koristi anotacije da bi generirao kod za razvojnog inženjera. Butterknife je razvio Jake Wharton, a koristi se da poštedi razvojnog inženjera od pisanja repetitivnih linija koda kao što su `findViewById` (R.id.view). Butterknife se može smatrati redukcijskim proširenjem. Omogućava fokus na logiku, umjesto fokusiranja na kopiranje koda, čime se smanjuje redundantno pisanje koda.

[31] URL: https://www.journaldev.com/10439/android-butterknife-example (07.10.2018.)

Glavna značajka Butterknife proširenja je povezivanje pogleda koji su deklarirani u datotekama izgleda(engl. *layout*) koristeći anotacije umjesto pronalaženja svakog pogleda prema njegovom ID-u koristeći metodu `findViewById`. Također se može koristiti i za povezivanje resursa, kao što su resursi Drawable, String, Boolean, Integer, Color i Dimensions. Još jedna zanimljiva značajka proširenja je vezanje grupe pogleda u jednu listu, što je korisno kad se želi izvršiti ista operacija na više pogleda, npr. sakriti sve tipke unutar aplikacije kamere kod fokusa.

ButterKnife Zelezny[32] je zanimljivo proširenje za Android Studio koje generira ButterKnife veze, držače pogleda i slušače (engl. *listener*). Nakon instalacije ovog proširenja dovoljno je otići na bilo koju aktivnost, fragment ili adapter i desnim klikom miša odabrati referenciranu datoteku resursa. Odabrati „Generate" iz opcija i tamo će se naći opcija Generiranja Butterknife injekcija. Svi pogledi i njihovi ID-ovi su vidljivi i moguće je odabrati samo one poglede ili slušače koje razvojni inženjer treba za aplikaciju.

3.4.2. Retrofit i OkHttpClient

Retrofit[33] je HTTP klijent za Android. Koristeći Retrofit u Androidu lako pozivamo i šaljemo JSON(Javascript object notation) odgovore s REST (Representational State Transfer) mrežnih API-eva. Razlikuje se od ostalih proširenja jer Retrofit daje platformu koja se lagano koristi i ne treba raščlanjivati JSON odgovor zato što to obavlja proširenje. U pozadini koristi GSON biblioteku da raščlani odgovore. Sve što korisnik treba napraviti je definirati java objekt koji će obaviti raščlanjivanje. Postoji šest modula serijalizacije koji se mogu koristiti s Retrofitom, a to su GSON, Jackson, Moshi, Protobuf, Wire, Simple XML, Scalars. To su pretvarači koji se bave serijalizacijom i deserijalizacijom podataka. Osim navedenih pretvarača moguće je napraviti prilagođeni pretvarač da procesuira protokole.

Za rad s Retrofitom potrebne su sljedeće tri klase:

- Model klasa koja se koristi kao JSON model
- Sučelje koje definira moguće HTTP operacije
- Retrofit.Builder klasa – instanca koja koristi sučelje i Builder API da dozvoli definiranje završne točke URL-a za HTTP operacije.

[32] URL: https://github.com/avast/android-butterknife-zelezny/blob/master/README.md (28.11.2018.)

[33] URL: http://www.gkmit.co/articles/retrofit-2-0-a-type-safe-http-client-for-android-and-java (07.10.2018.)

Svaka metoda sučelja predstavlja jedan mogući poziv API-ju. Mora imati HTTP anotaciju (GET, POST, itd.) da bi se definirao tip zahtjeva i relativni URL. Povratna vrijednost je odziv u Call objektu s tipom očekivanog rezultata.

Da bi se podesio URL mogu biti korišteni zamjenski blokovi i parametri upita. Zamjenski blok dodaje se URL-u s vitičastim zagradama. Pomoću anotacije @Path na parametrima metode vrijednost parametra vezana je za specifičan zamjenski blok. Parametri upita se dodaju s @Query anotacijom na parametre metode i automatski su dodani na kraj URL-a. Anotacija @Body govori Retrofitu da koristi objekt kao tijelo zahtjeva za poziv.[34]

Retrofit autentifikacija

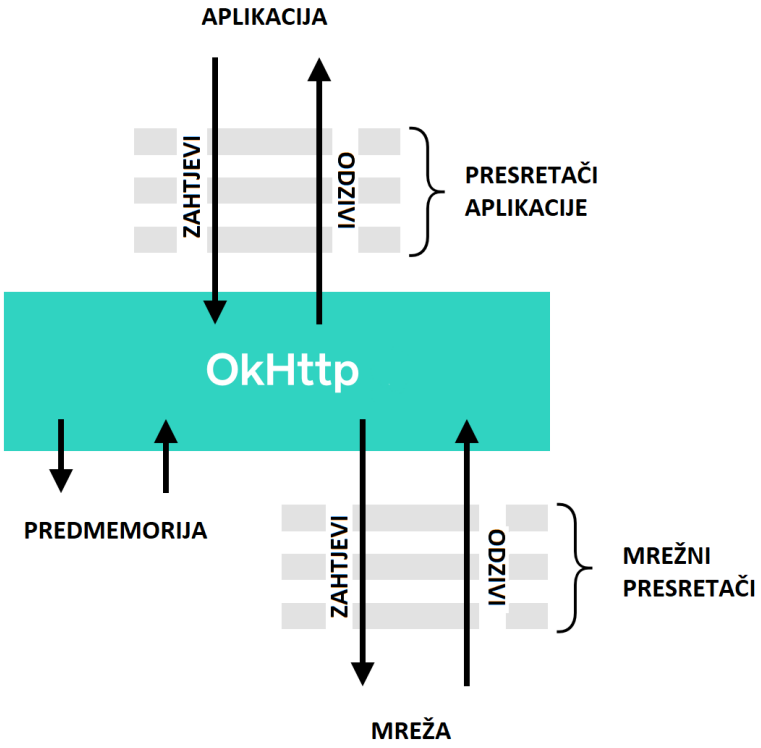
Retrofit podržava API pozive koji zahtijevaju provjeru valjanosti. Provjera valjanosti se može obaviti korištenjem korisničkog imena i lozinke (Osnovna provjera valjanosti HTTP-a) ili koristeći API token. Postoje dva načina kako se provjera valjanosti obavlja. Prva metoda je manipulacija zaglavlja zahtjeva pomoću anotacija. Druga mogućnost je korištenje OkHttp presretača.

Anotacijska provjera valjanosti obavlja se kad u pozivu zahtjeva tražimo korisničke detalje koji zahtijevaju provjeru. Dodavanjem novog parametra u definiciju API-ja Retrofitu se naređuje da doda polje Autorizacije u zaglavlje zahtjeva s vrijednošću koju dobije za vjerodajnice (engl. *Credentials*). Kod generacije osnovnih vjerodajnica za provjeru valjanosti može se koristiti OkHttp klasa Credentials sa svojom osnovnom(String, String) metodom. Ova metoda uzima korisničko ime i lozinku te vraća vjerodajnice za provjeru valjanosti.

OKHttpClient[35] je HTTP i HTTP/2 klijent za Android i Java aplikacije. OkHttp je zadužen za spajanje na server te slanje i dohvaćanje informacija. Presretači su značajka koju pruža OKHttp. Presretači omogućavaju dodavanja ponašanja prije slanja zahtjeva na mrežu i prije bavljenja s odzivom sa servera. Rad presretača prikazan je na slici 9.

[34] URL: http://www.vogella.com/tutorials/Retrofit/article.html (01.12.2018.)

[35] URL: https://soulesidibe.com/2017/02/25/okhttp-interceptors/ (01.12.2018.)



Slika 9. Prikaz rada OkHttp presretača[36]

3.4.3. Picasso

Picasso[37] je proširenje za Android koje se bavi slikama. Razvijen je i održavan od strane Square otvorenog koda. Bavi se učitavanjem i procesuiranjem slika. Pojednostavljuje proces prikaza slika iz vanjskih izvora. U većini slučajeva potrebno je samo par linija koda za implementaciju ovog proširenja. Odlično prikazuje slike dohvaćene s interneta. Proširenje upravlja svim stadijima procesa, od inicijalnog HTTP zahtjeva do spremanja slike u memoriju. Neke od mogućnosti Picassa su automatski stvoren preuzimatelj slika, spremanje u predmemoriju nekompresiranih slika, poništavanje zahtjeva te više preuzimanja u isto vrijeme. Podržava transformacije kao što su rotacija. Ako se aplikacija bazira na udaljene resurse važno je dodati sliku u obliku držača mjesta (engl. *placeholder*). Ta slika je prikazana odmah i zamijenjena kad Picasso završi s učitavanjem udaljene slike. Proširenje podržava dvije vrste slika u obliku držača mjesta. Uz metodu držača mjesta postoji i metoda greške koja prihvaća sliku držač mjesta. Picasso će pokušati preuzeti udaljenu sliku tri puta i prikazati sliku držač mjesta u slučaju da nije dohvatio udaljeni resurs.

[36] isto
[37] URL: https://code.tutsplus.com/tutorials/android-sdk-working-with-picasso--cms-22149 (07.10.2018.)

U sljedećem odjeljku biti će predstavljena tri glavna razloga za korištenje Picasso proširenja funkcionalnosti.[38]

Pojednostavljuje učitavanje i prikaz slika s eksternih URL-ova. Preuzimanje slike sa servera je jedna od najčešćih radnji u bilo kojoj aplikaciji, a u Android mrežnom API-ju to zahtijeva veliku količinu koda. Korištenjem Picassa to je postignuto u nekoliko linija koda. Treba uzeti u obzir da nije sve u preuzimanju slika s udaljene lokacije. Važna je i implementacija i logika spremanja slika u predmemoriju da bi se pružilo dobro korisničko iskustvo. Picasso podržava automatsko spremanje slika u predmemoriju. Transformacija slika je proces koji koristi puno sistemskih resursa. Ako se aplikacija treba baviti s transformacijom slika pri pokretanju, razvojni inženjer treba paziti na OutOfMemoryException. Picasso se bavi tom greškom tako da razvojni inženjer ne mora.

3.4.4. Espresso

Espresso[39] je razvojna cjelina (engl. *framework*) za Android koja omogućava lako pisanje pouzdanih korisničkih testova sučelja. Google je objavio Espresso razvojnu cjelinu u listopadu 2013. godine. Izdanje 2.0 Espresso je dio Android repozitorija podrške.

Espresso automatski sinkronizira testiranje radnje s korisničkim sučeljem aplikacije. Razvojna cjelina također osigurava da se aktivnost pokrene prije testova i dopušta testovima da čekaju dok se ne završe sve promatrane pozadinske aktivnosti. Namijenjen je za testiranje jedne aplikacije, ali se može koristiti za testiranje unutar više aplikacija. Ako se koristi za testiranje izvan aplikacije, mogu se obavljati testovi crne kutije zato što se ne može pristupiti klasama izvan aplikacije.

Ljudi su skloni greškama i u prirodi ljudi je da stvaraju softver koji je sklon greškama. Prije lansiranja aplikacije testiranjem se detektiraju greške napravljene tijekom razvoja. Uz detekciju grešaka ostvaruje se zadovoljstvo korisnika aplikacijom i osigurava se kvaliteta proizvoda što pomaže u stjecanju dobrih odnosa i povjerenja s korisnicima.

Android uglavnom podržava dvije vrste testiranja, a to su jedinični testovi i instrumentacijski testovi. Jedinični testovi su testiranje svake metode ili funkcije koda, npr. kod pozivanja funkcije s parametrom x, trebala bi vratiti y. Ova vrsta testova pokreće se na JVM-u lokalno

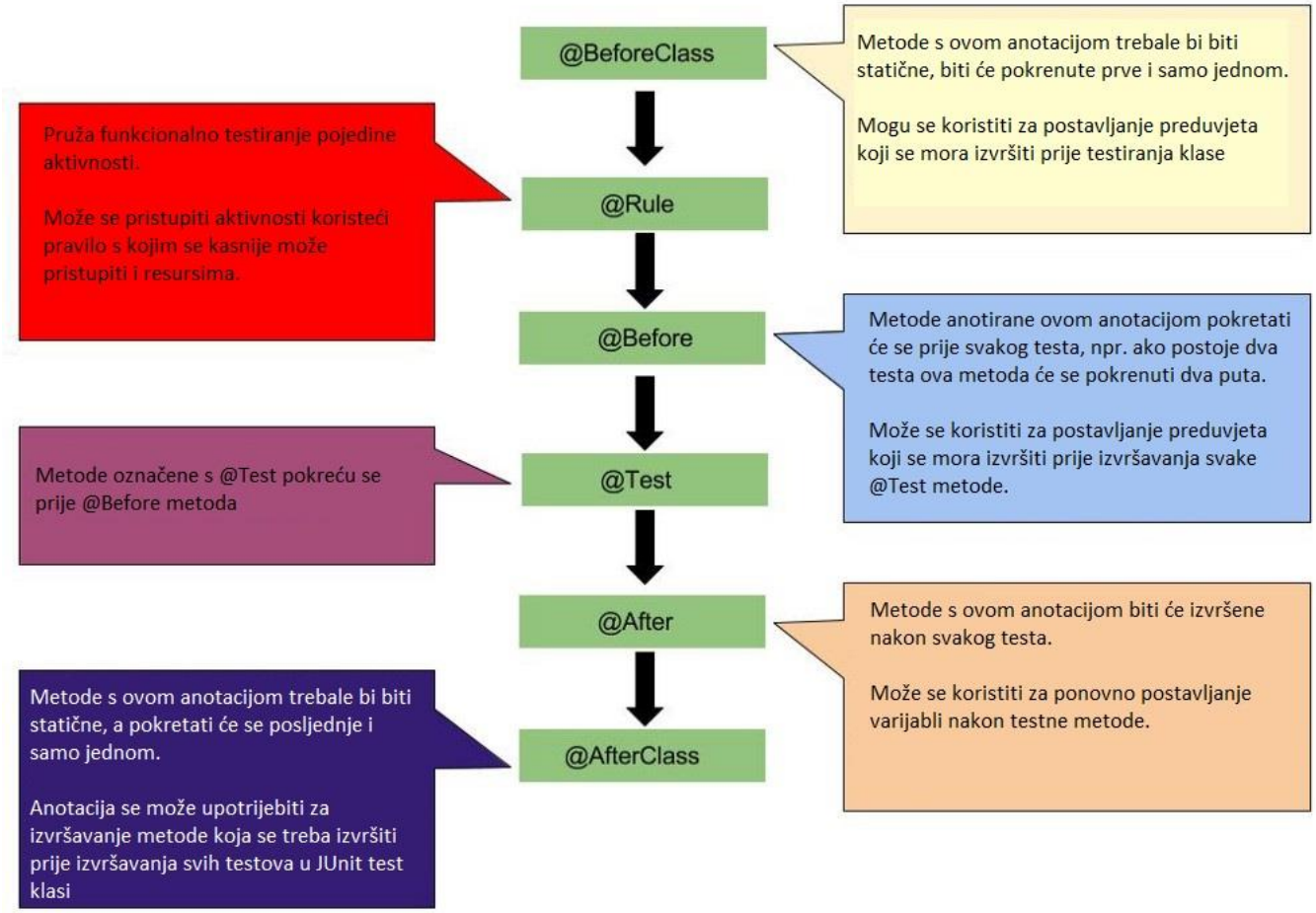
[38] URL: https://stacktips.com/tutorials/android/how-to-use-picasso-library-in-android (01.12.2018.)

[39] URL: http://www.vogella.com/tutorials/AndroidTestingEspresso/article.html#espresso_introduction (07.10.2018.)

bez potrebe za emulatorom ili fizičkim uređajem. Instrumentacijski testovi se koriste za testiranje Android okvira kao što su UI i SharedPreferences. Zato što nisu dio Android okvira pokreću se na uređaju ili emulatoru. Instrumentacijski testovi koriste zaseban .apk za potrebe testiranja. Svaki put kad se test pokreće Android Studio instalira .apk nad kojim se testovi provode. Nakon toga Android Studio instalira testni apk koji sadrži kod relevantan za testiranje. Postoji 6 anotacija koje se mogu primijeniti na metode unutar testnih klasa, a to su @Test, @Before, @BeforeClass, @After, @AfterClass i @Rule. Kod testiranja važno je:[40]

- Aktivnost će biti pokrenuta koristeći @Rule prije nego testni kod počne
- Po zadanim opcijama pravilo će biti inicijalizirano i aktivnost će se pokrenuti (onCreate, onStart, onResume) prije pokretanja svake @Before metode
- Aktivnost će biti uništena (onPause, onStop, onDestroy) nakon pokretanja @After metode koja se poziva nakon svake @Test metode
- Pokretanje aktivnosti može se odgoditi unutar ActivityTest pravila, ali u tom slučaju treba se ručno pokrenuti aktivnost prije testova.

[40] URL: https://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b (01.12.2018.)



Slika 10. Objašnjenje anotacija korištenih kod testiranja[41]

3.4.5. RxJava

RxJava[42] je postala jedna od najvažnijih vještina za razvijanje Android aplikacija. Definicija je da je to JVM implementacija reaktivnih ekstenzija. Službena dokumentacija opisuje reaktivne ekstenzije (ReactiveX) kao proširenje funkcionalnosti za slaganje asinkronih programa i programa upravljanih događajima korištenjem vidljivih sljedova.

Za početak treba definirati dva pojma spomenuta u opisu: asinkrono i upravljano događajima:

- Asinkrono znači da se različiti dijelovi programa pokreću simultano

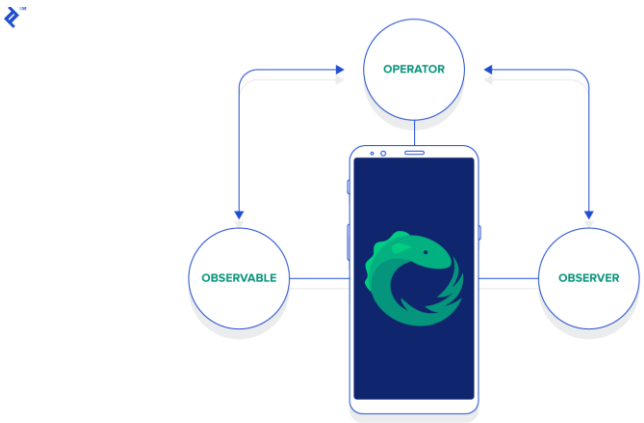
[41] URL: https://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b (01.12.2018.)
[42] URL: https://blog.mindorks.com/rxjava-anatomy-what-is-rxjava-how-rxjava-is-designed-and-how-rxjava-works-d357b3aca586 (03.12.2018.)

- Upravljano događajima znači da program izvršava kod baziran na događajima koji se generiraju dok je program pokrenut, npr. klikom na gumb izazove događaj, a zatim upravljač programa primi taj događaj i obavi tu radnju.

Reaktivna ekstenzija proširuje uzorak promatrača tako da podržava sljedove podataka ili događaja i dodaje operatore s kojima se slažu sljedovi. U isto vrijeme apstraktirane su stvari niske razine kao sinkronizacija, sigurnost niti (engl. *thread*), istovremena struktura podataka i ulaza i izlaza bez blokiranja.

Dokumentacija naziva ovaj proces funkcionalnim reaktivnim razvojem. ReactiveX je funkcionalan i reaktivan, ali funkcionalan i reaktivan razvoj je druga stvar. Jedna od glavnih razlika je što se funkcionalan i reaktivan razvoj bavi vrijednostima koje se kontinuirano mijenjaju, dok se ReactiveX bavi diskretnim vrijednostima koje se emitiraju.

RxJava je implementacija reaktivnih sljedova, a reaktivni sljedovi su koncept u razvoju koji se koristi za upravljanje asinkronih sljedova podataka bez blokiranja. Evoluirali su u specifikaciju koja se bazira na konceptu Objavljivača <T> i Pretplatnika <T>. Objavljivač je izvor događaja u T slijedu, a Pretplatnik je korisnik tih događaja. Pretplatnik se pretplaćuje na objavljivača prizivajući tvorničku metodu unutar Objavljivača koja će pogurati predmete <T> slijeda tako da započne novu pretplatu. Ovo se također zove i uzorak Reaktora. U svojoj srži Rxjava pojednostavljuje razvoj jer podiže razinu apstrakcije koja je povezana s nitima. Razvojni inženjer ne mora brinuti o detaljima izvođenja operacija na različitim nitima. Ovaj proces je poprilično izazovno izvesti točno i ako se ne implementira ispravno dovodi do grešaka koje je teško otkriti i riješiti.



Slika 11. Tri glavna koncepta RxJave[43]

[43] URL: https://www.toptal.com/android/functional-reactive-android-rxjava (03.12.2018.)

U svijetu RxJave sve se može modelirati kao slijed. Slijed emitira predmete i svaka emisija se može konzumirati i nadgledati. Apstrakcija slijeda se implementira s 3 glavna konstrukta prikazana na slici 11. Gledljivo (engl. *Observable*), Gledatelj (engl. Observer) i Operator. Gledljivo emitira predmete, a Gledatelj konzumira većinu tih predmeta. Emisije iz gledljivih objekata se mogu modificirati, transformirati i manipulirati lančanim pozivima na Operatora.

Neiskusni razvojni inženjeri imaju problema sa shvaćanjem rada RxJave, smatraju da su alati koje nudi suvišni i njihova upotreba bezvrijedna, ali oni inženjeri koji uzmu vremena da shvate Rx shvaćaju da pruža niz prednosti:[44]

- Intuitivnost – Rx koristi istu sintaksu za opise radnji kao što se koristi u funkcionalnom razvoju
- Deklarativnost – Ono što je potrebno može se izraziti s višom razinom apstrakcije
- Proširivost – Rx se može proširiti s prilagođenim metodama proširenja
- Složivost – Operatori u RxJavi se lagano slažu, a izvode teške operacije
- Konvertibilnost – Operatori u RxJavi mogu pretvarati tipove podataka filtriranjem, procesuiranjem i proširivanjem slijeda podataka.

3.4.6. Dagger

Dagger[45] je potpuno statičan okvir za injekciju zavisnosti koji se pokreće za vrijeme kompiliranja. Slično kao i RxJava, Dagger je poprilično težak za savladati, ali se uloženo vrijeme isplati. Injekcija zavisnosti je način pružanja manjih komponenata postojećem modelu i zalijepiti ih skupa s minimalnim trudom. Za primjer, ako postoji model Auto, mogu mu se dodati gume i jednostavno pružiti mogućnost implementacije zamjene guma u budućnosti, sve bez mijenjanja jedne linije koda unutar Auto modela. Kod razvoja većih aplikacija razvojni inženjeri ne bi trebali sami obavljati injekciju zavisnosti jer količina koda brzo raste i postaje vrlo težak za održavanje. Dagger pomaže izbjeći ovaj slučaj tako što stvara graf injekcije zavisnosti za vrijeme kompiliranja s anotacijskim procesuiranjem.

Anotacije koje se koriste u Daggeru opisane su u sljedećim odjeljcima.

[44] URL: https://jaxenter.com/5-reasons-use-rxjava-148982.html (03.12.2018.)

[45] URL: https://code.tutsplus.com/tutorials/dependency-injection-with-dagger-2-on-android--cms-23345 (03.12.2018.)

@Inject – Ovom anotacijom se zahtijevaju zavisnosti, odnosno razvojni inženjer govori Daggeru da anotirana klasa ili polje želi sudjelovati u injekciji zavisnosti. Zbog toga će Dagger konstruirati instance anotiranih klasa i zadovoljiti njihove zavisnosti.

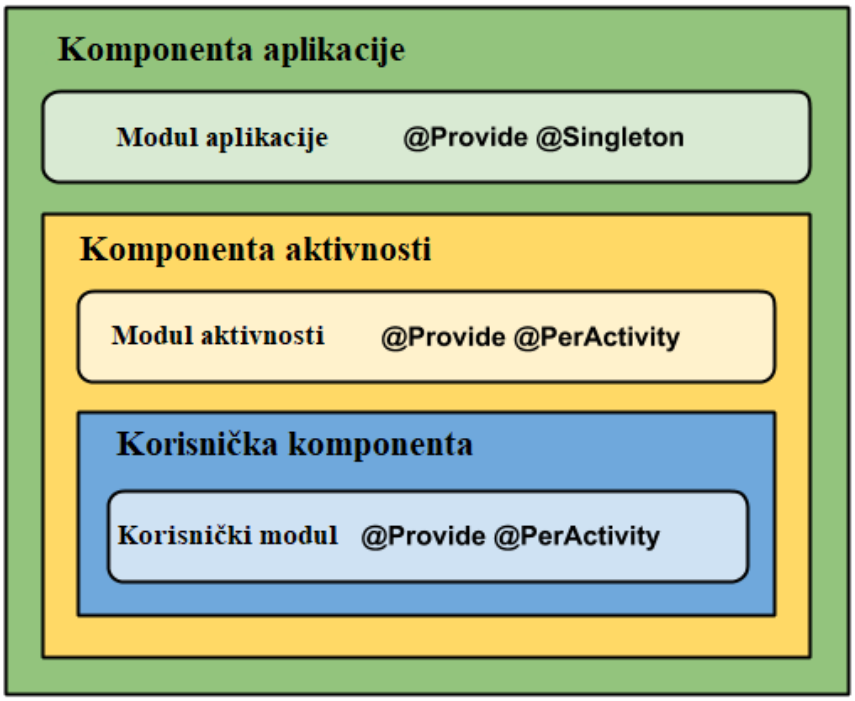
@Module – Moduli su klase čije metode pružaju zavisnosti. Kada razvojni inženjer anotira klasu s @Module, Dagger zna gdje pronaći zavisnosti da bi ih zadovoljio s konstrukcijom instanci klasa. Jedna od glavnih značajki modula je da moraju biti dizajnirani i podijeljeni zajedno.

@Provide – Unutar modula definiraju se metode koje sadrže ovu anotaciju koja govori Daggeru kako da konstruira i pruži zavisnosti.

@Component – Komponente su ustvari injektori, poveznica između @Inject i @Module, čija je glavna odgovornost spojiti te dvije komponente. Injektori daju instance svih definiranih tipova. Sve komponente znaju opseg zavisnosti koje pružaju kroz svoje module.

@Scopes – Opsezi su jako korisni, a Dagger ima konkretan način postavljanja opsega kroz anotacije. Nema potrebe da svaki objekt zna kako upravljati sa svojim instancama. Primjer opsega je klasa s prilagođenom @PerActivity anotacijom , tako da taj objekt „živi“ dok god je aktivnost „živa“.

@Qualifier – Ova anotacija se koristi kad je tip klase nedovoljan za identifikaciju zavisnosti. U slučaju Androida mnogo puta su razvojnom inženjeru potrebni različiti tipovi konteksta, tako da može definirati anotacije @ForApplication i @ForActivity. Kad se injektira kontekst može se reći Daggeru koji tip konteksta mu se pruža.



Slika 12. Struktura grafa injekcije zavisnosti[46]

Komponenta aplikacije, prikazana na slici 12, je komponenta čiji je životni vijek zapravo životni vijek aplikacije. Uporaba @Singleton anotacije za ovu komponentu ograničava ju na jednom po aplikaciji. Ovo je važna značajka koja opisuje kako se ponašaju komponente u Daggeru, ne otkrivaju tipove njihovih modula osim ako ih se eksplicitno učini dostupnima. Modul aplikacije pruža objekte koji žive tijekom životnog vijeka aplikacije. Komponenta aktivnosti živjeti će za vrijeme trajanja životnog vijeka aktivnosti. Modul aktivnosti izlaže aktivnost zavisnostima grafa. Razlog za to je uporaba konteksta aktivnosti unutar fragmenta. Korisnička komponenta je komponenta koja proširuje komponentu aktivnosti ali u opsegu @PerActivity. Korisnički modul je modul koji pruža suradnike povezane s korisnikom.

Prednosti Daggera:[47]

- Poboljšava performanse, uključujući i slučajeve po pojedinačnom pozivu
- Bolje generiran kod koji je čitak i lagan za pratiti
- Sve se obavlja kao konkretan poziv.

Nedostatci Daggera:

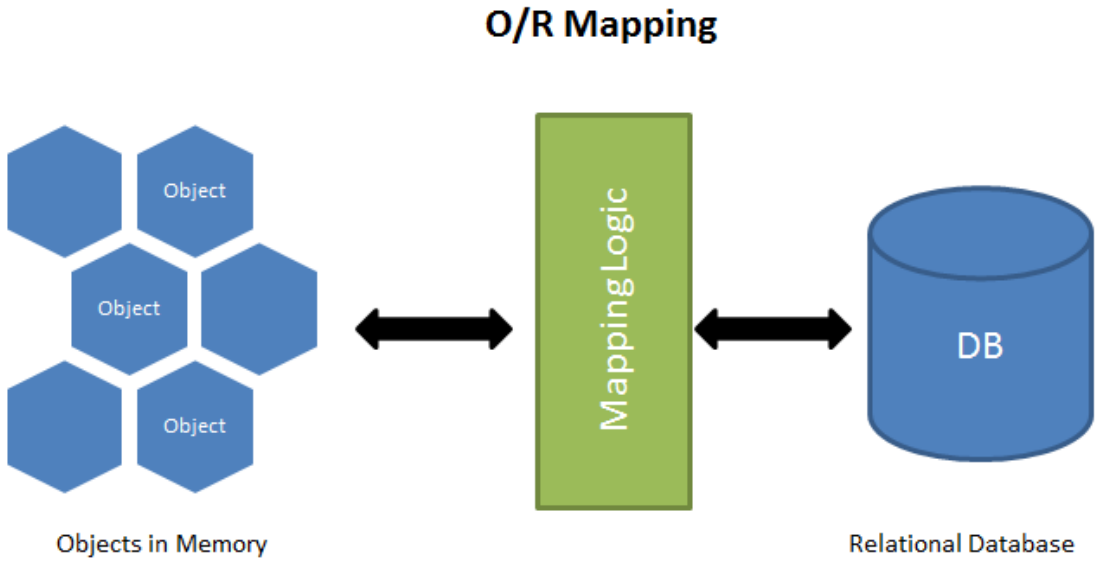
[46] URL: https://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/ (03.12.2018.)

[47] URL: https://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/ (03.12.2018.)

- Implementacija komponenti zahtijeva ponovnu izgradnju projekta, a svaka greška rezultira u nestanku klase (nije ni bila generirana)
- Metoda Inject() ima jaki tip asocijacije s metom injekcije. To je dobro za otkrivanje grešaka, ali komplicira uobičajenu praksu injektiranja iz osnovnih klasa.

4. ORM

Object-relational mapping[48] (ORM) je mehanizam koji omogućava adresiranje, pristup i manipulaciju objektima bez obzira u kakvoj su relaciji ti objekti s izvorima podataka. ORM dopušta razvojnim inženjerima da održavaju konstantan pogled na objekte u stvarnom vremenu, dok ih izvori isporučuju, a aplikacije koje im pristupaju se mijenjaju. Baziran na apstrakciji, ORM upravlja detaljima između seta objekata i relacijskih baza podataka, XML repozitorija ili drugih izvora podataka. Simultano skriva detalje koji se često mijenjaju ili povezana sučelja od razvojnih inženjera i koda koji stvaraju. ORM skriva i enkapsulira promjenu u izvoru podataka tako da kad se izvor podataka ili njihov API mijenjaju, samo ORM mora pratiti promjene, a ne aplikacije koje koriste ORM. Ovo omogućuje razvojnim inženjerima da iskoriste nove klase kako postaju dostupne i olakšava ekstenziju ORM-a na aplikacije koje se baziraju na ORM-u. Na slici 13 prikazana je razlika između objekata u memoriji i relacijskih baza podataka.



Slika 13. Prikaz razlike između objekata u memoriji i relacijske baze podataka[49]

[48] URL: https://searchwindevelopment.techtarget.com/definition/object-relational-mapping (12.11.2018.)

[49] URL: https://medium.com/@mithunsasidharan/should-i-or-should-i-not-use-orm-4c3742a639ce (24.11.2018.)

4.1. Prednosti i nedostatci ORM-a[50]

Debata za ili protiv korištenja ORM-a nije zasnovana na tehnologiji, već na vrijednostima. Informacija da li koristiti ORM ili ne ovisi o tome što je razvojnom inženjeru važnije, čist pristup podatcima ili čist kod.

Zagovornici čistog pristupa podacima smatraju da se softver treba koncentrirati na razini modela i odnositi se sa spremanjem podataka kao sporednom stvari. Oni će reći da se ORM treba koristiti zato jer dopušta opisivanje na visokoj razini. Bavi se s tim kako će podatci biti spremljeni i dohvaćeni iz baze podataka s malo koda i programskim jezikom.

Zagovornici čistog koda smatraju da se softver mora koncentrirati na razini perzistencije i tretirati kod kao sporednu stvar. Ovi razvojni inženjeri ne koriste ORM-ove jer smatraju da program ili aplikacija trebaju biti napisani tako da spremanje podataka i uzorci pristupa odgovaraju samom pristupu podatcima i da je sav pristup podatcima kontroliran i jasan.

Oba pristupa nisu apsolutno točna i nitko nije u pravu ili krivu. Oba pristupa mogu odvesti do lošeg i dobrog rješenja.

Kod korištenja ORM-a, objekti modela trebali bi biti što jednostavniji. Potrebno je biti oprezniji oko jednostavnosti, tako da objekti modela budu samo podaci. U protivnom dolazi do borbe s ORM-om da bi se osigurala poželjna perzistencija podataka, a ORM ne traži metode i svojstva koja ustvari ne postoje.

Ako se ORM ne koristi, potrebno je definirati ili DAO (engl. *Data Access Object*) ili metode perzistencije i upita da bi se izbjeglo povezivanje sloja modela i sloja perzistencije. U protivnom se SQL nađe u objektu modela i razvojni inženjer je prisiljen uključiti zavisnost u projekt.

Kad razvojni inženjer zna da će svi uzorci pristupa podacima biti jednostavni (kao što je obično dohvaćanje objekta), ali ih unaprijed ne poznaje, treba razmisliti o upotrebi ORM-a. ORM-i čine izgradnju kompleksnih upita zbunjujućim i teškim za otkrivanje grešaka u kodu, ali ušteda vremena je velika ako su upiti jednostavni.

Ako razvojni inženjer želi da aplikacija bude apsolutno najbrža što može, preporuka je ne koristiti ORM. Jedini način da svi upiti konzistentno budu brzi je pažljivo planiranje strukture baze podataka i upravljanje uzorcima pristupa podataka. Veliku pažnju treba usmjeriti na

[50] URL: https://medium.com/@mithunsasidharan/should-i-or-should-i-not-use-orm-4c3742a639ce (24.11.2018.)

odabir jednog načina spremanja podataka i pisanje upita koji su optimizirani za taj način spremanja podataka.

Kao glavne prednosti ORM-a[51] ističu se pružanje implementacije domena-model-uzorak, velika redukcija u količini koda te briga o kodu specifičnom za upravljanje bazom. ORM upravlja predmemorijom, te štedi resurse na nekoliko načina. Ubrzava vrijeme razvoja eliminacijom potrebe za repetitivnim SQL kodom, smanjuje vrijeme razvoja te smanjuje cijenu razvoja.

Ipak, ima i nezanemarivih nedostataka. Neki od nedostataka ORM-a jesu povećano vrijeme pokretanja zbog pripreme metapodataka, što nije pozitivno za aplikacije radne površine. Nedostatak je i velika krivulja učenja.

Relativno ga je teško za optimizirati i otkrivati greške u generiranom SQL-u, te nije prikladan za aplikacije bez čistog objektnog modela. Nadalje, nedostatak ORM-a je gubitak produktivnosti razvojnog inženjera koji uči s njime programirati. Razvojni inženjer ne razumije što kod zapravo radi i ima veću kontrolu kad koristi SQL.

S tehničke strane, ORM ima tendenciju da bude spor te ne može konkurirati SQL upitima kad se rade kompleksni upiti.

Popularni ORM-i

U ovom poglavlju biti će prikazani najpopularnija ORM rješenja koja se koriste u raznim projektima. Slično kao i s popularnim proširenjima funkcionalnosti ranije, ova ORM rješenja imaju odličnu dokumentaciju i veliki broj pogrešaka već ispravljenih te se u zajednici razvojnih inženjera preporučaju kao bolje opcije od ostalih ORM-a.

4.1.1. GreenDao

GreenDao[52] je Android ORM otvorenog koda koji olakšava razvoj za SQLite baze podataka. Oslobađa razvojne inženjere od bavljenja sa zahtjevima baza podataka na niskoj razini i pritom štedi vrijeme razvoja. SQLite je odlična relacijska baza podataka, ali pisanje SQL-a i parsiranje rezultata upita može se kategorizirati kao dosadan posao koji oduzima puno vremena. GreenDao mapira Java objekte prema tablicama baze podataka. Na ovaj način može se spremati, ažurirati, brisati i slati upite za Java objekte koristeći jednostavan objektno

[51] URL: https://stackoverflow.com/questions/4667906/the-advantages-and-disadvantages-of-using-orm (24.11.2018.)

[52] URL: http://greenrobot.org/greendao/ (12.11.2018.)

orijentirani API. Na stranici dokumentacije greenDao razvojni inženjeri tvrde da je on najbrži od svih ORM-a i ne radi kompromise u performansama. Koristeći greenDao većina entiteta može se umetnuti, ažurirati i učitati u omjeru nekoliko tisuća entiteta po sekundi. Na slici 14 prikazana je struktura rada greenDao ORM-a.

Slika 14. greenDao ORM[53]

GreenDao podržava enkriptirane baze podataka koje služe da bi se zaštitili osjetljivi podatci. Novije verzije Androida podržavaju sustavnu enkripciju, sam Android ne podržava enkripciju za datoteke baze podataka. Ako napadač uspije pristupiti datoteci baze podataka (tako da dobije korijenski pristup, eksploatacijom sigurnosne greške ili prevarom korisnika uređaja) ima pristup svim podatcima unutar baze. Korištenjem enkripcije s lozinkom dodaje dodatan sloj zaštite i sprječava napadača da otvori datoteku baze podataka.

Obilježja GreenDao ORM:

- Postoji od 2011. godine i korišten je u mnogo poznatih aplikacija
- Jednostavan i sažet API koji sadrži anotacije
- Malena biblioteka veličine manje od 150 kB koja sadrži samo Java.jar(bez nativnih dijelova koji koriste procesor)
- Jedan od najbržih ORM-a za Android, pokretan inteligentnim generiranjem koda
- Siguran i ekspresivan API upita - QueryBuilder koristi svojstva konstanta da se izbjegnu greške
- Snažna spajanja, upiti kroz entitete i lančana spajanja za kompleksne odnose
- Fleksibilni tipovi svojstava, koristi prilagođene klase da predstavi podatke u entitetu
- Enkripcija, podržava SQLCipher enkriptirane baze podataka.

4.1.2. OrmLite

Object Relational Mapping Lite[54] pruža laganu funkcionalnost za trajne Java objekte za SQL baze podataka dok izbjegava kompleksnost i dodatne procese koji dolaze sa standardnim ORM

[53] URL: http://greenrobot.org/greendao/ (12.11.2018.)
[54] URL: http://ormlite.com/sqlite_java_android_orm.shtml (24.11.2018.)

paketima. Podržava velik broj SQL baza podataka koristeći JBDC(Java Database Connectivity) i podržava SQL izvorne pozive API-jima Android operacijskog sustava.

Zbog nedostatka službene podrške za JDBC u Android operacijskom sustavu, ORMLite obavlja direktne pozive Android API-ju baze podataka da bi pristupio SQLite bazama podataka. U sljedećih sedam koraka opisano je kako pokrenuti ORMLite u Android okruženju.

1. Potrebno je napraviti vlastitu klasu pomoćnik baze podataka koja proširuje OrmLiteSQliteOpenHelper klasu. Ova klasa kreira i nadograđuje bazu podataka za vrijeme instalacije aplikacije i pruža DAO klase koje će koristiti ostale klase u aplikaciji. Pomoćnik mora implementirati metode onCreate i onUpgrade. OnCreate stvara bazu podataka kod prve instalacije aplikacije dok onUpgrade nadograđuje bazu podataka kad se aplikacija nadograđuje na novu verziju.
2. Pomoćnik može biti otvoren kroz sve aktivnosti aplikacije s istom SQLite vezom baze podataka koja se koristi na svim nitima. Ako se otvori više veza na istu bazu podataka, mogu se dobiti neočekivani rezultati. Preporuča se korištenje OpenHelperManagera da bi nadgledao korištenje pomoćnika. On će se stvoriti pri prvom pristupu i pratiti svaki put kada se koristi u kodu, a zatim će se zatvoriti posljednji put kad se koristi pomoćnik.
3. Kada je definiran pomoćnik baze podataka i ispravno je upravljan, potrebno je koristiti ga u klasama aktivnosti. Jednostavan način korištenja OpenHelperManagera je da proširuje OrmLiteBaseActivity za svaku klasu aktivnosti. Također postoje klase OrmLiteBaseListActivity, OrmLiteBaseService i OrmLiteBaseTabActivity. Ove klase pružaju pomoćniku zaštićeno polje i metodu getHelper() za pristup bazi podataka kada god je to potrebno.
4. Ako razvojni inženjer ne želi proširiti OrmLiteBaseACtivity i ostale bazne klase, onda je potrebno duplicirati njihovu funkcionalnost. Potrebno je pozvati OpenHelperManager.getHelper() na početku koda, spremiti pomoćnika i koristiti ga koliko je potrebno, a zatim pozvati OpenHelperManager.release() kada više nije potreban.
5. Podrazumijeva se ako se koristi OrmLiteBaseActivity ili ostale bazne klase, OpenHelperManager će detektirati klasu pomoćnika baze podataka kroz refleksiju. Još jedan način spajanja pripadajuće klase pomoćnika baze podataka je postavljanje imena klase u open_helper_classname vrijednosti koja je definirana u datoteci resursa.
6. Android izvorni tip SQLite baze podataka je SqliteAndroidDatabaseType i korišten je od strane baznih klasa interno.

7. Upozorenje! Potrebno je osigurati da se pozivi prema OpenHelperManager.getHelper() i release metodama obavljaju na pozadinskoj niti. U protivnom pristup bazi podataka prije nego što se otvori ili nakon što se zatvori dovodi do grešaka.

Kao glavna obilježja ORMLitea[55] ističu se jednostavno postavljanje klasa dodavanjem Java anotacija, snažna apstrakcija DAO klasa te automatska generacija SQL-a za stvaranje i brisanje tablica u bazi. Fleksibilan QueryBuilder omogućuje lagano obavljanje jednostavnih i kompleksnih upita, a ORMLite podržava MySQL, Postgre, Microsoft SQL Server, H2, Derby, HSQLDB i SQLite tehnologije. Također upravlja kompiliranim SLQ izjavama za repetitivne upite, nudi osnovnu podršku za transakcije u bazama podataka i daje podršku za konfiguraciju tablica i polja bez anotacija.

4.1.3. ActiveAndroid

ActiveAndroid[56] je ORM koji koristi stil aktivnog zapisa. Dopušta razvojnom inženjeru manipulaciju podatcima preko SQLite sustava upravljanja. Pruža brzu i jednostavnu upotrebu bez korištenja direktnih SQL upita. Razvojni inženjeri čak ne moraju znati SQL da bi koristili ovo proširenje. ActiveAndroid pruža korištenje zapisa u bazi podataka putem objekata, tako da razvojni inženjeri nemaju problema s konverzijom tabularnih zapisa u objekte i obrnuto. Klasa model predstavlja tablicu, a instanca klase predstavlja zapis u tablici. Upravljanje bazom podataka s ActiveAndroidom je puno jednostavnije i brže jer razvojni inženjeri mogu stvarati klase i anotacijom reći sustavu koja klasa predstavlja što u bazi podataka. ActiveAndroid obavlja sav posao povezan sa sređivanjem baze podataka tako da razvojni inženjeri ne moraju pisati CRUD metode sami. Korištenje ActiveAndroid proširenja[57] donosi sa sobom dobar skup prednosti. Lagan je za održavanje i reducira pogreške koje se većinom događaju zbog ljudskog faktora. Direktno utječe na broj linija koda i upola ga smanjuje. ActiveAndroid kao ORM pruža pristup prvo kodu, tako da se bazni model ne može napraviti od tablica.

4.1.4. Sugar ORM

Sugar ORM[58] je namijenjen da olakša interakciju sa SQLite bazama podataka u Android OS-u. Pruža mogućnost stvaranja baze podataka i jednostavne API-je za manipulaciju objekata.

[55] URL: https://android.jlelse.eu/lessons-learned-from-migrating-to-ormlite-7843eb55ec84 (04.12.2018.)

[56] URL: http://www.activeandroid.com/ (24.11.2018.)

[57] URL: https://guides.codepath.com/android/activeandroid-guide (04.12.2018.)

[58] URL: http://satyan.github.io/sugar/index.html (24.11.2018.)

Obavljanje CRUD operacija poprilično je jednostavno. Funkcije kao što su save(), delete() i findById() uključene su da olakšaju rad. Na slici 15 prikazane su metode koje se koriste za operacije unutar Sugar ORM-a.

Slika 15. Prikaz metoda Sugar ORM-a[59]

Napravljen je za razvoj Android aplikacija koje imaju poprilično velik i kompleksan model podataka i velik broj linija koda za operacije unutar baze podataka, a primarna zadaća im je dohvaćanje i iteracija po objektima.

Glavna obilježja SugarORM-a uključuju eliminaciju pisanja SQL upita za interakciju sa SQLite bazom podataka i upravljanje odnosima između objekata. SugarORM se brine o stvaranju baze podataka i pruža čist i jednostavan API za operacije unutar baze.

4.1.5. DBFlow

DBFlow[60] je biblioteka bazirana na Kotlinu koja je brza, efikasna i bogata mogućnostima. DBFlow koristi anotacijsko procesuiranje da generira SQLite kod i pruža moćan SQLite jezik upita. Razvijen je iz kolekcije najboljih značajki raznih biblioteka baza podataka. Iako nije najrazvikaniji ORM, brzo se izvodi, nema mnogo koda, ima dobru podršku za migraciju i unaprijed učitanu podršku za baze podataka. Jedan je od „najodraslijih“ ORM-a. Podržava

[59] isto
[60] URL: https://github.com/codepath/android_guides/wiki/DBFlow-Guide (24.11.2018.)

velik broj značajki baza podataka koji će smanjiti i učiniti boljim vrijeme provedeno kodirajući baze podataka. Podržava više baza podataka u isto vrijeme (u razdvojenim modulima) dok god ne dijele modele.

Obilježja DBFlowa su:[61]

- Jezik omotač za SQLite – Pruža nekoliko korisnih metoda, ekstenzija i generiranih pomagača iz kojih nastaje sažeta i tečna sintaksa upita
- Spremanje u predmemoriju – Podržava spremanje modela u predmemoriju što uvelike povećava brzinu rada
- Migracija – Migracije u DBFlowu su jednostavne, podržane su samo one koje pruža SQLite i razvojnom inženjeru omogućena je modifikacija podataka na strukturiran način tijekom migracije
- Više modela – Može se koristiti u knjižničnim projektima, u bilo kojem broju unutar projektnih modula istovremeno, ali modeli moraju biti u različitim bazama podataka
- Odnosi – Podržan je @ForeignKey koji uključuje više stranih ključeva za odnos 1-1. @OneToMany je odnos generiran od strane kompajlera između dvije tablice za upravljanje podatcima. @ManyToMany generira tablicu spajanja između dvije tablice
- Pogledi – Deklarirani su kao tablice i podržani kao virtualne tablice
- Modeli upita – Modeli upita nemaju poveznicu sa SQLiteom, ali su ekstremno korisni za prilagođene upite ili upite spajanja koji vraćaju set rezultata koji nisu mapirani za nijedan model koji se trenutno koristi za tablice
- Baze podataka s enkripcijom – Enkripcija za sigurnost, koristi se SQLCipher koji je odvojen i lagan za integraciju
- Indeksi – Značajka SQLitea koja drastično poboljšava vrijeme upita na velikim setovima podataka
- Reaktivnost – Promjene u bazama podataka lagano se prate pomoću sustava ModelNotifier
- Upravljanje transakcijama – Mogućnost stavljanja svih transakcija i dohvaćanja na istu pozadinsku nit za maksimalnu efikasnost i sprečavanje grešaka s korisničkim sučeljem
- Konverteri – Ako su podatci koji se koriste prilagodljivi, potrebno je definirati TypeConverter da ih mapira na SQLite tipove podataka.

[61] URL: https://agrosner.gitbooks.io/dbflow/ (10.12.2018.)

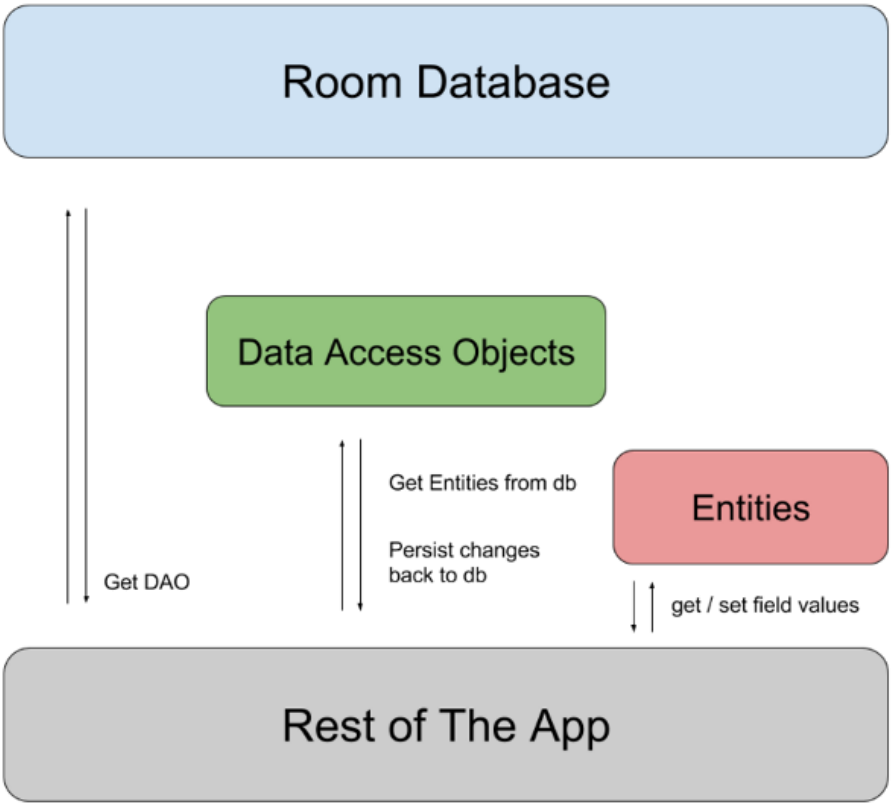
4.1.6. Room

Room[62] je biblioteka perzistencije koja pruža apstrakcijski sloj iznad SQLitea koji omogućava gladak pristup bazi podataka dok koristi punu snagu SQLitea. Postoje 3 osnovne komponente Rooma.

1. Entitet: Java ili Kotlin klasa koja predstavlja tablicu unutar baze podataka.
2. DAO: Dao ili Data Access Object je sučelje koje sadrži metode kao što su getData() ili storeData(). Te metode se koriste za pristup bazi podataka, a to sučelje biti će implementirano od strane Rooma.
3. Baza podataka: Apstraktna klasa koju obuhvaća RoomDatabase. Ovdje se definiraju entiteti (tablice) i broj verzije baze podataka. Sadrži držač (engl. *holder*) baze podataka i služi kao glavna pristupna točka za spajanje.

Postoji nekoliko ključnih aspekata Rooma koji se razlikuju od tradicionalnih ORM okvira. ORM okviri pružaju limitirani set upita, tablice su deklarirane kao Java objekti, a odnosi između tablica diktira tip upita koji se mogu obavljati. Kod Rooma su SQL umetanja, ažuriranja, brisanja i kompleksna spajanja deklarirana kao DAO. Podatci koje upiti vraćaju jednostavno su mapirani na Java objekt koji čuva te informacije u memoriji. Na ovaj način ne postoje pretpostavke o tome kako se može pristupiti podatcima. Definicije tablica su odvojene od upita koji se postavljaju. ORM-i tipično upravljaju odnosima jedan na jedan ili jedan na više određujući odnose između tablica i koriste sučelja ili set API-eva koji obavljaju SQL upite u pozadini. Na slici 16 prikazana je struktura Rooma.

[62] URL: https://developer.android.com/topic/libraries/architecture/room (24.11.2018.)



Slika 16. Struktura Room ORM-a[63]

Obilježja Rooma:

- Pruža provjeru za vrijeme kompiliranja
- Dobro se poklapa s LiveData, za nadgledanje uživo koristi se LiveData
- Testiranje raznih komponenata u Roomu je lako
- Lako za koristiti i implementirati
- Smanjuje količinu koda.

4.1.7. Sprinkles

Sprinkles[64] je biblioteka za smanjenje standardiziranog koda koja se bavi s bazama podataka u Android aplikacijama. Sprinkles dopušta SQL-u da radi ono u čemu je dobar, postavljanje kompleksnih upita. Sprinkles pomaže sa stvarima kao što su umetanje, ažuriranje i brisanje modela. Sprinkles pomaže s teškim zadatkom otpakiravanja pokazivača u model. Anotacije koje se koriste u Sprinklesu opisane su u sljedećem odjeljku.

[63] URL: https://medium.com/mindorks/room-kotlin-android-architecture-components-71cad5a1bb35 (20.12.2018.)
[64] URL: https://github.com/emilsjolander/sprinkles (24.11.2018.)

@Table – koristi se za asocijaciju model klase u SQL tablici.

@AutoIncrement – Koristi se za obilježavanje polja koje se automatski uvećava, ovo polje mora biti broj.

@Column – Koristi se za asocijaciju polja klase koja će biti SQL stupac.

@DynamicColumn – Koristi se za polje klase s dinamičkim upitima.

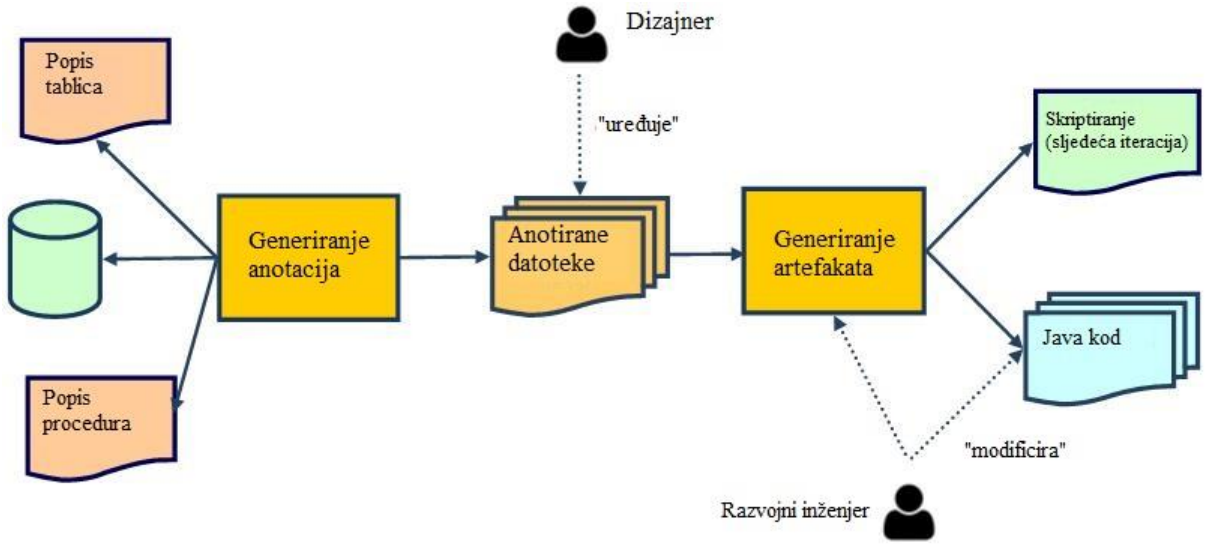
@Key – Koristi se da obilježi polje kao ključ. Više ključeva u klasi su dozvoljeni i rezultirati će kompozitnim ključem. Ključevi će najčešće biti mapirani direktno na primarne ključeve u bazi podataka.

Metoda za spremanje koristi se i za umetanje i za ažuriranje. Pravilna metoda biti će obavljena u ovisnosti s postojanjem modela u bazi podataka. Sve metode spremanja koriste provjeru da li model postoji u bazi podataka. Migracije se obavljaju korištenjem čistog SQL-a, što omogućava punu slobodu korištenja moćnih ograničenja koje se mogu staviti nad redove baze podataka. Dvije pomoćne metode pružaju oblik procesuiranja podataka prije i poslije migracije. Ovo može biti korisno kod kreiranja tablice s različitim svojstvima u kojima se trebaju zadržati podaci koji su prije bili spremljeni u sad obrisanoj tablici. Sprinkles se ne bavi s odnosima, tako je dizajniran. Potrebno je koristiti uobičajene načine za bavljenje s odnosima koje pruža SQL.

4.1.8. Freezer

Freezer[65] je generator koda koji konstruira sloj perzistentnosti u Java aplikacijama. DAO, DTO, tablice baza podataka i dokumentacija baza podataka. Tradicionalni DAO uzorak kroz godine je dokazao efektivnost upravljanja i pristupa relacijskim bazama podataka. DAO je definiran kao tradicionalan kad implementira, koristeći JDBC, operacije za pristup bazi podataka kao što su umetanje, odabir, brisanje i ažuriranje. Kada DAO vraća ili prima DTO(Data Transfer Object) kao parametar, taj DTO ima atribut za svaki stupac tablice kojim upravlja DAO. U početku ova vrsta DAO-a bili su pisani rukom, što je bio težak zadatak sklon pogreškama. Većina koda u DAO-u je repetitivna i zbog toga su se pojavili generatori koda da bi mogli stvoriti DAO iz deskriptora entiteta koristeći UML dijagram ili XML deskriptore. Na slici 17 prikazan je tok rada Freezer ORM-a.

[65] URL: https://www.theserverside.com/feature/Freezer-Putting-object-relational-mapping-ORM-tools-to-the-test (24.11.2018.)



Slika 17. Freezer[66]

Obilježja Freezera:

- Izbjegava potrebu za pisanjem generičkog koda za sloj perzistencije, sav generički kod za perzistenciju je automatski generiran
- Pruža razvojnom inženjeru kontrolu nad objektima, upitima i granicama transakcija koje se obavljaju
- Generirani kod koristi isključivo JDBC i ne zahtjeva upotrebu pokretača perzistentnosti pri pokretanju
- Dobra kontrola nad upitima olakšava optimizaciju i pruža lakše održavanje aplikacije
- Pruža interakciju sa spremljenim procedurama i funkcijama baze podataka
- Čini laganim komunikaciju između tablica i pogleda u aplikaciji kao Web servisi koji se mogu koristiti od strane drugih aplikacija
- Rezultirajuća aplikacija može se lako razviti kao mikroservis
- Intuitivan za učenje
- Prilagodljiv potrebama korisnika
- Generiranje koda događa se prije kompiliranja, time generirani DAO ne zahtjeva tumača (engl. *interpreter*) za vrijeme pokretanja
- Korištenje tradicionalnih DAO implicira manju krivulju učenja, a sljedeće značajke olakšavaju učenje Freezera: grafički alati, jednostavna sintaksa konfiguracije i dobra dokumentacija

[66] URL: http://softwaresouls.com/freezer/ (28.12.2018.)

- Mijenjajući implementaciju Freezerovog generatora modula moguće je prilagoditi kod specifičnim zahtjevima projekta.

4.1.9. Requery

Requery[67] je lagan i moćan ORM i generator SQL upita koji podržava RxJavu i značajno smanjuje količinu standardiziranog koda. Jedna od najvećih prednosti Requerya je podržavanje RxJave. Također ako razvojni inženjer poznaje RxJavu ne treba potrošiti mnogo vremena na učenje Requerya. Requery koristi anotacijsko procesiranje za vrijeme kompiliranja da generira modele entiteta klasa. Na Android OS-u ovo znači da se dobiva otprilike jednaka brzina čitanja objekata kroz upite kao da se koristit standardni pokazivač API. Kompilirane klase obavljaju posao s API-jem upita da bi iskoristili prednosti generiranih atributa. Odnosi se mogu definirati kao jedan na jedan, jedan na mnogo, mnogo na jedan i mnogo na mnogo u modelima koristeći anotacije. Kroz odnose može se navigirati u oba smjera.

Obilježja Requerya:

- Nema reflektivnosti
- Brzo pokretanje i rad
- Nema zavisnosti
- Tipkani jezik upita
- Generacija tablica
- Podržava Android (SQLite, RecyclerView, DataBinding, SQLCipher)
- Spremanje u predmemoriju
- Konverteri prilagođenog tipa.

4.2. Rješenja bez ORM-a

4.2.1. SQLite[68]

SQLite je biblioteka koja implementira SQL transakcijski mehanizam baze podataka koji je samostalan, bez servera i ne zahtjeva podešavanja. Kod SQLitea je na javnoj domeni i besplatan je za korištenje u bilo kojoj svrsi, privatnoj ili komercijalnoj. SQLite je najšire rasprostranjena i najrazvijanija baza podataka na svijetu s više aplikacija nego što se može izbrojati. SQLite je ugrađeni mehanizam baze podataka, a za razliku od većine SQL baza

[67] URL: https://github.com/requery/requery (24.11.2018.)

[68] URL: https://www.sqlite.org/about.html (03.01.2018.)

podataka nema odvojen server. SQLite čita i piše direktno na obične datoteke na disk. Kompletna SQL baza podataka s više tablica, indeksa, okidača i pogleda sadržana je u jednoj datoteci. SQLite je kompaktna biblioteka koja sa svim omogućenim značajkama teži manje od 600 kB ovisno o ciljanoj platformi. Postoji kompromis između upotrebe memorije i brzine.

SQLite generalno pokreće stvari brže što više memorije mu je omogućeno, ali radi poprilično dobro i u uvjetima s malo memorije. Testiran je prije svakog izdanja i prati ga reputacija pouzdanosti. Većina izvornog koda je posvećeno testiranju i verifikaciji. Automatizirani paket testova pokreće milijune testnih slučajeva koji uključuju stotine milijuna individualnih SQL upita. Transakcije su ACID (Atomic, Consistent, Isolated, Durable), čak i kad su prekinute sistemskim rušenjima ili nestankom struje. Sve ovo verificirano je automatiziranim testovima koji simuliraju rušenja sustava. Naravno uz sve ovo testiranje i dalje dolazi do grešaka, ali SQLite otvoreno i iskreno komunicira o svim pogreškama i pruža popis svih pogrešaka i promjena u kodu. Izvorna baza koda podržana je od strane međunarodnog tima razvojnih inženjera koji na SQLiteu rade puno radno vrijeme. Razvojni inženjeri nastavljaju proširivati mogućnosti SQLitea i unapređuju stabilnost i performanse dok u isto vrijeme održavaju kompatibilnost sa starijim verzijama.

Značajke SQLitea:

- Transakcije su atomske, konzistentne, izolirane i izdržljive čak i nakon pada sustava
- Nema potrebe za podešavanjem ili administracijom
- Implementacija SQL-a obavlja se sa svim značajkama i naprednim mogućnostima kao što su parcijalni indeksi, indeksi nad izrazima, JSON i uobičajeni izrazi nad tablicama
- Kompletna baza podataka spremljena je u jednu datoteku
- Podržava baze podataka veličine terabytea
- Maleni trag koda, manje od 600 kB
- API jednostavan za korištenje
- Brzina – u nekim slučajevima SQLite je brži od direktnih sustava datoteka
- Dobro dokumentiran izvorni kod
- Nema vanjskih zavisnosti
- Više platformski: Android, BSD, iOS, Linux, Max, Solaris, VxWorks i Windows.

4.2.2. Realm

Realm[69] baza podataka je nova vrsta mobilne baze podataka koja je razvijena od nule da bi pokretala moderne aplikacije. Lako je pomisliti da je Realm razvijen na leđima postojećih tehnologija ili da je samo moderni prijepis postojećeg proširenja funkcionalnosti. Kad se uzmu u obzir serverske baze podataka, nije bilo mnogo inovacija na klijentskoj strani. Uzimajući u obzir postojeća rješenja i nedostatke istih, Realm tim stvorio je svoj mehanizam za pohranu koji se bazira na performansama. Realm baza podataka nije povezana sa SQLiteom i nije samo SQL baza podataka. Cilja rješiti mnoge probleme koji postoje na tom polju. Također nije tip pohrane ključ – vrijednost, ova vrsta pohrane je odlična u određenim situacijama, ali razvojni inženjeri zaista žele raditi direktno s objektima u njihovom kodu. Realm baza podataka nije ORM.

ORM-i pretvaraju podatke kao što su SQL tablice u grafove objekata, tako da se mogu koristi u kodu. Realmov mehanizam za pohranu ne pretvara podatke u oblike objektnih grafova, tako da nema potrebe za ORM-om. Realm baza podataka perzistira objekte direktno u memoriju s najmanjom mogućom promjenom tipa ili strukture. S obzirom da nema mapiranja ili rasplitanja kompleksnih entiteta u realnom vremenu, Realm baza podataka je u mogućnosti da dohvati objekte iz memorije u upotrebu ekstremno brzo. Jedna od ključnih značajki Realma je što se pisanje i čitanje podataka može obavljati na glavnoj niti, bez bojazni blokiranja korisničkog sučelja. Osnova Realma pisana je u super brzom C++-u i mapira C++ objekte na disk. Zato jer je razvijen uz SDK-ove i API-eve ponašanje je vrlo slično načinu na koji mobilne aplikacije koriste podatke zbog čega se može dobiti lagani mentalni model kako što radi. Ovakva usklađenost podsjeća na iOS. Apple proizvodi hardver za iPhone i pruža razvojnim inženjerima najbolji mogući SDK da iskoriste većinu hardvera. Zbog toga pregledavanje duge liste kontakata na iPhoneu nikad ne zapinje, unatoč tome što hardver nije jak kao u Android uređajima. Zbog toga što je osnova Realma u C++-u, ima potencijal da se može pokrenuti bilo gdje, to jest nije ograničen na određenu platformu ili OS.

[69] URL: https://academy.realm.io/posts/realm-object-centric-present-day-database-mobile-applications/ (03.01.2018.)

4.3. Usporedba performansi[70]

U trenutku pisanja ovog rada prema web stranici Android Arsenal[71] postoji 46 ORM rješenja za Android OS. Za testiranje su odabrana najpopularnija rješenja uz SQLite i NOSQL pristup. Uzete su 3 varijante (jednostavno, kompleksno i balansirano), a za svaku od varijanti 4 operacije CRUD (engl. *Create, Read, Update, Delete*).

Modeli su postavljeni na sljedeći način:

```
public class Library{
    String address;
    String name;
}

public class Book{
    String author;
    String title;
    int pagesCount;
    int bookId;
    Library library;
}
public class Person{
    String firstName;
    String secondName;
    Date birthdayDate;
    String gender;
    long phone;
    Library library;
}
```

Testni slučajevi

Simple – jednostavno – 1 Library objekt s 1000 Book objekata

Complex – kompleksno – 1 Library objekt za 500 Book objekata i 400 Person objekata (5 Libraries, 2500 Books, 2000 Persons)

Balanced – balansirani – 1 Library objekt za 50 Book objekata i 50 Person Objekata (50 Libraries, 2500 Books, 2500 Persons)

Za svako testiranje punjenje predmemorija / lijena inicijalizacija je bila isključena, a izvedeno je u prosjeku 10 mjerenja po operaciji.

[70] URL: https://github.com/AlexeyZatsepin/Android-ORM-benchmark (24.11.2018.)

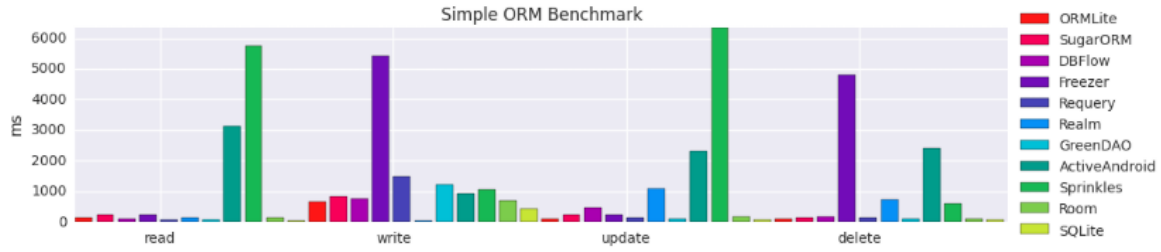
[71] URL: https://android-arsenal.com/tag/69?page=2&sort=created (22.11.2018.)

Tablica rezultata

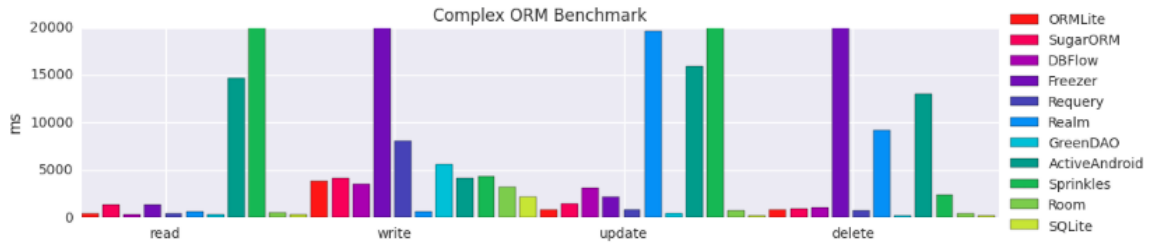
Tablica 2. Prikaz rezultata testiranja brzine pojedinih ORM-a, rezultati u milisekundama[72]

Vrsta mjerenja	Jednostavno mjerenje				Kompleksno mjerenje				Balansirano mjerenje			
ORM	**write**	**read**	**update**	**delete**	**write**	**read**	**update**	**delete**	**write**	**read**	**update**	**delete**
ORMLite	151	666	122	105	445	3836	857	811	1563	3426	724	728
SugarORM	245	842	252	152	1402	4129	1467	1003	2204	4397	1702	1197
Freezer	248	5430	240	4797	1337	78982	2221	22104	3255	134942	1887	29515
DBFlow	97	757	459	186	360	3534	3124	1044	1129	4653	5204	1268
Requery	87	1501	147	129	461	8057	861	802	1368	8002	886	763
Realm	151	29	1079	723	698	688	19666	9180	1522	210	21129	10006
GreenDAO	81	1238	117	97	357	5552	455	274	598	5905	504	315
ActiveAndroid	3123	930	2293	2423	14671	4165	15958	13023	17213	4653	19303	14642
Sprinkles	5766	1050	6364	605	25978	4334	65579	2428	27774	4526	37705	2519
Room	131	699	170	109	562	3201	717	403	1330	3532	790	507
SQLite	50	436	63	80	386	2155	192	284	1146	2313	213	318

Cjelokupni rezultati



Slika 18. Jednostavno ORM mjerenje[73]

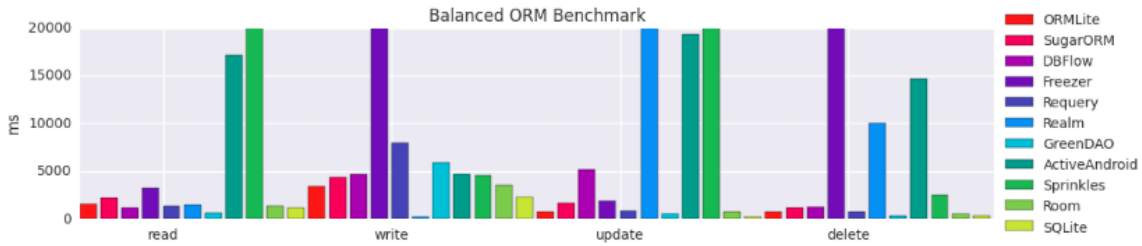


Slika 19. Kompleksno ORM mjerenje[74]

[72] URL: https://github.com/AlexeyZatsepin/Android-ORM-benchmark (24.11.2018.)

[73] isto

[74] isto



Slika 20. Balansirano ORM mjerenje[75]

4.3.1. Pregled rezultata

ORMLite ne oslobađa razvojnog inženjera od standardiziranog koda, ali je jedno od najbržih proširenja s ugrađenim spremanjem u predmemoriju, odgođenom inicijalizacijom i dobrom dokumentacijom. Unatoč svim prednostima, jedan od glavnih razloga korištenja ORM-a je da poštedi razvojnog inženjera od pisanja standardiziranog repetitivnog koda, što je u drugim proširenjima izvedeno bolje i time ubrzava vrijeme razvoja.

SugarORM ima prednost lakoće korištenja, dok je jedan od nedostataka pristup implementaciji. Ne radi s opcijom „Instant Run“ u Android Studiju i kod korištenja se ta opcija treba isključiti.

Freezer je relativno novo proširenje i nije jako rašireno. Nije pogodno za komplicirana manipuliranja s podacima i brzina rada je poprilično spora u usporedbi s drugim proširenjima.

DBFlow je jedno od najboljih proširenja po parametrima koji su korišteni u ovom testiranju. Ima jedno od najbržih brzina čitanja i pisanja i prosječnu brzinu ažuriranja i brisanja.

Requery ima kompliciraniji pristup instalaciji i šturu dokumentaciju, ali ako razvojni inženjer razumije kako raditi s tim proširenjem, onda problemi nestaju. Podržava razne baze podataka i bogato je funkcionalnostima.

Realm je relativno brzo proširenje, sve poveznice su jednostavno implementirane s obzirom na objektnu orijentiranost baze podataka. Odlična dokumentacija i jedno od najboljih opcija za spremanje podataka na mobilnim uređajima. Nedostatak ovog proširenja je što se veličina APK datoteke povećava za 2.5 MB.

GreenDao je fleksibilan i prikladan za korištenje u slučaju komunikacije jedan prema mnogo. Generacija koda je dosta neuredna i kao rezultat toga klase se popunjavaju s raznim metodama i komentarima.

[75] URL: https://github.com/AlexeyZatsepin/Android-ORM-benchmark (24.11.2018.)

ActiveAndroid nije pretjerano brzo proširenje, ali je rješenje koje se kroz svoje vrijeme postojanja pokazalo pouzdanim i dobrim. Poprilično je popularno i zbog toga se većina problema i grešaka riješila, a samim time se broj nerješivih problema smanjio na minimum.

Sprinkles se pokazao kao najlošije proširenje po performansama u ovom testiranju.

Room je interesantno rješenje prikazano na konferenciji Google I/O 2017 kao optimalno za rad s bazama podataka u Android OS. Po performansama je na vrhu i zato što je ovo rješenje prezentirano od strane Googlea, brzo postaje popularno i neće biti problema s nalaženjem rješenja za moguće probleme koji nastanu.

SQLite ima mnogo koda, duže vremena za pisanje, ali po brzini rada i fleksibilnosti je i dalje pobjednik.

4.3.2. Rezultati testiranja

Najbrži ORM-i su Realm, GreenDAO, ORMLite i Room, ali ako su performanse kritične za projekt, čisti SQLite s prilagođenim spremanjem u predmemoriju je i dalje najbolji odabir. Po drugim parametrima najbolji su Realm, DBflow i GreenDAO. Za manje projekte srednje kompleksnosti preporuča se korištenje DBFlowa ili GreenDAOa, a ako veličina APK datoteke nije važna onda Realm. Realm i Room su prikladni za velike projekte, a ako njihove značajke nisu dovoljne, ili se razvojni inženjer odluči za pristup bez ORM-a, onda se preporuča korištenje ugrađenog API-ja za SQLite.

5. Zaključak

Glavni zadatak ovog rada bio je prikaz proširenja funkcionalnosti za Android platformu s opširnijim prikazom ORM proširenja funkcionalnosti.

Dobra i korisna proširenja funckionalnosti otvorenog koda neizbježno će učiniti razvijanje Android aplikacija lakšim i bržim, što je prikazano u radu opisom najpopularnijih rješenja koje imaju velike zajednice razvojnih inženjera, a malo grešaka.

ORM proširenja funkcionalnosti su zanimljiv pristup bavljenja s bazama podataka koji se koristi u raznim primjenama, a u ovom radu ne zagovara se isključivo korištenje ORM rješenja, već su prikazane prednosti i nedostatci najpopularnijih ORM rješenja i dva pristupa bez ORM-a.

Ovisno o potrebama projekta treba odabrati rješenje koje će najbezbolnije rješiti predstavljeni problem, zbog toga ne postoji univerzalno idealan izbor. Potrebno je pažljivo definirati potrebe projekta i zaključiti koje je najbolje rješenje.

Razvoj Android aplikacija je specifičan i dinamičan te ovisi o vještinama razvojnog inženjera. Zajednica razvojnih inženjera na internetu je vrlo rasprostranjena i široka budući da su Android razvojni inženjeri aktivni na stručnim forumima kao što su Stackoverflow i Github. Budući da se Android ekosustav kontinuirano poboljšava i optimizira, može se očekivati da će se s njim poboljšavati i optimizirati proširenja funkcionalnosti.

U ovom radu prikazano je da pametno korištenje proširenja funkcionalnosti uvelike može olakšati razvoj aplikacija i pomoći razvojnom inženjeru da isporuči proizvod koji je stabilan i bez grešaka u kratkom vremenskom roku.

6. Literatura

1. ActiveAndroid. ActiveAndroid

   URL: http://www.activeandroid.com/ (24.11.2018.)

2. ActiveAndroid Guide. CodePath

   URL: https://guides.codepath.com/android/activeandroid-guide (04.12.2018.)

3. Android architecture components – Room and Kotlin. MindOrks

   URL: https://medium.com/mindorks/android-architecture-components-room-and-kotlin-f7b725c8d1d (24.11.2018.)

4. Android Butterknife example. JournalDev

   URL: https://www.journaldev.com/10439/android-butterknife-example (07.10.2018.)

5. Android developer portal with tools, libraries and apps. Android arsenal

   URL: https://android-arsenal.com/tag/69?page=2&sort=created (22.11.2018.)

6. Android OS & its features. Magivatech

   URL: http://magivatech.com/blog/info/android-versions (23.01.2019.)

7. Android developers. Android Studio

   URL: https://developer.android.com/studio/intro/ (10.01.2019.)

8. Android SDK: Working with Picasso

   URL: https://code.tutsplus.com/tutorials/android-sdk-working-with-picasso--cms-22149 (07.10.2018.)

9. Android testing part 1: Espresso basics. MindOrks

   URL: https://medium.com/mindorks/android-testing-part-1-espresso-basics-7219b86c862b (01.12.2018.)

10. Android user interface testing with Espresso – Tutorial. Vogella

    URL: http://www.vogella.com/tutorials/AndroidTestingEspresso/article.html#espresso_introduction (07.10.2018.)

11. Bluestacks

URL: https://www.bluestacks.com/ (27.09.2018.)

12. ButtterKnifeZelezny. Github

    URL: https://github.com/avast/android-butterknife-zelezny/blob/master/README.md (28.11.2018.)

13. CyanogenMod has now been installed on over 2 million devices. Android Police

    URL: https://www.androidpolice.com/2012/05/28/cyanogenmod-has-been-installed-over-2-million-times-doubles-install-numbers-since-january/ (14.12.2018.)

14. Dagger 2 to the rescue – Dependency injection. Medium

    URL: https://medium.com/@CStudio/dagger-2-to-the-rescue-dependency-injection-6b6033772c7c (03.12.2018.)

15. DBFlow. Agrosner Gitbooks

    URL: https://agrosner.gitbooks.io/dbflow/ (10.12.2018.)

16. DBFlow guide. CodePath

    URL: https://github.com/codepath/android_guides/wiki/DBFlow-Guide (24.11.2018.)

17. Dependency injection with Dagger 2 for Android. EnvatoTuts

    URL: https://code.tutsplus.com/tutorials/dependency-injection-with-dagger-2-on-android--cms-23345 (03.12.2018.)

18. Emulator. Techopedia

    URL: https://www.techopedia.com/definition/4788/emulator (27.09.2018.)

19. Enable multidex for apps with over 64k methods. Developer Android

    URL: https://developer.android.com/studio/build/multidex (23.01.2019.)

20. Five reasons to use RxJava in your projects. JaxEnter

    URL: https://jaxenter.com/5-reasons-use-rxjava-148982.html (03.12.2018.)

21. Freezer Persistence Framework. Software Souls

    URL: http://softwaresouls.com/freezer/ (28.12.2018.)

22. Freezer: Putting object relational mapping (ORM) tools to the test. TheServerSide

URL: https://www.theserverside.com/feature/Freezer-Putting-object-relational-mapping-ORM-tools-to-the-test (24.11.2018.)

23. Genymotion for Windows. Softonic

    URL: https://genymotion.en.softonic.com/ (23.01.2018.)

24. GenyMotion User Manual. GenyMotion

    URL: https://docs.genymotion.com/latest/pdf/PDF_User_Guide/Genymotion-2.12-User-Guide.pdf (27.09.2018.)

25. Getting started with Bluestacks 4. Bluestacks Help Center

    URL: https://support.bluestacks.com/hc/en-us/articles/360013732972-How-to-install-an-app-on-BlueStacks-4- (27.09.2018.)

26. Google makes Kotlin a first class language for Writing Android apps. TechCrunch

    URL: https://techcrunch.com/2017/05/17/google-makes-kotlin-a-first-class-language-for-writing-android-apps/ (14.12.2018.)

27. Google Mobile Services. Android

    URL: https://www.android.com/gms/ (13.12.2018.)

28. Google Play hits 1 million apps. Mashable

    URL: https://mashable.com/2013/07/24/google-play-1-million/?europe=true#gwIL7YDq_iqs (14.12.2018.)

29. Gradle user Manual. GradleDocs

    URL: https://docs.gradle.org/current/userguide/userguide.html (26.09.2018.)

30. Gradle vs. Maven. DZone

    URL: https://dzone.com/articles/gradle-vs-maven (27.09.2018.)

31. greenDao: Android ORM for your SQLite database. Greenrobot

    URL: http://greenrobot.org/greendao/ (12.11.2018.)

32. How many Java developers are there in the world. Plumbr

    URL: https://plumbr.io/blog/java/how-many-java-developers-in-the-world (30.09.2018.)

33. How to install plugins in Android Studio.

URL: https://medium.com/@adetayo_james/how-to-install-plugins-in-android-studio-f64f1d584438 (3.11.2018.)

34. How to use Picasso image loader library in Android. Stack Tips

URL: https://stacktips.com/tutorials/android/how-to-use-picasso-library-in-android (01.12.2018.)

35. Introduction to Android Studio IDE. MindOrks

URL: https://medium.com/mindorks/project-marble-android-studio-stable-release-3-3-d7b177ccbd2c (15.01.2018.)

36. Java. Techopedia

URL: https://www.techopedia.com/definition/3927/java (27.09.2018.)

37. Java vs. Kotlin – Android development. Medium

URL: https://medium.com/@snrawoof93/java-vs-kotlin-android-development-7550660dc2b0 (30.09.2018.)

38. Kotlin vs. Java: Which one you should choose for your next Android app. Netguru

URL: https://www.netguru.co/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app (30.09.2018.)

39. Lessons learned from migrating to ORMLite. AndroidPub

URL: https://android.jlelse.eu/lessons-learned-from-migrating-to-ormlite-7843eb55ec84 (04.12.2018.)

40. Meet RxJava: The missing Reactive programming library for Android. Toptal

URL: https://www.toptal.com/android/functional-reactive-android-rxjava (03.12.2018.)

41. Number of available applications in the Google Play Store. Statista

URL: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (13.12.2018.)

42. Object-relational mapping(ORM). TechTarget

URL: https://searchwindevelopment.techtarget.com/definition/object-relational-mapping (12.11.2018.)

43. OkHttp Interceptors. Soulesidibe

URL: https://soulesidibe.com/2017/02/25/okhttp-interceptors/ (01.12.2018.)

44. OrmLite – Lightweight Java ORM supports Android and SQLite. OrmLite

    URL: http://ormlite.com/sqlite_java_android_orm.shtml (24.11.2018.)

45. Performance comparison of Android ORM frameworks. Github

    URL: https://github.com/AlexeyZatsepin/Android-ORM-benchmark (24.11.2018.)

46. Plug-in(Computing). Wikipedia

    URL: https://en.wikipedia.org/wiki/Plug-in_(computing) (11.11.2018.)

47. Realm is an Object-centric modern database for mobile apps. Realm

    URL: https://academy.realm.io/posts/realm-object-centric-present-day-database-mobile-applications/ (03.01.2018.)

48. Requery. Github

    URL: https://github.com/requery/requery (24.11.2018.)

49. Retrofit 2.0, A type-safe HTTP Client for Android and Java. GKMIT

    URL: http://www.gkmit.co/articles/retrofit-2-0-a-type-safe-http-client-for-android-and-java (07.10.2018.)

50. Room – Kotlin, Android Architecture components. MindOrks

    URL: https://medium.com/mindorks/room-kotlin-android-architecture-components-71cad5a1bb35 (20.12.2018.)

51. Room persistence library. Developers Android

    URL: https://developer.android.com/topic/libraries/architecture/room (24.11.2018.)

52. RxJava Anatomy: What is RxJava, how RxJava is designed, and how RxJava works. MindOrks

    URL: https://blog.mindorks.com/rxjava-anatomy-what-is-rxjava-how-rxjava-is-designed-and-how-rxjava-works-d357b3aca586 (03.12.2018.)

53. Should developers use third-party libraries. Scalable path

    URL: https://www.scalablepath.com/blog/third-party-libraries/ (3.11.2018.)

54. Should I or should I not use ORM. Medium

URL: https://medium.com/@mithunsasidharan/should-i-or-should-i-not-use-orm-4c3742a639ce (24.11.2018.)

55. Sprinkles. Github

URL: https://github.com/emilsjolander/sprinkles (24.11.2018.)

56. SQLite

URL: https://www.sqlite.org/about.html (03.01.2018.)

57. Sugar ORM

URL: http://satyan.github.io/sugar/index.html (24.11.2018.)

58. Tasting Dagger 2 on Android. Fernando Cejas

URL: https://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/ (03.12.2018.)

59. The advantages and disadvantages of using ORM. StackOverflow

URL: https://stackoverflow.com/questions/4667906/the-advantages-and-disadvantages-of-using-orm (24.11.2018.)

60. Using Kotlin for Android development. Kotlin

URL: https://kotlinlang.org/docs/reference/android-overview.html (27.09.2018.)

61. Using Retrofit 2.x as REST client – tutorial

URL: http://www.vogella.com/tutorials/Retrofit/article.html (01.12.2018.)

62. What is Maven. Apache Maven Project

URL: https://maven.apache.org/what-is-maven.html (26.9.2018.)

7. Popis slika

8. Popis tablica

Proširenja funkcionalnosti Android platforme

Sažetak

U ovom diplomskom radu prikazana je Android platforma, najpopularnija proširenja funkcionalnosti i ORM proširenja funkcionalnosti, te njihova usporedba. Na početku je objašnjeno što je Android platforma, koje su značajke operacijskog sustava, programski jezici koji se koriste, alati izgradnje, te emulatori. Nakon toga prikazana su najpopularnija proširenja funkcionalnosti za Android platformu, s objašnjenjem kako i zašto se koriste i kako olakšavaju razvoj za Android platformu. U drugoj polovici rada objašnjeno je što su ORM-i i njihovi prednosti i nedostatci. Također su prikazana najčešće korištena ORM proširenja funkcionalnosti i dva rješenja koja ne koriste ORM, te je napravljena usporedba performansa između predstavljenih rješenja.

Ključne riječi: Android, proširenja funkcionalnosti, ORM, usporedba ORM-a

Plugins for Android

Summary

In this graduate thesis the Android platform, most popular Android plugins and ORM plugins are explained and later compared. At the beginning the Android platform is explained, what are its features, which programming languages are used along with build tools and emulators. After this the most popular plugins for Android are presented with an explanation of how and why they are used and how they make Android development easier. In the second half of this thesis ORMs are explained alongside their advantages and disadvantages. The most used ORM plugins are presented alongside two non ORM solutions, after which a detailed performance comparison is made.

Key words: Android, plugins, ORM, ORM comparison