UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

PUNIM DIPLOME

DRAFT

Tema:

Zhvillimi i aplikacionit ne Android për menaxhimin e bashkëudhëtimit ne taksi

Mentori:

Prof. Dr. Blerim Rexha

Kandidati:

Agon Hoxha

Prishtinë, Nëntor 2019

Abstrakt

Abstract

1. Hyrja

1.1. Hyrje

Aplikacionet marrin rol me te madhe ne jeten tone, gjithnje e me shume. Ne vend te komunikimeve te formes relativisht te vjeter, si thirrje dhe mesazhe permes rrjetes tradicionale qe perdorej ne telefona, eshte nje tentim i vazhdueshem qe te gjendet menyra me eficiente per te thjeshtesuar qeshtje te perditshmerise. Nje drejtim ne te cilen kjo ka ndodhur eshte edhe udhetimi. Prej navigimit permes telefonit, deri ne drejtim te vetures permes telefonit mobil. Ky punim nuk eshte ambicioz sa drejtimi i vetures permes telefonit mobil, mirepo tenton te ofroj zgjidhje dhe te thjeshtesoj nje qeshtje qe prek shume njerez tere kohen.

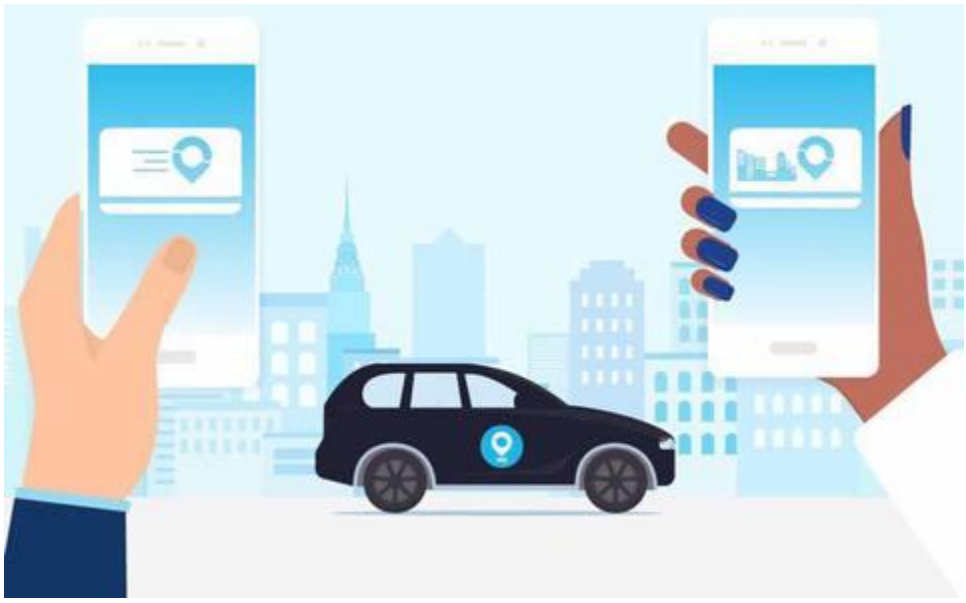


Figura 1. Perdorimi i ridesharing apps ne vend te vozitjes

Udhetimi i perbashket eshte ekonomikisht me i qendrueshem, kjo eshte pa dyshim nder arsyet qe aplikacionet ne lidhje me bashkeudhetim kane arritur nje lloj fame. Shembull i kesaj lloj aplikacioni, eshte Uber - aplikacion permes te cilit jane bere mbi 10 miliard udhetime, apo 14 milion udhetime qdo dite. [1] Shembull tjeter eshte edhe Lyft, qe gjersa nuk eshte afer nivelit te njejte me Uber, prap se prap ka nje perdorim te madh, me mbi 1 miliard udhetime deri me Shtator 2018. [2]

1.2. Motivimi

Ne Kosove, keto aplikacione nuk jane ne perdorim. Udhetimi i perbashket behet permes transportit urban, apo transportit me vetura qe defakto eshte ilegal. Se fundmi, organizimi i bashkeudhetimit ka kaluar online ne Facebook, permes grupeve si Hitchhiker Kosova - UDHË, ku qdokush mund te beje postim ne form te ofertave, tregojne se qfare rruge do kalojne, oren dhe shpesh e qartesojne edhe qmimin. [3]

Gjersa kjo eshte zgjidhje per kete problem, mendoj se ekziston nje zgjidhje me e mire. Ky paraqet motivimin e punimit - krijimi i nje aplikacioni qe do zevendesonte planifikimet e tilla neper grupe duke ofruar zgjidhje me te mire.

1.3. Përshkrimi i problemit

Perdorimi i platformave te tilla per kete qeshtje mund te jete zgjidhje e përkohshme, por zgjidhja qe e kane menduar shfaq disa probleme:

a. Perdor infrastrukture te nje platforme qe nuk eshte e menduar fare per kete qeshtje.
b. Mungon mundësia per shfaqje te sakte te pikënisjes dhe përfundimit.
c. Komunikimi mes personave qe pajtohen per udhëtim mund te behet vetëm përmes komentimit, apo platformës te Facebook per komunikim – Messenger, qe paraqet problem pasi qe zakonisht ndalon komunikimin ndërmjet personave nëse nuk janë miq ne platforme.

Zgjidhja qe do e ofroj ne kete punim, do tentoj qe te i zgjedh këto mangësi qe i ka metoda qe perdoret tani per tani, duke ofruar nje sistem qe eshte krijuar per pikërisht kete qellim, te përfshij harta dhe navigim ne vetvete qe te e lehtësoj tere procesin, si dhe te ofroj nje menyre komunikimi mes përdorueseve te aplikacionit.

Figura 2. Organizimi i bashkeudhetimit ne Facebook

2. Sistemi Operativ Android

2.1. Hyrje

Android eshte nje sistem operativ per pajisje te ndryshme, me fokus te veqante ne pajisje mobile te formes smartphone, mirepo edhe ne pajisje IoT, TV dhe PC. [4][5][6]. Fillimisht e zhvilluar nga Open Handset Alliance, por pastaj kalon ne pronesi te Google LLC ne Tetor te vitit 2007, e cila e ka vazhduar zhvillimin prej qe e ka marr nen pronesi. [7] Android ka qasje open-source rreth shumicen e qeshtjeve; ne fakt, pas qdo lansimi te versioni te Android, publikohet AOSP (Android Open Source Project) ne GitHub ku mund te shihet saktesisht se qka permban sistemi operativ, me disa perjashtime. [8] Pasi qe eshte open source, ka diqka qe sistemet operativ konkurrente si iOS veshtire qe mund te thuhet se e kane - nje perkrahje nga komuniteti. Ekzistojne komunitete si xda-developers qe merren me krijim te versioneve te modifikuara te Android, duke aplikuar ndryshime ne source code te sistemit, kernel dhe si rezultat, shpesh arrihet shfrytezimi me i mire i resurseve qe posedon pajisja se sa qe ka mundur te shfrytezohet permes versionit te Android qe ja u ka vendosur prodhuesi i pajisjes.

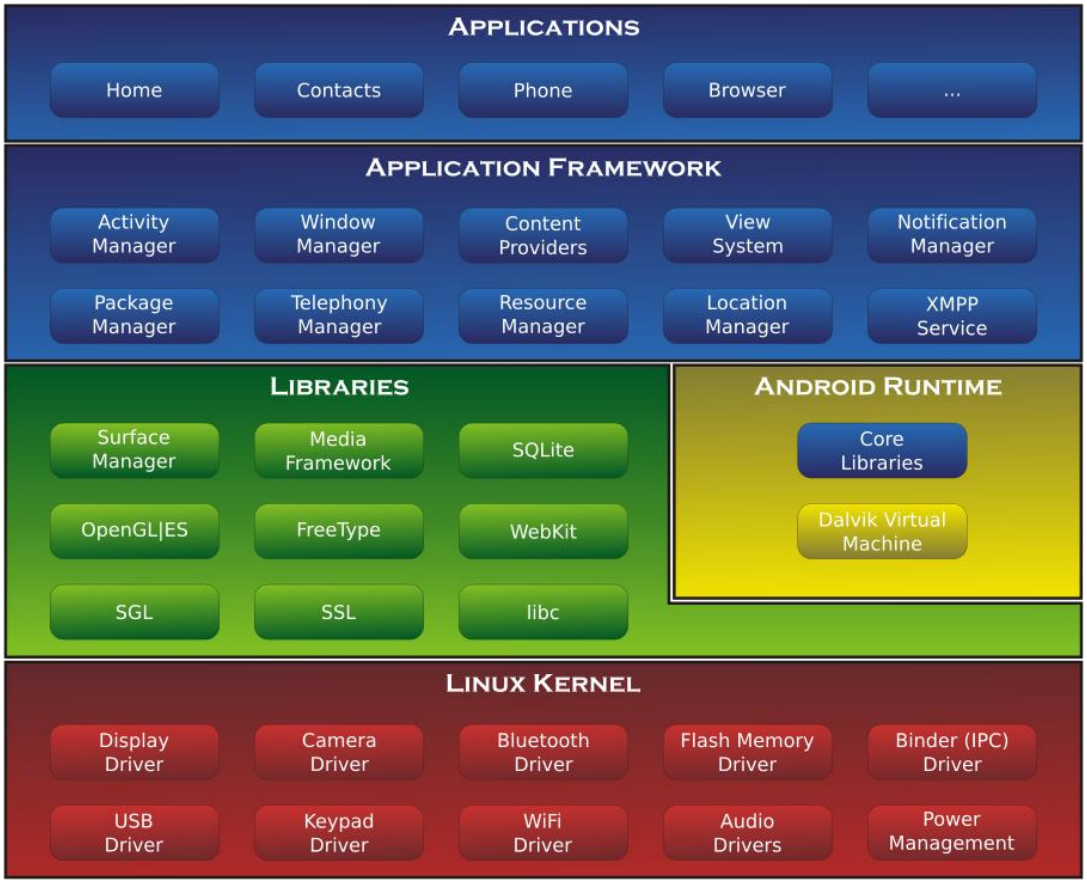


Figura 3. Arkitektura e Android

Kjo lloj veçorie qe gjithkush mund te krijoj versionin e vet te sistemit operativ Android ngjan me Linux distributions, kjo ka arsye: Android ne prapavije eshte Linux, me saktesisht kerneli qe perdor eshte kernel i modifikuar i Linux, dhe ka nje qasje te ngjajshme ne qasjen qe perdoruesi ka ne sistem operativ. Linux ishte pjese kyqe e Android prej fillimit te zhvillimit, dhe ende eshte i bazuar ne te, dhe zyrtar nga Google janë cituar duke thene se ne te ardhmen, vetëm se do rritet nderlidhshmeria mes Android dhe Linux. [9]

Me pare, Android perdorte emrin e ëmbëlsirave si emër te versioneve me renditje alfabetike, por qe nga versioni i fundit – qe ne kohe te shkrimit te këtij punimi eshte Android 10, nuk aplikohet me emërimi i

tille. Mirepo emërimi i tille eshte vetëm emërim komercial, pasi qe ne kuptim te zhvillimit, eshte me drejte te thuhet qe versioni i tanishëm eshte versioni 29, siq eshte sqaruar me poshtë përmes ilustrimit te versioneve ne tabel.

Emri komercial	Versioni komercial	Niveli API
(Pa emër)	1.0	1
Petit Four	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Eclair	2.0	5
	2.0.1	6
	2.1	7
Froyo	2.2	8
Gingerbread	2.3	9
	2.3.7	10
Honeycomb	3.0	11
	3.1	12
	3.2.6	13
Ice Cream Sandwich	4.0.0	14
	4.0.1	15
Jelly Bean	4.1	16
	4.2	17
	4.3	18
KitKat	4.4	19
	4.4.4	20
Lollipop	5.0	21
	5.1	22
Marshmallow	6.0	23
Nougat	7.0	24
	7.1	25
Oreo	8.0	26
	8.1	27
Pie	9.0	28
10	10.0	29

Tabela 1. Versionet e Android

Nje nder mangësitë me te shpeshta qe citohet rreth Android, eshte shpërndarja e versioneve. Me kete nënkuptojmë se sa perqind te pajisjeve ne treg per momentin kane cilin version. Zakonisht, përqindja e pajisjeve qe kane versionin e fundit eshte tejet e vogël. Kjo paraqet nje sfide per zhvillim te aplikacioneve, pasi qe duhet te merret nje vendim rreth cili nivel i API do duhet te përdorur. Nese perdoret version i vjetër i API, rritet numri i pajisjeve ku mund te instalohet aplikacioni, por do kemi me pak veçori dhe funksione qe mund te përdorim; ne anën tjetër, nëse perdoret version tejet i ri i API, zvogëlohet numri i pajisjeve ku mund te instalohet aplikacioni, por do kemi qasje ne veçori dhe funksione qe mund te na lehtësojnë punën dhe te rrisin performancen e aplikacionit.

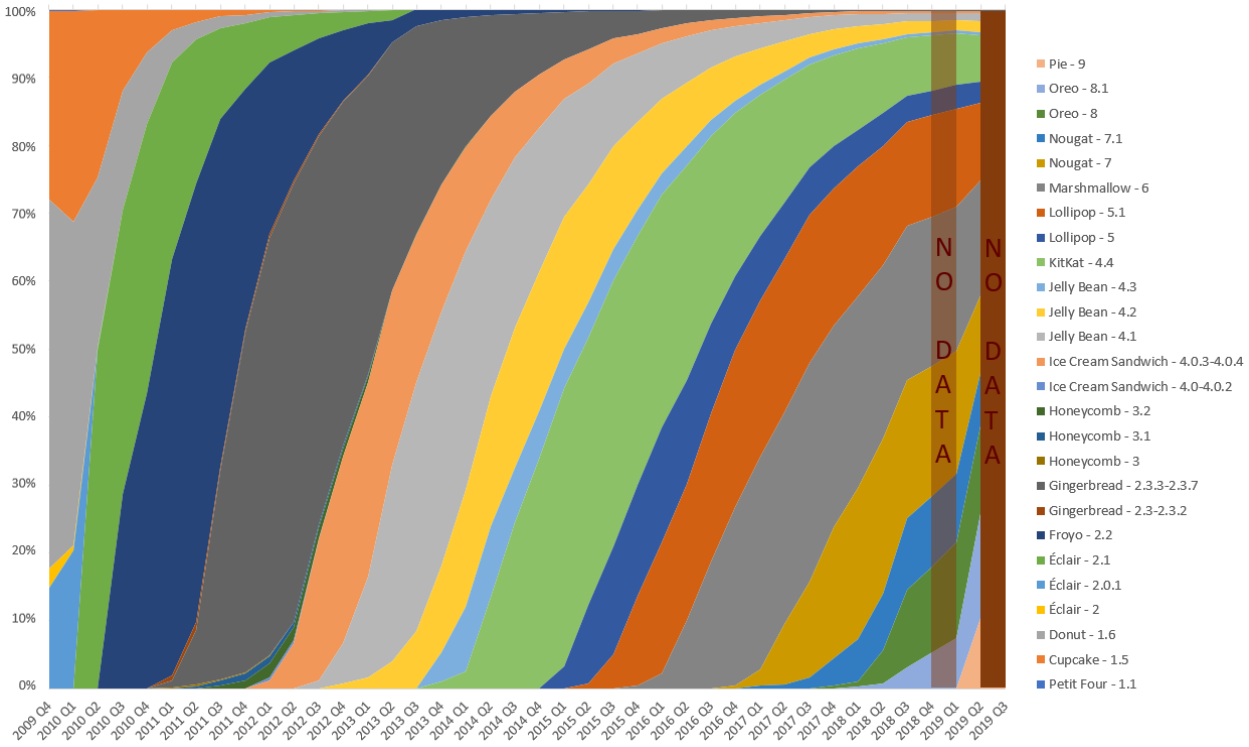


Figura 4. Shpërndarja e versioneve te Android

Per aplikacionin e krijuar, eshte perdorur versioni minimal prej API 21, apo komercialisht njohur si Lollipop. Me kete jemi siguruar qe aplikacioni te mund te instalohet ne afersisht 90% te pajisjeve Android ne përdorim, dhe njëkohësisht kemi marrur qasje ne disa veçori qe na duhen.

2.2. Komponentët e aplikacionit

Cdo aplikacion perbehet prej disa komponenteve: aktiviteteve, sherbimeve, broadcast receivers dhe content providers.

2.2.1. Aktivitetet

Nje aktivitet eshte nje njësi e vetme e aplikacionit ne te cilën përdoruesi vepron. Nje aktivitet kalon neper disa etapa, duke filluar me funksionin onCreate(). Prej atu mund te kalon neper disa etapa, apo te përfundoj (procesi te nderprehet).

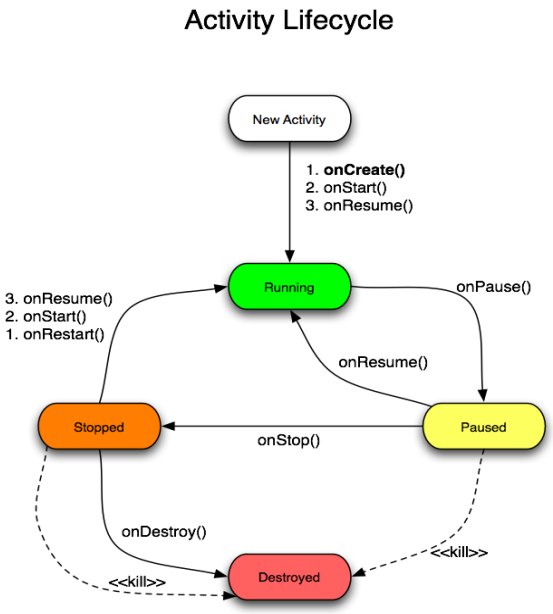


Figura 5. Etapat e jetes te aktivitetit

2.2.2. Servise

Serviset shfrytezohen per te kryer operacione qe marrin kohe te gjata ne prapavije, apo qe duhen te ndërmarrën operacione te caktuara vetem ne kushte te catktuara.

2.2.3. Broadcast Receiver

Broadcast Receiver janë komponentë qe lejojnë te përcaktojmë pergjigjje per ngjarje ne aplikacion apo ne tere sistemin. Qdo broadcast receiver i regjistruar menagjohen nga Android kur te ndodh ngjarja

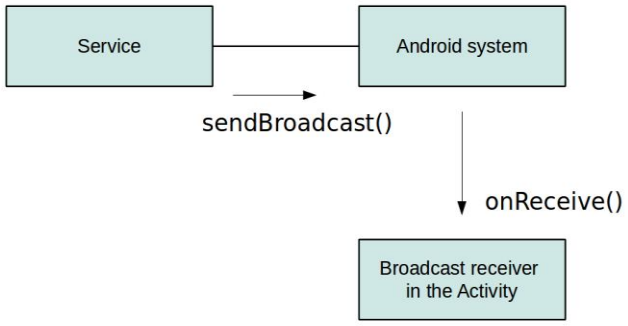


Figura 7. Shfrytezimi i Serviseve ne Android

e cekur. Shembull, nje aplikacion mund te deklaroj Broadcast Receiver per ACTION_BOOT_COMPLETED, dhe te cek nje funksion. Sapo te ndezët telefoni, ai funksion i aplikacionit tone do te aktivizohet. Receiver duhet te jete i regjistruar ne Manifestin e aplikacionit. Manifesti i aplikacionit ja u shpjegon sistemit operativ, Play Store dhe build tools, te dhënat esenciale mbi aplikacionin tone, si aktivitetet qe përmban, lejet (permissions) qe i duhen, shërbimet qe i ka, eventet ne te cilat do reagoj (broadcast receivers), etj.

2.2.4. Content Providers

Content Provider menagjon qasjen ne nje depo te te dhënave. Providers zakonisht japin nje UI te vet te cilën mund t'e manipulojmë ashtu si na duhet. Content providers zakonisht perdoren per dy raste: kur dëshirojmë qe te qasemi ne te dhëna qe aplikacioni jone përmban, apo te lejojmë qe aplikacioni jone te ndaj te dhëna me aplikacione tjera.

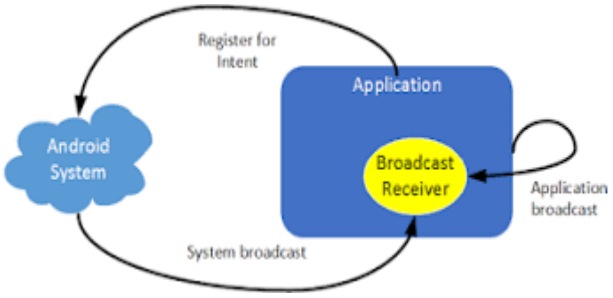


Figura 6. Funksionimi i Broadcast Receiver

2.3. Rast studimi: Kotlin vs. Java

2.3.1. Hyrje

Kur zhvillojmë aplikacione ne Android, kemi disa opsione sa i përket gjuhës qe dëshirojmë te përdorim:

1. Java
2. Kotlin
3. C#
4. C++
5. Python
6. Corona
7. Gjuhet skriptuese (HTML, CSS, Javascript)

Shumica e aplikacioneve qe krijohen, krijohen me Java, por dy vitet e fundit, ka pasur tentime qe te kalohet ne nje gjuhe tjetër – me saktësisht, ka pasur tentime te kalohet sa me shume ne gjuhen Kotlin. Sot, afro 16% te aplikacioneve ne tere Play Store e përdorin ne vend te Java. [10] Java dhe Kotlin kane përparësitë dhe mangësitë e veta qe do i shtjellojmë tani.

2.3.2. Java

Java është momentalisht gjuha me e përdorur programuese. Kjo është me arsye, pasi qe programet e shkruara ne Java mund qe te teknikisht te funksionojnë ne mbi 3 miliard pajisje. Me tej, është pohuar nga Oracle, qe aktivisht e zhvillon, qe janë mbi 21 miliard cloud-connected JVMs. [11] Fillimisht, krijuar nga Sun MicroSystems, është gjuhe e bazuar ne klasa, e orientuar ne objekte, dizajnuar te këtë

te implementuar sa me pak varësi. Është tentuar qe aplikacionet te shkruhen një here dhe te ekzekutohen kudo (koncepti WORA – Write Once, Run Anywhere). Version i tanishëm është Java 13, dhe Java 14 e planifikuar për publik ne Mars, 2020. Ndërsa Java 12 është ende përkrahur. Dizajni dhe zhvillimi i gjuhës Java është bazuar ne disa koncepte:

• Duhet te jete thjeshte, orientuar ne objekte dhe familjare.
Luan rol te një motori për pajisje te vogla dhe te mëdha kompjuterike, si i tille, ky është prioritet.

• Duhet te jete sigurte dhe pa probleme.
Java është fuqishëm për shkak te përkrahjes. Mund te përdoret ne sisteme te ndryshme. Përkrah menaxhim te memories dhe largim te mbeturinave ne memorie. Qe këto te mbesin te vërteta, versionet e reja duhen te jene me te sigurta dhe pa probleme. [12]

• Duhet te jete neutral ne arkitekture, dhe e paanshme.
Qe te mbetet neutral, kompiluesi gjeneron files neutral ne format te Java code (bytecode), qe mund te përdoret ne procesorë te ndryshëm me kusht qe te ketë Java runtime.

• Duhet te këtë performance te larte.
Java ka performance te larte për shkak te JIT - Just In Time compiler. JIT ndihmon ne kompilim te kodit ashtu siç është nevoja.

• Duhet te interpretohet, përkrah multithreading dhe te jete dinamike.
Interpretuesi i Java mund te ekzekutoj Java bytecodes direkt ne makine. Pasi qe përkrah multithreading, ajo gjend përdorim ne programe ku ka nevojë për performance te larte. Një multi-threaded program përmban pjese qe mund te ekzekutohen njëkohësisht dhe çdo pjese mund te merret me një detyre te caktuar ne te njëjtën kohe duke pasur përdorim optimal te resurseve. Gjersa Java është strikt ne lidhje me kompilim, ne faze te linking, Java është tejet dinamike. Klasat linkohen ashtu siç është nevoja, qofte edhe neper rrjete. Ne rast te HotJava Browser dhe aplikacione te ngjashme, kodi ekzekutueshëm mund te ekzekutohet nga kudo, qe lejon për ndryshime ne aplikacion qe nuk vërehet nga përdoruesi. Rezultati i kësaj janë web shërbime qe janë gjithnjë ne zhvillim; mund te mbesin inovativ dhe te reja, te marrin me shume konsumatorë. [12] Vjen ne versione te ndryshme, por versioni kryesor mund te përdoret pa pagese, përpos ne mjedis komercial.

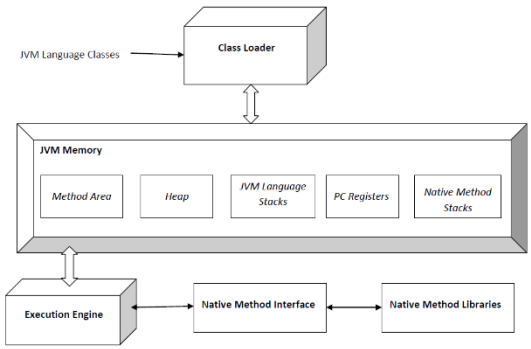


Figura 8. Java Virtual Machine

Java përdor javac compiler, qe kthen kodin nga Java ne Java bytecode. Java bytecode është ne relacion me Java files, si assembly code është me C files.

2.3.3. Kotlin

Kotlin është deklaruar si gjuha zyrtare për zhvillim ne Android ne Maj 2018, por është gjuha e trete e shtuar si 'e përkrahur' (e dyta ishte C). Është gjithashtu gjuha e preferuar nga Google për zhvillim. [13] Qëllimi nga fillimi ishte te jete një alternative për Java – është e dizajnuar qe te konkurroj me Java. [14] Është momentalisht e mirëmbajtur dhe zhvilluar nga JetBrains, dhe përkrahet nga Android Studio duke filluar me versionin 3.0. Gjithashtu është prezent ne disa tjera IDE te përdorura për zhvillim te aplikacioneve ne Android, si Eclipse dhe IntelliJ IDEA. Përdor Java Virtual Machine për ekzekutim, si vet Java, dhe si e tille, duhet te konvertohet ne Java bytecode. Ndryshe nga Java, qe përdor javac compiler, Kotlin përdor kotlinc compiler, por sidoqoftë rezultati përfundimtar është Java bytecode.

Evolucioni i Kotlin filloj ne Rusia ne 2011. Zhvilluesi i saj, JetBrains kërkonte një zëvendësim për Java për shkak te disa limitimeve qe hasnin ne Java. Për arsye te qarta, ata ishin te orientuar nga një gjuhë e bazuar ne Java. Prej gjuhëve te ndryshme qe përdorin Java Virtual Machine (JVM), konsideruan Scala si me te përshtatshëm për nevojat e tyre. Por sapo filluan përdorimin e Scala, vërtetuan se ishte e ngadalshme, si dhe kishte mungesë të një IDE të mirë. Për këtë arsye, JetBrains vendosi te krijoj gjuhen e tyre kompatibile me JVM. [15]

2.3.4. Studim Krahasues

2.3.4.1. Zgjerimi

Nëse kërkohet të shtoni disa veçori shtesë në një klasë, në shumicën e gjuhëve programuese, rrjedh një klasë e re. Një funksion shtesë është një funksion anëtar i një klase që përcaktohet jashtë klasës. Funksioni i shtrirjes mund të shtjellohet me shembullin e dhënë më poshtë. Si për shembull, ne kemi nevojë për një funksion i tipit String qe duhet të kthejë një String të ri me karakterin e parë dhe të fundit të hequr; kjo metodë nuk është e disponueshme në klasën String. Funksioni i shtrirjes i deklaruar jashtë klasës krijon një funksionalitet të klasës së specifikuar që shtrin funksionet e paracaktuara. Kjo behet si vijon:

```
fun String.removeFirstLastChar(): String = this.substring(1, this.length - 1)
fun main(args: Array<String>) {
    val myString = "Përshëndetje"
    println("Karakteri i pare është: $ myString.removeFirstLastChar()")
}
```

Kotlin siguron mundësinë për të zgjeruar një klasë me funksionalitet të ri pa pasur nevojë të trashëgoni nga klasa ose të përdorni ndonjë lloj modeli të projektimit siç është Dekoruesi. Kjo bëhet përmes deklaratave speciale të quajtura zgjatje. Kotlin mbështet funksionet shtesë dhe vetitë e zgjerimit. [16] Zgjidhja e Java për këtë është krijimi i mbështjellësve.

2.3.4.2. Kontrollimi i Perjashtimeve

Java përdorë try ... catch blloqe për të trajtuar përjashtime të kohës së funksionimit. Kryesisht përdor përjashtime të kontrolluara. Një përjashtim i kontrolluar është një lloj përjashtimi që duhet të kapet ose deklarohet në metodën në të cilën hidhet. Më poshtë është sintaksa e përdorur në Java:

```
try{
    // disa kod
}catch (e: SomeException){
    // handler
}finally{
    // bllok opsional
}
```

Mund të ketë zero ose më shumë blloqe catch dhe një ose asnjë bllok finally. Në Java, nëse ndonjë kod brenda një metode hedh një përjashtim të kontrolluar, atëherë metoda duhet të trajtojë përjashtimin ose duhet të specifikojë përjashtimin.

Kotlin nuk ka kontroll të përjashtuar. Të gjitha klasat e përjashtimeve në Kotlin janë pasardhës të klasës Throwable. [17] Në Kotlin, "hedhja" duhet të përdoret për të shkaktuar përjashtime.

```
fun fail(message: String): Nothing {
    throw IllegalArgumentException(message)
}
val s = person.name ?: throw IllegalArgumentException("Nuk ka emër")
```

Përjashtimet e kontrolluara mund të thyejnë logjikën ose rrjedhën e kodit. Veçanërisht në kode me shumë metoda kthyese, përdorimi i përjashtimeve të kontrolluara mund të krijoj rrjedhë të humbur të kodit. Dhe në rast të softuerit të madh, përjashtimet e kontrolluara çojnë në më pak produktivitet, dhe ne rastin me te mire, rritje minimale të cilësisë së kodit. [18]

2.3.4.3. Siguria ne Zero (Null Safety)

Një nga pengesat më të zakonshme në shumë gjuhë programimi, përfshirë Java, është që qasja te një anëtar i një referencë të pavlefshme do të rezultojë në një përjashtim të referencës së pavlefshme. Në Java kjo do të ishte ekuivalenti i një NullPointerException ose NPE për shkurt, gjithashtu i referuar si "gabimi miliardë dollarësh" nga personi me më influencë në zhvillim të Java. Kotlin përdor funksionin e quajtur Null Safety për të trajtuar situatën e treguesit NULL. Përveç nëse kërkohet në mënyrë të qartë, Kotlin nuk hedh një NullPointerException. Më poshtë është disa kod në Java që synojnë të hedhin një NullPointerException:

```
public static void main(String args[]) {
    String name= null;
    System.out.println(name.length());
}
```

Rezultati: NullPointerException.
Më poshtë është kodi ekuivalent ne Kotlin:

```
fun main(args: Array<String>){
    var name: String? = null println(name?.length)
}
```

Rezultati: null.

Për shkak të kësaj, në Java, një NullPointerException prish rrjedhën e aplikacionit, por nuk prish rrjedhën e aplikacionit në Kotlin - ai thjesht jep rezultatin si "null". Por, siç tregohet në kod, për të lejuar null, duhet të shpallim një ndryshore si të pavlefshme, duke përdorur "?". Për disa raste, ekziston një operator tjetër, '!! ', i cili mund të përdoret kur përjashtime të tregu esit të ngushtë duhet të raportohen. Kjo do të simulonte një NPE, sa herë që ndryshorja ka null si vlerën e saj dhe përdoret.

2.3.4.4. Kuptueshmeria

Kotlin preferohet për disa arsye për përdorim në zhvillim, por një nga më të mëdhenjtë është kuptueshmëria e kodit. Shembujt janë të panumërta ku një sasi e madhe e linjave të kodit në Java mund të bëhet me disa në Kotlin. Më poshtë është një shembull i tillë - një klasë që përdoret zakonisht në mësimet Java:

```
public class Person{
    private String name;
    public Person(String name){
        this.name=name;
    }
    public String getName(){
        return name;
    }
```

```
    public void setName(String name){
        this.name=name;
    }
}
```

Tani, le te shohim ekuivalentin e kësaj klase ne Kotlin:

```
    data class Person(val name: String)
```

Ky është vetëm një shembull shumë i thjeshtë se si disa rreshta të kodit të Java, kanë nevojë vetëm për një linjë kodi në Kotlin. Dallimi në gjatësinë e kodit nuk është zakonisht kaq ekstrem por mund të jetë afër tij.

2.3.4.5. Ngarkim i dembeluar (Lazy Loading)

Ngarkimi dembel përdoret në programet kompjuterike për të shtyrë fillimin e një objekti deri në pikën që duhet. Kështu, tipari i ngarkimit me dembelizëm zvogëlon kohën e ngarkimit. Kotlin na e jep këtë veçori, ndryshe nga Java. Në rast të Java, nuk ka ndonjë karakteristikë të tillë si ngarkimi dembel, kështu që një pjesë e madhe e përmbajtjes që nuk kërkohet ngarkohet gjatë fillimit të aplikacionit duke ngarkuar në përgjithësi aplikacionin më të ngadaltë.

2.3.4.6. Performanca

Një nga mënyrat e pakta për të analizuar performancën e një gjuhe programuese ndaj një tjetri, është duke u krahasuar me kriteret. Patrik Schwermer bëri një studim me këtë objektiv të saktë, duke përdorur teste standarde të gjuhës kompjuterike. Do të marrim aspektin e performancës së rahasimit nga puna e tij. [19] Ai drejtoi disa kritere, si Fasta benchmark, Fannkuch-Redux benchmark, N-body benchmark dhe Reverse-complement benchmark. Rezultatet e tyre janë paraqitur ne figurat me larte.

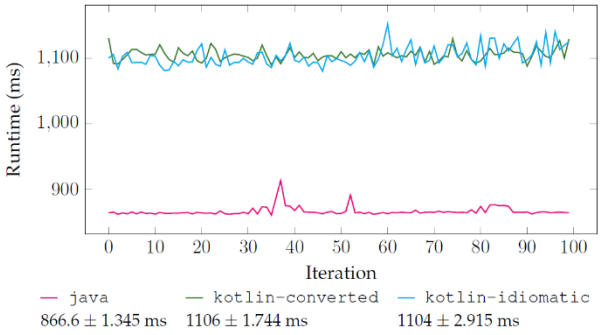


Figura 9. Fasta benchmark

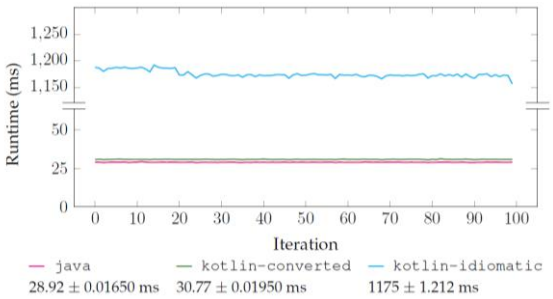


Figura 10. Fannkuch-Redux benchmark

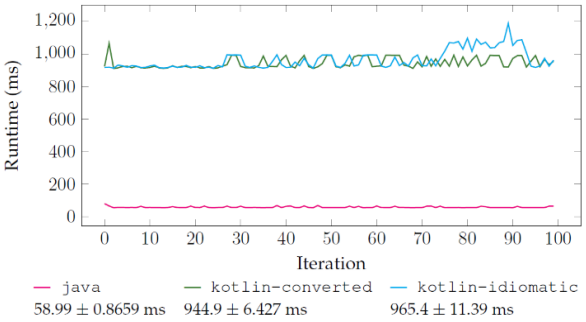


Figura 11. N-body benchmark

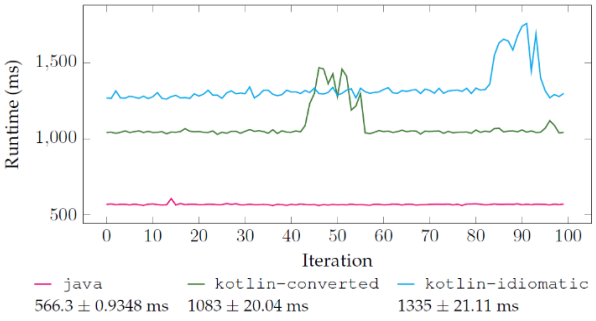


Figura 12. Reverse-complement benchmark

Deri më tani, të gjitha pikat që kemi bërë, kanë qenë në favor të Kotlin, sesa të Java. Sidoqoftë, kur bëhet fjalë për performancën, është e qartë që Kotlin mbetet pas Java. Përderisa ndryshimi nuk do të ishte me të vërtetë i dukshëm në aplikimet e botës reale, ai definitivisht ekziston. Arsyeja pse është atje, Patrik sugjeron është fakti që Java ka qenë gjuha zyrtare e Android shumë më gjatë dhe si e tillë, edhe vetë sistemi operativ është i optimizuar për Java. Në studimin e tij, ai gjithashtu vuri re që grumbullimi

i mbeturinave me Kotlin është shumë më pak efikas, gjë që mund të ketë ndikuar në rezultatet. Shtojeni faktin që shumë më tepër kohë është investuar në zhvillimin e vetë Java dhe që Kotlin kodi shndërrohet në Java, përpara se të shndërroheni në bytecode, ndryshe nga Java files, të cilat mund të përpilohen drejtpërdrejt ne Java bytecode, dhe shpjegimi për rezultatet bëhet më i qartë. Sipas gjasave ka shumë më shumë faktorë që ndikuan në rezultatin. Por pavarësisht, rezultatet janë ende unanime - Kotlin është më i ngadaltë.

2.3.4.7. Komuniteti

Kjo është fusha ku Java është më supreme se çdo gjuhë tjetër. Siç u tha më herët, ajo është gjuha e programimit më e popullarizuar - për shkak të kësaj, sa herë që mund të hasim një problem, ne mund të bëjmë një pyetje dhe gjasat janë tejet te larta të marrim përgjigje të shpejta. Nga ana tjetër, ndërsa komuniteti i Kotlin po rritet me një ritëm të shpejtë, nuk është askund afër asaj të Java - nuk është as realiste që ajo të arrihet ndonjëherë në të njëjtin nivel. Për shkak të kësaj, mund të jetë një vendim i vështirë për fillestarët se cila gjuhë do të fillojë të mësojë - Kotlin ka sintaksë më të thjeshtë dhe ka nevojë për më pak kod, por ka edhe më pak mbështetje të komunitetit.

2.3.4.8. Dallime tjera:

Kotlin ka disa veçori qe nuk i ka Java:

- raw types,
- type inference,
- proper function types,
- smart casts,
- primary and secondary constructors,
- range expressions,
- data classes,
- companion objects,
- co-routines,
- etc.

Në anën tjetër, Java ka:

- primitive types,
- static members,
- non-private fields,
- wildcard types,
- checked exceptions,
- etc. [20]

2.3.4.9. Perfundimi i Studimit Krahasues

Analizuam avantazhet dhe disavantazhet e Java dhe Kotlin... Java është një gjuhë shumë e njohur që përdoret gjerësisht në mesin e zhvilluesve. Zhvillimi i Android është vetëm një rast ku Java përdoret. Kështu, duke qenë fillestar, njohja e Java është më e dobishme sesa Kotlin pasi që zgjeron spektrin e mundësive.

Së dyti, ekziston një komunitet i madh i programuesve Java, që do të thotë se gjejmë përgjigje për çështje kritike të programimit kur jemi ngecur. Kjo është shumë e rëndësishme sepse, si fillestar, përballja me problemet teknike është një skenar i zakonshëm dhe ne mund të mos dimë se ku të drejtohemi kur kemi ngecje. Kur kërkojmë për problemet Java, gjasat janë tejet te larta qe të marrim përgjigje; nuk mund të thuhet e njëjta gjë për Kotlin, e cila është ende një gjuhë programimi e ardhshme. Ka edhe më shumë mësime, libra dhe kurse, falas dhe me pagesë, të cilat mund të na mësojnë zhvillimin e Android me Java, ndërsa ato janë të pakta për Kotlin.

Duke menduar nga këndvështrim te zhvilluesit, Kotlin do preferohej për arsyet e mëposhtme:

- Gjuha dhe mjedisi janë të "pjekur". Lëshimi i Kotlin ka kaluar nëpër shumë faza para versionit 1.0 ndryshe nga gjuhët e tjera të programimit.

- E bën zhvillimin e Android shumë më të lehtë. Kotlin e bën programimin më të lehtë dhe aplikacionet Android më të mira. Kotlin është një gjuhë moderne e programimit. Ajo hap dritaren për një numër të madh të mundësive për zhvilluesit e aplikacioneve Android, d.m.th. ai i bën zhvilluesit më produktivë.

- Kotlin ndihmon në zvogëlimin e gabimeve dhe pasi qe përkrah dështim sa më shpejt që të jetë e mundur, kjo lehtëson kërkimin e gabimeve. Për të shmangur gabimet e ekzekutimit dhe për të zvogëluar koston, dhe përpjekjen e nevojshme për rregullimin e gabimeve, përpiluesi Kotlin kryen shumë kontrolle.

- Kodi i Kotlinit është më i sigurt. Gabimet e zakonshme të programimit në dizajn mund të parandalohen lehtësisht duke përdorur Kotlin, duke rezultuar në më pak dështime të sistemit dhe prishje të aplikacioneve. Kjo vërteton se kodi Kotlin është në thelb më i sigurt se Java.

- Kotlin është shumë më i qarte se çdo gjuhë tjetër e programimit në shumë raste; na lejon të zgjidhim të njëjtat probleme me më pak linja kodesh. Kjo përmirëson mirëmbajtjen dhe lexueshmërinë e kodit, që do të thotë që inxhinierët mund të shkruajnë, lexojnë dhe ndryshojnë kodin në mënyrë më efektive dhe efikase.

- Kotlin është plotësisht i pajtueshëm me Java. Do kod që është shkruar në Java, funksionon po aq mirë në Kotlin. Zhvilluesit mund të zgjedhin të mbajnë kodin që përdorin në Java, ose t'i lënë IDE të kryejë përkthim automatik nga Java në Kotlin. Kjo ka për qëllim të ndihmojë në lehtësimin e migrimit. Android Studio jep mundësinë për ta kthyer të gjithë projektin në Kotlin me vetëm disa klikime. Për shkak të kësaj, përdorimi i të dyve (Java dhe Kotlin), në të njëjtën kohë është gjithashtu një mundësi.

- Tashmë është në përdorim nga Amazon Web Services, Pinterest, Coursera, Netflix, Uber, Square, Trello, Basecamp. Gjithashtu perdoret nga Corda - Corda është kompania përgjegjëse për aplikimet bankare, për banka si Goldman Sachs, Wells Fargo, JP Morgan, Deutsche Bank, UBS, HSBC, BNP Paribas (kompani pronare e TEB), Société Générale, etj. Kjo është një testament i mjaftueshëm që në përgjithësi Kotlin pranohet si opsioni më i sigurt midis të dyve.

Sidoqoftë, nëse po flasim për njerëz që sapo fillojnë në botën e programimit, të cilët po përpiqen të marrin një vendim midis Java dhe Kotlin, ose zhvilluesve, aplikacionet e të cilëve duhet të jenë sa më shpejtë që mund të jenë, ne do të duhet të rekomandojnë Java për shkak të disponueshmërisë në dokumentacion dhe rezultateve që janë nxjerrë nga standardet përkatëse. Por në çdo rast tjetër, rekomandohet Kotlin. Përfitimet e saj tejkalojnë të metat.

Per arsyet e cekura me larte – lehtësimin qe Kotlin jep per zhvilluesin, thjeshtësinë dhe kohen qe kursen, eshte përdorur si gjuha kryesore per aplikacionin. Ketu fjala kyqe eshte ‘gjuha kryesore’, pasi qe ne aplikacion përmban edhe kod ne gjuhën Java, pasi qe ka interoperabilitet mes te dyjave.

3. Rasti studimi: Menaxhimi i bashkëudhëtimit me taksi

3.1. Hyrje

Ne kuadër te punimit te diplomës, eshte zhvilluar aplikacioni per menaxhim te bashkeudhetimit me taksi ne sistemin operativ Android. Aplikacioni Eja Shkojme ofron platforme te vecante qe tenton te i plotësoj nevojat e shfaqura rreth qeshtjes te bashkeudhetimit. Aplikacioni përmban tri module: modulin e forumit, modulin e bisedave dhe modulin e udhëtimit. Moduli i forumit eshte thjeshte nje hapesire ne te cilën mund te krijohen postime ne lidhje me udhëtime; moduli i bisedave eshte nje hapesire ku mundësohet komunikimi mes përdorueseve te aplikacionit përmes tekstit; moduli i udhëtimit eshte nje version i thjeshtësuar i navigimit qe ofrojnë aplikacionet si Google Maps.

3.2. Veglat dhe libraritë e përdorura

3.2.1. Android Studio

Android Studio eshte IDE me e përdorur per zhvillim te aplikacioneve ne Android, mirëpo jo e vetmja, pasi qe ka edhe IDE tjera si Eclipse, IntelliJ IDEA, Xamarin, etj. Eshte e krijuar nga JetBrains, ne bashkëpunim me Google, e cila edhe e rekomandon. Android Studio përkrah gjuhet Java, Kotlin dhe C++, mirëpo gjithashtu përkrah edhe files qe perdoren per aplikacione si XML files, sbashku e pjesën e dizajnimit. Ekzistojne versione per sistemet operative Windows, MacOS, Linux, si dhe Chrome OS. Ne te njëjtën kohe, ka përshtatje te integruar per Gradle qe po thuajse qdo aplikacion ne Android e përdor.

Qka e veqonte ne fillim Android Studio, ishte integrimi i dizajnimit – kishte drag and drop features per krijim te dizajnit, mirëpo gjithashtu lejonte ndryshim te dizajnit përmes ndryshim te files te asociuara XML. Pra përmban editor te kodit jo vetëm per Java, Kotlin & C++, por edhe per XML. Editori i kodit ne fakt eshte nder përparësitë qe Android Studio ka kundrejt IDE-ve tjera per zhvillim te aplikacioneve ne Android pasi qe eshte optimizuar per te. Editori i kodit ofron edhe veqori te refaktorimit per pastrim dhe riformatim te kodit, veçori kjo qe mund te perdoret per te pare me lehte dhe thjeshtësuar rrjedhjen e kodit.

Gjithashtu, menaxhon instalimin, mirëmbajtjen dhe kontrollimin e sistemeve te simuluara përmes Android Emulator. Android Emulator eshte lansim i sistemit operativ Android, qe reagon ne te njëjtën menyre qe do reagonte nje telefon real. [21]

3.2.1.1. AndroidX

Pergjate procesit te zhvillimit te aplikacioneve ne Android, disa librari qe ishin ne përdorim, janë mbetur ende ne përdorim deri ne nje mase. Mirepo, përgjatë viteve dhe versioneve, janë shfaqur konflikte mes versioneve te librarive qe ishin ne versionet e vjetra, dhe versionet qe ishin ne versionet e reja te Android. Per t'e zgjidhur kete, eshte publikuar AndroidX. Ne te njëjtën kohe, krijon konsistence per emërime:

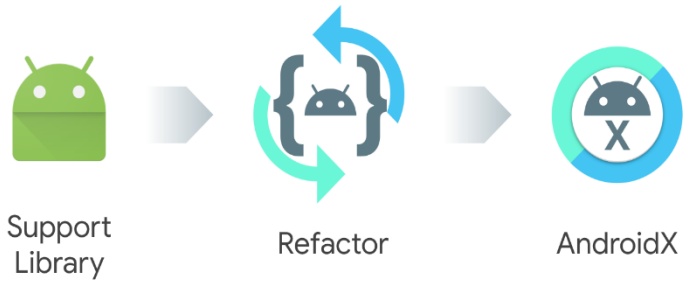


Figura 13. Konvertimi i Librarive

- Emërimet ne Support Libraries ishin shembull: android.support.v7.widget.RecyclerView, android.support.v7.widget.GridLayoutManager, android.support.v7.widget.LinearLayoutManager, com.android.support.constraint:constraint-layout.
- Emerimet ne AndroidX janë: androidx.recyclerview.widget.RecyclerView, androidx.recyclerview.widget.GridLayoutManager, androidx.recyclerview.widget.LinearLayoutManager, androidx.constraintlayout:constraintlayout.

Ne anën tjetër, zgjedhi edhe problemet mes Support Library v4 dhe Support Library v7, duke krijuar nje librari te unifikuar. Mirepo, nuk eshte pa te metat e veta – nuk mund te perdoren Support Libraries dhe AndroidX libraritë ne te njeten kohe, qe mund te pengoj me implementim te aplikacioneve. Kjo krijoj pak problem gjate krijimit te aplikacionit, pasi qe shumica e këshillave qe janë ne internet, janë

duke u referencuar ne Support Libraries, dhe si tille nuk janë kompatibil me implementim te AndroidX. [22]

3.2.2. Google Play Services

Pasi qe punimi eshte ne lidhje me udhëtime, nënkupton qe do perdoren metoda per përcaktim te lokacioneve, gjetje te lokacionit, shfaqje te lokacionit ne harte. Standard per tere këto, janë shërbimet e Google qe i ofron ne lidhje me lokacione per shkak te përdorimit tejet te larte te tyre. [23] Perdorimi i tyre behet përmes API qe Google jep qasje ne to. Ne kete punim, janë përdorur tri API:

1. Maps SDK for Android
2. Places API
3. Directions API

Per te pasur qasje ne to, duhet te regjistrohen per projekt dhe te gjenerohet nje API key, i cili pastaj perdoret ne aplikacion per identifikim te aplikacionit kur kërkohen shërbimet e lartecekura. Per nga ana e aplikacionit, ne Gradle duhen te përfshihen qe te trija libraritë relevante per këto API:

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
implementation 'com.google.android.gms:play-services-places:17.0.0' // places+directions
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Libraria e pare, na mundëson te shfrytëzojmë shërbimet e lokacionit qe i ka Google, qe janë me eficiente se te sistemit. Places API na mundëson gjetjen e lokacioneve prej emrit dhe kthimin e emrit, si Prishtine, ne format te kuptueshëm per Maps SDK, qe pastaj te vendoset ne harte ne piken e duhur. Directions API na mundëson marrjen e shtegut qe duhet kaluar prej pikënisjes deri ne destinacion – këto i marrim ne forme te JSON, te cilat pastaj mund t'i manipulojmë per nevojat tona. Maps SDK perdoret ne përgjithësi per qasje ne harta te Google.

3.2.3. Firebase

Firebase eshte nje grumbull shërbimesh qe ofrohen per zhvillim te aplikacioneve. Eshte platforme nen pronësi te Google, dhe e njohur si platforma me e përdorur per implementim te databazes ne aplikacione te sistemit operativ Android, duke u përdorur ne qindra mijëra aplikacione dhe pothuajse te gjitha aplikacionet e reja. [24] Edhe pse eshte e menduar veqanerisht per implementim ne Android, ekzistojnë metoda qe te perdoret edhe per aplikacione desktop. Nder shërbimet qe ofron janë databaza, analizime, raportime per dështime ne aplikacion, autentifikim, hapesire online, shërbime per mesazhe, etj. Nga këto,

3.2.3.1. Autentifikim

Aplikacioni i krijuar përdor autentifikim, dhe si furnizues te këtij shërbimi e përdor Firebase. Autentifikimi i lejuar per aplikacion eshte i zgjedhur vetëm per email dhe fjalëkalim, edhe pse Firebase ka edhe mënyra tjera per autentifikim si përmes numrit te telefonit, Facebook profilit, Google profilit, përdorim anonim, etj.

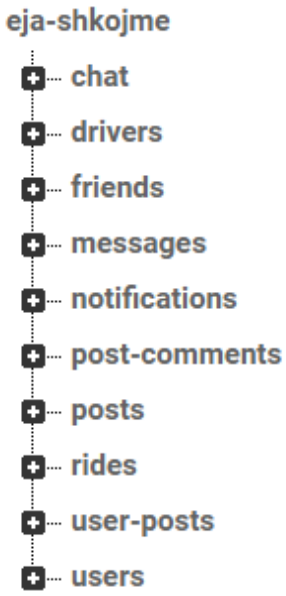


Figura 14. Pamje high-level e databazes per aplikacionin

3.2.3.2. Databaza

Databaza e përdorur per kete aplikacion përdor strukturën e paraqitur ne Figuren 14. Qdo gje qe ndodh ne aplikacion, ka te beje me databazen – qdo mesazh qe dërgohet, qdo miqësi qe krijohet, qdo postim, koment, person qe regjistrohet, qdo udhëtim qe planifikohet. Kjo mund te bie ne pyetje qeshtjen e privatësisë, qe ne fakt edhe paraqet problem. Ne kushte komerciale, do duhej qe te aplikohej ndonjë enkriptim i mesazheve, postimeve, komenteve, e te gjitha te dhënave qe te mos kete qasje ne to ndokush i paautorizuar, por duke marre parasysh natyrën e aplikacionit, kam vendosur qe te mos aplikohet nje gje e tille dhe tere informatat te ruhen ne plaintext.

3.2.3.3. Hapesira (Storage)

Firebase ofron mundësi per ruajtje te dhënave te ndryshme ne nje kontejner qe e quan 'bucket'. Ne rast tonin, bucket e Firebase eshte përdorur qe te ruajmë fotot e profilit te përdorueseve ne to. Mund te perdoret per qfaredo arsye tjetër, mirëpo nuk kam pare nevoj qe te perdoret per ruajtje te ndonjë gje tjetër.

3.2.3.4. Messaging

Sherbim i radhës te Firebase qe eshte përdorur, eshte Messaging, apo dërgimi dhe pranimi i mesazheve përmes Firebase. Kjo eliminon nevojën per server te veqante per komunikime, apo anashkalon mbështetje ne shërbime qe nuk janë nen kontroll tone direkte.

3.2.3.5. Core

Core eshte nje librari mbështetëse e Firebase. Gjersa ajo nuk eshte përdorur drejtperdrejte, eshte e nevojshme qe disa komponentë tjera te përdorura te Firebase te funksionojnë ne rregull.

3.2.3.6. GeoFire

GeoFire eshte shërbim relativisht i ri i Firebase qe ende nuk figuron ne faqen e tyre kryesore. Permes GeoFire, lokacionet ne vend qe te ruhen sipas koordinatave te tyre, ruhen ne databazen tone përmes nje identifikuesi te veqante ne forme te nje stringu. Shembull: lokacioni 'Podujeva', ne vend qe te ruhet me koordinata 42.9108 N dhe 21.1956 E, ruhet si 'keSNFTivsnUPIAIoR0YL2zjKYi63'. Kjo na mundëson manipulim me te lehte me lokacione se sa qe do mundësohej përmes koordinatave pasi qe me nuk kemi me dy variabla te tipeve double, por vetëm nje variable string qe tregon saktësisht vendin. Pra, na furnizon me eficience te njejte sikur koordinatat, mirëpo na lehtëson procesin e manipulimit me te dhëna. [25]

3.2.4. Librarite e krijueseve tjerë

Nisur nga parimi qe na eshte mësuar gjate studimeve, 'nuk keni nevoj me zbulu rrotën, mjafton te dini si te perdoret', e kam pare si te arsyeshme qe te perdoren librari te krijueseve tjerë qe te lehtësohet krijimi dhe funksionalizimi i plote i aplikacionit. Permes përdorimit te tyre, koha eshte kursyer dhe pasi qe janë librari tejet shume te përdorura dhe te testuara, mund te jemi me te sigurte se sa nëse do i implementonim vete vetitë qe i kane këto librari.

3.2.4.1. Picasso

implementation 'com.squareup.picasso:picasso:2.71828 '

Nje nder libraritë e para te futura ne përdorim eshte Picasso nga SquareUp. Kjo librari mundëson manipulim me te lehte te fotografive brenda aplikacionit. Lehteson punën pasi qe mund te perdoret ne kombinim me Recycler dhe Adapters, dhe ne te njëjtën kohe eshte ne gjendje te marr fotografi nga interneti, te i transformoj fotot ne baze te nevojës, te i vendos ne cache, dhe tere kete me shfrytëzim minimal te resurseve te memories. Kjo gjene përdorim ne shfaqjen e fotove te profilit te qdo përdoruesi. [26]

3.2.4.2. Material DateTime Picker

implementation 'com.wdullaer:materialdatetimepicker:4.2.3'

Percaktimi i datës dhe kohës per fillim te nisjes te udhëtimit behet përmes Material DateTime Picker. Kjo librari na lehtëson kete pune, dhe ne te njëjtën kohe ka nje dizajn tërheqës qe përdor edhe sistemi operativ Android. [27]

3.2.4.3. Image Cropper

implementation 'com.theartofdev.edmodo:android-image-cropper:2.7.0'

Kur përdoruesit ngarkojnë foto per profil, shpesh eshte nevoja qe ato te prehen ne menyre qe te pershtaten me standardin qe e kemi aplikuar – duhen te jene katror ne menyre qe te mos shkaktojnë probleme. Kete e arrijmë përmes Image Cropper, qe detyron përdoruesin foton e ngarkuar te e prej ne formatin tone te parapërcaktuar. Image Cropper ofron edhe formate tjera, jo vetëm katror (1:1), si 16:9, 4:3, form te lire, etj, mirëpo per nevojat tona, na eshte nevojitur 1:1, prandaj ate e kemi aplikuar. [28]

3.2.4.4. CircleImageView

implementation 'de.hdodenhof:circleimageview:3.0.1'

CircleImageView eshte nder arsyet pse Image Cropper e ka formën te përcaktuar si 1:1. Kjo pasi qe kur te shfaqen fotot, e kemi përcaktuar qe te shfaqen ne forme rrethore, dhe kthimi i nje fotoje qe nuk pershatet me formatin 1:1 ne forme rrethore, doli se eshte problematike. Fotot ne aplikacion ne forum dhe vende tjera, përmes kësaj librarie shfaqen ngjashëm siq shfaqen ne aplikacione si Facebook, Instagram, Twitter, etj, pra ne forme rrethore. [29]

3.2.4.5. Compressor

implementation 'id.zelory:compressor:2.1.0'

Kur përdorues ngarkon nje foto, ajo foto eshte rëndom e shkrepur me kamer dhe si e tille ka nje madhësi jo edhe relativisht te vogël. Po te lejohej përdorimi i fotove te tilla, nëse aplikacioni ngarkohet me shume përdorues, kushdo qe qaset ne forum, telefoni i tyre do duhej te shkarkonte te gjitha fotot e përdorueseve ne kualitetin ashtu siq i kane ngarkuar. Kete duhet anashkaluar, siq e anashkalojnë edhe aplikacionet tjera, përmes kompresimit te fotos. Fotoja e ngarkuar kompresohet dhe ne vend qe te perdoret versioni i ngarkuar, perdoret versioni i kompresuar duke rritur shpejtësinë e aplikacionit. Eficienca e kompresimit ka variacion, mirëpo sipas krijuesit mund te rangoj deri 100 here me pak vend te nxënë. Ne testimet e bëra, ky numër ishte me afër 75, duke kompresuar nje fotografi me 7.5MB ne

vetëm 88kB, por eshte me se mjaftueshem. Kjo përpos qe lehtëson aplikacionin, mund te ruaj edhe buxhetin pasi qe Firebase aplikon kosto varësisht prej sasisë te GB te përdorur [30]

3.2.4.6. OkHttp

implementation 'com.squareup.okhttp3:okhttp:4.2.2'

OkHttp eshte nje projekt per nje HTTP client me eficient, përkrah kërkesa per HTTP 2.0, dhe përkrah dërgim te kërkesave te shumeta përmes te njëjtit socket connection. Eshte perfshire ne aplikacion pasi qe lejon dërgimin e kërkesave me header qe mund ta manipulojmë lirisht, dhe eshte duhur nje i tille. [31]

3.2.4.7. JODA-Time

implementation 'net.danlew:android.joda:2.10.3'

JODA-Time eshte API e krijuar qe te ofroj klasa me te mira ne lidhje me koha dhe data se sa ofronte java.util qe eshte e perfshire, qasje me te lehte dhe funksione me te vyeshme. [32]

3.2.4.8. Retrofit

implementation 'com.squareup.retrofit2:retrofit:2.6.2'
implementation 'com.squareup.retrofit2:converter-scalars:2.6.2'

Retrofit eshte librari qe mundëson trajtimin e nje API te caktuar sikur te ishte nje klase. Ne rastin tone, kjo eshte përdorur qe te kemi qasje me te lehte ne GoogleAPI. [33]

3.3. Struktura e aplikacionit

Aplikacioni përmban nje numër te aktiviteteve, adaptereve, ViewHolders, fragmente, ndihmës (utilities). Lansimi i aplikacionit fillon me aktivitetin per marrje te qasjes (login). Prej atu, mund te kalohet ne nje numër tjetër te aktiviteteve, varësisht prej veprimeve. Veprimi me i zakonshëm do ishte shkrimi i email dhe fjalëkalimit, duke marrur qasje ne aplikacion. Kur te kete nevoje per ndogje ne lidhje me lokacion, apo shfaqje te ha rtave, nëse nuk i eshte dhene leja per qasje ne lokacion, do i kërkohet qasja ne sensoret e GPS.

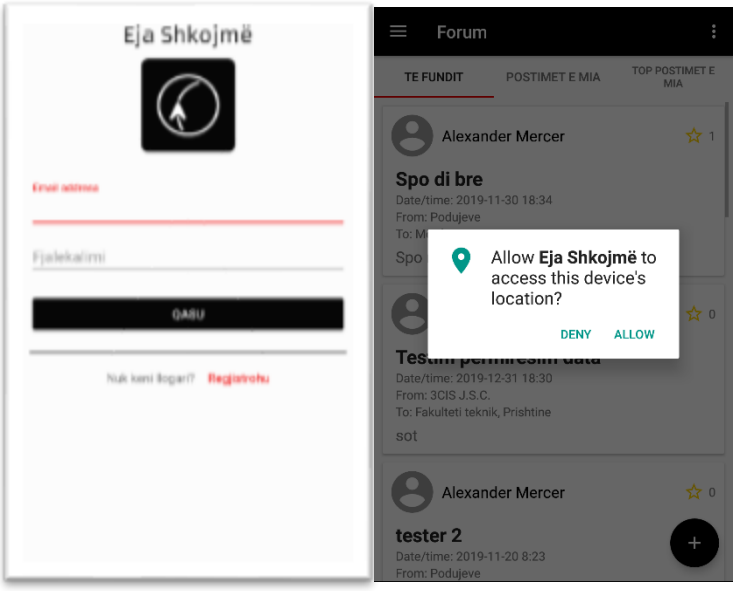


Figura 15. Hapat e pare ne aplikacion

3.4. Forumi

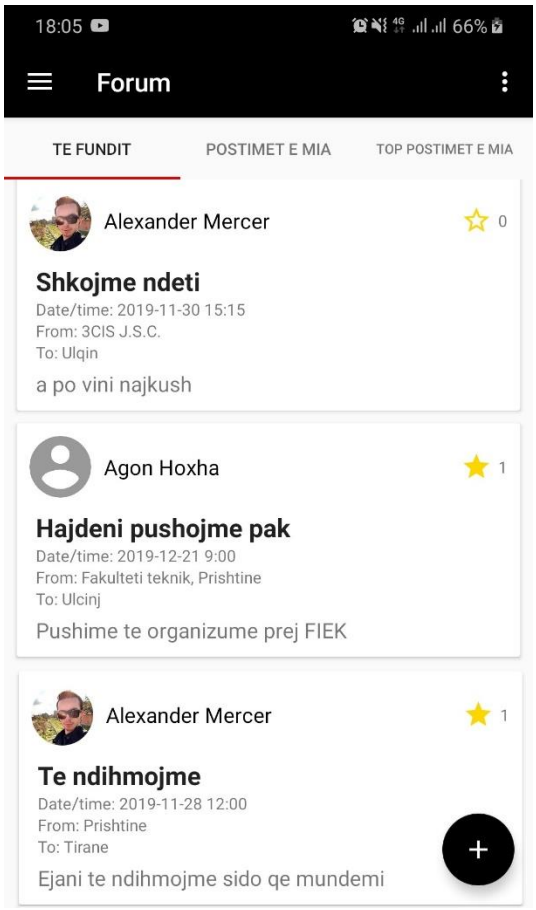


Figura 16. Pamja kryesore

Pjesa kryesore e aplikacionit eshte forumi. Ne forum shihen postimet e kaluara, mund te shihen vetëm postimet e përdoruesit qe eshte logged in, apo te i sheh top postimet, qe llogariten ne baze te likes. Nga forumi, mund edhe te qasemi ne postimet qe janë postuar, nëse dëshirojmë te komentojmë, apo mund edhe te fillojmë te krijojmë postim te ri. Forumi funksionon përmes te klasës ForumFragment – fragment te cilit iu eshte deleguar ngarkimi i postimeve te fundit, postimet e përdoruesit dhe top postimet e përdoruesit, gje qe behet me klasat per postime te fundit, postimet te përdoruesit dhe top postimet e përdoruesit respektivisht. Te gjithë këto klasa dhe fragmente kane varësi nga PostListFragment – nje nder klasat e vetme e shkruar ne Java, qe i merr te dhënat nga Firebase varësisht prej scenarios, dhe i vendost kthen te dhënat e duhura neper RecyclerViews tek fragmentet e lartecekura qe me pastaj e bëjnë shfaqjen. PostListFragment merr qasje ne Firebase përmes , po si edhe klasat tjera qe kane qasje ne Firebase databaze, përmes referencës te instancës te databazes, apo ne kod, kjo shkruhet si vijon:

mDatabase = FirebaseDatabase.getInstance().getReference();

Adapteri qe perdoret per te marre te dhënat, me saktësisht FirebaseRecyclerAdapter, i merr te dhënat nga databaza, krijon pamjen, por edhe përcakton funksionet kur përdoruesi te klikon ne postim apo ne emër te postuesit. Ne rast se klikon ne postim, aksioni delegohet tek MainActivity qe e ka funksionin e definuar per hapje te postimit, kurse nëse klikon ne emër te postuesit, e merr përsipër vet pasi qe vetëm i duhet te shfaq opsionin per te hapur profilin e përdoruesit. Kjo behet me kodin si vijon:

```
viewHolder.itemView.setOnClickListener(v ->
Objects.requireNonNull(mainActivity).onViewPostBtnClicked(postKey));
viewHolder.authorView.setOnClickListener(v -> {
    PopupMenu popup = new PopupMenu(getContext(), viewHolder.authorView);
    popup.inflate(R.menu.menu_user_action);
    popup.setOnMenuItemClickListener(item -> {
        if (item.getItemId() == R.id.profile_action) {
            Intent intent = new Intent(getActivity(), ProfileActivity.class);
            intent.putExtra("user_id", model.uid);
            startActivity(intent);
        }
        return false;
    });
    popup.show();
});
```

Perkthyer ne gjuhe te kuptueshme per njerëz, nëse klikon ne postim, ekzekuto funksionin onViewPostBtnClicked, dhe nëse klikohet ne personin qe e ka postuar, hap opsionin qe te shkohet tek profili i personit.

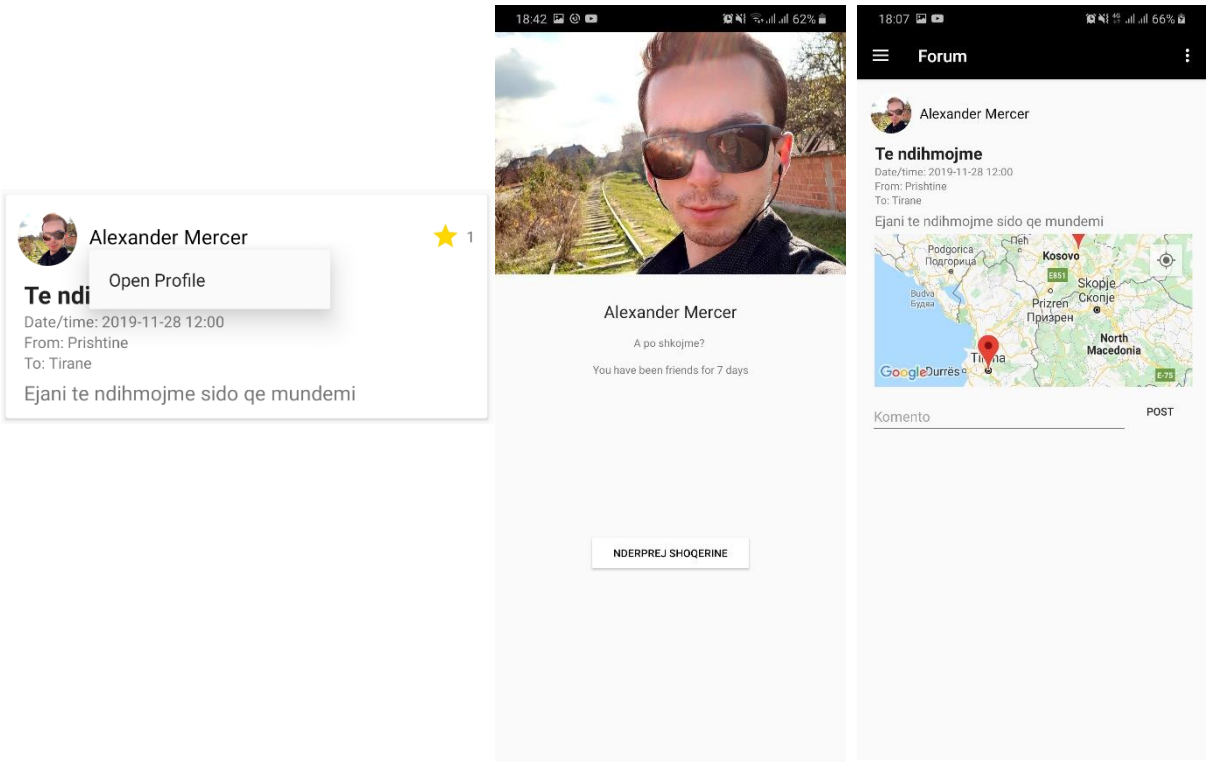


Figura 17. Hapja e profilit, apo postimi me i detajuar

Nga screenshot i mesëm mund te shihet nje buton rreth ndërprerjes te shoqërisë, njëkohësisht ne figurën 14 (ne përmbajtje te databazes) mund te shihet nje tabel 'friends' –ne aplikacionin e krijuar eshte mundësuar krijimi i shoqërisë ne format te ngjajshem sikur qe e kane edhe faqet tjera – përdoruesi mund te dërgoj kerkese per shoqëri, dhe personit qe ja dërgon mund t'e pranoj apo refuzoj. Krijimi i shoqërisë nuk eshte i nevojshëm per komunikim apo ndonjë gje tjetër, mirëpo nëse je miq me personin qe dëshiron te flasësh, e ke me lehte t'e gjesh, siq do shpjegohet tek seksioni i komunikimit. Tani per tani, kthehemi tek forumi.

Krijimi postimeve te reja eshte relativisht thjeshte i bere dhe po ashtu funksionon përmes fragmenteve. Mirepo, ne kete rast, na duhen tri fragmente, dhe rrjedhimisht, tri dizajnë:

1. NewPostFragment me dizajn te aplikuar ne fragment_new_post,
2. SetDateTimeFragment me dizajn te aplikuar ne fragment_setdatetime,
3. SetPickPointFragment me dizajn te aplikuar ne fragment_setpickpoint.

NewPostFragment thjesht jep dy hapësira – nje hapesire per titull, dhe nje tjetër per përmbajtje te postimit, si dhe na jep butonin per next, buton ky qe ne fakt vetëm se i barte te dhënat e mbledhura ne fragmentin e radhës qe eshte fragmenti per përcaktim te datës dhe kohës. Ne forme te njejte, edhe ky fragment ka dy hapësira, mirëpo sapo te preket cilado hapet Material DateTime Picker, qe i jep mundësi përdoruesit te zgjedh datën dhe kohen. Se fundmi, kemi fragmentin per caktim te pikave kyqe, i cili prap ka vetëm dy fusha qe përdoruesi te shkruaj piken prej nga do niset dhe destinacionin qe e ka. Me poshtë eshte paraqitur grafikisht ajo qka u tha deri me tani.

Material DateTime Picker aktivizohet bazuar ne kodin e me poshtëm:

```
mPickDateText!!.setOnClickListener {
    val now = Calendar.getInstance()
    DatePickerDialog(activity!!, DatePickerDialog.OnDateSetListener { _: DatePicker?, year: Int,
```

```
month: Int, dayOfMonth: Int -> Log.d("Orignal", "Got clicked")
            mPickDateText!!.setText("""$year-${month + 1}-$dayOfMonth""")
        },
        now[Calendar.YEAR],
        now[Calendar.MONTH],
        now[Calendar.DAY_OF_MONTH]
    ).show()
}
mPickTimeText!!.setOnClickListener {
    val now = Calendar.getInstance()
    TimePickerDialog(activity,TimePickerDialog.OnTimeSetListener { _: TimePicker?, hour: Int,
minute: Int ->Log.d("Original", "Got clicked")
            val formatedMinute = String.format("%02d", minute)
            mPickTimeText!!.setText("$hour:$formatedMinute")
        },
        now[Calendar.HOUR_OF_DAY],
        now[Calendar.MINUTE],
        true
    ).show()
}
```

Sapo qe përdoruesi te klikoj posto, do validohen se pari pikënisja dhe destinacioni – se ne fakt ekzistojnë dhe mund te shfaqen ne harte, dhe thirret funksioni writeNewPost, i cili merr te dhënat e verifikuara, dhe i vendos ne databaze nen tabelën posts.

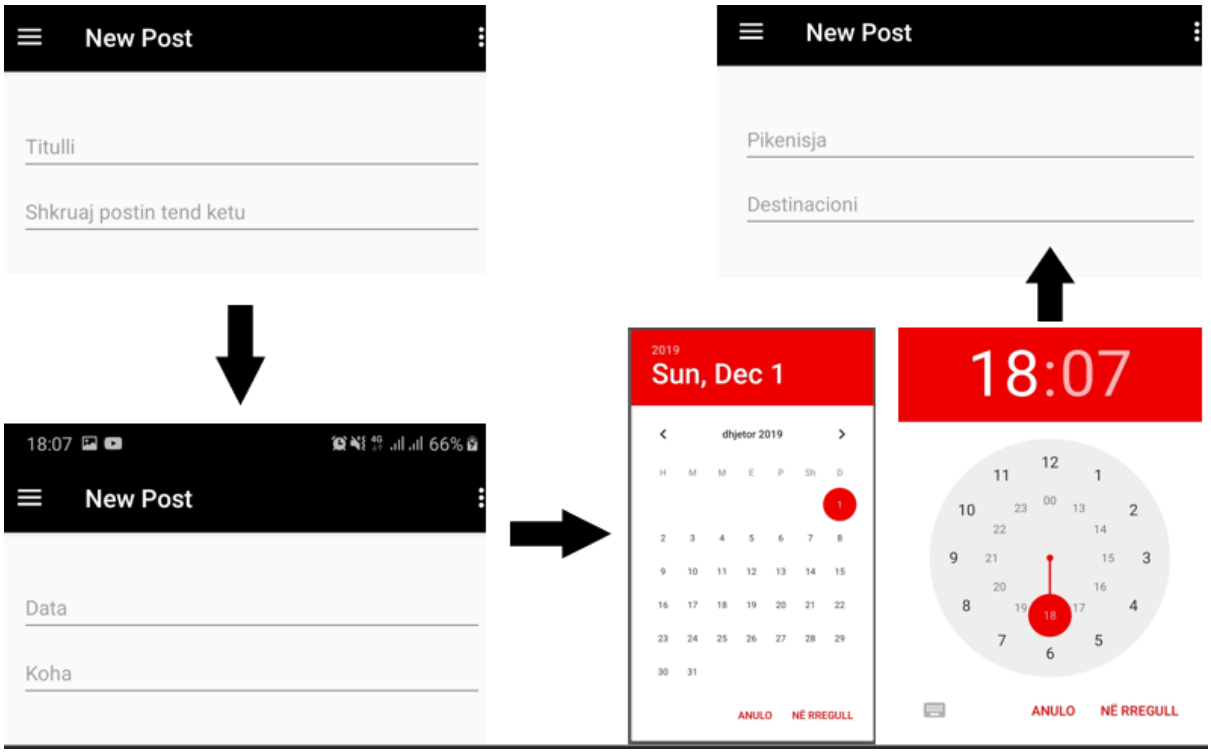
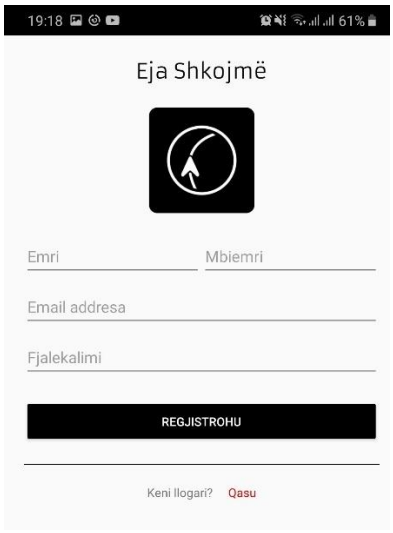


Figura 18. Procesi per krijim te postimit

3.4.1. Regjistrimi

Nese përdoruesi nuk eshte i regjistruar, qasja nuk do jete e mundshme, dhe do duhet te regjistrohet.

Regjistrimi eshte tentuar te behet sa me i thjeshte dhe intuitiv, me ndihmesa per përdoruesin qe i tregone se qfare te dhëna kërkohen nga ai/ajo. Per regjistrim mjafton te jepet emri, mbiemri, email adresa dhe



fjalëkalimi qe do e përdor per qasje. Natyrisht, eshte përdorur validim i emailes qe te sigurohemi se eshte email reale, mirëpo me tej nuk jemi siguruar se kemi te bëjmë me individ real.

Sapo te i mbush te dhënat, dhe te prek butonin per regjistrim, do shtohet ne databaze dhe do krijohet mundësia per te qe te qaset me email dhe fjalëkalim.

```
    mAuth!!.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this@SignupActivity) { task:
Task<AuthResult?> ->
        if (!task.isSuccessful) {
            // ...
        } else {
            // ...
                @Suppress("DEPRECATION") val deviceToken =
FirebaseInstanceId.getInstance().token
            writeNewUser(userId, name, email, deviceToken)
        }
    }
```

Kurse writeNewUser eshte definuar si vijon:

```
private fun writeNewUser(userId: String, name: String, email: String, device_token: String?) {
    val user = User(name, email, device_token)
    mDatabase!!.child("users").child(userId).setValue(user).addOnCompleteListener { task:
Task<Void?> ->
        if (task.isSuccessful) {
            val intent = Intent(this@SignupActivity, MainActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
}
```

Prej procesit te krijimit te përdoruesit u permenden vetëm këto te dyja pasi qe janë pjesa me kryesore.

3.4.2. Profili dhe veprimet me te

Siq u cek me larte, qdo përdorues e ka profilin e vet. Tek profilet kemi disa qeshtje qe duhet cekur, përpos qe e kane mundësinë per krijim shoqërie. Siq mund te vërehet me larte, dhe eshte përmend tek libraritë e përdorura, ekziston edhe mundësia qe përdoruesit te vendosin foton e tyre ne profil.

Nese nuk eshte përcaktuar ndonjë foto paraprakisht, si foto e zakonshme perdoret ajo qe shihet ne foto. Po te preket butoni per ndrrim te fotos, do hapet mundësia per te zgjedhur foton qe dëshiron. Ky veprim delegohet tek vet sistemi operativ pasi qe ka aktivitet te vecant sistemi operativ per importim te nje file te caktuar, mjafton ne t'i tregojmë se qfare formati te te dhënave pranojmë – ne kete rast pranojmë vetëm foto, andaj type do duhet specifikuar si "image/*". Ne kod kjo duket si vijon:

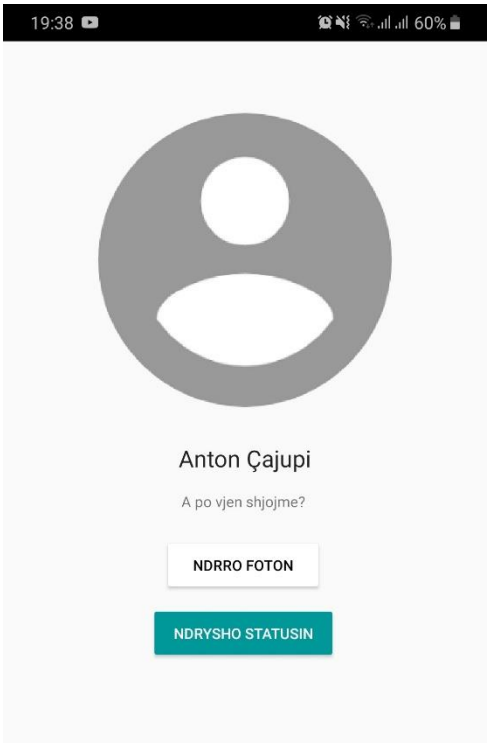


Figura 19. Pamja e profilit nga vete perdoruesi

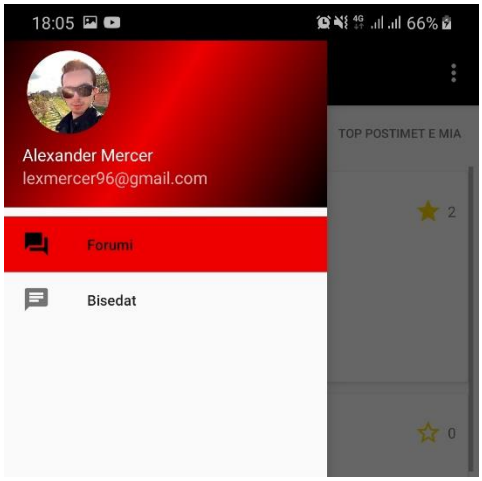


Figura 20. Navigation Drawer

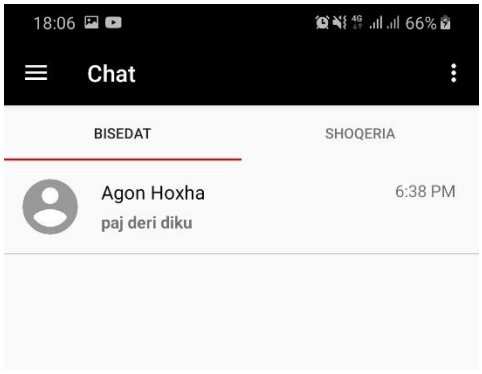


Figura 21. Pamja e Bisedave

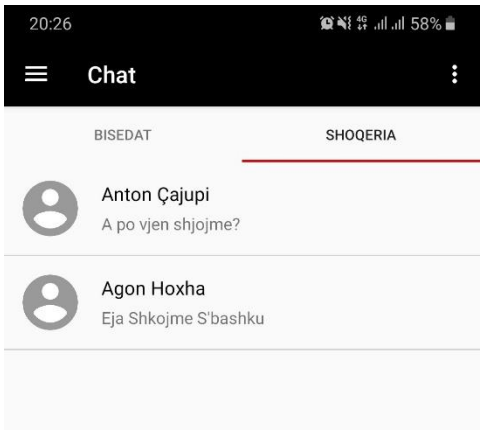


Figura 22. Pamja e Shoqerise

```
mImageButton?.setOnClickListener {
    val intent = Intent()
    intent.type = "image/*"
    intent.action = Intent.ACTION_GET_CONTENT
 startActivityForResult(Intent.createChooser(intent, "Select image"), SELECT_IMAGE)
}
```

Pasi qe te zgjedhet nje foto nga perdoruesi do duhet te prehet ne menyre qe te pershtatet me nevojat tona – pra te jete formatit 1:1. Siq u specifikua tek librarite, kjo behet përmes Image Cropper, me funksionin:

```
CropImage.activity(uri)
        .setAspectRatio(1, 1)
        .setMinCropWindowSize(500, 500)
        .start(this)
```

Pastaj do kompresohet përmes Image Compressor te përcaktuar tek libraritë e përdorura:

```
val thumbBitmap = Compressor(this)
        .setMaxWidth(200)
        .setMaxHeight(200)
        .setQuality(75)
        .compressToBitmap(thumbFilepath)
```

Dhe ngarkohet ne Firebase bucket qe ekziston enkas per aplikacionin tone. Perpos imazheve, profilet karakterizohen edhe me status te tyre. Qdo profil ka statusin e vet, mirëpo ndryshimi i statusit delegohet tek nje aktivitet tjetër – tek ChangeStatusActivity. Eshte aktivitet qe thjeshte ndryshon nje string ne databaze te Firebase, andaj nuk shoh ndonjë arsye t'e përfshij ndonjë pjese nga kodi apo përbërja e këtij aktiviteti.

Siq u tha me herët, përdoruesit kane mundësi te krijojnë shoqëri, mirëpo kane mundësi qe edhe t'e nderprejne ate shoqëri – kjo arrihet përmes butonit qe shihet ne Figuren 17. Ngjajshem me ChangeStatusActivity, edhe kjo eshte vetëm nje ndryshim i vogël ne databaze, andaj nuk e shoh si te arsyeshme te diskutohet me ne thellësi.

3.5. Komunikimi

Nese hapim menynë qe zakonisht i referohet si Navigation Drawer, shohim nje opsion tjetër përpos te Forumit – behet fjale per Bisedat.

Tek Bisedat, do te merremi me tri klasa qe kane rendesi te veqante:

1. ConversationFragment
2. ChatActivity
3. ChatFragment

Ngjajshem si ForumFragment i cili ishte përgjegjës qe te thirre te gjitha klasat e duhura dhe funksionet e duhura per te shfaqur dhe menaxhuar postimet, ConversationFragment merret me qeshtje te ngjajshme, vetëm se tani ka te beje me bisedat qe jene bere, se qka do ndodh kur te preket biseda, dhe per te shfaqur komplet listën e shoqërisë pasi qe shoqëria mund te shihet tek Bisedat. Funksioni kryesor

i këtij fragmenti eshte te beje gati bisedat, lexon nga databaza se me kenë keni biseduar dhe mesazhi i fundit shfaqet ngjajshem me platforma te ndryshme.

Lista e shoqërisë përmban te gjithë profilet me te cilat përdoruesi ka krijuar mardhenie shoqërie. Teksti qe shfaqet nen emër tek Shoqeria, eshte statusi qe e kane përcaktuar.

Nese preket ndokush ne Shoqeria thjesht hapet profili i atij personi dhe mund te shihet para sa ditësh eshte krijuar miqësia, dhe mund te zgjidhet opsioni per dërgim mesazhi apo per ndërprerje te miqësisë. Nese preket ndonjë nga bisedat, kalohet drejt ne bisede. Per nga ana e kodit, thirret ChatActivity, duke shtuar ne Intentin qe perdoret per ate thirrje ID te personit dhe emrin e tij. Pasi te thirret ChatActivity, e marrim pamjen ne Figuren 23.

Parimisht, merret statusi se nëse ka qene online kohe te fundit apo jo, duke kontrolluar parametrin e duhur ne Firebase. Nese po, tregon qe eshte online, nëse jo e paraqet vlerën qe eshte e ruajtur ne po te njëjtën variabël duke treguar se kur perhere te fundit ishte online.

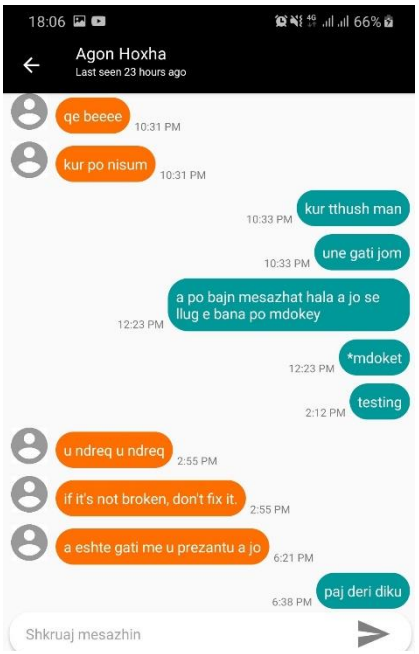


Figura 23. Pamja e Bisedës

Pasi qe te behet kjo, duhen te shkarkohen nga Firebase te gjithë mesazhat qe janë dërguar me pare dhe te shfaqen ato, se bashku me kohen kur janë dërguar. Tere kjo behet thjesht me lexim te te dhënave nga databaza dhe nuk ka ndonjë funksion te veqante, andaj nuk do shtjellohet me ne detaje.

Per te dërguar mesazhe, merret funksioni i posaçëm sentMessage, i cili pasi qe sigurohet qe nuk eshte mesazh i thate, procedon me marrje te qasjes ne direktoriumet e duhura ne databaze. Pasi qe te behet kjo, krijon objekt, si vijon:

```
val messageMap: MutableMap<String, Any?> = HashMap()
messageMap["message"] = message
messageMap["type"] = "text"
messageMap["timestamp"] = ServerValue.TIMESTAMP
messageMap["from"] = uid
messageMap["seen"] = false
```

Objekti messageMap eshte i definuar ashtu qe si index, te kete nje String, dhe si vlere te tij mund te kete qkado, perfshire edhe 'null'. Futja ne databaze behet paksa me ndryshe se sa rastet e tjera:

```
val userMessagePush = mDatabase!!.child("messages")
        .child(uid).child(mChatUser!!).push()
val pushId = userMessagePush.key
mDatabase!!.child("chat").child(uid).child(mChatUser!!).child("seen").setValue(true)
mDatabase!!.child("chat").child(uid).child(mChatUser!!).child("timestamp").setValue(ServerValue.T
IMESTAMP)
mDatabase!!.child("chat").child(mChatUser!!).child(uid).child("seen").setValue(false)
mDatabase!!.child("chat").child(mChatUser!!).child(uid).child("timestamp").setValue(ServerValue.T
IMESTAMP)
mDatabase!!.updateChildren(messageUserMap) { databaseError: DatabaseError?, _:
DatabaseReference? ->
    if (databaseError != null) {Log.d(TAG, databaseError.message)}
}
```

3.6. Zhvillimi i mëtutjeshëm

4. Diskutime dhe konkluzione

5. Shtesat

6. Bibliografia

[1] https://www.businessofapps.com/data/uber-statistics – Statistikat rreth përdorimit te Uber

[2] https://www.businessofapps.com/data/lyft-statistics – Statistikat rreth përdorimit te Lyft

[3] https://www.facebook.com/groups/343190843152164 – Grup ne Facebook per Ridesharing

[4] https://developer.android.com/things – Zhvillim i aplikacioneve Android IoT pajisje

[5] https://developer.android.com/tv – Zhvillimi i aplikacioneve per Android TV

[6] https://www.android-x86.org/ – Version i Android per PC

[7] https://www.cnet.com/news/google-buys-android – Blerja e Android nga Google

[8] https://android.googlesource.com – Publikimi i Android source code

[9] https://arstechnica.com/gadgets/2019/11/google-outlines-plans-for-mainline-linux-kernel-support-in-android/ – Plane per rritje te nderlidhshmerise te Android me Linux

[10] https://www.appbrain.com/stats/libraries/details/kotlin/kotlin – Statistika mbi përdorim te Kotlin

[11] https://www.oracle.com/java – Oracle

[12] https://www.oracle.com/technetwork/java/intro-141325.html – Targetet per zhvillim te Java

[13] https://techcrunch.com/2017/05/17/google-makes-kotlin-a-first-class-language-for-writing-android-apps – Kotlin deklarohet si gjuhe zyrate per zhvillim te aplikacioneve ne Android

[14] https://zeroturnaround.com/rebellabs/jvm-languages-report-extended-interview-with-kotlin-creator-andrey-breslav – Intervist me krijuesin e Kotlin

[15] https://www.netsolutions.com/insights/why-kotlin-is-the-future-of-android-app-development – E ardhmja e zhvillimit ne Android

[16] https://kotlinlang.org/docs/reference/extensions.html – Zgjerimet ne Kotlin

[17] https://code.tutsplus.com/tutorials/kotlin-from-scratch-exception-handling--cms-29820 – Implementimi i përjashtimeve ne Kotlin

[18] http://jonnyzzz.com/blog/2017/02/15/catchall – Perjashtimet e Kontrolluara

[19] Performance Evaluation of Kotlin and Java On Android Runtime, Patrik Schwermer

[20] https://kotlinlang.org/docs/reference/comparison-to-java.html – Dallime mes Java dhe Kotlin

[21] https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html – Lansimi i Android Studio

[22] https://developer.android.com/jetpack/androidx – AndroidX

[23] https://themanifest.com/app-development/popularity-google-maps-trends-navigation-apps-2018 – Statistika mbi përdorimin e Google Maps shërbimeve

[24] https://www.appbrain.com/stats/libraries/details/firebase/firebase – Statistika mbi përdorimin e Firebase ne aplikacione

[25] https://github.com/firebase/geofire-android – GeoFire

[26] https://square.github.io/picasso – SquareUp Picasso

[27] https://github.com/wdullaer/MaterialDateTimePicker – Material DateTime Picker

[28] https://github.com/ArthurHub/Android-Image-Cropper/wiki – Image Cropper

[29] https://github.com/hdodenhof/CircleImageView - CircleImageView

[30] https://github.com/zetbaitsu/Compressor – Image Compressor

[31] https://square.github.io/okhttp – OkHttp

[32] https://www.joda.org/joda-time – JODA-Time

[33] https://square.github.io/retrofit – Retrofit