

UNIVERSITETI I PRISHTINËS
FAKULTETI I INXHINIERISË ELEKTRIKE DHE
KOMPJUTERIKE



PUNIM DIPLOME

Tema:

Zhvillimi i aplikacionit në Android për menaxhimin e
bashkëudhëtimit në taksi

Mentori:

Prof. Dr. Blerim Rexha

Kandidati:

Agon Hoxha

Prishtinë, Dhjetor 2019

Abstrakt

Si rezultat i zhvillimit të teknologjisë mobile dhe kushteve të ndryshme ekonomike të njerëzve, bashkëudhëtimi është bërë një metodë e përhapur e transportit. Mirëpo, në vendin tonë, është bërë metodë e përhapur pa pasur ndonjë platformë të veçantë për këtë çështje. Përdoruesit e kësaj metode të transportit ju është duhur të përdorin platforma që nuk janë menduar fare për këtë qëllim, apo të përdorin mënyra ilegale për transport. Në shtetet tjera, tashmë ekzistojnë platforma të veçanta për këtë qëllim, mirëpo në Kosovë jo. Ky punim do tentoj që ta ndryshoj këtë, duke krijuar një aplikacion që do të tentoj të jetë platformë e veçantë për bashkëudhëtim. Platforma e tillë duhet të jetë në gjendje të mbështes tregun e madh Kosovar në këtë drejtim, të jetë lehtë, të lehtësoj planifikimet dhe udhëtimet e përdorueseve, dhe tërë këto në mënyrën më moderne dhe eficiente të mundshme.

Abstract

As a result of the development of mobile technology and varying economic conditions, ridesharing has become a commonly used method of transportation. However, in our country, while it has become a widely used method, this has happened without a special platform designed for this purpose. The users of this method of transportation, have had to use platforms that were never intended for this purpose, or resort to illegal methods of travel. In other countries, there are already various platforms for ridesharing, but not here. With this app, that intends to be changed, by making an app that will try to become the method people will use for ridesharing. Said platform will have to support a considerable number of users, have to be intuitive, simplify planning and travel for users, and all this, needs to be done in the most modern and efficient way.

Përmbajtja

1.	Hyrja	1
1.1.	Hyrje	1
1.2.	Motivimi	1
1.3.	Përshkrimi i problemit	2
2.	Sistemi Operativ Android	3
2.1.	Hyrje	3
2.2.	Komponentët e aplikacionit	5
2.2.1.	Aktivitetet	6
2.2.2.	Shërbimet.....	6
2.2.3.	Broadcast Receiver	6
2.2.4.	Content Providers	7
2.3.	Rast studimi: Kotlin vs. Java	7
2.3.1.	Hyrje	7
2.3.2.	Java	7
2.3.3.	Kotlin	8
2.3.4.	Studim Krahasues	9
3.	Rasti studimi: Menaxhimi i bashkëudhëtimit me taks.....	15
3.1.	Hyrje	15
3.2.	Veglat dhe libraritë e përdorura	15
3.2.1.	Android Studio	15
3.2.2.	Google Play Services.....	16
3.2.3.	Firebase.....	16
3.2.4.	Libraritë e krijueseve tjerë.....	18
3.3.	Struktura e aplikacionit	19
3.4.	Forumi.....	20
3.4.1.	Regjistrimi	23
3.4.2.	Profili	24
3.5.	Komunikimi	26
4.	Diskutime dhe konkluzione.....	29
5.	Shtesat	30
5.1.	Lista e shkurtesave.....	30
5.2.	Lista e figurave	30
5.3.	Lista e tabelave	31
5.4.	Lista e pjesëve të kodit.....	31
6.	Bibliografia	32

1. Hyrja

1.1. Hyrje

Aplikacionet marrin rol më të madhe në jetën tonë, gjithnjë e më shumë - në vend të komunikimeve të formës relativisht të vjetër, si thirrje dhe mesazhe përmes rrjetës tradicionale që përdorej në telefona, është një tentim i vazhdueshëm që të gjendet mënyra me eficientë për të thjeshtësuar çështje të përditshmërisë. Një drejtim në të cilën kjo ka ndodhur është edhe udhëtimi. Prej navigimit përmes telefonit, deri në drejtim të veturës përmes telefonit mobil. Ky punim nuk është aq ambicioz sa drejtimi i veturës përmes telefonit mobil, mirëpo tenton të ofroj zgjidhje dhe të thjeshtësoj një çështje që prek shumë njerëz tërë kohën.

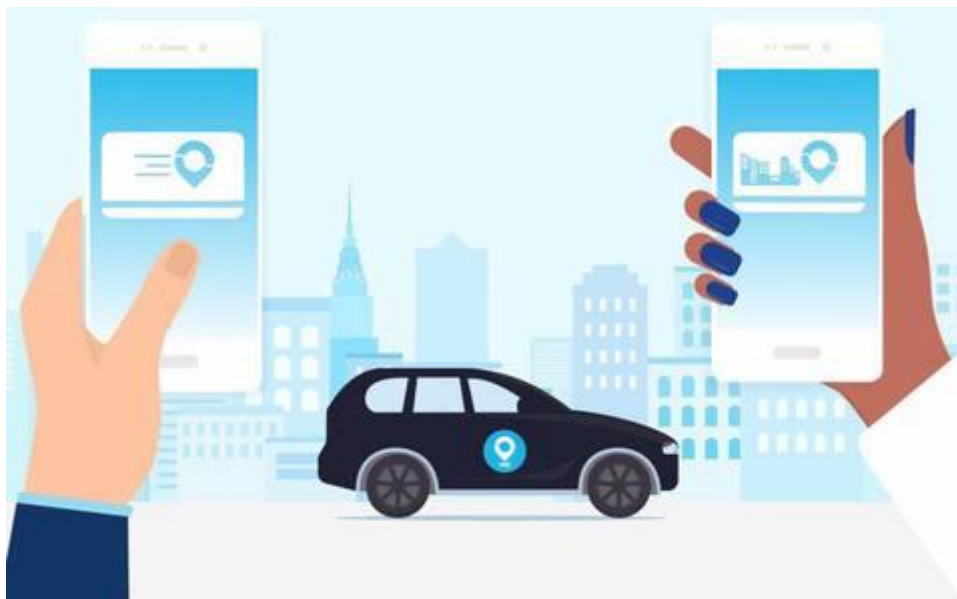


Figura 1. Përdorimi i aplikacioneve për bashkë udhëtim në vend të vozitjes [1]

Udhëtimi i përbashkët është ekonomikisht me i qëndrueshëm, kjo është pa dyshim ndër arsyet që aplikacionet në lidhje me bashkëudhëtim kanë arritur një lloj fame. Shembull i kësaj lloj aplikacioni, është Uber - aplikacion përmes të cilit janë bërë mbi 10 miliard udhëtime, apo 14 milion udhëtime çdo ditë [2]. Shembull tjetër është edhe Lyft, që gjersa nuk është afër nivelit të njëjtë me Uber, prapë se prapë ka një përdorim të madh, me mbi 1 miliard udhëtime deri më Shtator 2018 [3]. Përdorimi i aplikacioneve për bashkëudhëtim shihet si alternativë me shumë përparësi dhe pak anë negative, që është arsyeja pse lloj lloj njerëz i përdorin, dhe koncepti është mjaftë i thjeshtë, dhe mund të paraqitet me një figure të vetme, si me lartë në figurën 1.

1.2. Motivimi

Në Kosovë, këto aplikacione nuk janë në përdorim. Udhëtimi i përbashkët bëhet përmes transportit urban, apo transportit me vetura që faktikisht është ilegal. Së fundmi, organizimi i bashkëudhëtimit ka kaluar online në Facebook, përmes grupeve si Hitchhiker Kosova - UDHË, ku çdokush mund të bëjë postim në formë të ofertave, tregojnë se çfarë rruge do kalojnë, orën dhe shpesh e qartësojnë edhe çmimin siç është shfaqur në figurën 2.

Gjersa kjo është zgjidhje për këtë problem, mendoj se ekziston një zgjidhje më e mirë. Ky paraqet motivimin e punimit - krijimi i një aplikacioni që do zëvendësonte planifikimet e tilla nëpër grupe duke ofruar zgjidhje më të mirë.

1.3. Përshkrimi i problemit

Përdorimi i platformave të tilla për këtë çështje mund të jetë zgjidhje e përkohshme, por zgjidhja që e kanë menduar shfaq disa probleme:

- a. Përdor infrastrukture të një platforme që nuk është e menduar fare për këtë çështje.
- b. Mungon mundësia për shfaqje të sakte të pikënisjes dhe përfundimit.
- c. Komunikimi mes personave që pajtohen për udhëtim mund të bëhet vetëm përmes komentimit, apo platformës të Facebook për komunikim – Messenger, që paraqet problem pasi që zakonisht ndalon komunikimin ndërmjet personave nëse nuk janë miq në platforme.

Zgjidhja që do e ofroj në këtë punim, do tentoj që të i zgjedh këto mangësi që i ka metoda që përdoret tani për tani, duke ofruar një sistem që është krijuar për pikërisht këtë qëllim, të përfshij harta dhe navigim në vetvete që të e lehtësoj tërë procesin, si dhe të ofroj një mënyrë komunikimi mes përdoruesve të aplikacionit.

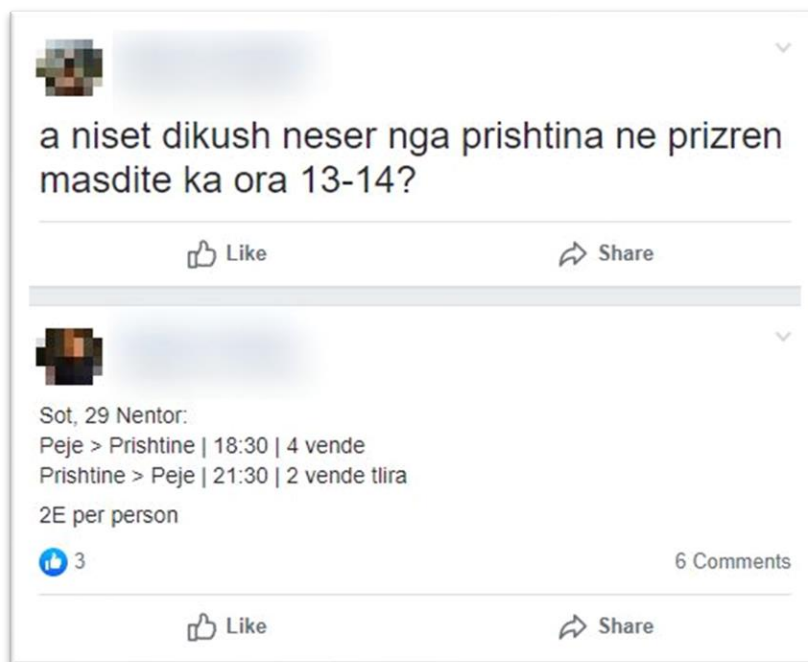


Figura 2. Organizimi i bashkë udhëtimit ne Facebook [3]

2. Sistemi Operativ Android

2.1. Hyrje

Android është një sistem operativ për pajisje të ndryshme, me fokus të veçantë në pajisje mobile të formës smartphone, mirëpo edhe në pajisje IoT, TV dhe PC [5][6][7]. Fillimisht e zhvilluar nga Open Handset Alliance, por pastaj kalon në pronësi të Google LLC në Tetor të vitit 2007, e cila e ka vazhduar zhvillimin prej që e ka marrë nën pronësi [8]. Android ka qasje open-source rreth shumicën e çështjeve; në fakt, pas çdo publikimi të versionit të Android, publikohet edhe AOSP (Android Open Source Project) në GitHub ku mund të shihet saktësisht se çka përmban sistemi operativ, me disa përjashtime [9]. Përbërja e sistemit operativ është paraqitur në figurën 3. Pasi që është open source, ka diçka që sistemet operativ konkurrense si iOS vështirë që mund të thuhet se e kanë - një përkrahje nga komuniteti. Ekzistojnë komunitete si xda-developers që merren me krijim të versioneve të modifikuara të Android, duke aplikuar ndryshime në source code të sistemit, kernel dhe si rezultat, shpesh arrihet shfrytëzimi më i mirë i resurseve që posedon pajisja se sa që ka mundur të shfrytëzohet përmes versionit të Android që ja u ka vendosur prodhuesi i pajisjes.

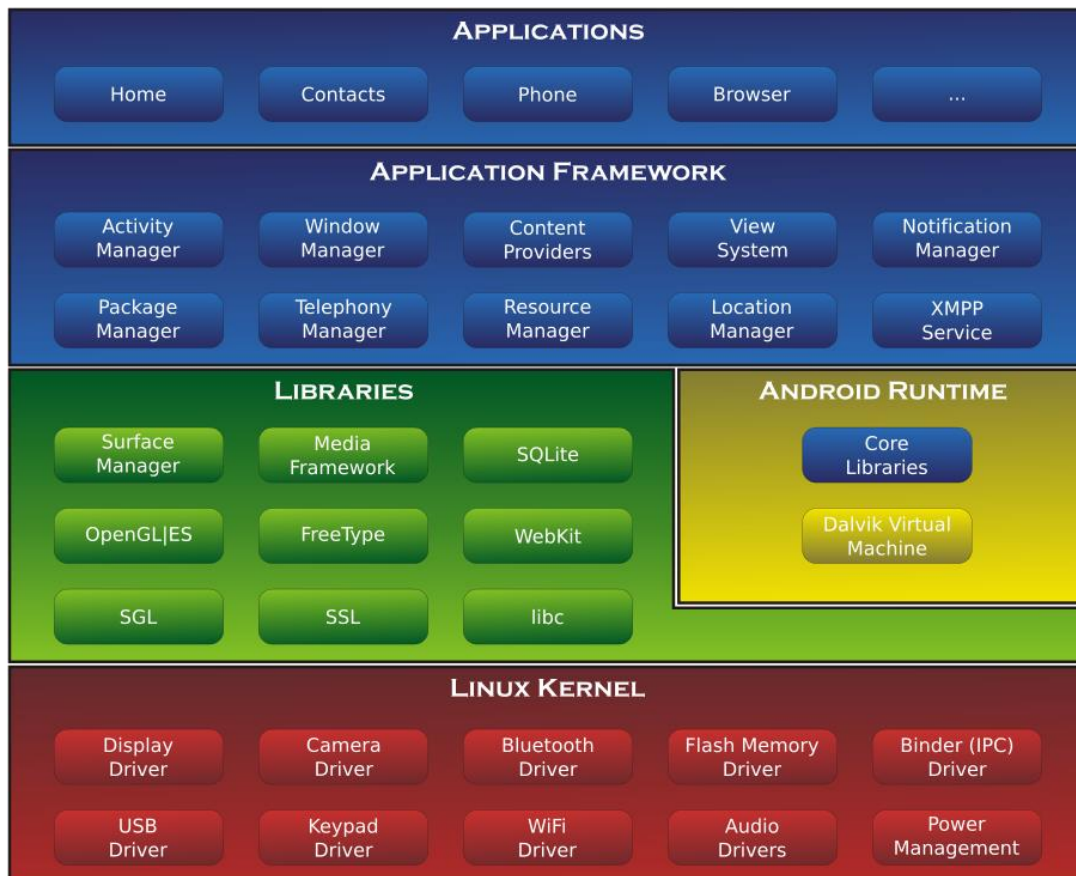


Figura 3. Arkitektura e Android [10]

Kjo lloj veçorie që gjithkush mund të krijoj versionin e vet të sistemit operativ Android ngjan me Linux distributions, kjo ka arsye: Android në prapavijë është Linux, më saktësisht kerneli që përdor është kernel i modifikuar i Linux, dhe ka një qasje të ngjashme në qasjen që përdoruesi ka në sistem operativ. Linux ishte pjesë kyçe e Android prej fillimit të zhvillimit, dhe ende është i bazuar në të, dhe zyrtar nga

Google janë cituar duke thënë se në të ardhmen, vetëm se do rritet ndërlidhshmëria mes Android dhe Linux [11].

Më parë, Android përdorte emrin e ëmbëlsirave si emër të versioneve me renditje alfabetike, por që nga versioni i fundit – që në kohë të shkrimit të këtij punimi është Android 10, nuk aplikohet më emërimi i tillë. Mirëpo emërimi i tillë është vetëm emërim komercial, pasi që në kuptim të zhvillimit, është më drejtë të thuhet që versioni i tanishëm është versioni 29, siç është sqaruar në Tabelën 1.

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 [11]

Një ndër mangësitë me të shpeshta që citohet rreth Android, është shpërndarja e versioneve. Me këtë nënkuptojmë se sa përqindje të pajisjeve në treg për momentin kanë cilin version. Zakonisht, përqindja e pajisjeve që kanë versionin e fundit është tejet e vogël. Kjo paraqet një sfidë për zhvillim të aplikacioneve, pasi që duhet të merret një vendim rreth cili nivel i API do duhet të përdorur. Nëse përdoret version i vjetër i API, rritet numri i pajisjeve ku mund të instalohet aplikacioni, por do kemi më pak veçori dhe funksione që mund të përdorim; në anën tjetër, nëse përdoret version tejet i ri i API, zvogëlohet numri i pajisjeve ku mund të instalohet aplikacioni, por do kemi qasje në veçori dhe

funksiione që mund të na lehtësojnë punën dhe të rrisin performancën e aplikacionit. Përdorimi i versioneve të ndryshme përgjatë viteve dhe publikimeve është paraqitur grafikisht në figurën 4.

Për aplikacionin e krijuar, është përdorur versioni minimal prej API 21, apo njohur me lehtë si ‘Lollipop’. Me këtë jemi siguruar që aplikacioni të mund të instalohej në afërsisht 90% të pajisjeve Android në përdorim, dhe njëkohësisht kemi siguruar qasje në disa veçori që na duhen.

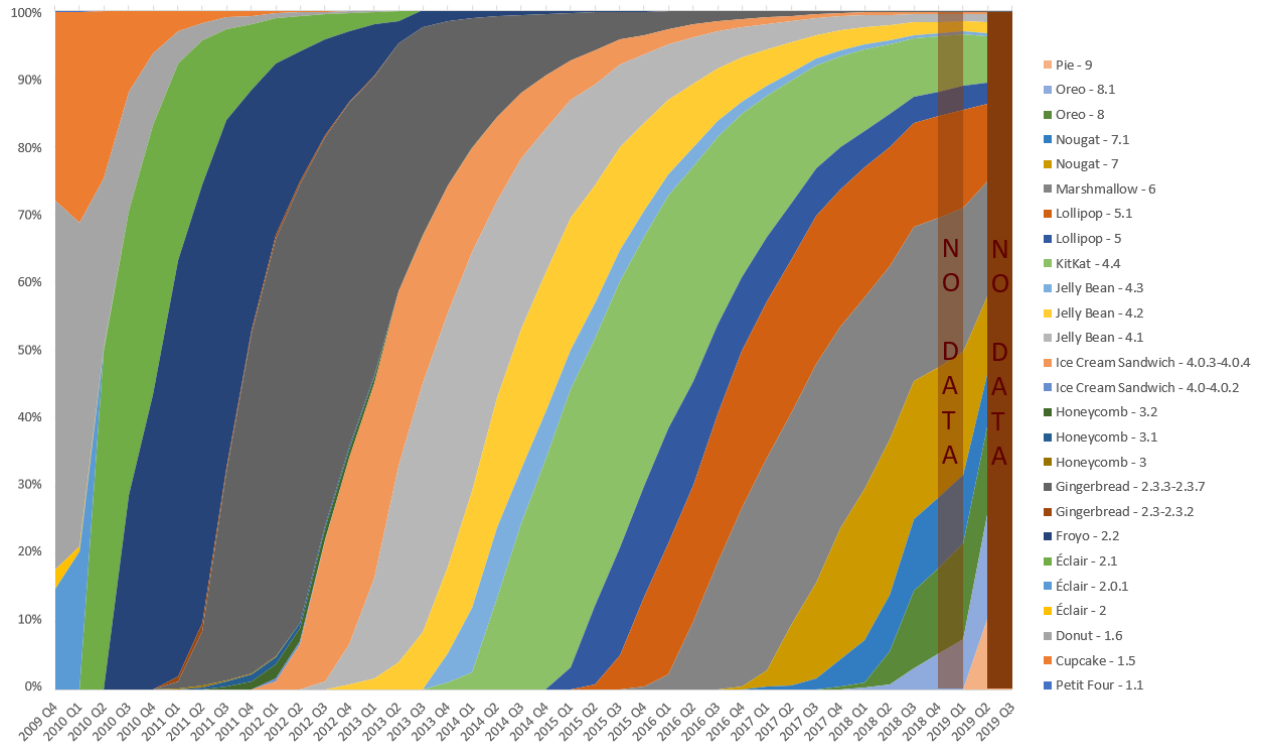


Figura 4. Shpërndarja e versioneve të Android [11]

2.2. Komponentët e aplikacionit

Cdo aplikacion në sistemin operativ Android përbëhet prej disa komponenteve kryesore: aktiviteteve, shërbimeve, broadcast receivers dhe content providers. Në vazhdim do specifikojmë çdo njëren në detaje dhe rolin që e kanë.

Activity Lifecycle

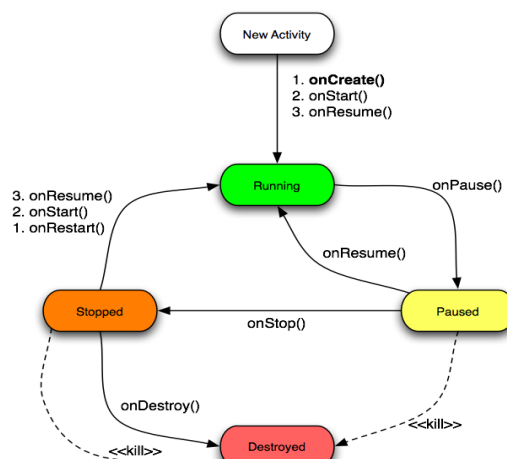


Figura 5. Etapat e jetës të aktivitetit [12]

2.2.1. Aktivitetet

Një aktivitet është një njësi e vetme e aplikacionit në të cilën përdoruesi vepron. Një aktivitet kalon nëpër disa etapa, duke filluar me funksionin onCreate(), siç është paraqitur në figurën 5. Prej atë mund të kalon nëpër disa gjendje si Running, Paused, Stopped apo Destroyed. Në gjendjen Running ekzekutimi behet normalisht; në gjendjen Paused, siç le të kuptoj emri, është pushuar ekzekutimi, prej nga mund të kthehet në gjendjen Running përmes onResume(); në gjendjen Stopped, ekzekutimi është ndalur dhe nuk mund të vazhdoj ekzekutimi, vetëm të filloj prapë nga e para apo të përfundoj.

2.2.2. Shërbimet

Shërbimet shfrytëzohen për të kryer operacione që marrin kohë të gjatë në prapavijë, apo që duhen të ndërmarrën operacione të caktuara vetëm në kushte të caktuara. Roli i shërbimit është ndërlidhur me broadcast receiver, siç është paraqitur në figurën 6.

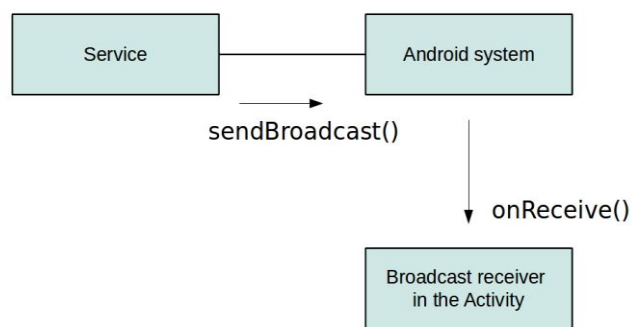


Figura 6. Shfrytëzimi i shërbimit në Android

2.2.3. Broadcast Receiver

Broadcast Receiver janë komponentë që lejojnë të përcaktojmë përgjigje për ngjarje në aplikacion apo në tërë sistemin. Çdo broadcast receiver të regjistruar menaxhohen nga Android kur të ndodh ngjarja e cukur. Shembull, një aplikacion mund të deklaroj Broadcast Receiver për ngjarjen 'ACTION_BOOT_COMPLETED', dhe të cek një funksion që duhet ekzekutuar në atë moment. Kjo do rezultonte që sapo të ndezët telefoni, ai funksion i aplikacionit tonë do të aktivizohet. Receiver duhet të jetë i regjistruar në Manifestin e aplikacionit. Manifesti i aplikacionit ja shpjegon sistemit operativ, Play Store dhe build tools, të dhënat esenciale mbi aplikacionin tonë, si aktivitetet që përmban, lejet (permissions) që i duhen, shërbimet që i ka, ngjarjet në të cilat do reagoj (broadcast receivers), etj. Broadcast receivers mund të ju përgjigjen thirrjeve që vijnë nga sistemi, si shërbimet që janë të sqaruara më lartë, apo edhe thirrjeve që bëhen nga vet përbrenda aplikacionit, siç është paraqitur në figurën 7.

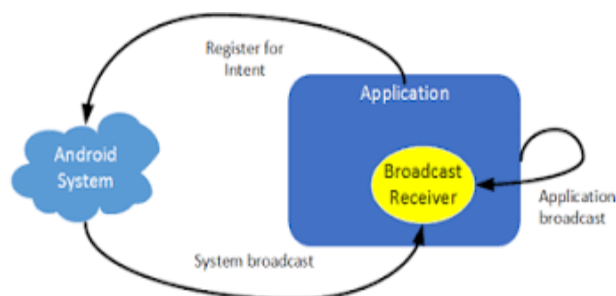


Figura 7. Funksionimi i Broadcast Receiver [13]

2.2.4. Content Providers

Content Provider menaxhon qasjen në një depo të të dhënave. Providers zakonisht japin një UI të vet të cilën mund të e manipulojmë ashtu si na duhet. Content providers zakonisht përdoren për dy raste: kur dëshirojmë që të qasemi në të dhëna që aplikacioni jonë përmban, apo të lejojmë që aplikacioni jonë të ndaj të dhëna me aplikacione tjera.

2.3. *Rast studimi: Kotlin vs. Java*

2.3.1. Hyrje

Kur zhvillojmë aplikacione në Android, kemi disa opsione sa i përket gjuhës që dëshirojmë të përdorim:

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

Shumica e aplikacioneve që krijohen, krijohen me Java, por dy vitet e fundit, ka pasur tentime që të kalohet në një gjuhë tjetër – më saktësisht, ka pasur tentime të kalohet sa më shumë në gjuhën Kotlin. Sot, afro 16% të aplikacioneve në tërë Play Store e përdorin në vend të Java [14]. Java dhe Kotlin kanë përparësitë dhe mangësitë e veta që do i shtjellojmë tani.

2.3.2. Java

Java është momentalisht gjuha më e përdorur programuese. Kjo është me arsye, pasi që programet e shkruara në Java mund që teknikisht të funksionojnë në mbi 3 miliard pajisje. Më tej, është pohuar nga Oracle, që aktivisht e zhvillon gjuhën Java, se ekzistojnë mbi 21 miliard cloud-connected JVMs [15]. Fillimisht, krijuar nga Sun Microsystems, është gjuhë e bazuar në klasa, e orientuar në objekte, dizajnuar të këtë të implementuar sa më pak varësi. Është tentuar që aplikacionet të shkruhen një herë dhe të ekzekutohen kudo (koncepti WORA). Version i tanishëm është Java 13, dhe Java 14 është e planifikuar për publik në Mars, 2020. Ndërsa Java 12 është ende duke u përkrahur. Dizajni dhe zhvillimi i gjuhës Java është bazuar në disa koncepte:

- Duhet të jetë thjeshtë, orientuar në objekte dhe familjare.

Luan rol të një motori për pajisje të vogla dhe të mëdha kompjuterike, si i tillë, ky është prioritet.

- Duhet të jete sigurtë dhe pa probleme.

Java është fuqishëm për shkak të përkrahjes. Mund të përdoret në sisteme të ndryshme. Përkrah menaxhim të memories dhe largim të mbeturinave në memorie. që këto të mbesin të vërteta, versionet e reja duhen të jene me të sigurtë dhe pa probleme [14].

- Duhet të jete neutral në arkitekture, dhe e paanshme.

Që të mbetet neutral, kompiluesi krijon files neutral në format të Java code (bytecode), që mund të përdoret në procesorë të ndryshëm me kusht që të ketë Java runtime.

- Duhet të ketë performancë të lartë.

Java ka performancë të lartë për shkak të JIT. JIT ndihmon në kompilim të kodit ashtu siç është nevoja, dhe nuk vendos objekte në memorie nëse nuk është nevoja.

- Duhet të interpretohet, përkrah multithreading dhe të jetë dinamike.

Interpretuesi i Java mund të ekzekutoj Java bytecodes direkt në makinë. Pasi që përkrah multithreading, ajo gjen përdorim në programe ku ka nevojë për performancë të lartë. Një multithreaded program përmban pjesë që mund të ekzekutohen njëkohësisht dhe çdo pjesë mund të merret me një detyrë të caktuar në të njëjtën kohe duke pasur përdorim optimal të resurseve. Gjersa Java është strikt në lidhje me kompilim, në fazë të linking, Java është tejet dinamike. Klasat linkohen ashtu siç është nevoja, qoftë edhe nëpër rrjetë. Në rast të HotJava Browser dhe aplikacione të ngjashme, kodi ekzekutueshëm mund të ekzekutohet nga kudo, që lejon për ndryshime në aplikacion që nuk vërehet nga përdoruesi. Rezultati i kësaj janë web shërbime që janë gjithnjë në zhvillim; mund të mbesin inovativ dhe të reja, të marrin më shumë konsumatorë [16]. Vjen në versione të ndryshme, por versioni kryesor mund të përdoret pa pagesë, përpos për projekte komerciale. Java përdor javac compiler, që kthen kodin nga Java në Java bytecode. Java bytecode është në relacion me Java files, si assembly code është me C files.

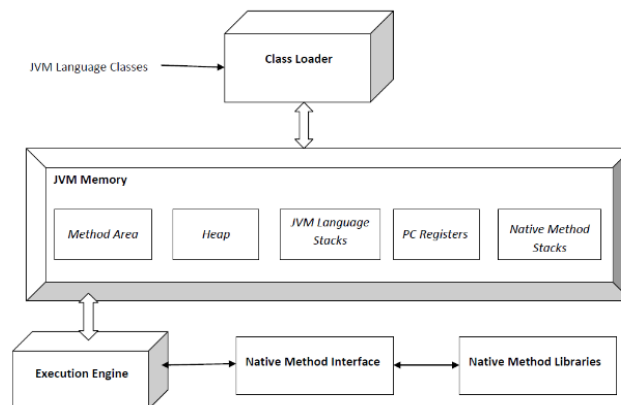


Figura 8. Java Virtual Machine [17]

2.3.3. Kotlin

Kotlin është deklaruar si gjuha zyrtare për zhvillim në Android në Maj 2018, por është gjuha e tretë e shtuar si 'e përkrahur' (e dyta ishte C). Është gjithashtu gjuha e preferuar nga Google për zhvillim [18]. Qëllimi nga fillimi ishte të jetë një alternative për Java – është e dizajnuar që të konkurroj me Java [19]. Është momentalisht e mirëmbajtur dhe zhvilluar nga JetBrains, dhe përkrahet nga Android Studio duke filluar me versionin 3.0. Gjithashtu është prezent në disa tjera IDE të përdorura për zhvillim të aplikacioneve në Android, si Eclipse dhe IntelliJ IDEA. Përdor Java Virtual Machine për ekzekutim, si vet Java, dhe si e tillë, duhet të konvertohet në Java bytecode. Menyra e funksionimit të Java Virtual Machine është paraqitur simbolikisht në figurën 8. Ndryshe nga Java, që përdor javac compiler, Kotlin përdor kotlinc compiler, por sidoqoftë rezultati përfundimtar është Java bytecode. Evolucioni i Kotlin filloj në Rusia në 2011. Zhvilluesi i saj, JetBrains kërkonte një zëvendësim për Java për shkak të disa limitimeve që hasnin në Java. Për arsye të qarta, ata ishin të orientuar nga një gjuhë e bazuar në Java. Prej gjuhëve të ndryshme që përdorin JVM, konsideruan Scala si më të 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 të krijoj gjuhën e tyre të përshtatshme me JVM [20].

2.3.4. Studim Krahases

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 që duhet të kthejë një String të ri me karakterin e parë dhe të fundit të hequr; kjo metodë nuk gjendet në klasën String. Funksioni i shtrirjes i deklaruar jashtë klasës krijon një funksionalitet të klasës së specifikuar që shtrin funksionet e paracaktuara. Kjo bëhet si në kodin 1.

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

Kodi 1. Zgjerimi në Kotlin

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 [21]. Zgjidhja e Java për këtë është krijimi i mbështjellësve.

2.3.4.2. Kontrollimi i Përrjashtimeve

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. Në kodin 2 është sintaksa që përdoret në Java.

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

Kodi 2. Shembull kodi në Java për try dhe catch

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. Në Kotlin, "hedhja" duhet të përdoret për të shkaktuar përjashtime, si në kodin 3.

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

Kodi 3. Mënyra e krijimit të përjashtimit në Kotlin

Përjashtimet e kontrolluara mund të thyjnë logjikën ose rrjedhën e kodit. Veçanërisht në kode me shumë metoda kthyesë, 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 në rastin me të mire, rritje minimale të cilësisë së kodit [22].

2.3.4.3. Siguria në 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 NPE për shkurt, gjithashtu i referuar si "gabimi miliardë dollarësh" nga komuniteti por edhe 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. Në kodin 4, synojnë të hedhin një NullPointerException në Java.

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

Kodi 4. Shkaktim i një NullPointerException në Java

Rezultati: NullPointerException.

Ne kodin 5 është i njëjti funksion, mirëpo në Kotlin.

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

Kodi 5. Kodi ekuivalent me Kodin 4, në Kotlin

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 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. Kuptueshmëria

Kotlin preferohet për disa arsye për përdorim në zhvillim, por një nga arsyet më të mëdha 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. Në kodin 6 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;
    }
}
```

Kodi 6. Klasa Personi me getters dhe setters për variablën 'name'

Në kodin 7, është i njëjti funksion, mirëpo në Kotlin.

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

Kodi 7. Ekuivalenti i Kodit 6, në Kotlin

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 dembel (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 kritere. Patrik Schwermer bëri një studim me këtë objektiv të saktë, duke përdorur teste standarde të gjuhës kompjuterike. Do të marrim aspektin e performancës së krahasimit nga puna e tij [23]. Ai drejtoi disa kritere, si Fasta benchmark, Fannkuch-Redux benchmark, N-body benchmark dhe Reverse-complement benchmark. Rezultatet e tyre janë paraqitur në figurën 9.

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. Arsyetja pse është atje, Patrik sugjeron është fakti që Java ka qenë gjuha zyrtare e Android shumë më gjatë dhe si e tillë, edhe

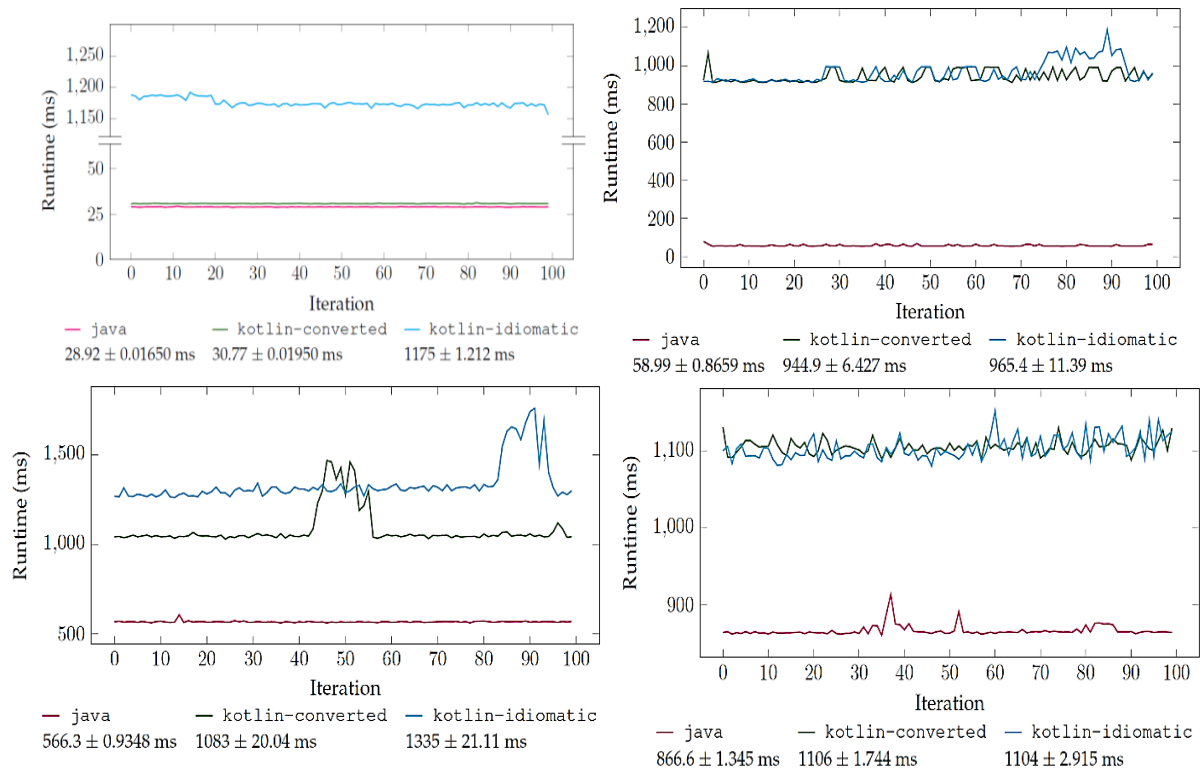


Figura 9. Benchmarks mes Java dhe Kotlin [23]

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 në 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 tjetër 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 të 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 që 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. [24]

2.3.4.9. Përfundimi i Studimit Krahases

Analizuar disa nga avantazhet dhe disavantazhet e Java dhe Kotlin, dhe mund t'ju tregojmë se Java është një gjuhë shumë e njohur në mesin e zhvilluesve, mirëpo zhvillimi i aplikacioneve në 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 programerëve në 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 në mund të mos dimë se ku të drejtohem kur kemi ngecje. Kur kërkojmë për problemet Java, gjasat janë tejet të larta që të marrim përgjigje; nuk mund të thuhet e njëjta gjë për Kotlin, e cila është ende një gjuhë programimi në zhvillim e sipër. Ka edhe më shumë mësim, 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 këndvështrimi i zhvilluesit, Kotlin do preferohej për arsye të 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 që 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 qartë 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. 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 përdoret 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, në do të duhet të rekomandojnë Java për shkak të dokumentacion tejet të madhe dhe rezultateve që janë nxjerrë nga standardet përkatëse. Por në çdo rast tjetër, rekomandohet Kotlin. Përfitimet e saj tejkalojnë të metat.

Për arsyet e cekura me lartë – lehtësimin që Kotlin jep për zhvilluesin, thjeshtësinë dhe kohën që kursen, është përdorur si gjuha kryesore për aplikacionin. Këtu fjala kyçe është ‘gjuha kryesore’, pasi që në aplikacion përmban edhe kod në gjuhën Java, pasi që është plotësisht e mundshme përdorimi i të dyjave në të njëjtën kohë.

3. Rasti studimi: Menaxhimi i bashkëudhëtimit me taksi

3.1. Hyrje

Në kuadër të punimit të diplomës, është zhvilluar aplikacioni për menaxhim të bashkëudhëtimit me taksi në sistemin operativ Android. Aplikacioni Eja Shkojmë ofron platformë të veçantë që tenton të i plotësoj nevojat e shfaqura rreth çështjes të bashkëudhëtimit. Aplikacioni përmban tri module: modulën e forumit, modulën e bisedave dhe modulën e udhëtimit. Moduli i forumit është thjeshtë një hapësirë në të cilën mund të krijohen postim në lidhje me udhëtime; moduli i bisedave është një hapësirë ku mundësohet komunikimi mes përdoruesve të aplikacionit përmes tekstit; moduli i udhëtimit është një version i thjeshtësuar i navigimit që ofrojnë aplikacionet si Google Maps.

3.2. Veglat dhe libraritë e përdorura

3.2.1. Android Studio

Android Studio është IDE me e përdorur për zhvillim të aplikacioneve në Android, mirëpo jo e vetmja, pasi që ka edhe IDE tjera si Eclipse, IntelliJ IDEA, Xamarin, etj. Është e krijuar nga JetBrains, në bashkëpunim me Google, e cila edhe e rekomandon. Android Studio përkrah gjuhët Java, Kotlin dhe C++, mirëpo gjithashtu përkrah edhe files që përdoren për aplikacione si XML files, së bashku me pjesën e dizajnit. Ekzistojnë versione për sistemet operative Windows, MacOS, Linux, si dhe Chrome OS, dhe në të njëjtën kohe, ka përshtatje të integruar për Gradle që po thajse çdo aplikacion në Android e përdor.

Çka e veçonte në fillim Android Studio, ishte integrimi i dizajnit – kishte drag and drop features për krijim të dizajnit, mirëpo gjithashtu lejonte ndryshim të dizajnit përmes ndryshim të files të asociuara XML. Pra përmban editor të kodit jo vetëm për Java, Kotlin & C++, por edhe për XML. Editori i kodit në fakt është ndër përparësitë që Android Studio ka kundrejt IDE-ve tjera për zhvillim të aplikacioneve në Android pasi që është optimizuar për të. Editori i kodit ofron edhe veçori të refaktorimit për pastrim dhe riformatim të kodit, veçori kjo që mund të përdoret për të parë më lehtë dhe thjeshtësuar rrjedhjen e kodit.

Gjithashtu, menaxhon instalimin, mirëmbajtjen dhe kontrollimin e sistemeve të simuluar përmes Android Emulator. Android Emulator është ekzekutim i sistemit operativ Android, që reagon në të njëjtën mënyrë që do reagonte një telefon real [25].

3.2.1.1. AndroidX

Përgjatë procesit të zhvillimit të aplikacioneve në Android, disa librari që ishin në përdorim, janë mbetur ende në përdorim deri në një masë. Mirëpo, përgjatë viteve dhe versioneve, janë shfaqur konflikte mes versioneve të librarive që ishin në versionet e vjetra, dhe versionet që ishin në versionet e reja të Android. Për të e zgjidhur këtë, është publikuar AndroidX. AndroidX përpos që harmonizon konfliktet, krijon edhe konsistencë për emërime:

- Emërimet në 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`.

- Emërimet në AndroidX janë: `androidx.recyclerview.widget.RecyclerView`, `androidx.recyclerview.widget.GridLayoutManager`, `androidx.recyclerview.widget.LinearLayoutManager`, `androidx.constraintlayout.constraintlayout`.

Në anën tjetër, ishte edhe zgjidhje për problemet mes Support Library v4 dhe Support Library v7, duke krijuar një librari të unifikuar. Mirëpo, nuk është pa të metat e veta – nuk mund të përdoren Support Libraries dhe AndroidX libraritë në të njëjtën kohe, që mund të pengoj me implementim të aplikacioneve, gjë që krijoj problem gjatë krijimit të aplikacionit, pasi që shumica e këshillave që janë në internet, janë duke u referencuar në Support Libraries, dhe si tille nuk janë përshtatshme me implementim të AndroidX [26]. Për të e konvertuar projektin në AndroidX, kërkohet refaktorim që simbolikisht paraqitur në figurën 10.

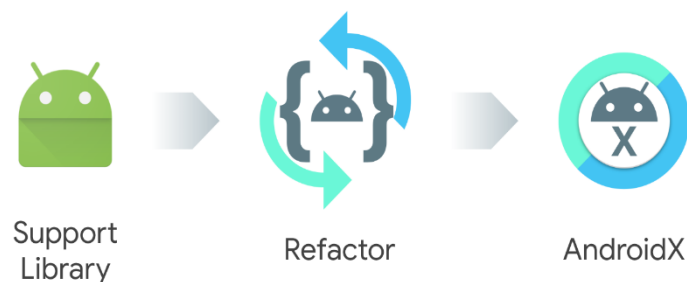


Figura 10. Konvertimi i Librarive [26]

3.2.2. Google Play Services

Pasi që punimi është në lidhje me udhëtime, nënkupton që do përdoren metoda për përcaktim të lokacioneve, gjetje të lokacionit, shfaqje të lokacionit në hartë. Standard për tërë këto, janë shërbimet e Google që i ofron në lidhje me lokacione për shkak të përdorimit tejet të lartë të tyre [27]. Përdorimi i tyre bëhet përmes API që Google jep qasje në to. Në këtë punim, janë përdorur tri API:

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

Për të pasur qasje në to, duhet të regjistrohen për projekt dhe të gjenerohet një API key, i cili pastaj përdoret në aplikacion për identifikim të aplikacionit kur kërkohen shërbimet e lartë cekura. Për nga ana e aplikacionit, në Gradle duhen të përfshihen që të trija libraritë relevante për këto API.

Libraria e parë, na mundëson të shfrytëzojmë shërbimet e lokacionit që i ka Google, që janë me eficientë se të sistemit. Places API na mundëson gjetjen e lokacioneve prej emrit dhe kthimin e emrit, si Prishtinë, në format të kuptueshëm për Maps SDK, që pastaj të vendoset në harte në pikën e duhur. Directions API na mundëson marrjen e shtegut që duhet kaluar prej pikënisjes deri në destinacion – këto i marrim në formë të JSON, të cilat pastaj mund t'i manipulojmë për nevojat tona. Maps SDK përdoret në përgjithësi për qasje në harta të Google.

3.2.3. Firebase

Firebase është një grumbull shërbimesh që ofrohen për zhvillim të aplikacioneve. Është platformë nën pronësi të Google, dhe e njohur si platforma më e përdorur për implementim të databazës në aplikacione të sistemit operativ Android, duke u përdorur në qindra mijëra aplikacione dhe pothuajse të gjitha aplikacionet e reja [28]. Edhe pse është e menduar veçanërisht për implementim në Android,

ekzistojnë metoda që të përdoret edhe për aplikacione desktop. Nder shërbimet që ofron janë databaza, analizime, raportime për dështime në aplikacion, autentifikim, hapësirë online, shërbime për mesazhe, etj.

3.2.3.1. Autentifikim

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

3.2.3.2. Databaza

Databaza e përdorur për këtë aplikacion përdor strukturën e paraqitur në Figurën 14. Çdo gjë që ndodh në aplikacion, ka të bëjë me databazën – çdo mesazh që dërgohet, çdo miqësi që krijohet, çdo postim, koment, person që regjistrohet, çdo udhëtim që planifikohet. Kjo mund të bie në pyetje çështjen e privatësisë, që në fakt edhe paraqet problem. Në kushte komerciale, do duhej që të aplikohesh enkriptim i mesazheve, postimeve, komenteve, e të gjitha të dhënat që të mos ketë qasje në to ndokush i paautorizuar, por duke marrë parasysh natyrën e aplikacionit, kam vendosur që të mos aplikohet një gjë e tillë dhe tërë informatat të ruhen në plaintext.



Figura 11. Pamje high-level e databazës për aplikacionin

3.2.3.3. Hapësira (Storage)

Firebase ofron mundësi për ruajtje të dhënave të ndryshme në një kontejner që e quan 'bucket'. Në rast tonin, bucket e Firebase është përdorur që të ruajmë fotot e profilit të përdoruesve në to. Mund të

përdoret për çfarëdo arsye tjetër, mirëpo nuk kam parë nevojë që të përdoret për ruajtje të ndonjë gjë tjetër.

3.2.3.4. Messaging

Shërbim i radhës të Firebase që është përdorur, është Messaging, apo dërgimi dhe pranimi i mesazheve përmes Firebase. Kjo eliminon nevojën për server të veçantë për komunikime, apo anashkalon mbështetje në shërbime që nuk janë direkt nën kontroll tonë.

3.2.3.5. Core

Core është një librari mbështetëse e Firebase. Gjersa ajo nuk është përdorur drejtpërdrejtë, është e nevojshme që disa komponentë tjerë të përdorura të Firebase të funksionojnë në rregull.

3.2.3.6. GeoFire

GeoFire është shërbim relativisht i ri i Firebase që ende nuk figuron në faqen e tyre kryesore. Përmes GeoFire, lokacionet në vend që të ruhen sipas koordinatave të tyre, ruhen në databazën tone përmes një identifikuesi të veçantë në formë të një stringu. Shembull: lokacioni ‘Podujeva’, në vend që të ruhet me koordinata 42.9108 N dhe 21.1956 E, ruhet si ‘keSNFTivsnUPIAIoR0YL2zjKYi63’. Kjo na mundëson manipulim me të lehtë me lokacione se sa që do mundësohej përmes koordinatave pasi që më nuk kemi dy variabla të tipeve double, por vetëm një variable string që tregon saktësisht vendin. Pra, na furnizon me efikasitet të njëjtë sikur koordinatat, mirëpo na lehtëson procesin e manipulimit me të dhëna [29].

3.2.4. Libraritë e krijueseve tjerë

Nisur nga parimi që na është mësuar gjatë studimeve, ‘nuk keni nevojë të zbuloni rrotën, mjafton të dini si të përdoret’, e kam parë si të arsyeshme që të përdoren librari të krijueseve tjerë që të lehtësohet krijimi dhe funksionalizimi i plotë i aplikacionit. Përmes përdorimit të tyre, koha është kursyer dhe pasi që janë librari tejet shume të përdorura dhe të testuara, mund të jemi më të sigurtë se sa nëse do i implementonim vetë vetitë që i kanë këto librari.

3.2.4.1. Picasso

Një ndër libraritë e para të futura në përdorim është Picasso nga SquareUp. Kjo librari mundëson manipulim me të lehtë të fotografive brenda aplikacionit. Lehtëson punën pasi që mund të përdoret në kombinim me RecyclerView dhe Adapters, dhe në të njëjtën kohë është në gjendje të marr fotografi nga interneti, të i transformoj fotot në bazë të nevojës, të i vendos në cache, dhe tërë këtë me shfrytëzim minimal të resurseve të memories. Kjo gjenë përdorim në shfaqjen e fotove të profilit të çdo përdoruesi [30].

3.2.4.2. Material DateTime Picker

Përcaktimi i datës dhe kohës për fillim të nisjes të udhëtimit bëhet përmes Material DateTime Picker. Kjo librari na lehtëson këtë punë, dhe në të njëjtën kohë ka një dizajn tërheqës që përdor edhe sistemi operativ Android [31].

3.2.4.3. Image Cropper

Kur përdoruesit ngarkojnë foto për profil, shpesh është nevoja që ato të prehen në mënyrë që të përshatën me standardin që e kemi aplikuar – duhen të jenë katror në mënyrë që të mos shkaktojnë probleme. Këtë e arrijmë përmes Image Cropper, që detyron përdoruesin foton e ngarkuar të e prej në

formatin tonë të parapërcaktuar. Image Cropper ofron edhe formate tjera, jo vetëm katror (1:1), si 16:9, 4:3, formë të lire, etj., mirëpo për nevojat tona, na është nevojitur 1:1, prandaj atë e kemi aplikuar [32].

3.2.4.4. CircleImageView

CircleImageView është arsyeja pse Image Cropper e ka formën të përcaktuar si 1:1. Kjo pasi që kur të shfaqen fotot, e kemi përcaktuar që të shfaqen në forme rrethore, dhe kthimi i një fotoje që nuk përshtatet me formatin 1:1 në formë rrethore, doli se është problematike. Fotografitë në aplikacion në forum dhe vende tjera, përmes kësaj librerie shfaqen ngjashëm siç shfaqen në aplikacione si Facebook, Instagram, Twitter, etj, pra në formë rrethore [33].

3.2.4.5. Compressor

Kur përdoruesi ngarkon një foto, ajo foto është rëndom e shkrepur me kamerë dhe si e tillë ka një madhësi jo edhe relativisht të vogël. Po të lejohej përdorimi i fotove të tilla, nëse aplikacioni ngarkohet më shumë përdorues, kushdo që qaset në forum, telefoni i tyre do duhej të shkarkonte të gjitha fotot e përdorueseve në kualitetin ashtu siç i kanë ngarkuar. Këtë duhet anashkaluar, siç e anashkalojnë edhe aplikacionet tjera, përmes kompresimit të fotos. Fotoja e ngarkuar kompresohet dhe në vend që të përdoret versioni i ngarkuar, përdoret versioni i kompresuar duke rritur shpejtësinë e aplikacionit. Eficienca e kompresimit ka variacion, mirëpo sipas krijuesit mund të rangoj deri 100 herë më pak vend të nxënë. Në testimet e bëra, ky numër ishte më afër 75, duke kompresuar një fotografi me 7.5MB në vetëm 88kB, por është më se mjaftueshëm. Kjo përpos që lehtëson aplikacionin, mund të ruaj edhe buxhetin pasi që Firebase aplikon kosto varësisht prej sasisë të GB të përdorur [34].

3.2.4.6. OkHttp

OkHttp është një projekt për një HTTP klient me eficient, përkrah kërkesa për HTTP 2.0, dhe përkrah dërgim të kërkesave të shumëta përmes të njëjtit socket connection. Është përfshirë në aplikacion pasi që lejon dërgimin e kërkesave me header që mund ta manipulojmë lirisht, dhe është duhur një i tille [35].

3.2.4.7. JODA-Time

JODA-Time është API e krijuar që të ofroj klasa me të mira në lidhje me koha dhe data se sa ofronte java.util që është e përfshirë, qasje më të lehtë dhe funksione më të vryeshme [36].

3.2.4.8. Retrofit

Retrofit është librari që mundëson trajtimin e një API të caktuar sikur të ishte një klase. Në rastin tonë, kjo është përdorur që të kemi qasje më të lehtë në Google API [37].

3.3. *Struktura e aplikacionit*

Aplikacioni përmban një numër të aktiviteteve, adapterëve, ViewHolders, fragmente, ndihmës (utilities). Hapja e aplikacionit fillon me aktivitetin për marrje të qasjes (login). Prej aty, mund të kalohet në një numër tjetër të aktiviteteve, varësisht prej veprimeve. Veprimi më i zakonshëm do ishte shkrimi i email dhe fjalëkalimit, duke marr qasje në aplikacion. Kur të këtë nevojë për ndogjë në lidhje me lokacion, apo shfaqje të hartave, nëse nuk i është dhënë leja për qasje në lokacion, do i kërkohet qasja në sensorët e GPS.

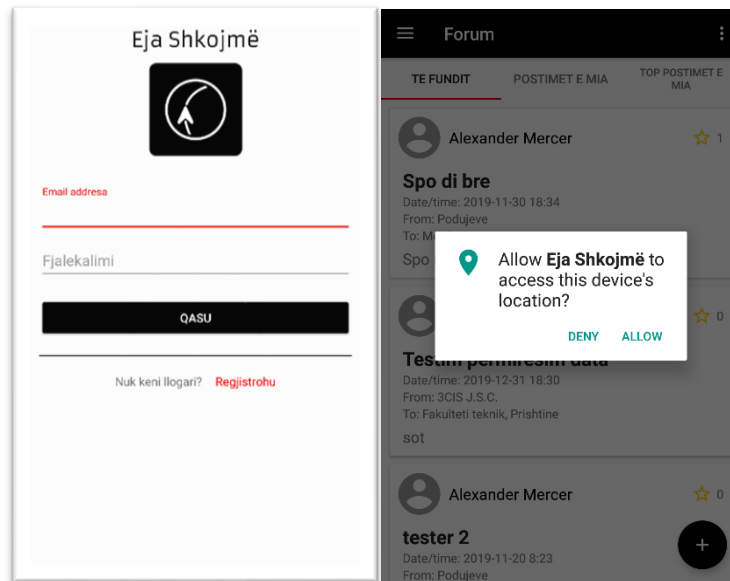


Figura 12. Hapat e parë në aplikacion

3.4. Forumi

Pjesa kryesore e aplikacionit është forumi, i paraqitur në figurën 13. Në forum shihen postimet e kaluara, mund të shihen vetëm postimet e përdoruesit që është i qasur, apo të i sheh top postimet, që llogariten në baze të likes. Nga forumi, mund edhe të qasemi në postimet që janë postuar, nëse

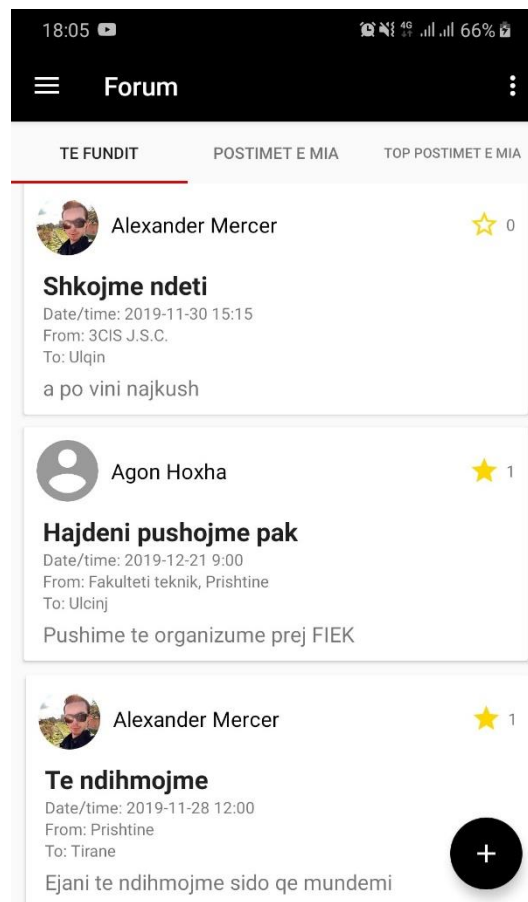


Figura 13. Pamja kryesore

dëshirojmë të komentojmë, apo mund edhe të fillojmë të krijojmë postim të ri. Forumi funksionon përmes të klasës ForumFragment – fragment të cilit iu është deleguar ngarkimi i postimeve të fundit, postimet e përdoruesit dhe top postimet e përdoruesit, gjë që bëhet me klasat për postime të fundit, postimet të përdoruesit dhe top postimet e përdoruesit respektivisht. të gjithë këto klasa dhe fragmente kanë varësi nga PostListFragment – një ndër klasat e vetme të shkruar në Java, që i merr të dhënat nga Firebase varësisht prej skenarit, dhe kthen të dhënat e duhura nëpër RecyclerViews tek fragmentet e lartë cekura që më pastaj e bëjnë shfaqjen. PostListFragment merr qasje në Firebase, po si edhe klasat tjera që kanë qasje në Firebase databazë, përmes referencës të instancës të databazës – kjo bëhet si në kodin 8.

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

Kodi 8. Krijimi i lidhjes me Firebase

Adapteri që përdoret për të marrë të dhënat, më saktësisht FirebaseRecyclerAdapter, i merr të dhënat nga databaza, krijon pamjen, por edhe përcakton funksionet kur përdoruesi të klikon në postim apo në emër të postuesit. Në rast se klikon në postim, aksioni delegohet tek MainActivity që e ka funksionin e

```
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();
});
```

Kodi 9. Hapja e Profilit

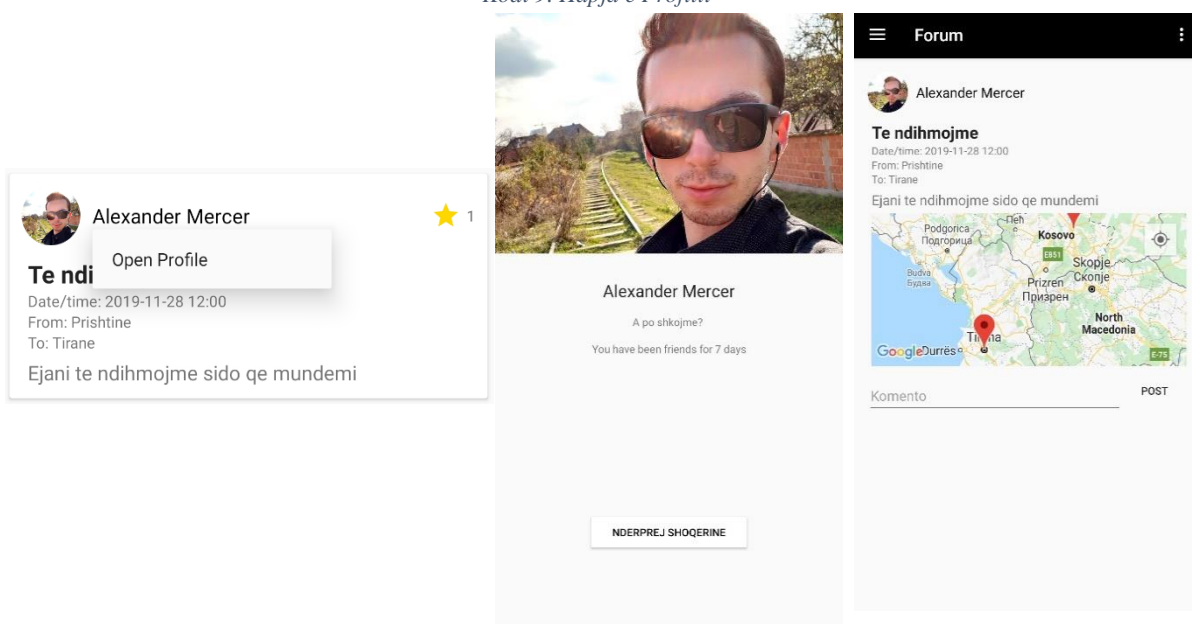


Figura 14. Hapja e profilit, apo postimi më i detajuar

definuar për hapje të postimit, kurse nëse klikon në emër të postuesit, e merr përsipër vet pasi që vetëm i duhet të shfaq opsionin për të hapur profilin e përdoruesit - kjo bëhet me kodin 9, dhe shfaq si rezultat pamjet në figurën 14.

Përkthyer në gjuhë të kuptueshme për njerëz, nëse klikon në postim, ekzekuto funksionin `onViewPostBtnClicked`, dhe nëse klikohet në personin që e ka postuar, hap opsionin që të shkohet tek profili i personit.

Nga screenshot i mesëm mund të shihet një buton rreth ndërprerjes të shoqërisë, njëkohësisht në figurën 14 (në përmbajtje të databazës) mund të shihet një tabelë ‘friends’ – kjo është pasi që në aplikacionin e krijuar është mundësuar krijimi i shoqërisë në format të ngjashëm sikur që e kanë edhe faqet tjera: përdoruesi mund të dërgoj kërkesë për shoqëri, dhe personit që ja dërgon mund të e pranoj apo refuzoj. Krijimi i shoqërisë nuk është i nevojshëm për komunikim apo ndonjë gjë tjetër, mirëpo nëse je miq me personin që dëshiron të flasësh, e ke me lehtë të e gjesh, siç do shpjegohet tek seksioni i komunikimit. Tani për tani, kthehemi tek forumi.

Krijimi postimeve të reja është relativisht thjeshtë i bere dhe po ashtu funksionon përmes fragmenteve. Mirëpo, në këtë rast, na duhen tri fragmente, dhe rrjedhimisht, tri dizajne:

1. `NewPostFragment` me dizajn të aplikuar në `fragment_new_post`,
2. `SetDateTimeFragment` me dizajn të aplikuar në `fragment_setdatetime`,
3. `SetPickPointFragment` me dizajn të aplikuar në `fragment_setpickpoint`.

`NewPostFragment` thjesht jep dy hapësira – një hapësirë për titull, dhe një tjetër për përmbajtje të postimit, si dhe na jep butonin për next, buton ky që në fakt vetëm se i bartë të dhënat e mbledhura në fragmentin e radhës që është fragmenti për përcaktim të datës dhe kohës. Në formë të njëjtë, edhe ky

```
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 formattedMinute = String.format("%02d", minute)
        mPickTimeText!!.setText("$hour:$formattedMinute")
    },
        now[Calendar.HOUR_OF_DAY],
        now[Calendar.MINUTE],
        true
    ).show()
}
```

Kodi 10. Përcaktimi i opsioneve të Material DateTime Picker

fragment ka dy hapësira, mirëpo sapo të preket cilado hapet Material DateTime Picker, që i jep mundësi përdoruesit të zgjedh datën dhe kohën. Së fundmi, kemi fragmentin për caktim të pikave kyçe, i cili prapë ka vetëm dy fusha që përdoruesi të shkruaj pikën prej nga do nisët dhe destinacionin që e ka. Në figurën 15 është paraqitur grafikisht ajo çka u tha deri më tani.

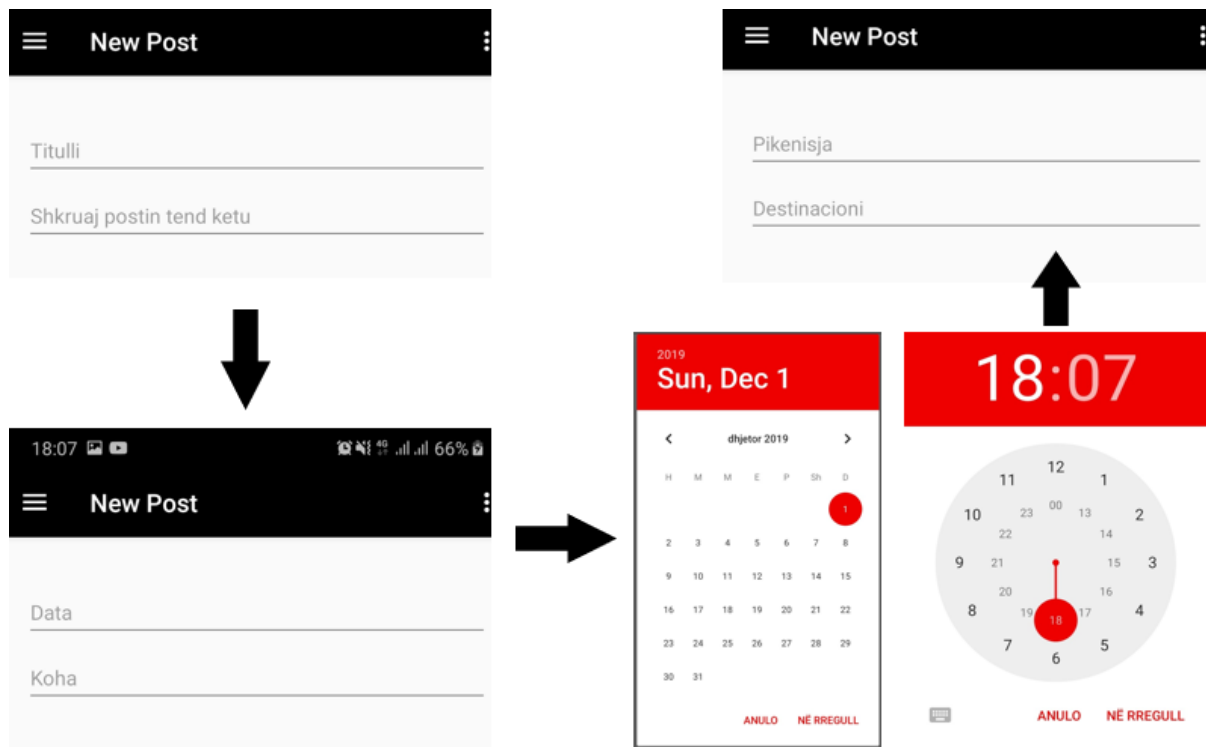


Figura 15. Procesi për krijim të postimit

Material DateTime Picker aktivizohet bazuar në kodin 10.

Sapo që përdoruesi të klikoj posto, do validohen së pari pikënisja dhe destinacioni – se në fakt ekzistojnë dhe mund të shfaqen në hartë, dhe thirret funksioni writeNewPost, i cili merr të dhënat e verifikuara, dhe i vendos në databazë nën tabelën posts.

3.4.1. Regjistrimi

Figura 16. Regjistrimi në aplikacion

Nëse përdoruesi nuk është i regjistruar, qasja nuk do jetë e mundshme, dhe do duhet të regjistrohet, përmes aktivitetit të paraqitur në figurën 16.

Regjistrimi është tentuar të bëhet sa më i thjeshtë, me ndihmesa për përdoruesin që i tregojnë se çfarë të dhëna kërkohen. Për regjistrim mjafton të jepet emri, mbiemri, email adresa dhe fjalëkalimi që do e përdor për qasje. Natyrisht, është përdorur validim i emailës që të sigurohemi se është email reale, mirëpo më tej nuk jemi siguruar se kemi të bëjmë me individ real.

Sapo të i mbush të dhënat, dhe të prek butonin për regjistrim, do ekzekutohet kodi 11.

```
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)
    }
}
```

Kodi 11. Krijimi i përdoruesit të ri

Kurse writeNewUser është definuar si në kodin 12:

```
private fun writeNewUser(userId: String, name: String, email: String,
device_token: String?) {
    val user = User(name, email, device_token)
    FirebaseDatabase!!.child("users").child(userId).setValue(user)
    .addOnCompleteListener { task: Task<Void?> ->
        if (task.isSuccessful) {
            // ...
        }
    }
}
```

Kodi 12. Funksioni për krijim të përdoruesit të ri

Prej procesit të krijimit të përdoruesit u përmenden vetëm këto të dyja pasi që janë pjesa me kryesore.

3.4.2. Profili

Siç u cek më lartë, çdo përdorues e ka profilin e vet. Tek profilet, e shfaqur në figurën 17, kemi disa çështje që duhet cekur, përpos që e kanë mundësinë për krijim shoqërie. Nga figura, lirisht mund të shihet se ekziston edhe mundësia që përdoruesit të vendosin foton e tyre në profil.

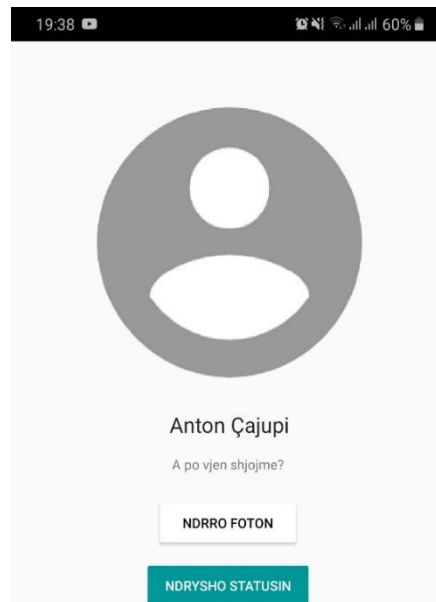


Figura 17. Pamja e profilit nga vetë përdoruesi

Nëse nuk është përcaktuar ndonjë foto paraprakisht, si foto e zakonshme përdoret ajo që shihet në foto. Po të preket butoni për ndërrim të fotos, do hapet mundësia për të zgjedhur foton që dëshiron. Ky veprim delegohet tek vet sistemi operativ pasi që ka aktivitet të veçantë sistemi operativ për importim të një file të caktuar, mjafton në t'i tregojmë se çfarë formati të të dhënave pranojmë – në këtë rast pranojmë vetëm fotografi, andaj type do duhet specifikuar si “image/*” - këtë e kryen kodi 13:

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

Kodi 13. Zgjedhja e fotografisë për profil

Pasi që të zgjidhet një fotografi nga përdoruesi do duhet të prehet në mënyrë që të përshtatet me nevojat tona – pra të jetë formatit 1:1. Siç u specifikua tek libraritë, kjo bëhet përmes Image Cropper, me kodin 14.

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

Kodi 14. Thirrja e aktivitetit për prerje të fotografisë

Pastaj do kompresohet përmes Image Compressor të përcaktuar tek libraritë e përdorura, me kodin 15.

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

Kodi 15. Kompresimi i fotografisë

Dhe ngarkohet në Firebase bucket që ekziston enkas për aplikacionin tonë. Përpos imazheve, profilet karakterizohen edhe me status të tyre. Çdo profil ka statusin e vet, mirëpo ndryshimi i statusit delegohet tek një aktivitet tjetër – tek `ChangeStatusActivity`. Është aktivitet që thjeshtë ndryshon një string në databazë të Firebase, andaj nuk shoh ndonjë arsye të e përfshirë ndonjë pjesë nga kodi apo përbërja e këtij aktiviteti.

Siç u tha me herët, përdoruesit kanë mundësi të krijojnë shoqëri, mirëpo kanë mundësi që edhe të e shkëpusin atë shoqëri – kjo arrihet përmes butonit që shihet në Figurën 17. Ngjashëm me `ChangeStatusActivity`, edhe kjo është vetëm një ndryshim i vogël në databazë, andaj nuk e shoh si të arsyeshme të diskutohet më në thellësi.

3.5. *Komunikimi*

Nëse hapim menynë që zakonisht i referohet si Navigation Drawer, shfaqur në figurën 18, shohim një opsion tjetër përpos të Forumit – bëhet fjale për Bisedat.

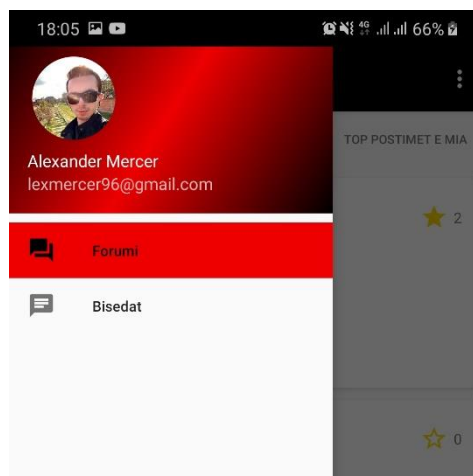


Figura 18. Navigation Drawer

Tek Bisedat, të shfaqura në figurën 19, do të merremi me tri klasa që kanë rëndësi të veçantë:

1. `ConversationFragment`
2. `ChatActivity`
3. `ChatFragment`

Ngjashëm si ForumFragment i cili ishte përgjegjës që të thirrte të gjitha klasat e duhura dhe funksionet e duhura për të shfaqur dhe menaxhuar postimet, ConversationFragment merret me çështje të ngjashme, vetëm se tani ka të bëjë me bisedat që janë bërë, se çka do ndodh kur të preket biseda, dhe për të shfaqur komplet listën e shoqërisë pasi që shoqëria mund të shihet tek Bisedat. Funkcioni kryesor i këtij fragmenti është të bëjë gati bisedat, lexon nga databaza se më kë keni biseduar dhe mesazhi i fundit shfaqet ngjashëm me platforma të ndryshme.

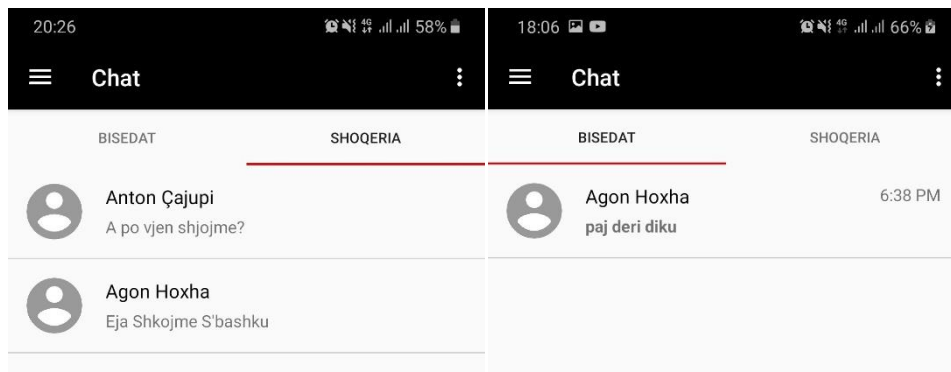


Figura 19. Opsionet tek Biseda

Lista e shoqërisë përmban të gjithë profilet me të cilat përdoruesi ka krijuar marrëdhënie shoqërie. Teksti që shfaqet nën emër tek Shoqëria, është statusi që e kanë përcaktuar.

Nëse preket ndokush në Shoqëria thjesht hapet profili i atij personi dhe mund të shihet para sa ditësh është krijuar miqësia, dhe mund të zgjidhet opsioni për dërgim mesazhi apo për ndërprerje të miqësisë. Nëse preket ndonjë nga bisedat, kalohet drejt në bisede. Për nga ana e kodit, thirret ChatActivity, duke shtuar në Intentin që përdoret për atë thirrje identifikuesin e personit dhe emrin e tij.

Parimisht, merret statusi se nëse ka qenë online kohe të fundit apo jo, duke kontrolluar parametrin e duhur në Firebase. Nëse po, tregon që është online, nëse jo e paraqet vlerën që është e ruajtur në po të njëjtën variabël duke treguar se kur për herë të fundit ishte online.

Pasi që të bëhet kjo, duhen të shkarkohen nga Firebase të gjithë mesazhet që janë dërguar më parë dhe të shfaqen ato, së bashku me kohën kur janë dërguar. Tërë kjo bëhet thjesht me lexim të të dhënave nga databaza dhe nuk ka ndonjë funksion të veçantë, andaj nuk do shtjellohet më në detaje.

Për të dërguar mesazhe, merret funksioni i posaçëm sendMessage, i cili pasi që sigurohet që nuk është mesazh i thatë, procedon me marrje të qasjes në direktoriumet e duhura.

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

Kodi 16. Përbërja e komponentit të përdorur për mesazhe

Objekti messageMap, i krijuar në kodin 16, është i definuar ashtu që si index, të këtë një String, dhe si vlerë të tij mund të këtë çfarëdo formati të të dhënave, përfshirë edhe 'null'. Futja në databazë bëhet paksa me ndryshe se sa rastet e tjera, sipas kodit 17. Kjo e tera rezulton që të mundësohet komunikimi mes përdoruesve, siç është shfaqur në figurën 20.

```
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.TIMESTAMP)
mDatabase!!.child("chat").child(mChatUser!!).child(uid)
    .child("seen").setValue(false)
mDatabase!!.child("chat").child(mChatUser!!).child(uid)
    .child("timestamp").setValue(ServerValue.TIMESTAMP)
mDatabase!!.updateChildren(messageUserMap) { databaseError: DatabaseError?,
    _: DatabaseReference? -> if (databaseError != null) {Log.d(TAG,
    databaseError.message)}}}
```

Kodi 17. Kodi për vendosje të mesazheve në databazë

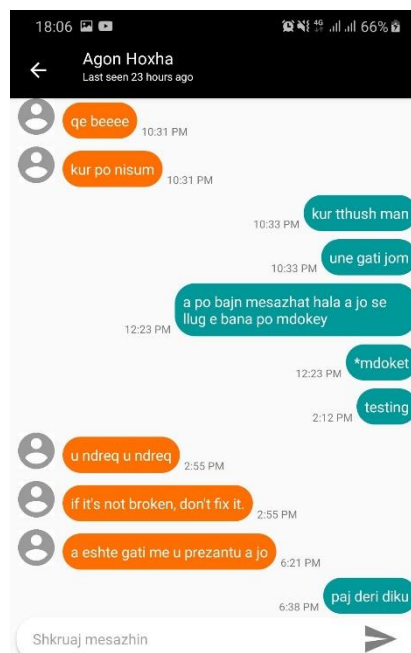


Figura 20. Pamja e Bisedës

4. Diskutime dhe konkluzione

Qëllimi i këtij punimi ishte krijimi i aplikacionit për menaxhim të bashkëudhëtimit në taksi, në sistemin operativ Android. Konsideroj se kjo është bërë me sukses, edhe pse ka hapësire për zhvillim të mëtejshëm, si në aspekt të mundësisit të komunikimit përmes mënyrave tjera si me zë apo me pamje video, edhe pse është e diskutueshme se a do ishte vërtetë e nevojshme për një aplikacion të tillë. Gjithashtu, do ishte vështirë të implementohej. Një tjetër drejtim në të cilin do mund të behet një zhvillim është nga ana e navigimit – mënyra se si është tani, është tejet bazike mirëpo plotëson nevojat bazike. Sidoqoftë, qëllimi është përmbushur, dhe aplikacioni mund të publikohet për përdorim të lirë. Aplikacioni i krijuar do të mundësoj një zgjidhje më të mire se sa opsionet e tanishme, si grupet në Facebook apo udhëtimi duke pritur në rrugë që të vij një automobil. Në të njëjtën kohë, siç është demonstruar, është mundësuar komunikimi mes krijuesit të postimit dhe pasagjereve potencial, mirëpo edhe komunikimi mes pasagjereve, jo vetëm përmes interaksionit në komente, por edhe në komunikim direkt.

Moduli i Forumit, që njëkohësisht është moduli kryesor, tani ju paraqet përdoruesve një platformë të veçantë. Moduli i Bisedave, ju paraqet përdoruesve të aplikacionit të zhvilluar, një mënyrë për komunikim. Moduli i Navigimit, që nuk është edhe aq i përfillur në këtë punim pasi që nuk është edhe aq i zhvilluar, iu ndihmon nëse kanë problem për gjetje të pikënisjes.

Është marre nën konsiderate edhe krijimi i aplikacionit njëkohësisht në sistemin operativ iOS, mirëpo për shkak të temës të limituar në sistemin operativ Android dhe kohës të limituar për realizim, kjo nuk është përfillur.

5. Shtesat

5.1. Lista e shkurtesave

IoT – Internet of Things

TV – Television

PC – Personal Computer

LLC – Limited Liability Company

AOSP – Android Open Source Project

API – Application Programming Interface

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

JVM – Java Virtual Machine

WORA – Write Once, Run Anywhere

JIT – Just in Time

IDE – Integrated Development Environment

NPE – Null Pointer Exception

XML – Extended Markup Language

SDK – Software Development Kit

JSON – JavaScript Object Notation

kB – Kilo Byte

MB – Mega Byte

GB – Giga Byte

HTTP – Hypertext Transfer Protocol

GPS – Global Positioning System

5.2. Lista e figurave

Figura 1. Përdorimi i aplikacioneve për bashkë udhëtim në vend të vozitjes [1]	1
Figura 2. Organizimi i bashkë udhëtimit ne Facebook [3]	2
Figura 3. Arkitektura e Android [10]	3
Figura 4. Shpërndarja e versioneve të Android [11]	5
Figura 5. Etapat e jetës të aktivitetit [12]	5
Figura 6. Shfrytëzimi i shërbimit në Android	6

Figura 7. Funksionimi i Broadcast Receiver [13]	6
Figura 8. Java Virtual Machine [17]	8
Figura 9. Benchmarks mes Java dhe Kotlin [23]	12
Figura 10. Konvertimi i Librarive [26]	16
Figura 11. Pamje high-level e databazës për aplikacionin	17
Figura 12. Hapat e parë në aplikacion	20
Figura 13. Pamja kryesore	20
Figura 14. Hapja e profilit, apo postimi më i detajuar	21
Figura 15. Procesi për krijim të postimit	23
Figura 16. Regjistrimi në aplikacion	23
Figura 17. Pamja e profilit nga vetë përdoruesi	25
Figura 18. Navigation Drawer	26
Figura 19. Opsionet tek Biseda	27
Figura 20. Pamja e Bisedës	28

5.3. Lista e tabelave

Tabela 1. Versionet e Android [11]	4
--	---

5.4. Lista e pjesëve të kodit

Kodi 1. Zgjerimi në Kotlin	9
Kodi 2. Shembull kodit në Java për try dhe catch	9
Kodi 3. Mënyra e krijimit të përjashtimit në Kotlin	10
Kodi 4. Shkaktim i një NullPointerException në Java	10
Kodi 5. Kodi ekuivalent me Kodin 4, në Kotlin	10
Kodi 6. Klasa Personi me getters dhe setters për variablën ‘name’	11
Kodi 7. Ekuivalenti i Kodit 6, në Kotlin	11
Kodi 8. Krijimi i lidhjes me Firebase	21
Kodi 9. Hapja e Profilit	21
Kodi 10. Përcaktimi i opsioneve të Material DateTime Picker	22
Kodi 11. Krijimi i përdoruesit të ri	24
Kodi 12. Funksioni për krijim të përdoruesit të ri	24
Kodi 13. Zgjedhja e fotografisë për profil	25
Kodi 14. Thirrja e aktivitetit për prerje të fotografisë	25
Kodi 15. Kompresimi i fotografisë	26
Kodi 16. Përbërja e komponentit të përdorur për mesazhe	27
Kodi 17. Kodi për vendosje të mesazheve në databazë	28

6. Bibliografia

- [1] Përdorimi i aplikacioneve për bashkëudhëtim në vend të udhëtimit, Online: <https://www.kristensenlaw.com/blog/2018/08/Systemic-liabilities-within-rideshare-programs.shtml>, qasur më: Dhjetor, 2019
- [1] Statistikat rreth përdorimit të Uber, Online: <https://www.businessofapps.com/data/uber-statistics>, qasur më: Dhjetor, 2019
- [2] Statistikat rreth përdorimit të Lyft, Online: <https://www.businessofapps.com/data/lyft-statistics>, qasur më: Dhjetor, 2019
- [3] Grup në Facebook për bashkëudhëtim, Online: <https://www.facebook.com/groups/343190843152164> –qasur më: Dhjetor, 2019
- [4] Zhvillim i aplikacioneve Android IoT pajisje, Online: <https://developer.android.com/things>, qasur më: Dhjetor, 2019
- [5] Zhvillimi i aplikacioneve për Android TV, Online: <https://developer.android.com/tv>, qasur më: Dhjetor, 2019
- [6] Version i Android për PC, Online: <https://www.android-x86.org/>, qasur më: Dhjetor, 2019
- [7] Blerja e Android nga Google, Online: <https://www.cnet.com/news/google-buys-android>, qasur më: Dhjetor, 2019
- [8] Publikimi i Android source code, Online: <https://android.googlesource.com>, qasur më: Dhjetor, 2019
- [9] Plane për rritje në ndërlidhshmërinë të Android me Linux, Online: <https://arstechnica.com/gadgets/2019/11/google-outlines-plans-for-mainline-linux-kernel-support-in-android/>, qasur më: Dhjetor, 2019
- [10] Arkitektura e sistemit operativ Android, Online: [https://en.wikipedia.org/wiki/Android_\(operating_system\)#Software_stack](https://en.wikipedia.org/wiki/Android_(operating_system)#Software_stack) qasur më: Dhjetor, 2019
- [11] Versionet e sistemit operativ Android, https://wikivisually.com/wiki/Android_version_history, qasur më: Dhjetor, 2019
- [12] Ubaya Huda, Design of Prototype Payment Application System With Near Field Communication (NFC) Technology based on Android, Online: https://www.researchgate.net/publication/274314022_Design_of_Prototype_Payment_Application_System_With_Near_Field_Communication_NFC_Technology_based_on_Android, qasur më: Dhjetor, 2019
- [13] Hyrje në Android, Online: <https://www.educba.com/introduction-to-android/> qasur më: Dhjetor, 2019
- [14] Statistika mbi përdorim të Kotlin, Online: <https://www.appbrain.com/stats/libraries/details/kotlin/kotlin>, qasur më: Dhjetor, 2019
- [15] Web faqja e Oracle, Online: <https://www.oracle.com/java>, qasur më: Dhjetor, 2019
- [16] Caqet për zhvillim të mëturshëm të Java, Online: <https://www.oracle.com/technetwork/java/intro-141325.html> –qasur më: Dhjetor, 2019
- [17] Sqarime mbi Java Virtual Machine, Online: https://www.tutorialspoint.com/java_virtual_machine/java_virtual_machine_tutorial.pdf, qasur më: Dhjetor, 2019

- [18] Kotlin deklarohet si gjuhë zyrtare për zhvillim të aplikacioneve në Android, Online: <https://techcrunch.com/2017/05/17/google-makes-kotlin-a-first-class-language-for-writing-android-apps>, qasur më: Dhjetor, 2019
- [19] Intervista me krijuesin e Kotlin, Online: <https://zeroturnaround.com/rebellabs/jvm-languages-report-extended-interview-with-kotlin-creator-andrey-breslav>, qasur më: Dhjetor, 2019
- [20] Supozime mbi të ardhmen e zhvillimit të aplikacioneve në Android, Online: <https://www.netsolutions.com/insights/why-kotlin-is-the-future-of-android-app-development>, qasur më: Dhjetor, 2019
- [21] Zgjerimet në Kotlin, Online: <https://kotlinlang.org/docs/reference/extensions.html>, qasur më: Dhjetor, 2019
- [22] Implementimi i përjashtimeve në Kotlin, Online: <https://code.tutsplus.com/tutorials/kotlin-from-scratch-exception-handling--cms-29820>, qasur më: Dhjetor, 2019
- [23] Performance Evaluation of Kotlin and Java On Android Runtime, Patrik Schwermer, Online: <http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1231573&dswid=-682>, qasur më: Dhjetor, 2019
- [24] Dallime mes Java dhe Kotlin, Online: <https://kotlinlang.org/docs/reference/comparison-to-java.html>, qasur më: Dhjetor, 2019
- [25] Publikimi i Android Studio, Online: <https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>, qasur më: Dhjetor, 2019
- [26] Shpjegime rreth AndroidX, Online: <https://developer.android.com/jetpack/androidx>, qasur më: Dhjetor, 2019
- [27] Statistika rreth përdorimit të Google Maps shërbimeve, <https://themanifest.com/app-development/popularity-google-maps-trends-navigation-apps-2018>, qasur më: Dhjetor, 2019
- [28] Statistika mbi përdorimin e Firebase në aplikacione, <https://www.appbrain.com/stats/libraries/details/firebase/firebase>, qasur më: Dhjetor, 2019
- [29] GeoFire repository, Online: <https://github.com/firebase/geofire-android>, qasur më: Dhjetor, 2019
- [30] SquareUp Picasso repository, Online: <https://square.github.io/picasso>, qasur më: Dhjetor, 2019
- [31] Material DateTime Picker repository, Online: <https://github.com/wdullaer/MaterialDateTimePicker>, qasur më: Dhjetor, 2019
- [32] Image Croppër repository, Online: <https://github.com/ArthurHub/Android-Image-Cropper/wiki>, qasur më: Dhjetor, 2019
- [33] CircleImageView repository, Online: <https://github.com/hdodenhof/CircleImageView>, qasur më: Dhjetor, 2019
- [34] Image Compressor repository, Online: <https://github.com/zetbaitsu/Compressor>, qasur më: Dhjetor, 2019
- [35] OkHttp repository, <https://square.github.io/okhttp>, qasur më: Dhjetor, 2019
- [36] JODA-Time website, Online: <https://www.joda.org/joda-time>, qasur më: Dhjetor, 2019
- [37] Retrofit repository, Online: <https://square.github.io/retrofit>, qasur më: Dhjetor, 2019