UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U NOVOM SADU

Igor Medenica

Jedno rešenje za prenos audio-video podataka preko P2P protokola

MASTER RAD

Novi Sad, 2017.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		Игор Меденица	
Ментор, **МН**:		Проф. др Никола Тескић	
Наслов рада, **НР**:		Једно решење за пренос аудио-видео података преко П2П протокола	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2017**	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)			
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Аудио-видео пренос, Јава, Андроид, мултимедијални садржај, П2П, заштита података	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Овај рад приказује једно решење снимања, преноса и репредукције мултимедијалног садржаја на Андроид мобилном телефону.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:		21.12.2017	
Чланови комисије, **КО**:	Председник:	доц. др Милан Бјелица	
	Члан:	проф. др Горан Стојановић	Потпис ментора
	Члан, ментор:	проф. др Никола Теслић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Igor Medenica	
Mentor, **MN**:		PhD Nikola Teslić	
Title, **TI**:		One software solution for transferring audio-video data through p2p protocol	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2017	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)			
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		video-audio transfer, Java, Android, Multimedia, P2P, data protection	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper presents a solution for recording, transferring, and reproducing multimedia content on an Android mobile phone.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		21.12.2017	
Defended Board, **DB**:	President:	Phd Milan Bjelica	
	Member:	Phd Goran Stojanović	Menthor's sign
	Member, Mentor:	PhD Nikola Teslić	

Zahvalnost

Zahvalnost na podršci dugujem mentoru, prof. dr Nikoli Tesliću, redovnom profesoru Fakulteta tehničkih nauka Univerziteta u Novom Sadu.

Podjednaku zahvalnost na korisnim informacijama, sugestijama i savetima dugujem Nikoli Vraniću.

U Novom Sadu, decembar 2017. godine Igor Medenica

SADRŽAJ

1. Uvod 8

2. Teorijske osnove 11

2.1 Opis platforme koja je korisćena 11

2.2 Point-to-Point 19

2.2.1 Podela prema načinu ispisivanja podataka 20

2.2.1.1 Centralizovane (hibridne) p2p mreže 20

2.2.1.2 Decentralizovane p2p mreže 20

2.2.2 Podela prema načinu spajanja učesnika 20

2.2.2.1 Strukturirane p2p mreže 21

2.2.2.2 Nestrukturirane peer-to-peer mreže 21

2.3 TCP/IP 21

3. Osnovni koncepti sistema 22

3.1 Uslužilac 24

3.2 Klijent 24

3.3 Baza podataka 25

3.4 Šema baze podataka 25

3.5 Algoritam za video i audio prenos 26

3.6 Algoritam za komunikaciju 27

3.6.1 Logovanje i registracija korisnika 27

3.6.2 Prikaz svih korisnika i svih kamera 28

3.6.3 Prenos tekstualne datoteke 29

3.6.4 Prenos video datoteke 31

4. Implementacija 32

4.1 Serverski deo ... 32
4.1.1 Slanje lokalne IP adrese baznom modulu ... 33
4.1.2 Snimanje i kreiranje video datoteka ... 33
4.1.3 Kreiranje tekstualne datoteke ... 34
4.1.4 Response datoteke ... 34
4.2 Klijentski deo ... 34
4.2.1 Komunikacija sa baznim modulom ... 35
4.2.2 Zahtevanje tekstualnog fajla od servera ... 35
4.2.3 Zahtevanje video datoteka od servera i njihovo prikazivanje ... 36
4.2.3.1 Učitavanje datoteke titla ... 37
4.2.3.2 Neprekidno puštanje video datoteke ... 37
4.2.3.3 Neprekidna reprodukcija niza video zapisa ... 37
4.2.3.4 Napredne kompozicije ... 37
4.3 Bazni deo ... 38
4.4 Ostale funkcije aplikacije ... 40
4.4.1 Registrovanje korisnika ... 40
4.4.2 Logovanje korisnika ... 40
4.4.3 Spisak svih korisnika ... 41
4.4.4 Zaštita podataka ... 42
4.4.5 Prenos podataka ... 42
4.4.5.1 json_encode() ... 43
4.4.5.2 json_decode() ... 43
5. Testiranje ... 44
5.1 Opis radnog okruženja ... 44
5.2 Komunikacija između Android aplikacije i baze podataka ... 45
5.3 Rad sa kamerom ... 46
5.4 Rad Http servera ... 47
6. Zaključak ... 49
7. Literatura ... 51

SPISAK SLIKA

Slika 1. Slojevi Android platforme 13

Slika 2. Životni ciklus Android aktivnosti 14

Slika 3. Životni ciklus Android servisa 16

Slika 4. Šema point-to-point mreže 19

Slika 5. Prikaz hibridne point-to-point mreže 20

Slika 6. TCP/IP i OSI model 21

Slika 7. Osnovni prikaz komunkacije 22

Slika 8. SQL dijagram za MySQL bazu podataka 26

Slika 9. Dijagram algoritma za video i audio prenos 27

Slika 10. Primer dijagrama za logovanje korisnika 28

Slika 11. Primer dijagrama za prikaz korisnika 29

Slika 12. Primer dijagrama za prenos playlist.txt 30

Slika 13. Primer dijagrama za prenos MPEG_4 video datoteka 31

Slika 14. Tabele users, cameras i log u MySQL bazi podataka 39

Slika 15. Forma za registrovanje korisnika 40

Slika 16. Forma za logovanje korisnika 41

Slika 17. Ispis svih registrovanih korisnika 41

Slika 18. Enkriptovani zapisi u tabeli korisnika 42

Slika 19. Testovi komunikacije između Android aplikacije i baznog modula.....46

Slika 20. Prikaz svih testova za rad sa kamerom.....47

Slika 21. Testovi provere rada http servera.....48

SPISAK TABELA

Tabela 1. Slojevi Android platform 12

Tabela 2. Metode životnog ciklusa Aktivnosti 15

Tabela 3. Metode životnog ciklusa Servisa 18

Tabela 4. Redosled komunikacije između Android uređaja uslužioca 23

Tabela 5. Redosled komunikacije Android uređaja klijenta sa ostalim delovima sistema 23

Tabela 6. Funkcije na korisničkom delu aplikacije 24

Tabela 7. Tabela prikaza entiteta u bazi podataka 25

Tabela 8. Delovi sistema za prenos video i audio sadržaja 32

Tabela 9. Tabela prikaza funkcija serveskog modula 33

Tabela 10. Tabela prikaza funkcija klijentskog dela 35

Tabela 11. Spisak PHP skripti koje se nalaze u baznom delu 39

Tabela 12. Tabela testova komunikacije Android aplikacije i baze podataka 45

Tabela 13. Tabela testova rada sa kamerom 46

Tabela 14. Tabela testova rada http servera 48

SKRAĆENICE

P2P - *peer to peer*, model komunikacije preko interneta

CPU - *Central Processor Unit*, Centralni procesoor

JDK - *Java Development Kit*

DVK - *Dalvik Virtual Machine*

IDE - *Integrated development environment*, integrisano razvojno okruženje

MPEG - *Moving Picture Experts Group*

TCP - *Transmission Control Protocol*

1. Uvod

Multimediji ili multimedija (eng. *multimedia*), sadržaj koji upotrebljava kombinaciju različitih oblika sadržaja. Stoga se razlikuju od medija koji upotrebljavaju samo rudimentarne kompjuterske prikaze kao npr. isključivo tekstove ili tradicionalne oblike štampanog ili ručno proizvedenog materijala. Multimediji uključuju kombinaciju teksta, zvuka, slika, animacije, videa. Koriste se tamo gde se čovek pojavljuje kao posmatrač ili korisnik elektronske informacije bilo koje vrste. Oni kod korisnika poboljšavaju, pojačavaju i pospešuju razumevanje i prijem informacija. Dobro dizajnirana i realizovana multimedija može da bude zabavna i korisna za dalju upotrebu.

Za prenos multimedijalng sadržaja potreban je sistem koji omogućava prenos podataka sa odgovarajućeg izvora. Ako se sadržaj istovremeno prenosi i prikazuje, neophodno je da sistem ima obezbeđen veći prostor za skladištenje podataka i propusni opseg, odnosno mogućnost prenosa velikog broja bita u jedinici vremena. Tada govorimo o sistemima koji vrše prenos u realnom vremenu. To je takozvani uživo (eng: *live*) prenos, odnosno direktan prenos preko mreže. Pri zahtevanju sadržaja koji je većeg kvaliteta i zahteva veću propusnu moć pri manjim brzinama sadržaj prvo se zapisuje na strani prijemnika, a potom prikazuje. Ovakvi sistemi rade van realnog vremena ili u odloženom vremenu.

Pomoću Point-to-Point tehnologije obavlja se direktna komunikaciju između dva uređaja. Pomenuta tehnologija se primenjuje kod BitTorrent [1] i Skype [2]. Za ovu tehnologiju nije potreban server za skladištenje ili distribuciju datoteka. Komunikacija se vrši između dva ili više uređaja što je prednost zbog smanjivanja komunikacije sa udaljenim serverom. P2P u ovom slučaju Point-to-Point sistem se upotrebljava za anonimno usmeravanje saobraćaja. Kao što je pokazano u radovima [3][4] arhitektura otežava dva čvora na različitim privatnim mrežama da međusobno kontaktiraju, što je često važno za komunikacijske protokole P2P koji se koriste u

aplikacijama poput telekonferencije i online igara. Potrebno da takve protokole učinimo da rade bez problema potrebno je prisustvo NAT-a. Jedna od najefikasnijih metoda uspostavljanja komunikacije između domaćina na različitim privatnim mrežama poznata je kao "probijanje rupa" (eng. *hole punching*). Ova tehnika se široko koristi u aplikacijama baziranim na UDP-u, ali u suštini istu tehniku radi i za TCP. Za razliku od onoga što njegovo ime može predložiti, probijanje rupa ne ugrožava sigurnost privatne mreže. Umesto toga, probijanje rupa omogućava aplikacijama da funkcionišu u okviru osnovne bezbednosne politike većine NAT-ova, što efektivno signalizira NAT-u na putu da se sesije komunikacije između peer-to-peer-a "pozivaju" (eng. *solicited*) i na taj način treba prihvatiti. Probijanje rupa za UDP i TCP i opisuje ključne aspekte aplikacije i ponašanja NAT-a koji prave udaranje rupa. Nažalost, tehnika prelazaka ne radi sa svim postojećim NAT-om, jer ponašanje NAT-a nije standardizovano. U ovim radovima predstavljeni su neki eksperimentalni rezultati koji vrednuju podršku za udaranje rupa u trenutnim NAT-ovima. Podaci proizilaze iz rezultata koje korisnici šalju putem Interneta tako što pokreću alatku "NAT Check" preko različitih NAT-ova različitih proizvođača. Iako su tačke podataka prikupljene od "samo-selektivne" (eng. *self-selecting*) korisničke zajednice i možda nisu reprezentativne za istinsku distribuciju NAT implementacija raspoređenih na Internetu.

Izbegavanjem navedenih problema u ovom radu je dodata udaljena baza podataka u kojoj se zapisuju lokalne ip adrese. Svaki multimedijalni sadržaj kao što su zvuk, slika, grafika, tekst se pretvaraju u digitalni oblik, odnosno prestavljeni su nizom brojeva.

Postoji veliki broj primera upotrebe ove tehnologije prilikom korišćenja sistema za video nadzor, kao i razmene video i audio sadržaja u realnom vremenu. Socijalne mreže su nedavno uvrstile u svoju primenu ovaj vid prenosa podataka[5].

U radovima [6] [7] opisana su tri važna problema u prenosu videa u realnom vremenu. To su dinamičan propusni opseg, kašnjenje i gubici. U našem istraživanju pokušavamo da rešimo problem prenosa video i audio sadržaja u lokalnoj mreži između dva uređaja. Cilj našeg istraživanja je smanjenje gubitaka u realnom vremenu koji nastaju prilikom video i audio prenosa između dva uređaja, poboljšavajući sigurnost u razmeni podataka. U radu je napravljena Android aplikacija koja ima klijentsku i uslužnu stranu i udaljenu bazu podataka. Komunikacija između klijenata i uslužioca vrši se pomoću tekstualne liste. Putanje do video fajlova na uslužiočevom uređaju koje se nalaze u ovom fajlu omogućavaju klijentu da reprodukuje pomoću ExoPlayer-a na svom uređaju. Ovaj rad se sastoji od šest poglavlja.

U poglavlju 2 opisana su radna okruženja koja su korišćenjena, opis platforme Android, kao i životni ciklus aktivnosti i servisa.

U poglavlju 3 naveden je opis rešenja problema koji je prikazan dijagramima, kao i pronalaženje ključnih problema.

U poglavlju 4, dat je detaljan opis implementacije softverskog rešenja za Android uređaje i rad sa udaljenom bazom podataka pomoću PHP skript jezika.

Poglavlje 5 se bavi rezultatima merenja ranije opisanim metrikama za ocenu kvaliteta predloženog rešenja.

U poglavlju 6, dat je zaključak istraživanja sa predloženim rešenjima daljeg razvoja.

2. Teorijske osnove

2.1 Opis platforme koja je korisćena

Android predstavlja programsku podršku namenjenu konkretno za mobilne uređaje, koji u sebi uključuje operativni sistem, sprežni sloj za povezivanje Java i C programskog jezika i pojedine aplikacije koje su dostupne korisnicima. U današnje vreme ovo je najrasprostanjeniji sistem za mobilne uređaje, ali su sve češće dostupna određena prilagođenja operativnog sistema Android platfome za druge uređaje koji rade u realnom vremenu, poput: mobilnih telefona, tablet računara, televiziskih prijemnika i drugih [8]. Sistem je slojevito organizovan i sastoji se iz sledećih slojeva :

Slojevi Android sistema	Opis
Sloj jezgra	Obezbeđuje prava pristupa, rad sa sistemskim datotekama, rad sa procesima
Radni sloj	Obezbeđuje grafičku podršku, podrška multimedijalnim funkcijama, sigurnost i skladištenje višim slojevima sistema
Sloj programskih okvira	Obezbeđuje Aplikativnom sloju upotrebu funckija iz Radnog sloja

Aplikativni sloj	Najviši sloj na kome se nalaze korisničke aplikacije

Tabela 1. Slojevi Android platform

Sloj jezgra (eng. *Kernel layer*) je zadužen za obezbeđivanje prava pristupa, rukovanje sistemom datoteka, rukovanje fizičkim i logičkim resursima, rukovanje procesima. Ovaj sloj se direktno oslanja na hardver i baziran je na Linux-u uz modifikacije koje obezbeđuju dodatne funkcionalnosti nephodne za ispravan rad ostatka Androida.

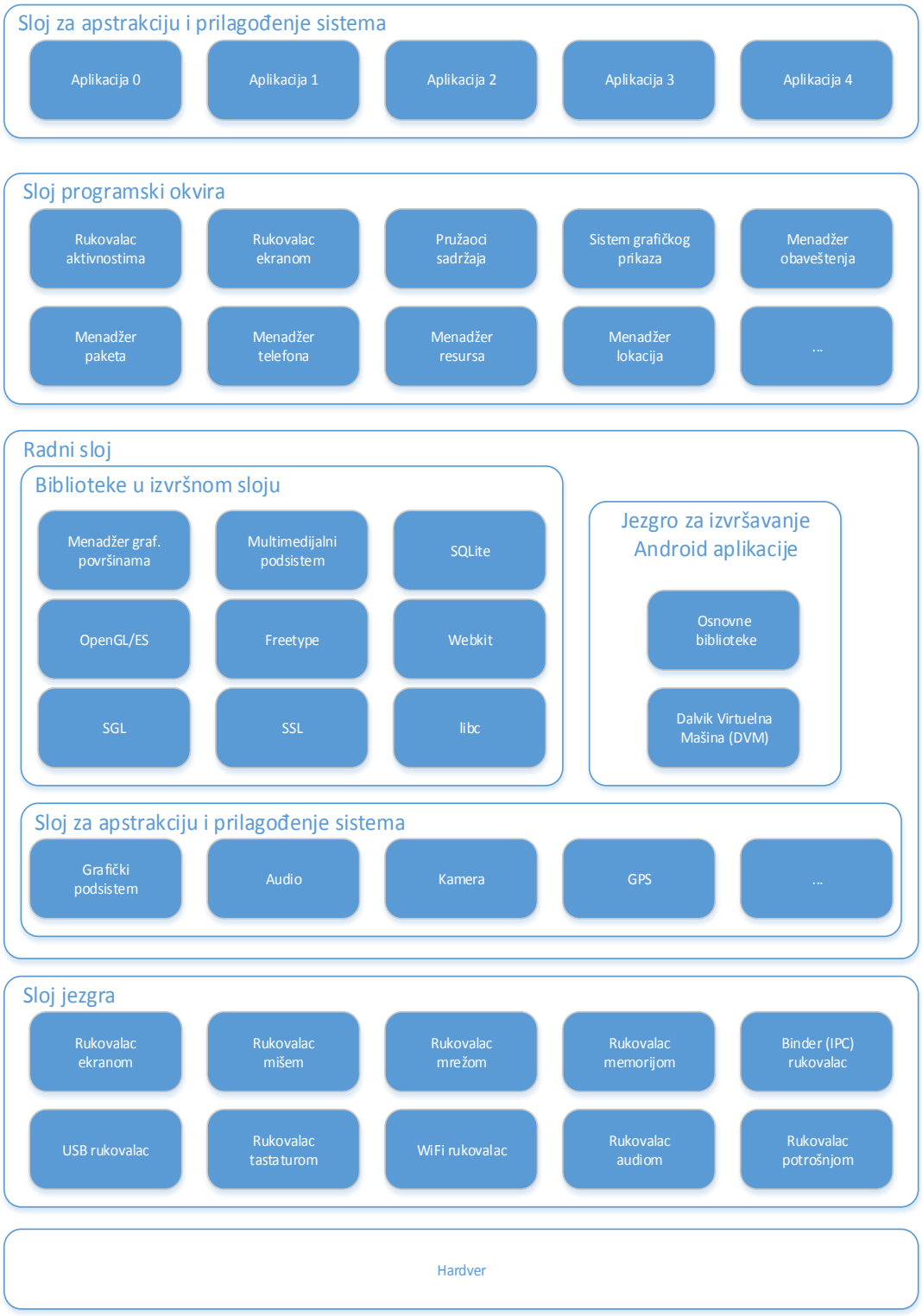
Radni sloj (eng. *Runtime layer*) se sastoji od sloja za apstrakciju i prilagođenje sistema, skupa biblioteka u izvršnom obliku i jezgra za izvršavanje Android aplikacija. Značajne osobine svih biblioteka u ovom sloju je direktno izvršavanje na ciljnoj platformi (end. *native code*) i obezbeđivanje usluga na višim slojevima kao što je grafička podrška, podrška multimedijalnim funkcijama, sigurnost, skladištenje podataka, itd. Sve biblioteke se smatraju sistemskim i Android aplikacije nemaju direktan pristup njima. Izvršavanje aplikacije je realzovano pomoću virtualne mašine koja je u stanju da izvrši bajt-kod Dalvik virtuelne mašine. Dalvik virtuelna mašina se koristi u Android sistemima do verzije 5.0, dok ART mehanizam u novijim verzijama. Dalvik virtuelna mašina prevodi bajt-kod u mašinski kod tokom izvršavanja aplikacije (eng. *Just In Time*), dok ART mehanizmom bajt-kod se prevodi u mašinski prilikom instalacije aplikacije. ART obezbeđuje bolje performanse, ali zahteva veći prostor za skladištenje.

Sloj programskih okvira je sloj između aplikativnog sloja i radnog sloja koji omogućava upotrebu funkcija iz radnog sloja u aplikacijama. Mehanizam koji se koristi za interakciju sa izvršnim slojom je JNI (eng. *Java Native Interface*). Cela programska sprega (API) spada u ovaj sloj.

Aplikativni sloj predstavlja najviši sloj na kome se nalaze korisničke aplikacije. Android aplikacije razvijaju se u najčešćće u programskom jeziku Java.

Sledećom slikom (Slika 1.) prikazana je opšta arhitektura Androida. Ovim dijagramom su prikazani svi slojevi sistema.

Android aplikacije omogućavaju vizuelizaciju funkcionalnosti celokupnog sistema i sakriva implementaciju ostalih slojeva i krajnji korisnik nema uvid u način njihovog funkcionisanja.

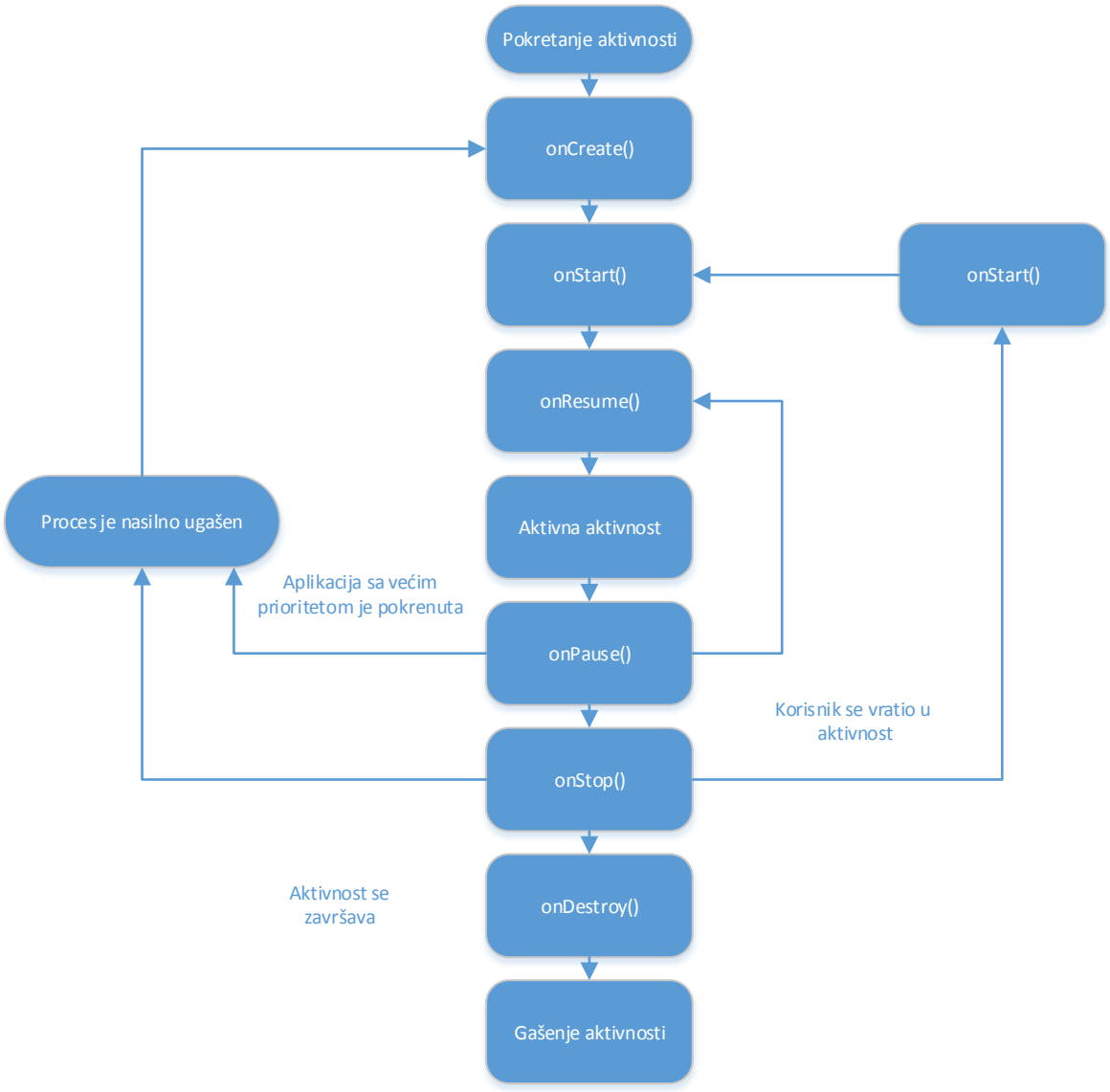


Slika 1. Slojevi Android platforme

Najvažniji gradivni blokovi svake Android aplikaciju su aktivnost (eng. Activity) i servis (eng. Service). Svaki gradivni blok ima posebnu namenu i funkcionalnost, a zajedno određuju opšte ponašanje celokupne aplikacije.

Aktivnost je blok u okviru aplikacije koji omogućuje interakciju sa krajnjim korisnikom, gde korisnik zadaje odgovarajuće akcije putem podržanih periferija kao što su dodir, miš, tastatura, itd. Svaka aktivnost sadrži prozor koji se koristi za iscrtavanje korisničke sprege.

Prozor obično zauzima ceo ekran, ali postoji mogućnost da bude postavljen u ravni iznad drugih prozora (eng. *Z-ordering*). Android aplikacija se sastoji iz više povezanih aktivnosti, o čemu će detaljnije biti reči u poglavlju 4.



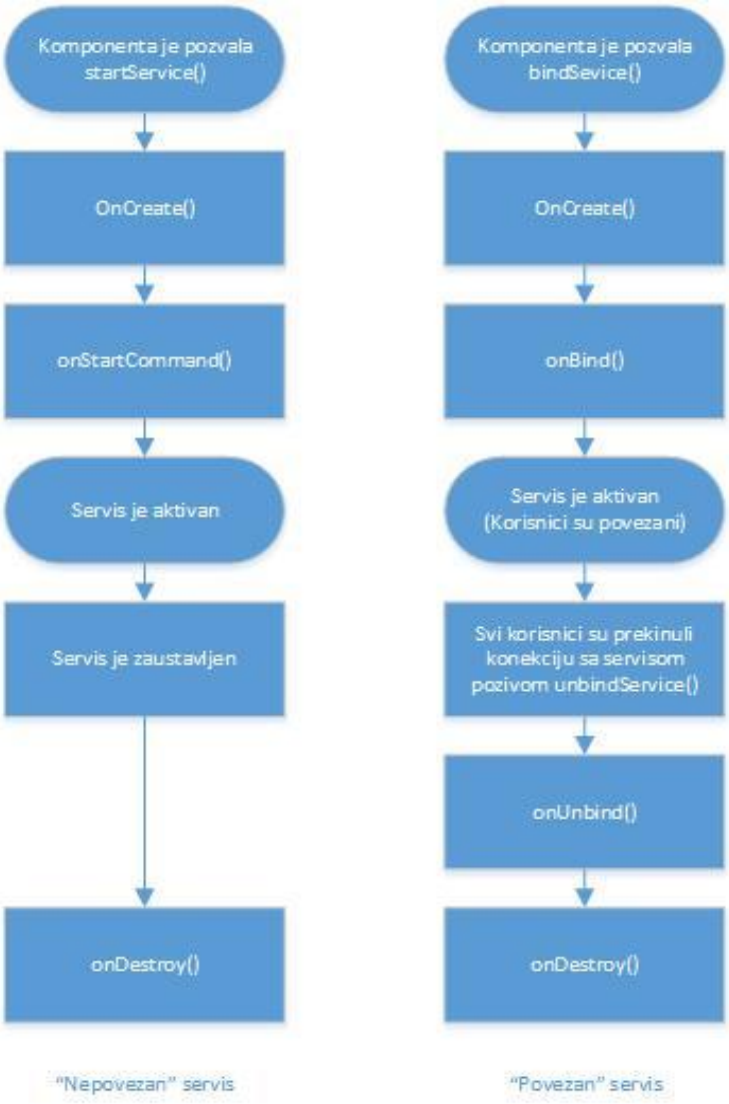
Slika 2. Životni ciklus Android aktivnosti

Aktivnost se definiše nasleđivanjem klase Activity, koja je definisana u okviru Android razvojnog okruženja i implementiranjem određenih povratnih metoda koje definišu životni ciklus svake aktivnosti (*Slika 2*).

U sledećoj tabeli prikazane su metode kreiranja, restartovanja, pokretanja, zaustavljanja aktivnosti.

Metode	Opis
onCreate()	Poziva se kada se aktivnost prvi put pokreće, tj. stvara. U okviru ove metode se obično vrše inicijalizacije svih važnih polja i grafičkih komponenti.
onRestart()	Poziva se pre ponovnog pokretanja, nakon prethodnog zaustavljanja aktivnosti. Uvek je praćena pozivom onStart() metode.
onStart()	Poziva se pre nego što aktivnost postane vidljiva krajnjem korisniku.
onResume()	Poziva se pre nego što aktivnost započne interakciju sa korisnikom. Uvek je praćena onPause() metodom.
onPause()	Poziva se pre završavanja ili pauziranja aktivnosti. Obično se koristi za skladištenje važnih podataka i određenih stanja. Mogu je slediti pozivi metoda onResume() ako će se aktivnost ponovo pokrenuti, ili onStop() ako je aktivnost u potpunosti završena.
onStop()	Poziva se kada aktivnosti nije vidljiva za korisnika.
onDestroy()	Poziva se pre zaustavljanja i uništavanja aktivnosti. Ujedno predstavlja poslednji poziv koji aktivnost dobije u toku životnog ciklusa.

Tabela 2. Metode životnog ciklusa Aktivnosti



Slika 3. Životni ciklus Android servisa

Android servis je komponenta koja se, za razliku od Android aktivnosti, izvršava u pozadini i služi za izvršavanje složenih i dugih operacija ili za interakciju sa drugim procesima (eng. IPC – Inter Process Communication). Servis može posedovati dva stanja, tj. dva režima rada:

- pokrenut servis - Servis je pokrenut kada ga neka aplikativna komponenta pokrene pozivom startService() metode. Kada je jednom pokrenut, servis se izvršava u pozadini, iako komponenta koja ga je pokrenula prestane da bude aktivna. Obično pokrenut servis izvršava jednu operaciju i nema povratnu vrednost.
- povezan servis – Nakon poziva bindService() metode sa strane neke aplikativne komponente, omogućuje se komunikacija sa pokrenutim Android servisom. Na

ovaj način realizuje se sprega usluživač - korisnik, koja omogućava različitim komponentama da komuniciraju sa samim servisom. Ovim načinom omogućeno je slanje zahteva, prijem rezultata, pa čak i komunikacija sa drugim procesima u okviru operativnog sistema. Ovakav servis se izvršava sve dok postoji veza sa aplikativnom komponentom. Više različitih komponenti može biti povezano na isti servis, tada servis prestaje sa izvršavanjem, kada sve komponente prekinu interakciju sa njim pozivom funkcije unbindService().

Servis se definiše nasleđivanjem ugrađene Android klase Service i implementiranjem metoda za definisanje po našanja servisa tokom životnog ciklusa. Takođe neophodno je obezbediti rukovanje mehanizmom za komunikaciju sa drugim aplikativnim blokovima.

Na slici (*Slici 3.*) prikazan je dijagram koji opisuje životni ciklus Android servis komponente za oba režima rada. U sledećoj tabeli su objašnjene metode za kreiranje, pokretanje, povezivanje servisa.

Metode	Opis
onStartCommand()	Sistem poziva ovu metodu kada neka aplikativna komponenta, npr. aktivnost, pozovme startService() metodu. Izvršavanjem ove metode servis se pokreće i može se beskonačno dugo izvršavati u pozadini, naravno servis može biti zaustavljen pozivom metode stopSelf() ili stopService(). Servis može biti nasilno zaustavljen od strane operativnog sistema u slučaju nedostatka memorije
onBind()	Proziva se kada neka komponenta ili proces želi da se poveže sa servisom pozivom metode bindService(). U okviru ove metode, ukoliko se želi omogućiti povezivanje sa drugim komponentama neophodno je definisati korisničku spregu i povratnu vrednost koji mora biti instanca IBinder klase. IBinder je ugrađe na korisnička sprega Android operativnog sistema, koja omogućuje

	komunikaciju visokih performansi, kao i deljenje podataka između različitih procesa.
onCreate()	Sistem poziva ovu metodu pri prvom startovanju servis komponente. U okviru ove metode se obično vrši inicijalizacija modula, bitnih za ispravno funkcionisanje samog servisa.
onDestroy()	Poziva se za zaustavljanje servisa u trenutku kada više nije potreban. Neophodno je u okviru ove metode izvršiti denicijalizaciju i čišćenje nepotrebnih resursa, kako bi se sprečilo nepotrebno trošenje memorije (eng. *memory leak*).

Tabela 3. Metode životnog ciklusa Servisa

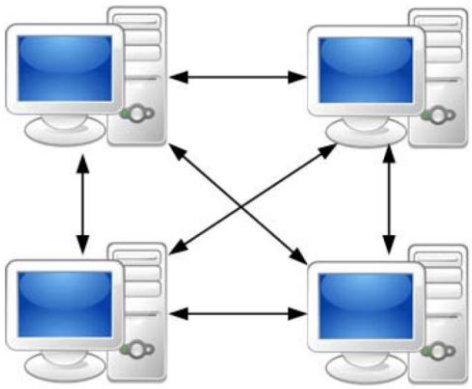
Da bi se korisničke aplikacije povezale sa servisom potrebno je definisati spregu kojom se odvija komunikacija između klijenta i servisa. Definisanje programske sprege se obavlja korišćenjem odgovarajućeg namenskog jezika (eng. *AIDL - Android Interface Definition Language*). Kao što je već rečeno, u Android operativnom sistemu jedan proces ne može jednostavno da pristupi memorijskom prostoru drugog procesa. AIDL sprega poseduje mogućnost raščlanjivanja složenih objekata i tipova podataka na primitivne tipove, koje operativni sistem može da deli između različitih procesa. Svaki složeni objekat koji se prenosi kroz IBinder spregu mora podržati protokol raščlanjivanja na primitivne tipove, i ponovnog sastavljanja u složen objekat (eng. *Parcelable).*Važno je napomenuti da se za potrebe programske podrške ovog rešenja koristi servis komponenta, koja dozvoljava komunikaciju sa drugim procesima. Na taj način se uspostavlja sprega između različitih slojeva aplikativnog dela programske podrške.

Aplikacija ili servis napisan u Java programskom jeziku tok instalicaije prevodi bajt kod u mašinski kod. Ovakav pristup se koristi od verzije 5.0. Ukoliko postoji potreba da aplikacija napisana u Java programskom jeziku, komunicira sa nekim modulom napisanom u C programskom jeziku, neophodno je koristiti JNI spregu (eng. *Java Native Interface*), kako bi se obezbedilo ispravno pozivanje C koda iz Java programa i prosleđivanje podataka u oba smera.

JNI sprega definiše način na koji komuniciraju moduli napisani u Java i C kodu. Sprega je nezavisna od platforme, omogućava korišćenje deljenih biblioteka i doprinosi u efikasnosti i performansama celokupnog sistema. JNI omogućava lak pristup razvojnom okruženju i razvojnoj podršci, koju omogućava operativni sistem Android platforme. Zbog toga JNI predstavlja prirodan način za povezivanje sistemskih i aplikativnih komponenti u operativnom sistemu Android platforme. Kao što je navedeno u ovom poglavlju operativni sistem Android platforme je prvi sloj programske podrške iznad fizičke arhitekture.

2.2 Point-to-Point

Mrežna arhitektura Point-to-Point sastoji se od učesnika koji dele deo svojih resursa koji se mogu koristiti putem interkonekcije. Ovi mrežni resursi su dostupni drugim učesnicima mreže bez potrebe za centralnim kontrolerima kao što su Serveri ili hostovi. Učesnici mreže su jednaki. Svi učesnici imaju jednaka prava pristupa i raspodele resursa. Na sledećoj slici je prikazana šema P2P mreže [9] (Slika 4.).



Slika 4. Šema point-to-point mreže

P2P mrežnu arhitekturu je moguće zamisliti tako da su na istom računaru postavljeni uslužilac i klijent, tj. svaki računar može istovremeno primati i davati resurse drugim učesnicima pripadajuće odgovarajuće mreže. Upravo to ukazuje na brzinu rada point-to-point mreža, jer svaki učesnik mreže mora poznavati mrežnu adresu drugog učesnika kako bi mu mogao pristupiti. Takav način rada značajno usporava rad samog mrežnog sistema ukoliko se radi o velikom broju učesnika [10].

Point-to-point mreže je moguće podeliti prema:

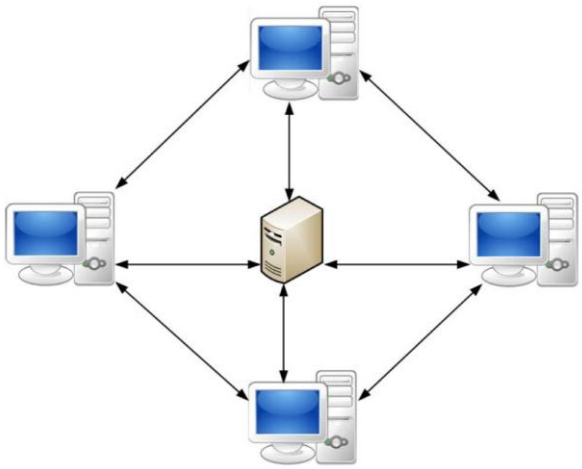
- načinu ispisivanja podataka koji se razmjenjuju (*eng. File listing*)
- načinu spajanja učesnika mreže (*eng. Node connection*)

2.2.1 Podela prema načinu ispisivanja podataka

Način ispisivanja podataka u P2P mrežama od velike je važnosti jer određuje brzinu rada mreže. Ispisivanjem podataka pomoću središnjeg uslužioca je ubrzan rad mreže jer učesnik preuzima popis traženih podataka sa uslužioca (što nije slučaj kod decentralizovanih mreža). Kao i kod svakog mrežnog sistema, poželjna je što veća brzina prenosa podataka, što kod ovakvog tipa mreže često nije slučaj.

2.2.1.1 Centralizovane (hibridne) p2p mreže

Kod centralizovane mreže popis dostupnih podataka se nalazi na središnjem uslužitelju, što uvelike rad mreže. Mreže koje rade na ovaj način smatraju se hibridnim, zato jer sadrže uslužitelja i time odstupaju od arhitekture konvencionalnih P2P mreža.



Slika 5. Prikaz hibridne point-to-point mreže

2.2.1.2 Decentralizovane p2p mreže

Decentralizirane p2p mreže u svojoj arhitekturi ne sadrže središnje uslužitelje, već samo učesnike. Kod decentralizovanih p2p mreža svaki učesnik je povezan s mnogo drugih učesnika mreže, te od njih preuzima popis dostupnih podataka. Svaki učesnik je u mogućnosti da uzima tuđe i daje vlastite resurse drugim učesnicima komunikacije.

2.2.2 Podela prema načinu spajanja učesnika

P2P mreže su prema načinu spajanja učesnika podeljene na:

- strukturirane *p2p* mreže
- nestrukturirane *p2p* mreže

2.2.2.1 Strukturirane p2p mreže

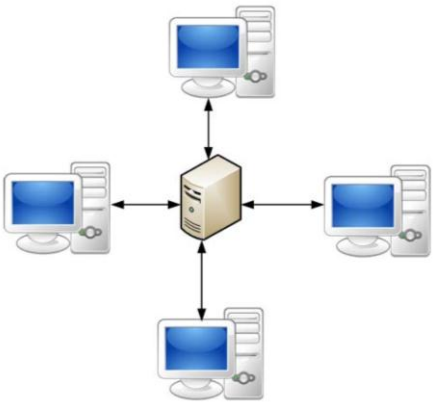
Strukturirane p2p mreže koriste algoritme koji osiguravaju da svaki učesnik može pomoću pretrage (*eng. Search*) pronaći željene podatke kod drugog učesnika mreže. Algoritmi u strukturiranim mrežama osiguravaju najveću učinkovitost i brzinu mreže. Najčešća vrsta strukturiranih p2p mreža su mreže koje koriste distribuirane raspršne tablice (*eng. Distributed hash table - DHT*), gdje je svaki podatak obeležen pripadajućom raspršnom funkcijom (*eng. Hash function*).

2.2.2.2 Nestrukturirane peer-to-peer mreže

Nestrukturirane p2p mreže nastaju proizvoljnim međusobnim spajanjem učesnika. Kada učesnik mreže u nestrukturiranim p2p mrežama želi pronaći neki podatak, upit se šalje svim dostupnim učesnicima kako bi se pronašao što veći broj koji imaju isti podatak. Nedostatak nestrukturiranih mreža jeste to da neki upiti nemaju rezultata. Ukoliko su u pitanju podaci koji se nalaze kod velikog broja učesnika upit će biti nađen, ali ukoliko se radi o retkim podacima, postoji velika mogućnost da učesnik neće uspjeti pronaći tražene podatke.

2.3 TCP/IP

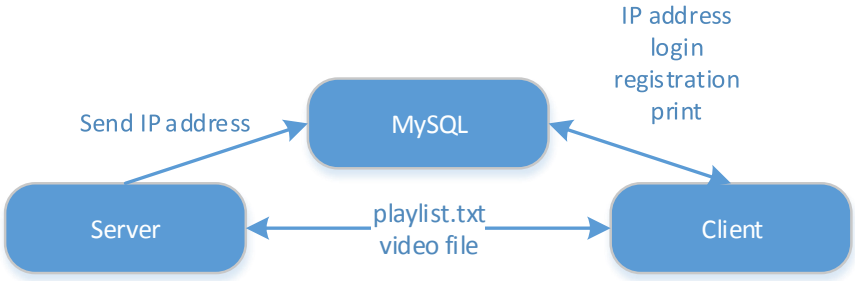
Druga stvar je upotreba TCP/IP protokola [11] kao način komunikacije. Ova grupa protokola naziv je dobila po dva najvažnija protokola TCP (Transmission Control Protocol) i IP (Internet Protocol). TCP/IP omogućuje komunikaciju preko različitih, međusobno povezanih mreža, i danas je najrasprostranjeniji protokol na lokalnim mrežama. Na njemu se zasniva i globalna mreža, internet. Preko ovog protokola vrši se razmena podataka između uslužioca i korisnika sa baznim modulom. Prikaz šeme na Slici 6.



Slika 6. TCP/IP i OSI model

3. Osnovni koncepti sistema

Za rešavanje opisanog problema odabrana je uslužno-korisnička arhitektura, gde uslužnu i korisnička stranu predstavljaju Android aplikacije na pametnim mobilnim uređajima. Komunikacija između dve strane reguliše se preko tekstualnog fajla u kome se nalazi lista putanja do video materijala. U ovom slučaju P2P komunikacija između uređaja označava direktnu komunikaciju između uređaja u razmeni podataka. Upotrebom AES algoritma [12] se enkriptuju podaci izmedju Android aplikacije na kojoj se pokreće server i udaljene baze podataka, kao i između klijenta i uslužioca. Grub prikaz arhitekture sistema nalazi se na slici 7.



Slika 7. Osnovni prikaz komunkacije

Na slici se može primetiti da uslužilac (eng: server) i klijent (eng: client) razmenjuju poruke. Uslužilac i korisnik vrše kumunikaciju sa udaljenim baznim modulom (MySQL). Za implementaciju baznog modula odabran je PHP skript jezik i SQL, dok se za ostale module podrazumeva programski jezik Android aplikacija - Java. Ključni problemi u ovom sistemu su prenos video i audio datoteka između uređaja u povremenom ažuriranju tekstualne datoteke sa putanjama fajlova do prekida prenosa. Sledećim redosledom je pokazan način komunikacije između Android uređaja uslužioca koji započinje snimanje video i audio materijala:

Koraci	Komunikacija
Korak 1	registracija korisnika u aplikaciji i slanje parametara udaljenoj bazi podataka
Korak 2	logovanje korisnika u aplikaciji i verifikacija poslatih podataka
Korak 3	započinjanje snimanja i pokretanje lokalnog server
Korak 4	slanje IP adrese bazi podataka
Korak 5	kreiranje tekstualnog fajla pod nazivom playlist.txt u memoriji Android telefona na lokaciji instalirane aplikacije
Korak 6	upisivanje putanje video fajla u format mp4 na svakih 5 sekundi

Tabela 4. Redosled komunikacije između Android uređaja uslužioca

Server odgovara drugim Android uređajima na traženi tip fajla. Odgovara sa tekstualnim fajlom ako se traži fajl sa ekstenzijom .txt, a zahtevanjem .mp4 odgovara video fajlom. Bilo kojim drugim zahtevanje ne navedene ekstenzije fajla u odgovoru se šalje tekstualni fajl.

Komunikacija između Android uređaja klijenta sa ostalim delovima sistema regulisanja je sledećim redosledom:

Koraci	Komunikacija
Korak 1	registracija korisnika u aplikaciji i slanje parametara udaljenoj bazi podataka
Korak 2	logovanje korisnika u aplikaciji i verifikacija poslatih podataka
Korak 3	zahtevanje liste svih aktivnih prenosa iz baze podataka
Korak 4	šalje se zahtev bazi podataka za IP adresu izabranog prenosa
Korak 5	konektovanje na lokalni server
Korak 6	slanje zahteva za playlist.txt
Korak 7	puštanje video sadržaja u ExoPlayer-u

Tabela 5. Redosled komunikacije Android uređaja klijenta sa ostalim delovima sistema

Zahtev koji je naveden u tački 6. se šalje ponovo kada se završi puštanje svih fajlova koji su zabeleženi u postojećem tekstualnom fajlu.

3.1 Uslužilac

Uslužna strana predstavlja server na kome se pokreće snimanje video i audio sadržaja korišćenjem Camera2 API [13]. Pokretanjem Android aplikacije pokreće se server na IP adresi koja je dodeljena tom uređaju u lokalnoj mreži na portu 8089. NanoHTTPD [14] je mali integrisani server koji poseduje osnovne funkcije. Prva komunikacija se izvršava između uslužioca i baze podataka. Uslužilac prilikom pokretanja servera šalje svoju IP adresu skripti na udaljenom serveru koja upisuje u bazu podataka.

Pomoću tajmera (eng: *CountDownTimer*) se određuje snimanje fajla na svakih pet sekundi. Kreira se tekstualni fajl u kome se nalaze putanje poslednjih pet datoteka.

Uslužilac pri završenoj komunikaciji sa bazom podataka očekuje zahtev od korisnika. Prilikom uspešne konekcije na server počinje razmena poruka.

3.2 Klijent

Na korisničkom delu Android java aplikacije omogućene su sledeće funkcije:

Funkcije korsiničkog dela	Opis funkcija
Prikaz svih uslužilaca	Lista svih aktivnih prenosa sa prikazivanjem naziva i ip adrese uslužioca
Dodavanje korisnika	Dodavanje novih korisnika u bazu podataka sa parametrima korisničko ime, email adresa i šifra
Promena informacija naloga	Promena korisničkog imena, email adrese i šifre korisnika koji je prijavljen na aplikaciju
Spisak svih korisnika	Lista svih registrovanih korisnika sa prikazivanjem korisničkog imena i email adrese
Pregled svakog stream-a	Prikazivanje aktivnog prenosa video materijala putem ExoPlayer-a

Tabela 6. Funkcije na korisničkom delu aplikacije

Odabirom prve opcije u listi ponuđenih opcija u meniju pod nazivom prikaz svih kamera (eng: *All cameras*) predstavlja se lista svih aktivnih prenosa video i audio sadržaja u lokalnoj mreži. Druga opcija omogućava započinjanje prenosa video i audio sadržaja koja predstavlja uslužioca, dok odabirom treće opcije vraća se lista svih onih koji su korisnici ove aplikacije.

Prilikom započete video i audio reprodukcije, na strani uslužioca, iz liste korisnik se prebacuje na ekran na kome se nalazi ExoPlayer [15] i započinje prikazivanje zabeleženog materijala sa uslužiočeve kamere.

Prilikom izabranog prenosa korisniku se šalje lista putanja upisana u tekstualni fajl. Plejer koristeći ovu listu pušta video materijal po redosledu iz liste. U ovom tekstualnom fajlu je upisana lista od pet medija. Nakon završenog poslednjeg medija šalje se zahtev serveru za novu listu koja sadrži novih pet, ako je snimanje toliko trajalo. Korisnik ima mogućnost da zahteva premotavanje prenosa unazad.

3.3 Baza podataka

Pod baznim modulom se podrazumeva udaljena baza podataka (*MySQL*). Prvo se obrađuje svaki GET zahtev za upis lokalne IP adrese u tabelu svih aktivnih kamera pomoću skripte (*add_camera.php*) koja je implementirana PHP skript jezikom na serveru koji se nalazi van lokalne mreže.

Svi zahtevi su kriptovani sa različitim ključem i tako se upisuju u bazu podataka. Do dekripcije podataka dolazi u korisničkoj aplikaciji.

3.4 Šema baze podataka

Na SQL dijagramu (Slika 8.) prikazana je master baza podataka koja se sastoji iz tri entiteta:

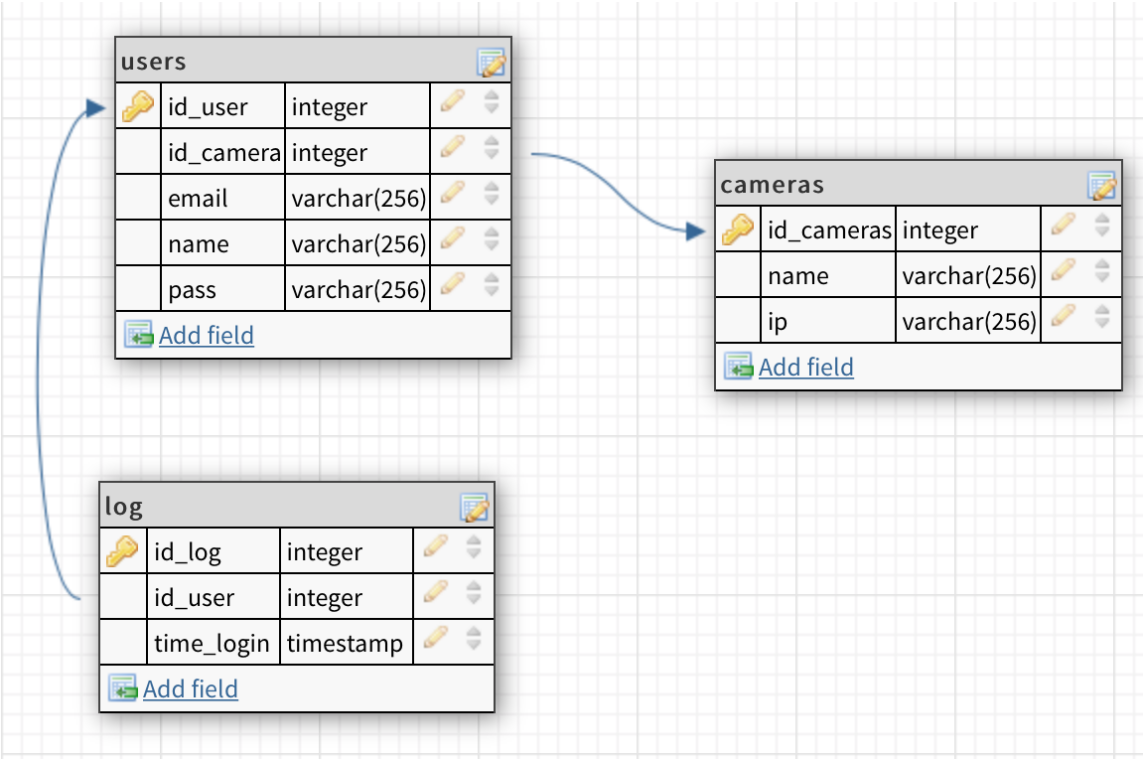
Entiteti	Opis sadržaja
Tabele korisnika (users)	Tabela svih korisnika aplikacije
Tabele kamera (cameras)	Tabela svih kamera koje aktivno vrše prenos video sadržaja
Tabele logovanja (log)	Tabela u kojoj se beleži vreme i datum prijavljivanje korisnika na aplikaciju

Tabela 7. Tabela prikaza entiteta u bazi podataka

Tabela korisnika ili users sastoji od atributa id_user koji predstavlja primarni ključ (primary key) tip polja je integer i email, ime (name), šifra (password) korisnika su tip varchar od maksimalnog broja karaktera 256. Pri registraciji korisnika se u ovu tabelu beleže zapisi.

Veza između entiteta cameras i users je jedan – jedan (1,1) što znači da je maksimalna kardinalnost 1 na svakom putu koji vodi iz jedne veze, svakom elementu cameras skupa pridružen je najviše jedan element iz skupa users i obratno, svakom elementu skupa users pridruzen je najviše jedan element skupa cameras. U tabeli cameras se beleže ime (name) i ip adresa kamera.

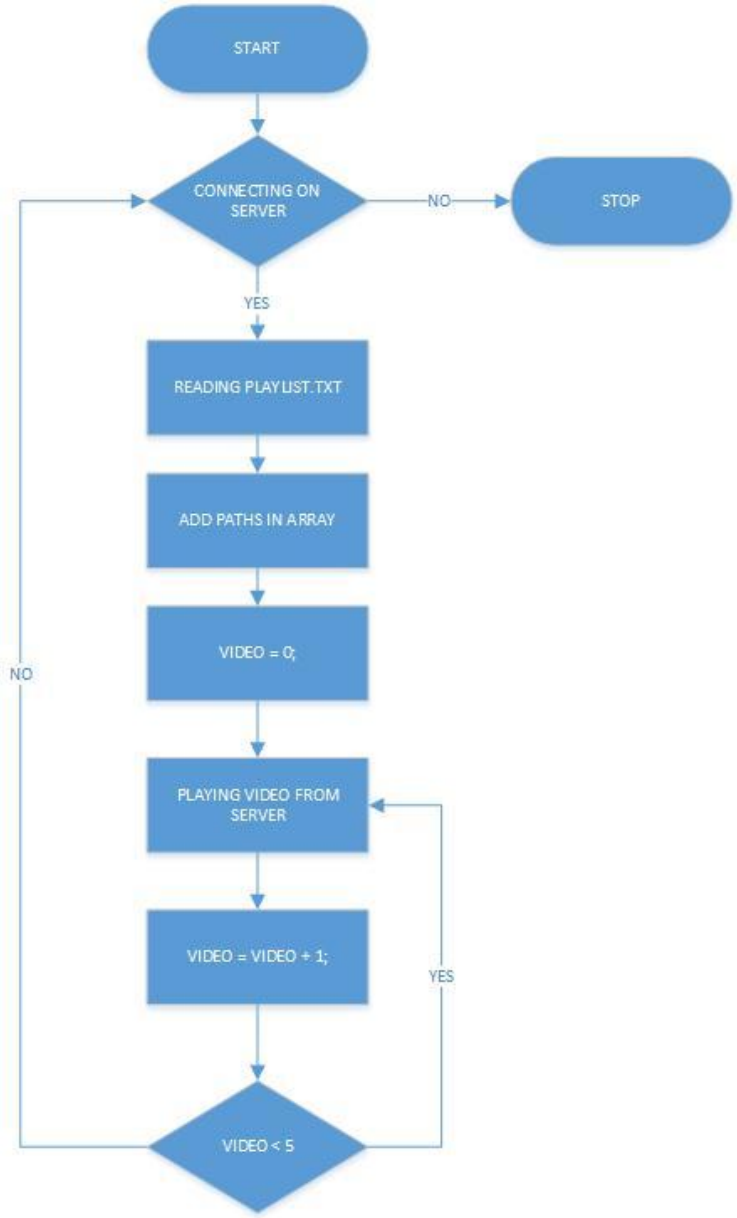
Veza između tabela log i users je isto jedan – jedan. Log tabela beleži vreme i datum kada se korisnik ulogovao.



Slika 8. SQL dijagram za MySQL bazu podataka

3.5 Algoritam za video i audio prenos

Na Slici 9. je prikazan dijagram algoritma za prenos video i audio sadržaja. Ako je server pokrenut moguće je uspostaviti konekciju i tražiti zahtev za čitanje tekstualnog fajla pod nazivom playlist.txt. Putanje do fajlova se dodaju u niz i počinje puštanje jednog po jednog video fajla. Pri završetku liste od pet puštenih video materijala osvežava se lista sa novih pet putanja, ako je prenos i dalje aktivan.



Slika 9. Dijagram algoritma za video i audio prenos

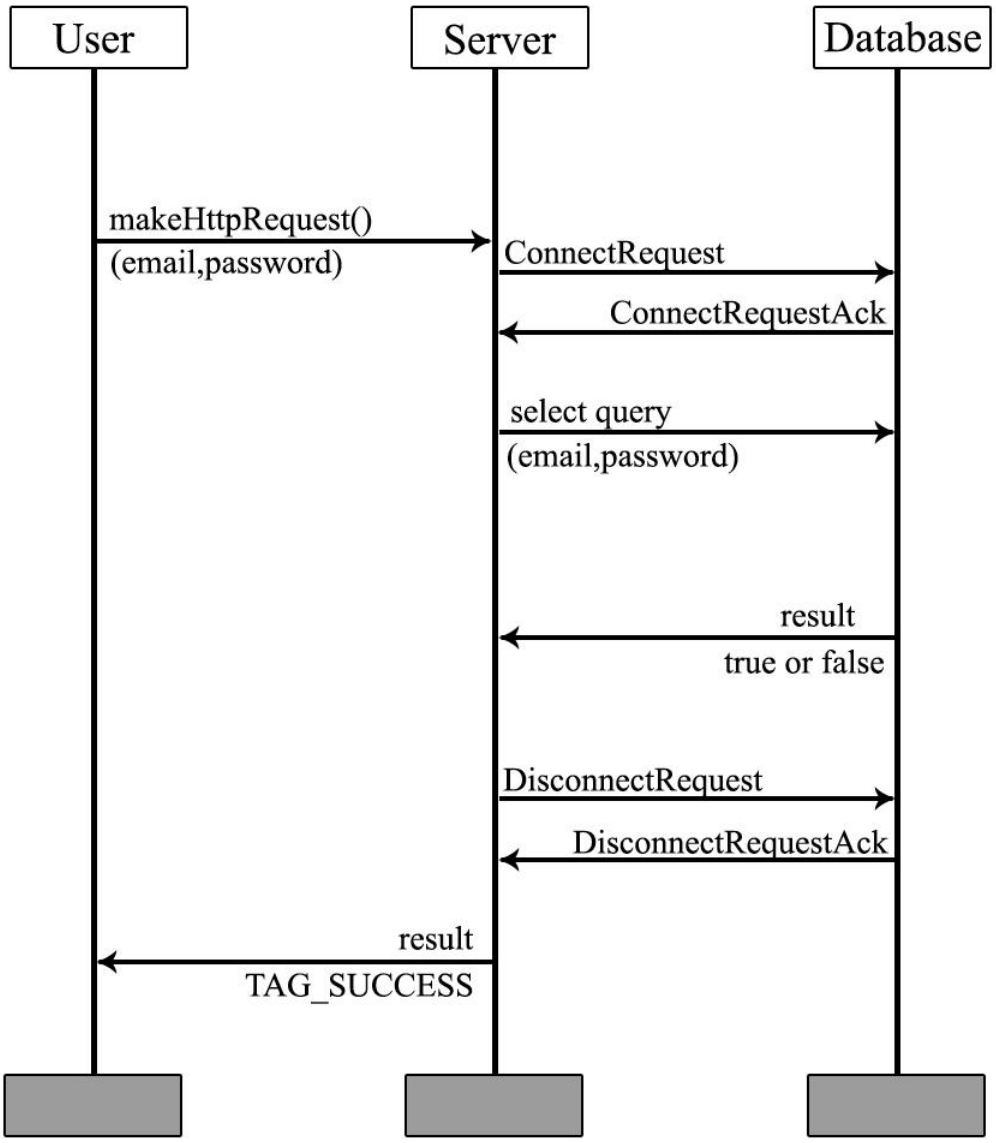
3.6 Algoritam za komunikaciju

3.6.1 Logovanje i registracija korisnika

Na slici 10. prikazano je rešenje načina komunikacije između korisnika (User) sa udaljenim serverom (Server). Server je posrednik između korisnika i baze podataka. Korisnik prvo šalje podatke za logovanje korišćenjem json strukture pomoću funkcije makeHttpRequest() sa parametrima email i šifra.

Server vrši konekciju na bazu podataka i pri uspešnoj konekciji pomoću SELECT upita sa poslatim parometrima dobija odgovor od baze da li je validan i da li postoji traženi zapis. Ovaj odgovor se šalje korisniku u obliku Stringa TAG_SUCCESS koji ima vrednost 1 ako je zapis postojan ili vrednost 0 ako zapis nije pronađen. Dijagram koji pokazuje komunikaciju pri

registraciji između korisnika i baze podataka je sličan prikazu dijagrama za logovanje korisnika. U ovom dijagramu se mesto SELECT upita šalje INSERT INTO upit za upisivanje zapisa u bazu podataka. Kada je zapis upisan u bazu dobija se odgovor da je uspešno izvršen upit ili nije.



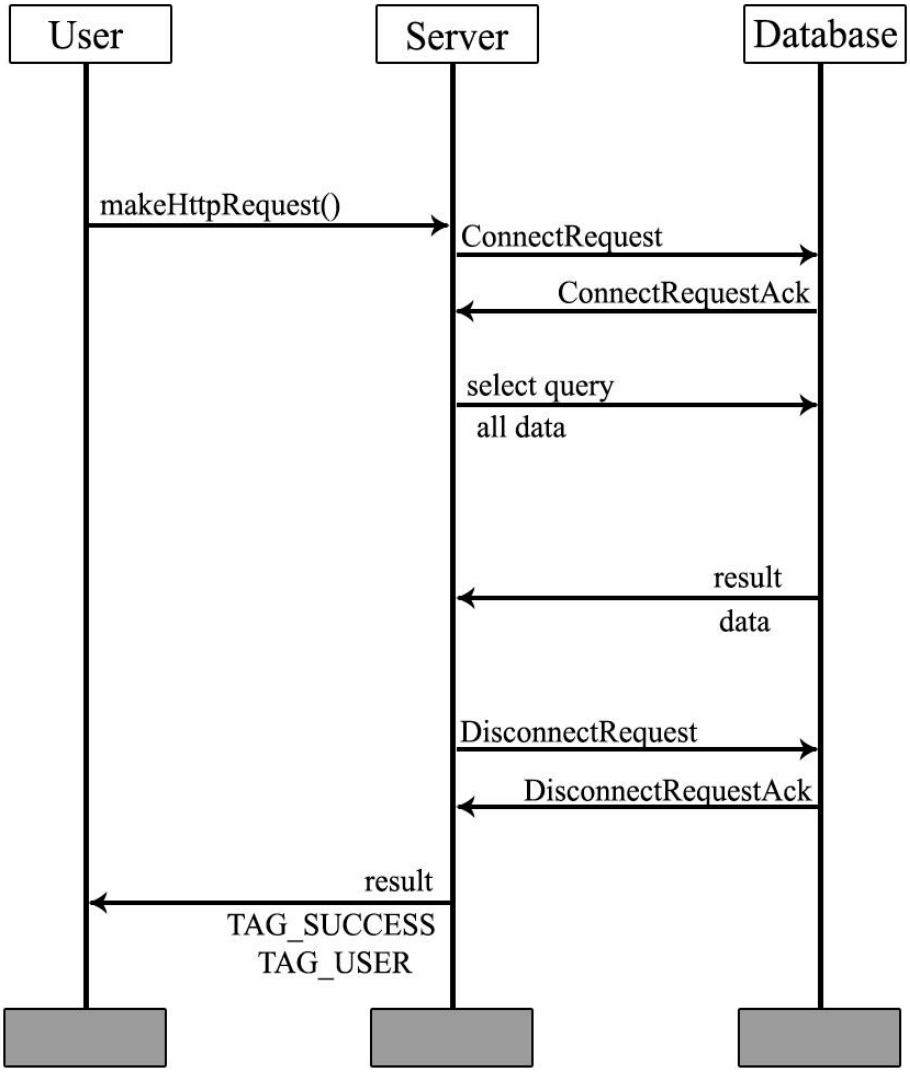
Slika 10. Primer dijagrama za logovanje korisnika

3.6.2 Prikaz svih korisnika i svih kamera

Na slici 11. prikazan je dijagram na kome se zahteva lista svih korisnika od baze podataka. Korisnik prvo šalje GET zahtev serveru, koji prvo izvršava konekciju sa bazom podataka i pri povratnoj informaciji da je uspešno konektovan šalje SELECT upit sa zahtevom svih kolona iz tabele korisnika. Rezultat dobijen od baze podataka u obliku liste zapisa korišćenjem JSON

strukture vraća korisniku. TAG_SUCCESS kao i u predhodnom delu ima vrednost ili 1 ili 0, dok TAG_USER dobijamo listu user sa id, email i imenom korisnika.

Dijagram prikaza svih kamera je identičan predhodnom dijagramu prikaza svih korisnika, samo što u SELECT upitu zahtevamo sve zapise iz tabele cameras i kao odgovor dobijamo listu svih upisanih kamera sa njihovim zadatim nazivom i ip adresom.

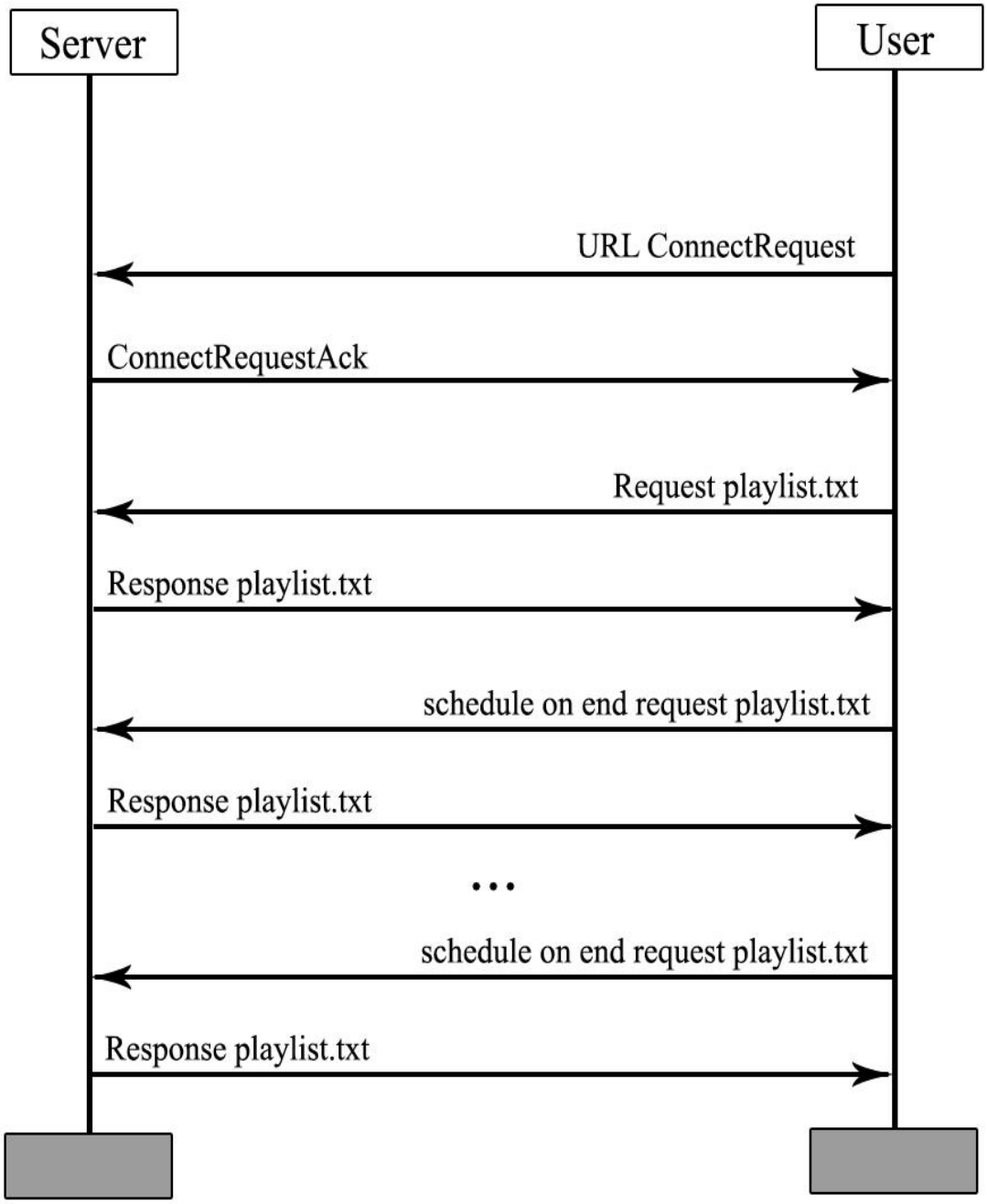


Slika 11. Primer dijagrama za prikaz korisnika

3.6.3 Prenos tekstualne datoteke

Prikaz komunikacija između lokalnog servera ili telefona koji vrši reprodukciju video sadržaja i korisnika ili strane koja gleda taj sadržaj. Prvo se izvršava konekcija na lokalnu ip adresu sa zadatim portom. Po uspešnoj konekciji korisnik zahteva tekstualni fajl pod nazivom playlist.txt .

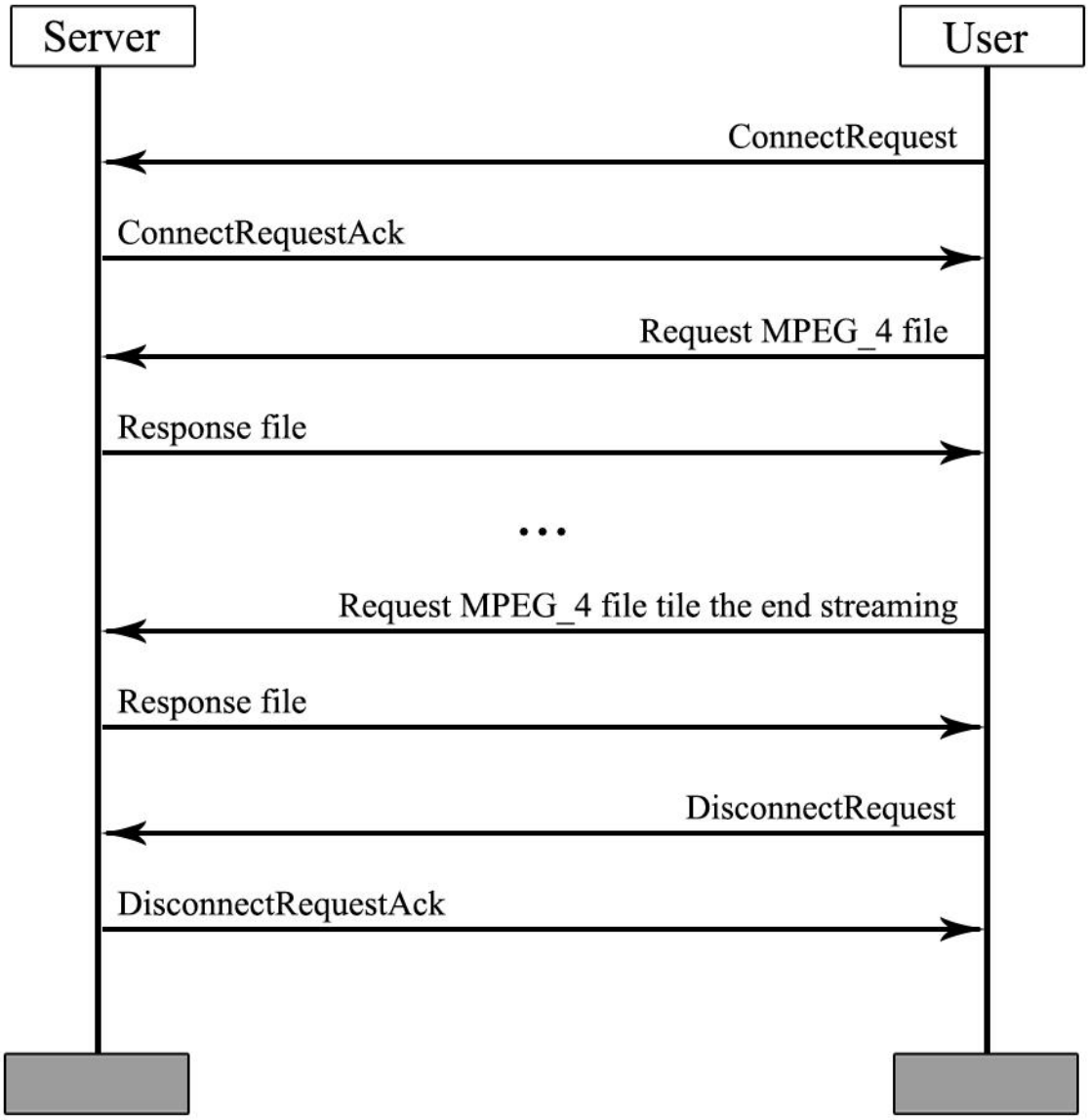
Dalja komunikacija se ponavlja pri završetku puštanja sadržaja iz stare liste. Primer dijagrama na Slici 12.



Slika 12. Primer dijagrama za prenos playlist.txt

3.6.4 Prenos video datoteke

Dijagramom na Slici 13. pokazujemo način razmene video fajla sa lokalnim serverom koji se pušta na korisnikovom uređaju. Nakon uspešnog konektovanja zahteva se video fajl od servera. Preuzimanje fajlova se izvršava sve dok je snimanja na serveru aktivno. Prilikom stopiranja snimanja raskida se konekcija između korisnika i servera.



Slika 13. Primer dijagrama za prenos MPEG_4 video datoteka

4. Implementacija

Ceo sistem se sastoji iz tri dela kao što smo već napomenuli u predhodnom poglavlju, a to su :

Delovi sistema	Opis
Serverski deo	U ovom delu sistema server odgovara na klijentov zahtev za fajl sa ekstenzijom .txt ili .mp4
Klijentski deo	Ovaj deo vrši reprodukciju video fajlova koje je preuzeo od servera
Bazni deo	Izvršava se na udaljenoj bazi podataka u kojoj se beleže sve aktivni prenosi i spisak svih korisnika koji mogu da im pristupe

Tabela 8. Delovi sistema za prenos video i audio sadržaja

U ovom delu ćemo pristupiti implementaciji svakog modula i kako je rešen način komunikacije između modula. Prikazani su primeri realzovanih najbitnih delova modula i problemi do kojih je došlo tokom implementacije.

4.1 Serverski deo

Ovaj deo predstavlja deo sistema, koji prima poruke od klijenta, na osnovu kojih daje odgovarajući odgovor. Pri pokretanju server modula čekaju se novi zahtevi klijenta. Na ovom delu izvršavaju se sledeće funkcije:

Funkcije	Opis
Slanje lokalne IP adrese baznom modulu	Upisivanje ip adrese uslužioca u bazu podataka
Snimanje video i audio datoteka	Klikom na dugme “Stream” poziva se funkcija startRecordingVideo()
Memorisanje video i audio datoteka	Video sadržaj se snima na svakih pet sekundi u internu memoriju
Kreiranje tekstualnog fajla na osnovu predhodnog sadržaja	Tekstualni fajl se kreira iznova na svakih pet snimljenih videa, dok se u među vremenu lista dopunjava
Slanje odgovora korisniku u obliku txt datoteke	Upotreba NanoHttpd servera za komunikaciju i slanje tekstualnog fajla korisniku
Slanje odgovora player-u na klijentskoj strani u obliku željenih video i audio datoteka	Korisnik na osnovu putanja koje je dobio u sadržaju tekstualnog fajla, vrši reprodukciju videa u ExoPlayer-u

Tabela 9. Tabela prikaza funkcija serveskog modula

4.1.1 Slanje lokalne IP adrese baznom modulu

Pokretanjem dela za početak prenosa video datoteka preko lokalne WiFi mreže između klijent mobilnog uređaja i server mobilnog uređaja P2P komunikacijom pokreće se server na serverskom delu aplikacije. Server koji se koristi je NanoHttpd koji ima funkciju usluživanja na zahtev korisnika, a u ovom slučaju je razmena video i audio i tekstualnih datoteka. Da bi klijent mogao da započne komunikaciju sa serverom, ip adresa servera mora da bude poznata. S obzirom da je dinamička ip adresa, svakim novim pokretanje vrši se komunikacija između ovog i baznog modula.

4.1.2 Snimanje i kreiranje video datoteka

Korisnik na pritisak dugmeta iz donjeg dela ekrana započinje snimanje videa i čime je potrebno učiniti izmene na grafičkom interfejsu koje se rade pomoću GUIThread-a.

Zbog dosta poteškoća, jer je korišćena nova verzija android OS Nougat, za rad sa kamerom korišćena je biblioteka Camera2 API. Ona omogućava snimanje video materijala i njihova podešavanja, kao automatsko fokusiranje videa. Video se snima u formatu mp4. Video datoteka se snima na svakih pet sekundi čime se prekida snimanje MediaRecorder.stop(), putanja datotke se stavlja na null vrednost i poziva se opet metoda startRecordingVideo(). Na početku ove

metode se poziva setUpMediaRecorder() kojom se podešava format video fajla u ovom slučaju MPEG_4, broj frejmova u sekundi koji je stavljen da bude 30 po uobičajenim podešavanjima kamere, visina i širina video datoteke. Kao video enkoder se koristi H264, a za audio enkoder AAC. Zavisnosti od položaja telefona se menja prikaz.

4.1.3 Kreiranje tekstualne datoteke

Datoteka se kreira na putanji gde je instalirana aplikacija. Korišćenjem abstraktne klase CountDownTimer na svakih pet sekundi se poziva metoda saveFile() kojom se snima video datoteka u trajanju od pet sekundi. Ovaj metoda za parometre traži putanju snimljene datoteke, kontekst i redni broj videa. Pomoću rednog broja određuje se da li će putanja biti upisana u trenutni tekstualni fajl ili se kreira novi sa tom putanjom. Na svakih pet snimljenih video materijala u trajanju od pet sekundi kreira se novi tekstualni fajl. Korišćenjem metode openFileOutput() kreira se novi fajl. Kao parometri se prosleđuju naziv fajla koji je u ovom slučaju nepromenljiv (playlist.txt) i Context.MODE_PRIVATE koji označava da se fajl opet kreira nevezano da li je već bilo nešto u istom fajlu na toj putanji. Dok Context.MODE_APPEND označava da se nastavlja upisivanje u postojeći i da se ne kreira novi fajl.

4.1.4 Response datoteke

Odgovor servera na klijentov zahtev može biti u obliku tekstualne txt datoteke ili video mp4 datoteke. Zahtevanjem određene datoteke od strane korisnika pomoću substring funkcije se saznaje ekstenzija datoteke na osnovu koje u switch petlji klijent dobija odgovor željene datoteke. Ekstenzija datoteke je bitna zbog meta oznake (MIME format) za video datoteke video/mp4, a za tekstualne datoteke text/plain. Koristi se Response funkcija NanoHTTPD biblioteke koja kao parometre koristi status (Response.Status.OK), meta oznaku i traženu datoteku koju smo pristupili funkcijom openFileInput() koja kao argument zahteva putanju na kojoj se ona nalazi. Za putanju tekstualnog fajla uvek koristimo isti fajl, dok se za video koristi traženi fajl iz putanje koji je zahteva korisnik.

4.2 Klijentski deo

Ovaj deo aplikacije izvršava se u delu za reprodukciju video datoteka i čine ga sledeće funkcije:

Funkcije	Opis
Komunikacija sa baznim modulom	Selektovanjem dela sa prikazom svih aktivnih prenosa pokreće se u pozadini preuzimanje liste svih kamera iz baze podataka
Zahtevanje tekstualnog fajla od servera	Konektovanjem na uslužiočev server zahteva se playlist.txt
Zahtevanje video datoteka od servera	Na osnovu liste putanja iz tekstulanog fajla šalje se zahtev za video fajlove
Prikaz dobijenih datoteka	Reprodukcija se realizuje preko ExoPlayer

Tabela 10. Tabela prikaza funkcija klijentskog dela

4.2.1 Komunikacija sa baznim modulom

U layout-u all_cameras se prikazuje lista svih kamera. Pri izboru odgovarajućeg servera za reprodukciju videa u realnom vremenu obavezna je komunikacija zbog lokalne adrese servera. U klasi LoadAllCameras je obezbeđena komunikacija sa baznim modulom iz klijentskog dela aplikacije. Korisnik slanjem GET zahteva php skripti get_all_cameras.php i kao povratnu informaciju dobija od servera u obliku json-a koji se parsira. Na osnovu dobijene adrese moguće je ostvarivanje komunikacije sa serverom. Parametar TAG_SUCCESS ako ima vrednost 1 označava da u tabeli cameras ima zabeleženih aktivnih strimova, dok vrednost 0 da je tabela prazna. Ostali parametri su id, name i ip. Kreira se ArrayList-a u kojoj se nalazi HashMap-a Stringova u kojoj su upisani svi podaci iz dobijenog odgovora iz tabele kamera. Ispisivanjem liste kamera u pozadini se pokreće nova nit.

4.2.2 Zahtevanje tekstualnog fajla od servera

Pomoću url-a playliste prvo se otvara konekcija sa serverom korišćenjem metode HttpURLConnection. Kreira se BufferReader koji pomoću InputStreamReader-a isčitava tekstualnu datoteku. U svakom redu se nalazi po jedna putanja koja određuje lokaciju video fajla na serveru. Ova komunikacija se ponavlja periodično zbog promene sadržaja playliste. Pomoću abstrakte klase ScheduledExecutorService kreira se nova nit sa ponovnim učitavanjem. Završetkom puštanja videa, sa određenih putanja, iz tekstualnog fajla koristi se predhodno navedena klasa pozivajući metodu scheduledAtFixedRate().

4.2.3 Zahtevanje video datoteka od servera i njihovo prikazivanje

Nakon preuzete playliste u .txt formatu pokrećemo novu nit u pozadini da bi sadržaj tekstualnog fajla u ovom slučaju putanje do video fajlova na serveru dodelili nizu. Reprodukcija medija se vrši korištenjem biblioteke ExoPlayer. Prvo se mora kreirati odgovarajući MediaSource. Da bi omogućili puštanje medija fajla sa ekstenzijom .mp4 kreiramo ExtractorMediaSource. Pogodan je za medijske fajlove formata MP4, WEBM, MKV, M4A, MPEG-TS and AAC. Primer kreiranja je prikazan sa zadatim Uri.

```
MediaSource videoSource = new ExtractorMediaSource(mp4VideoUri,dataSourceFactory,
extractorsFactory, null, null);
```

Prvo je potrebno kreirati plejer za puštanje sadržaja. Kreiramo SimpleExoPlayer koji nam daje raznovrsne funkcije, kao što je pauziranje i premotavanje video materijala, do pojačavanja i smanjivanja audio zapisa u video sadržaju. U layout delu koristi se SimpleExoPlayerView kao što je prikazano u primeru ispod.

```
<com.google.android.exoplayer2.ui.SimpleExoPlayerView
android:id="@+id/player_view"
android:focusable="true"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_below="@+id/resolution_textView"
android:layout_marginTop="10dp" />
```

Nakon što je kreiran plejer View se dodeljuje njegova inastanca. Koristimo još dva parametra pri kreiranju ExtractorMediaSource, a to su DataSourceFactory i ExtractorsFactory. Pomoću DataSourceFactory omogućeno je da se vidi koji je mediji učitan. Jedan od parametara je DefaultBandwidthMeter koji meri protok tokom puštanja videa sa servera, dok je moguće staviti null mesto ovog parametra. ExtractorsFactory služi za parsiranje video materijala.

Biblioteka ExoPlayer takođe pruža implementaciju DASH (DashMediaSource), SmoothStreaming (SsMediaSource) i HLS (HlsMediaSource).

Pored implementacije MediaSource opisane gore, ExoPlayer biblioteka takođe obezbeđuje MergingMediaSource, LoopingMediaSource i ConcatenatingMediaSource. Ovo su primeri implementacije MediaSource-a koji omogućavaju kompleksniju reprodukciju kroz kompoziciju. Opisali smo niz slučajeva uobičajene upotrebe. Iako su sledeći primeri opisani u kontekstu reprodukcije videa, oni se podjednako primjenjuju na audio reprodukciju i za reprodukciju svih podržanih vrsta medija.

4.2.3.1 Učitavanje datoteke titla

S obzirom na video datoteku i zasebnu datoteku titlova, MergingMediaSource se može koristiti za spajanje u jedan izvor za reprodukciju. Prvo učitavamo video fajl kao u predhodnom primeru, potom učitavamo datoteku sa titlom i spajamo ove dve datoteke.

```
MediaSource videoSource = new ExtractorMediaSource(videoUri, ...);
MediaSource subtitleSource = new SingleSampleMediaSource(subtitleUri, ...);
MergingMediaSource mergedSource = new MergingMediaSource(videoSource,
subtitleSource);
```

4.2.3.2 Neprekidno puštanje video datoteke

Ponavljanje video datoteka se može napraviti koristeći LoopingMediaSource. Sledeći primer omogućava neograničeno ponavaljanje videa. Takođe je moguće odrediti broj konačnih petlji prilikom kreiranja LoopingMediaSource-a. Prvo učitavamo video fajl i ponavljamo ga.

```
MediaSource source = new ExtractorMediaSource(videoUri, ...);
LoopingMediaSource loopingSource = new LoopingMediaSource(source);
```

4.2.3.3 Neprekidna reprodukcija niza video zapisa

ConcatenatingMediaSource omogućava sekvencijalnu reprodukciju dva ili više pojedinačnih MediaSources. Sledeći primer reprodukuje dva video zapisa u nizu. Prelazak između izvora je besprekoran. Ne postoji uslov da se izvori koji se konektuju nalaze u istom formatu (npr. video datoteku koja sadrži 480p H264 sa onom koja sadrži 720p VP9). Izvori mogu čak biti različitih tipova (npr. povezivanje video sa audio streamom).

```
MediaSource firstSource = new ExtractorMediaSource(firstVideoUri, ...);
MediaSource secondSource = new ExtractorMediaSource(secondVideoUri, ...);
ConcatenatingMediaSource concatenatedSource = new
ConcatenatingMediaSource(firstSource, secondSource);
```

4.2.3.4 Napredne kompozicije

Moguće je dodatno kombinuju kompozitni MediaSources za više neobičnih slučajeva upotrebe. S obzirom na dva video zapisa A i B, sledeći primjer pokazuje kako se

LoopingMediaSource i ConcatenatingMediaSource mogu koristiti zajedno za petlje (A, A, B) na neodređeno vrijeme.

```
MediaSource firstSource = new ExtractorMediaSource(firstVideoUri, ...);
MediaSource secondSource = new ExtractorMediaSource(secondVideoUri, ...);
LoopingMediaSource firstSourceTwice = new LoopingMediaSource(firstSource, 2);
ConcatenatingMediaSource concatenatedSource = new
ConcatenatingMediaSource(firstSourceTwice, secondSource);
LoopingMediaSource compositeSource = new LoopingMediaSource(concatenatedSource);
```

Sledeći primer je ekvivalentan, pokazujući da postoji više načina za postizanje istog rezultata.

```
MediaSource firstSource = new ExtractorMediaSource(firstVideoUri, ...);
MediaSource secondSource = new ExtractorMediaSource(secondVideoUri, ...);
ConcatenatingMediaSource concatenatedSource = new
ConcatenatingMediaSource(firstSource, firstSource, secondSource);
LoopingMediaSource compositeSource = new LoopingMediaSource(concatenatedSource);
```

Važno je izbegavati korištenje iste instance MediaSource više puta u sastavu, osim ako je izričito dozvoljeno prema dokumentaciji. Upotreba firstSource dvaput u gore navedenom primeru je jedan od takvih slučajeva, jer Javadoc za ConcatenatingMediaSource izričito navodi da su dozvoljeni duplirani unosi. Uopšteno govoreći, grafikon predmeta koji se formiraju u kompoziciji treba da bude drvo. Dozvoljeno je korišćenje više ekvivalentnih primera MediaSource u sastavu.

Korišćenjem for petlje omogućili smo puštanje svih medija fajlova koji su navedeni u tekstualnoj datoteci. Upotrebom ConcatenatingMediaSource vršimo neprekidnu reprodukciju video datoteka sa servera. Pre samo puštanja pripremamo player sa prosleđenim instancom na spojene video datoteke iz niza.

```
player.prepare(concatenatingMediaSource);
```

4.3 Bazni deo

Ovaj deo se izvršava na udaljenom serveru na otvoreno besplatnom hostingu sa neogradničenim prenosom podataka i prostorom do 5 GB. Ovaj modul sastoji se od Mysql baze podataka i skripti koje regulisu zahteve dobijene od Android aplikacije. Komunikacije između ovog modula sa klijentskim i serverskim odvija se preko već pomenutog standarda JSON.

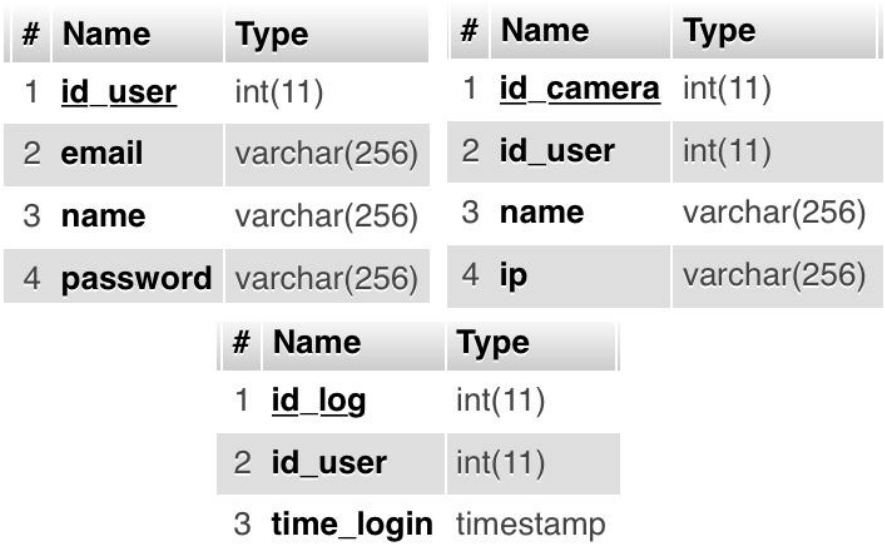
Skripte koje se nalaze na server su:

PHP skripte	Opis
create_user.php	Skripta za upisivanje novih korisnika u bazu podataka
create_camera.php	Skripta za upisivanje novih aktivnih prenosa u bazu podataka
login.php	Skripta za proveru unetih podataka u formu sa postojećim u bazi podataka
all_users.php	Skripta za ispisivanje svih registovanih korisnika
get_all_cameras.php	Skripta za ispisivanje svih aktivnih prenosa
get_camera_details.php	Skripta za ispisivanje ip adrese i imena određenog prenosa
db_config.php	Skripta za konektovanje na bazu podataka

Tabela 11. Spisak PHP skripti koje se nalaze u baznom delu

Tabele koje se koriste za beleženje podataka su users i cameras. Prikaz structure tabela na Slici 14.

Jedinstveni broj koji označava korisnika je id, a kameru cid. Ovi brojevi su stavljeni da imaju funkciju AUTO_INCREMENT čime se svaki put povećava id za jedan.



Slika 14. Tabele users, cameras i log u MySQL bazi podataka

4.4 Ostale funkcije aplikacije

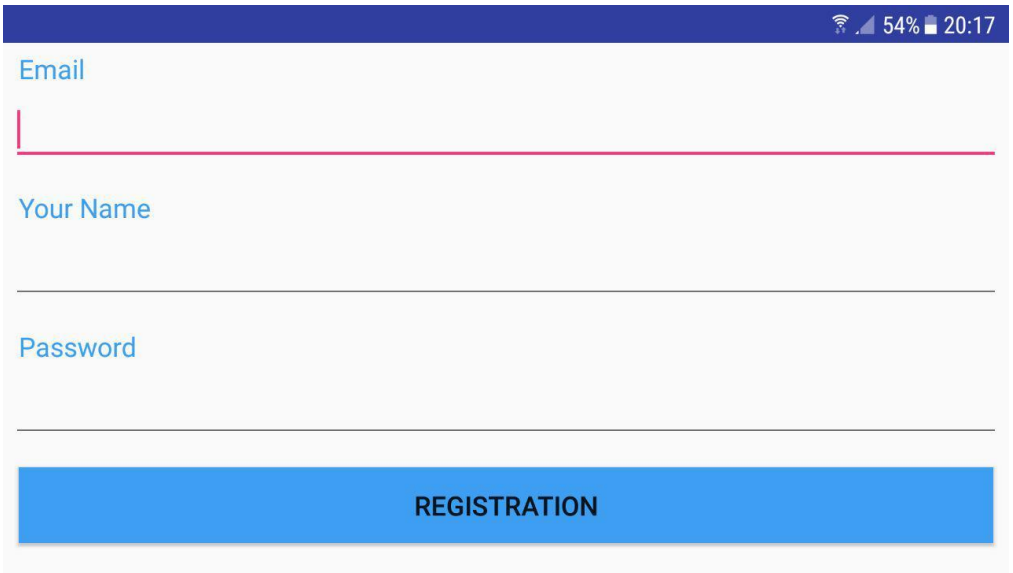
Pored osnovnih funkcija u aplikaciji postaje dodatne aktivnosti kao što su:

- registrovanje korisnika
- logovanje korisnika
- spisak svih korisnika
- zaštita podataka
- prenos podataka

4.4.1 Registrovanje korisnika

Korisnik pomoću forme upisuje email, ime i šifru. Ovi podaci se upotrebom AES algoritma pod određenim ključem enkriptuju i šalju baznom modulu. Slanjem POST zahteva php skripti create_user.php upisuju se email, name i password korisnika. U ovoj skripti se preuzimanjem parametara izvršava SQL upit za upisivanje (npr. INSERT INTO `users` (`email`, `name`, `pass`) VALUES ('$email', '$name', '$password')).

Završetkom registracije vraća se TAG_SUCCESS koji sa vrednosti 1 kreira novi Intent (LoginActivity) i korisnik se prebacuje na layout login na kome se nalazi forma za logovanje, dok je vrednost 0 označava da je korisnik već registrovan. Prikaz ovog prozora na Slici 15.

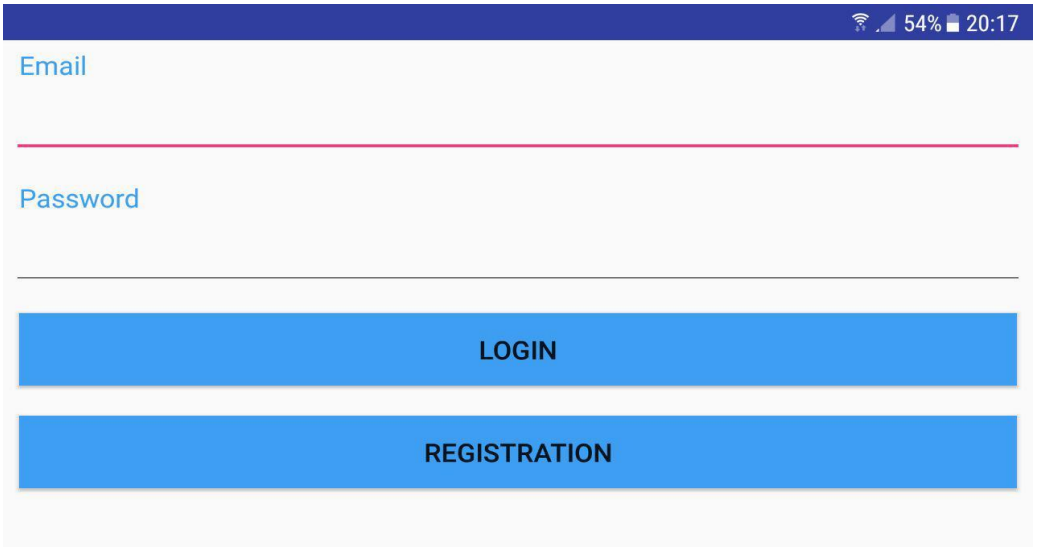


Slika 15. Forma za registrovanje korisnika

4.4.2 Logovanje korisnika

Već registrovani korisnici imaju mogućnost logovanja. Ovo predstavlja početnu aktivnost aplikacije. Podaci koji se unesu se enkriptuju sa istim algoritmom kao u registracionoj formi . U komunikaciji sa baznim modulom uneseni podaci se upoređuje sa već enkriptovanim podacima

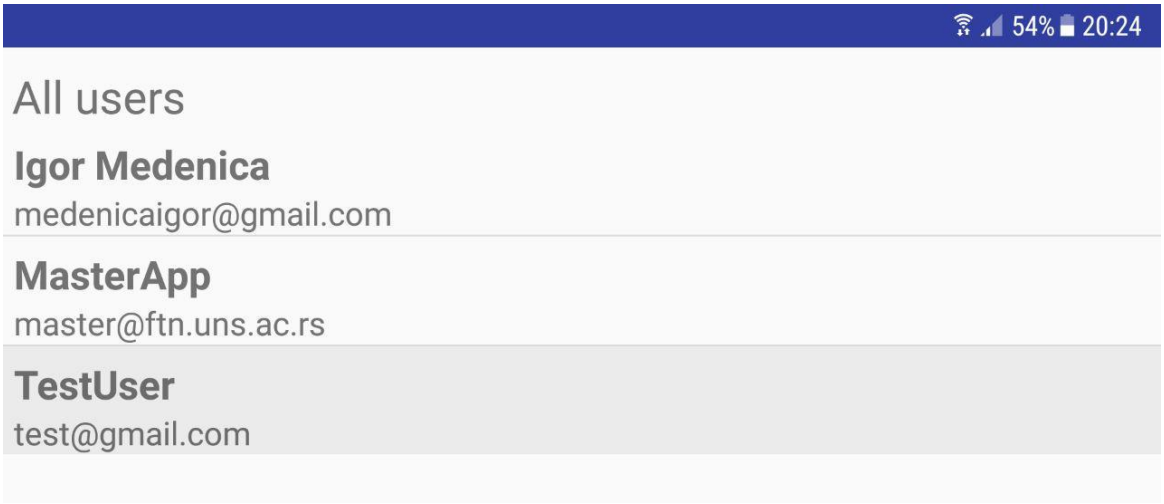
koji su smešteni u udaljenom delu aplikacije. Komunikacija sa Mysql bazom podataka i aplikacijom izvršava se pomoću php skripte koja je smeštena na serveru pod nazivom login.php.



Slika 16. Forma za logovanje korisnika

4.4.3 Spisak svih korisnika

Ovaj layout je sličan layout-u spisak svih kamera samo što se komunikacija vrši pomoću php skripte all_users.php. Pokretanjem ovog dela aplikacije pokreće se Activity koji u pozadini učitava podatke iz baze podataka (LoadAllUsers). Dobijamo niz korisnika sa parometrima id, email i ime. Kao i do sada što je objašnjeno ako je TAG_SUCCESS vrednosti 1 dobijamo navedeni niz podataka u obliku json strukture, dok za vrednost 0 obaveštavamo korisnika da nema registrovanih korisnika što je nemoguće s obzirom da bi korisnik video listu registrovanih korisnika mora da se uloguje. Prikaz na Slici 17.



Slika 17. Ispis svih registrovanih korisnika

4.4.4 Zaštita podataka

U maju 2002. godine Napredni enkripcijski standard (AES) [13], po prijedlogu Nacionalnog instituta za standarde i tehnologiju Sjedinjenih Američkih Država (NIST), postao je zvanični algoritam za enkripciju dokumenata u federalnim organima, dok je DES više nije bio dovoljno siguran standard. Definisanjem zahteva koje traženi algoritam mora ispunjavati otklonjeni su neki problemi i prigovori koji su postojali prilikom izbora DES-a. Ovaj algoritam je simetričan blok enkripcija sa podržanom dužinom bloka od 128 bita i podržanim ključevima dužine 128, 192 i 256 bita. Upotreba ovog algoritma (AESCrypt.class) u ovoj aplikaciji enkripcija i dekripcija se vrši na strani korisnika. Korišćenjem ovih funkcija pored prosleđivanja stringa, prosleđuje se i ključ koji predstavlja korisnikova email adresa. Primer enkriptovanih podataka korisnika na Slici 18.

id	email	name	pass
1	cd0kWtpfdEqVbob4uJrnLILCr4LY/mJsRDBFGUfTD/Y=	aQLkR56NcSEwf9aHFVd62g==	xwoPyx4iurF48/bZvfRQ2A==
2	RfRkWPi0T8On7kIPew8/EcwT4LQyUOwzJFB1jH1o9LU=	LyAMWaK534jAyUWdpIYM0Q==	NhddHmooHsXANdh3hnRBcA==
3	RhWSEWDjB8cs62WtmjAgBw==	NNQavY8TxwVCN9vE2HHQiQ==	B99/7c/RcGFFv6uvpdoLGw==

Slika 18. Enkriptovani zapisi u tabeli korisnika

4.4.5 Prenos podataka

JSON (JavaScript Object Notation) je jedan od lakših tekstualnih otvorenih standarda dizajniran za čitljivu razmenu podataka. Ekstenzija datoteke s podacima u JSON-ovom formatu je .json, dok je meta oznaka (MIME format) application/json.

JSON je format koji polako zamenjuje XML jer ima nekolicinu prednosti u odnosu na XML. JSON ne koristi tagove pa stoga ima kraći kod koji je lakši za pisanje i razumljiviji za čitanje. Mada je najbitnija prednost JSON-a u odnosu na XML je ta što se JSON parsira kroz standardnu JavaScript funkciju dok se XML parsira kroz XML parser.

Kroz php smo koristili dve funkcije JSON-a:

- json_encode()
- json_decode()

Obe funkcije rade sa UTF-8 kodiranjem.

4.4.5.1 json_encode()

Ova funkcija uzima niz i vraća string reprezentaciju argumenta u JSON-ovom formatu ili false ako je došlo do greške. Parametri funkcije su:

• mixed $value je podatak koji treba da se formatira u JSON-ov format, može biti običan niz ili dvodimenzionalni niz kodiran sa UTF-8

• $options je princip kojeg se funkcija pridržava pri formatiranju. Ukoliko se ne navede koristi defaultni normal princip. Može biti: JSON_HEX_QUOT, JSON_HEX_TAG, JSON_HEX_AMP, JSON_HEX_APOS, JSON_NUMERIC_CHECK, JSON_PRETTY_PRINT, JSON_UNESCAPED_SLASHES,

JSON_FORCE_OBJECT, JSON_PRESERVE_ZERO_FRACTION, JSON_UNESCAPED_UNICODE, JSON_PARTIAL_OUTPUT_ON_ERROR

4.4.5.2 json_decode()

Ova funkcija na osnovu string reprezentacije argumenta u JSON-om formatu formira odgovarajući objekat/asocijativni niz ili vraća NULL u slučaju greške.

Parametri funkcije su:

• json_string − encodirani niz slova koji mora da bude UTF-8 kodiran.

• assoc − je bolean-ova vrednost. Kada je TRUE vraća niz, a u suprotnom vraća objekat

• depth − ceo broj koji specificira dubinu rekurzije

• options − Ceo broj type bitmask of JSON decode, JSON_BIGINT_AS_STRING is supported.

5. Testiranje

Rađena su tri dela testiranja aplikacije. U prvom delu se radi testiranje komunikacije između Android aplikacije i baze podataka. Drugi deo testiranja obuhvata proveravanje rada kamere i memorisanje fajlova u memoriju. I u trećem delu testiranja se proverava rad Http servera. Radnog okruženje u kojem je vršeno testiranje je Android Studio.

Nakon ispitivanja ovih funkcionalnosti aplikacije, na osnovu dobijenih rezultata, može se zaključiti da svi modulu ispravno funkcionišu i da je izabran ispravan mehanizam komunikacije izemđu njih.

5.1 Opis radnog okruženja

Tokom razvoja klijentskog i serverskog dela, korišćeno je okruženje Android Studio i MySQL kao baza podataka. Android Studio je oficijalni IDE za razvoj Android aplikacija, koji se zasniva na poznatom okruženju IntelliJ IDEA [16]. Tokom razvoja sistema pojavila se nova verzija Android Studio, dok mi koristimo poslednju korišćenu verziju Android Studio-a.

Phpmyadmin je php alatka napravljena za potrebe administriranja MySQL sistema baza podataka putem veba. Ovaj alat se stalno razvija i trenutno postoji na četrdeset sedam različitih jezika. Funkcije koje ovaj alat omogućava su:

- kreiranje baze podatak
- brisanje baze podataka
- izvođenje operacija nad tabelama
- izvođenje operacija nad poljima
- rad sa sql upitima
- rukovođenje sa ključevima nad poljima

5.2 Komunikacija između Android aplikacije i baze podataka

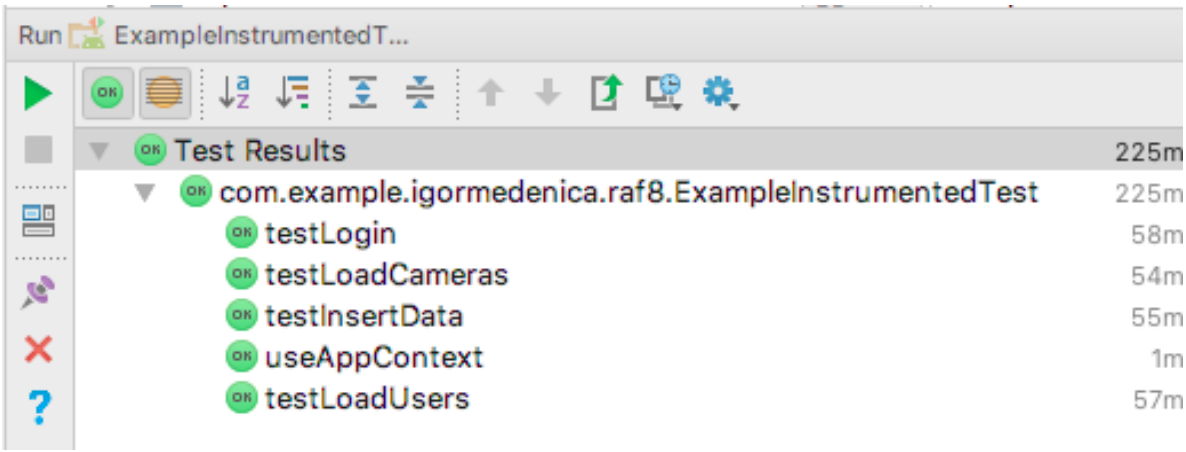
Komunikacija između Android aplikacije i baze podataka je ključna u radu celog sistema koji je opisan u predhodnim poglavljima. Kako bi se u potpunosti ispitala ispravnost komunikacije i upisivanje podataka u bazu podataka, potrebno je ispitati slanjem GET i POST zahteva sa određenim parameterima.

Naziv testa	Opis testa
testInsertData()	Testiranje dela aplikacije za registraciju korisnika. U ovom delu proverava se rad PHP skripte create_user i pravilno prosleđivanje parametara iz aplikacije. Ovakav tip testa važi i za upisivanje novih aktivnih prenosa, samo što se upisivanje vrši u drugu tabelu.
testLogin()	Test logovanja korisnika na aplikaciju. Ovim testom se proverava rad PHP skripte login, kao i komunikacija između Android aplikacije i baze podataka.
testLoadUsers()	Test učitavanja svih registrovanih korisnika u aplikaciji. Slanjem GET zahteva proverava se da li postoji i jedan registrovan korisnik ili je tabela prazna.
testLoadCameras()	Ovim testom se proverava učitavanje svih aktivnih prenosa slanjem GET zahteva PHP skripti get_all_cameras

Tabela 12. Tabela testova komunikacije Android aplikacije i baze podataka

Na sledećoj slici (Slika 19.) prikazano je uspešno testiranje komunikacije između Android aplikacije i MySQL baze podataka.

Svi testovi koji su rađeni ujedno su proveravali ispravnost enkriptovanja podataka i prilikom učitavanja podataka iz baznog dela koristi se funkcija za dekriptovanje.



Slika 19. Testovi komunikacije između Android aplikacije i baznog modula

5.3 Rad sa kamerom

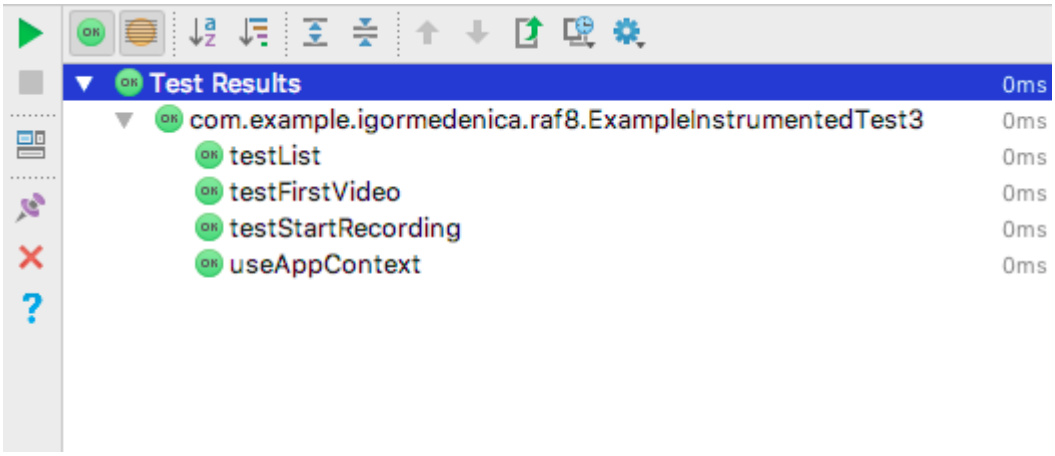
Implementacija funkcionalnosti rada sa kamerom je smeštena na uslužiočev modul. Pomoću ovog modula započinje se prenos video materijala u lokalnoj mreži.

Kako bi u potpunosti ispitali ispravnost ovog modula potrebno je testirati funkcije za započinjanje snimanja i prenosa video sadržaja, snimanje videa u memoriju uslužioca i kreiranje nove video liste na svakih pet snimljenih videa. U sledećoj tabeli opisani su testovi (Tabela 13.).

Naziv testova	Opis
testStartRecording()	Ovim testiranjem se proverava započinjanje snimanja videa na strani uslužioca. Prilikom pritiska dugmeta “Start Stream” poziva se funkcija startRecordingVideo()
testFirstVideo()	Ovaj test proverava da li se video sadržaja snima u memoriju na svakih pet sekundi. Video se snima u internu memoriju uslužiočevog Android uređaja u mp4 formatu.
testList()	Ovim testom je ispitano snimanje nove video liste na svakih pet videa. Prilikom kreiranja nove liste briše se prethodni sadržaj i kreira se novi tekstualni fajl pod istim imenom.

Tabela 13. Tabela testova rada sa kamerom

Na sledećoj slici (Slika 20.) su prikazani svi testovi za proveru rada Android aplikacije sa kamerom.



Slika 20. Prikaz svih testova za rad sa kamerom

5.4 Rad Http servera

Mogućnost prenosa multimedijalnih sadržaja sa jednog uređaja na drugi u lokalnoj mreži zahteva postojanje servera. Korišćenjem NanoHTTPD biblioteke omogućen je prenos MPEG-4 i TXT fajlova.

Ispitivanja ovih funkcionalnosti obuhvata više različitih ispitanih slučajeva:

- rada servera
- zahtevanja i ispravnost TXT liste video putanja
- zahtevanje video sadržaja

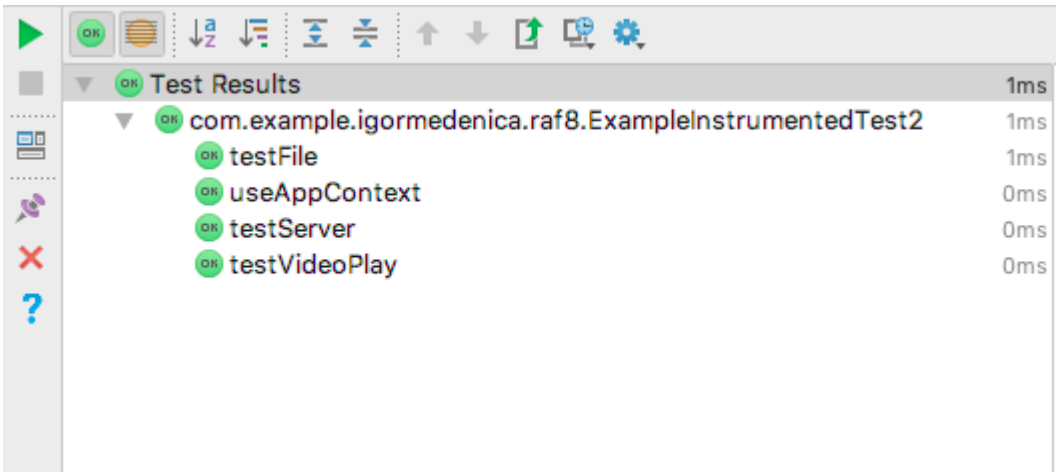
Ovim testovima se ispituje komunikacija i rad serverskog dela ili uslužioca i klijentskog dela Aplikacije. U sledećoj tabeli prikazane su testovi (Tabela 14.).

Naziv testova	Opis
testServer()	Proverava se kreiranje servera pri pokretanju video i audio prenosa na portu 8089
testFile()	Šalje se zahtev serveru za playlist.txt koja sadrži putanje do videa. Proverava se sadržaj tekstualnog fajla kod klijenta i uslužioca.

testVideoPlay()	Na osnovu ispitivanja sadržaja tekstualnog fajla, ispitujemo zahtevanje i puštanje video sadržaja. Video se reprodukuje pomoću ExoPlayer-a.

Tabela 14. Tabela testova rada http servera

Na sledećoj slici (Slika 21.) su prikazani svi testovi rada NanoHTTPD servera.



Slika 21. Testovi provere rada http servera

6. Zaključak

Ovaj rad se bavi jednim rešenjem prenosa video i audio datoteka upotrebom P2P tehnologije. Ovakav vid komunikacije baziran je između dva ili više Android pametnih telefona. Da bi bio olakšan proces uspostavljanja komunikacije između uređaja, koristi se udaljeni server sa bazom podataka. Prvo se beleže ip adrese koje su dodeljene od strane rutera u lokalnoj mreži.

Kompletno rešenje, koje je opisano u ovom radu, za prenos video datoteka sa integrisanim audio zapisima korišćenjem tekstualnih fajlova u obliku playliste sastoji se od tri dela ili modula:

Klijentski deo – Android deo aplikacije koji prilikom logovanja omogućava klijentu da gleda nove prenose video i audio sadržaja u realnom vremenu, pregled svih prenosa ukoliko postoji više i pregled svih registrovanih korisnika.

Serverski deo – Ovaj deo Android aplikacije predstavlja započinjanje snimanja i prenosa ovog sadržaja, pri kome se pokreće lokalni server sa pristupom njegovom fajl sistemu.

Bazni deo – Omogućava ispisivanje svih informacija u klijentskom delu aplikacije, kao i upisivanje novih korisnika u bazu podataka i provera postojanja u istoj. Pre komunikacije klijenta i servera, podaci o serveru se upisuju u bazu podataka na osnovu koje klijent saznaje adresu i nastavlja dalju komunikaciju sa serverom. Svi zapisi u bazi podataka su enkriptovani AES algoritmom.

Ideja za prenos video i audio sadržaja u realnom vremenu nije nova i postoji dosta implementacija koje se koriste kao open source i komercijalne verzije. Ovim radom je pokazan primer implementacije jednog sistema upotrebom Java programskog jezika i PHP skript jezika. Zbog ne klasičnih zahteva aplikacije u upotrebu su ušle javno dostupne biblioteke koje nisu osnovojnoj ponudi Android biblioteka. Zbog nove verzije Android sistema (API 24, Nougat verzija: 7.0) promenjen je način upotrebe kamere zbog čega je korišćena nova biblioteka

Camera2 API, dok na korisničkoj strani, zbog puštanja video i audio datoteka, iskorišćena je ExoPlayer biblioteka. Kao lokalni fajl server sa odgovarajućim funkcijama upotrebljena je NanoHTTPD biblioteka.

Dalji razvoj ovakvog sistema mogao bi se odnositi na promenu tekstualne datoteke sa svim putanjama, do željenih datoteka sa M3U playlist-om koju automatski server prepoznaje i pušta zabeležene materijale u realnom vremenu sa minimalnim kašnjenjem. Ovaj predlog bi mogao da se realizuje konvertovanjem MPEG-4 fajla u FMPEG-4 (Fragmented MPEG-4) koji je podržan od strane ove vrste protokola.

Još jedna mogućnost daljeg razvoja na osnovu ispitivanja i testiranja rada ovog sistema je dodavanje mogućnosti pristupa direktnom prenosu bez snimanja tekstualnih datoteka. Mogućnost rešavanje je upotreba DataSource u ExoPlayeru plejeru, čime ostvarujemo direktnu komunikaciju sa minimalnim kašnjenjima.

7. Literatura

[1] BitTorrent, http://www4.ncsu.edu/~kksivara/sfwr4c03/projects/4c03projects/NELake-Project.pdf , poslednji put posećeno 17. Decembra 2017.

[2] Skype, https://krutikakamilla.wordpress.com/2012/10/23/skype-technically/ , poslednji put posećeno 17. Decembra 2017.

[3] Andrew Biggadike, Daniel Ferullo, Geoffrey Wilson, and Adrian Perrig. NATBLASTER: Establishing TCP connections between hosts behind NATs. In *ACM SIGCOMM Asia Workshop*, Beijing, China, April 2005.

[4] Jeffrey L. Eppinger, "TCP connections for P2P apps: A software approach to solving the NAT problem", Technical Report CMU-ISRI-05-104, Carnegie Mellon University, January 2005.

[5] https://media.fb.com/2016/04/12/introducing-the-facebook-live-api/

[6] N. Efthymiopoulos, S. Tombros, A. Christakidis, Spyros Denazis, "Enabling live video streaming services realization in telecommunication networks using P2P technology", 2011.

[7] C. R. Plouffe, "Examining "peer-to-peer" (P2P) systems as consumer-to-consumer (C2C) exchange", 2008.

[8] dr Ištvan Pap, dr Nemanja Lukić, "Projektovanje I arhitekture softverskih sistema", FTN Izdavaštvo, Novi Sad, 2015.

[9] Sve o P2P arhitekturi ,http://www.webopedia.com/DidYouKnow/Internet/2005/peer_to_peer.asp, poslednji put posećeno 17. Decembra 2017.

[10] Prednosti i nedostaci P2P mreža, http://freedom-to-tinker.com/2005/10/10/cost-tradeoffs-p2p/ , poslednji put posećeno 17. Decembra 2017.

[11] TCP/IP protokol, http://es.elfak.ni.ac.rs/rmif/Materijal/TCP-IP.pdf , poslednji put posećeno 17. Decembra 2017.

[12] AES algoritam, https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf , poslednji put posećeno 17. Decembra 2017.

[13] Camera2 API, https://developer.android.com/reference/android/hardware/camera2/package-summary.html , poslednji put posećeno 17. Decembra 2017.

[14] "NanoHTTPD", https://github.com/NanoHttpd/nanohttpd, poslednji put posećeno 17. Decembra 2017.

[15] "ExoPlayer", https://google.github.io/ExoPlayer/guide.html, poslednji put posećeno 29. Oktobra 2017.

[16] IntelliJ IDEA, https://www.jetbrains.com/idea/features/ , poslednji put posećeno 17. Decembra 2017.