Izrada Android aplikacije za prijenos video signala

Benke, Ivan

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:890532

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ANDROID APLIKACIJA ZA PRIJENOS VIDEO SIGNALA

Diplomski rad

Ivan Benke

Osijek,2018.

SADRŽAJ

1. UVOD 1
1.1 Zadatak rada 1
2. ANDROID OPERACIJSKI SUSTAV 2
2.1 Povijest 2
2.2 Verzije android operacijskog sustava 2
2.3 Arhitektura 3
3. RAZVOJNI ALATI I TEHNOLOGIJE 7
3.1 Android Studio 7
3.2 SDK 7
3.3 Video 8
3.4 H.264 9
3.5 Flv 9
3.6 RTMP 9
3.7 HLS 10
3.8 Nginx 11
4. ANDROID APLIKACIJA 12
4.1 Kamera 12
4.2 Manifest deklaracija 12
4.3 Priprema razvojnog okruženja 13
4.4 Razvoj aplikacije 16
5. SERVERSKA APLIKACIJA 23
5.1 Kreiranje RTMP servera 23
5.2 Kreiranje web stranice 26
6. ZAKLJUČAK 28
LITERATURA 29
SAŽETAK 30
ABSTRACT 31
ŽIVOTOPIS 32

1. UVOD

Internet postaje sve brži i dostupan je svugdje, pa je tako omogućen i prijenos video signala visoke kvalitete. Sve je više servisa koji pružaju uslugu prijenosa videa kao što su Netflix, Twitch, Youtube. U zadnjih par godina na većim platformama koje se bave dijeljenjem videa, implementirana je mogućnost za prijenos videa uživo. Uz internet također su rasprostranjeni i mobilni uređaji koji imaju kvalitetne kamere i osim za slikanje i snimanje mogu se koristiti za razne namjene. Spojimo li sve zajedno moguće je napraviti vlastitu aplikaciju za prijenos video signala na neki udaljeni poslužitelj (engl. Server) i od poslužitelja do bilo kojeg klijenta (engl. Client).

1.1 Zadatak rada

Potrebno je opisati mogućnosti i način pristupa ugrađenoj kameri "pametnog" telefona/tableta. Napraviti aplikaciju za distribuciju video signala prema drugim uređajima na lokalnoj ili globalnoj mreži. Aplikacija se pokreće na android uređaju te se odabire ime prijenosa (engl. Stream) i uspostavlja konekcija sa poslužiteljem. Poslužitelj prima video i distribuira ga na druge izvore, u ovom slučaju ga šalje na web stranicu. Klijenti pristupaju stranici i imaju mogućnost gledanja video prijenosa.

2. ANDROID OPERACIJSKI SUSTAV

Android je prvi sustav otvorenog koda na mobilnim uređajima. Dizajniran je za mobilne uređaje i temelji se na Linux kernelu. Najviše se pojavljuje na pametnim telefonima i tabletima, a proširio se i na televizore, satove, automobile, igraće konzole, fotoaparate i ostalu elektroniku. Kod pametnih telefona i tableta odnosno uređaja osjetljivih na dodir, koriste se geste koje su slične sa aktivnostima koje se koriste u stvarnom svijetu. Najviše se koristi dodirivanje i manipulacija objektima koji su prikazani na zaslonu,a tu je i korištenje virtualne tipkovnice s kojom se unosi tekst [1].

2.1 Povijest

Chris White, Nick Sears, Rich Miner i Andy Rubin su osnovali Android Inc. u listopadu 2003. godine. Cilj im je bio da razvijaju programe koje bi koristili pametni mobilni uređaji, a u obzir bi uzimali korisnikovu lokaciju i postavke. Poslije 2. godine razvoja kupljeni su od strane Googlea. Android je predstavljen 5. Studenog 2007. godine zajedno s osnivanjem OHA (engl. Open Handset Alliance). Cilj OHA je da za mobilne uređaje stvori javni standard. Google je okupio 34 različite tvrtke, a neke od njih bave se proizvodnjom mobilnih telefona, programiranjem aplikacija, telekomunikacijskim mrežama i slično. Prvi uređaj koji je dolazio sa Android OS-om je T-Mobile G1, koji je proizvodio HTC. Ovim potezom počela je utrka sa ostalim platformama koje su bile dostupne na tržištu, a najvažnije su: Windows Phone, iOS, Symbian. Google je želio da Android bude otvoren i besplatan pa je cjeloviti kod koji je licenciran Apache licencom, dostupan od 21. listopada 2008. godine [1].

2.2 Verzije android operacijskog sustava

Verzija 1.0 i 1.1 nisu objavljene pod određenim kodnim nazivima. Od verzije 1.5 (Cupcake) krenula su kodna imena abecednim redom, a najnovija verzija je Android 9 Pie koji je izašao 6. kolovoza 2018. godine [2].

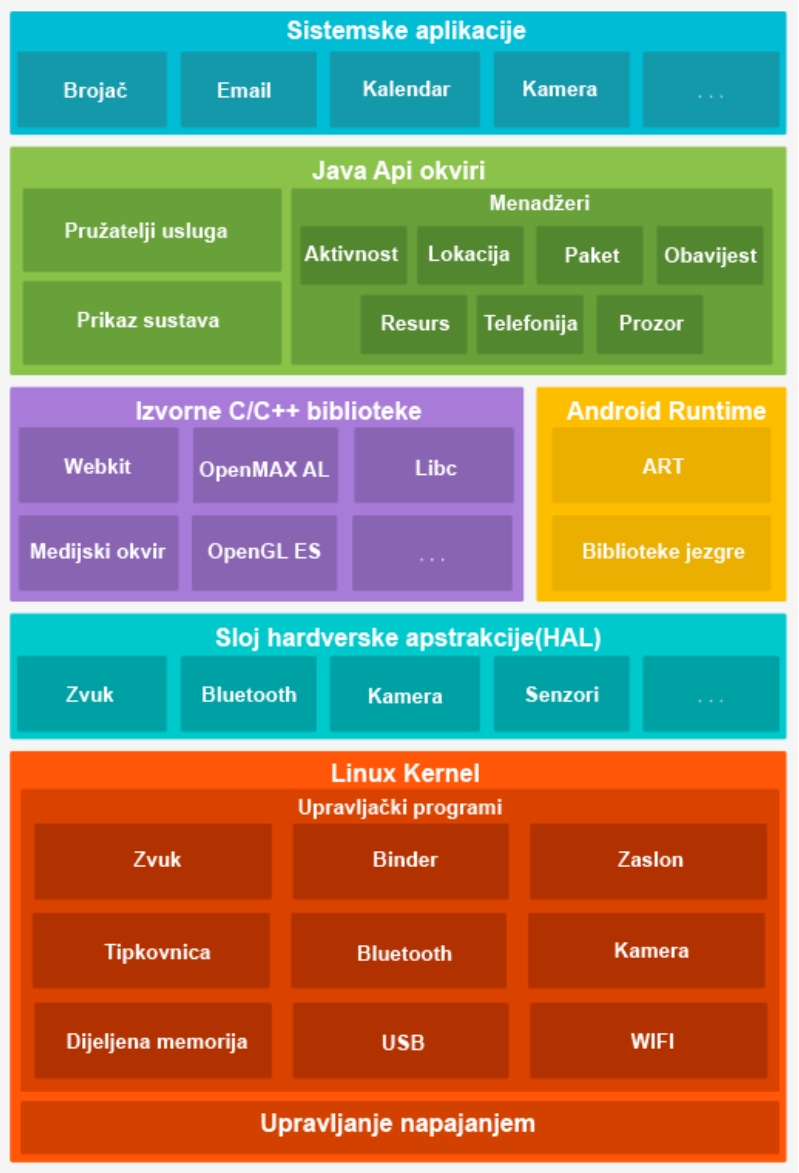
Slika 2.1 prikazuje udio pojedinih verzija android sustava u rujnu 2018. godine. Najzastupljenija verzija je Android 6.0 kodnog imena Marshmallow sa 22,7%, a slijedi ga Android 7.0 Nougat sa 20.4%. Za posljednju verziju još nema dostupnih podataka.

VERZIJA	KODNO IME	API	DISTRIBUCIJA
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.6%
4.3		18	0.5%
4.4	KitKat	19	7.8%
5.0	Lollipop	21	3.6%
5.1		22	14.7%
6.0	Marshmallow	23	21.6%
7.0	Nougat	24	19.0%
7.1		25	10.3%
8.0	Oreo	26	13.4%
8.1		27	5.8%

Slika 1.1. Verzije Android OS-a (izvor: https://developer.android.com/about/dashboards.)

2.3 Arhitektura

Android operacijski sustav sastoji se od nekoliko dijelova i slojeva. Svaki pojedini sloj ima svoje karakteristike i uloge, ali slojevi nisu jasno odvojeni već se najčešće preklapaju jedni sa drugima. Android operacijski sustav baziran je na Linux Kernel 2.6 jezgri, a napisana je u C ili C++ jeziku. Razlog njegove upotrebe je laka prenosivost, sigurnost i ostale značajke. Nije potrebno brinuti se o hardverskim karakteristikama pojedinih uređaja što omogućuje implementaciju Androida na velik raspon uređaja. Aplikacije su većinom pisane u Javi koristeći Android SDK, a moguće je koristiti i Android NDK pa se pišu u C ili C++ jeziku. Prednost pisanja u C ili C++ je brzina, a mana je složenost pisanja koda. U skorije vrijeme dostupan je novi programski jezik Kotlin koji se izvodi na java virtualnoj mašini. Arhitektura Android operacijskog sustava je prikazana je slikom 1.2 [1].



Slika 1.2. Android arhitektura (izvor: https://developer.android.com/guide/platform.)

Linux Kernel - Temelj Android platforme je Linux kernel. Na primjer Android Runtime (ART) oslanja se na Linux kernel za temeljne funkcionalnosti kao što su threading i low-level memory management [3]. Korištenjem Linux kernela Androidu se omogućuje da koristi ključne značajke sigurnosti i omogućava proizvođačima da razviju upravljačke programe za poznati kernel.

HAL (engl. Hardware Abstraction Layer) - HAL pruža standardno sučelje koje implementiraju proizvođači hardvera. HAL se sastoji od nekoliko biblioteka modula, a svaki modul implementira sučelje za harversku komponentu. Neki od najpoznatijih modula su audio ili kamera modul. Nakon što framework API zatraži poziv da dođe do hardvera uređaja, sustav Android učitava biblioteku modula te hardverske komponente.

ART (engl. Android Runtime) – kod uređaja koji upotrebljavaju android inačicu 5.0 (api 21) ili više, aplikacije se pokreću u vlastitim procesima i imaju vlastite instance ART-a. ART je napisan da pokreće različite virtualne strojeve na uređajima koji imaju nisku razinu memorije izvršavanjem DEX datoteka. Formata dizajniranog posebno za Android koji je optimiziran za minimalni memorijski otisak.

Neke od glavnih značajki ART-a uključuju sljedeće:

- AOT (engl. ahead of time) i JIT (engl. just in time) sastavljanje

- Poboljšana akumulacija smeća (CG)

- poboljšana podrška za uklanjanje pogrešaka

Prije Android verzije 5.0, Dalvik je bio Androidov runtime. Ako aplikacija ispravno radi na ART-u, trebala bi raditi i na Dalviku, ali obrnuto ne mora biti istinito.

Izvorne C/C++ biblioteke - Višejezgrene komponente i usluge Androida, primjerice ART i HAL, izgrađene su na temelju izvornih biblioteka napisanih u C i C++. Android platforma omogućuje Java framework api-je kako bi se u aplikacijama prikazale funkcije nekih od tih izvornih biblioteka. Jedan od primjera je OpenGL ES, kojem je moguće pristupiti kroz Android framework Java OpenGL api, da bi se uključila podrška za crtanje i upravljanje 2D i 3D grafikom u svojoj aplikaciji. Ako izrađujete aplikaciju koja zahtijeva C ili C++ kod, Android NDK možete upotrijebiti za pristup nekom od tih izvornih biblioteka platformi izravno iz izvornog koda.

Java API framework - Cijeli skup značajki Android OS-a je dostupan kroz api-je koji su napisani u Java jeziku. U ovim api-jima forimirani su blokovi koji su potrebni za stvaranje Android aplikacija pojednostavljenjem ponovne uporabe osnovnih, modularnih komponenti sustava i usluga, a uključuju sljedeće:

- Bogat i proširiv sustav prikaza koji možete koristiti za izradu korisničkog sučelja aplikacije, uključujući popise, mreže, teksts okvire, gumbe, pa čak i ugrađeni web preglednik.

- Upravitelj resursa (engl. Resource Manager) omogućuje pristup resursima koji nisu kodovi, npr. lokalizirani nizovi, grafike i datoteke izgleda.

- Upravitelj obavijesti (engl. Notification Manager) koji omogućuje svim aplikacijama prikazivanje prilagođenih upozorenja na statusnoj traci.

- Upravitelj aktivnosti (engl. Activity Manager) koji se bavi životnim ciklusom aplikacija i pruža zajedničku navigaciju unatrag.

- Davatelji sadržaja (engl. Content Providers) koji pruža aplikaciji pristup informacijama iz druge aplikacije, kao što je kontakt aplikacija ili aplikacija za dijeljenje vlastitih podataka.

Razvojni programeri imaju puni pristup istim framework API-jima koje koriste aplikacije Android sustava.

Aplikacije sustava - Android dolazi s nizom osnovnih aplikacija za e-poštu, SMS poruke, kalendar, internet preglednik, kontakti i još mnogo toga. Aplikacije uključene u platformu nemaju posebni status među aplikacijama koje korisnik odabere za instalaciju. Tako aplikacija treće strane može postati zadani SMS messenger, zadana tipkovnica ili internetski pretraživač.

Sustavne aplikacije funkcioniraju i kao korisničke aplikacije i za pružanje ključnih mogućnosti do kojih razvojni programeri mogu doći iz vlastite aplikacije. Npr. ako aplikacija zahtjeva slanje SMS poruku, izgradnja te funkcije nije potrebna, nego je moguće pozvati bilo koju SMS aplikaciju koja se već koristi kako bi poslala poruku.

3. RAZVOJNI ALATI I TEHNOLOGIJE

Skup programskih alata s kojima se kreiraju aplikacije za hardver, operacijski sustav, određene programe, programsko okruženje i slično. Kako bi obogatili aplikacije sa naprednim funkcionalnostima, većina razvojnih programera implementira određene setove za razvoj softvera. Sastoje se od skupine gotovih programskih rješenja koji imaju mogućnost komunikacije sa specifičnom programom ili platformom. Na primjer, razvoj Android aplikacije zahtijeva SDK s Javom, iOS aplikacije iOS SDK s Swiftom itd.

3.1 Android Studio

Android Studio je službeni Android alat i sadrži alate za izradu aplikacija na različitim vrstama android uređaja. Predstavljen u svibnju 2013. godine, a dostupan je od prosinca 2014. godine. Temelji se na programskoj podršci (engl. Software) tvrtke JetBrains i prilagođen je za razvoj Android aplikacija. Sistemski zahtjevi prema tab. 3.1.

Tablica 3.1. Sistemski zahtjevi za android studio

Kriteriji	Opis
OS verzija	Windows 7 ili kasniji Mac OS X 10.9.5 ili kasniji GNOME or KDE desktop
RAM	Minimalno 3 GB rama, a preporučeno 8 GB
Diskovni prostor	500 MB za android studio i minimalno 1.5 GB za android SDK
Java verzija	JDK 8
Rezolucija	1200x800

3.2 SDK

Android SDK (engl. software development kit) je skup razvojnih alata [4]. U to su uključeni emulator, debugger, biblioteke, primjeri koda, dokumentacija i tutoriali. SDK pohranjuje sve podatke o aplikaciji što uključuje izvorni kod, metapodatke i manifest u obliku arhive sa ekstenzijom .apk.

3.3 Video

Video je elektronički medij za snimanje, kopiranje, reprodukciju, emitiranje i prikaz pokretnih vizualnih medija [5]. Video je razvijen za mehaničke televizijske sustave koji su brzo zamijenjeni sustavima katodne cijevi (CRT), a trenutno su u upotrebi LED i LCD televizori. Glavne karakteristike video sustava su rezolucija, omjer slike, brzina osvježavanja, broju boja i ostalo. Postoje digitalne i analogne vrste videa, a neki od najčešćih prijenosnih media za njih su: magnetske vrpce, optički diskovi, računalne datoteke i prijenos video signala putem interneta.

Metoda video kompresije

Ako je video nekomprimiran on pruža maksimalnu kvalitetu, ali da bi to prikazali moramo imati visoku brzinu prijenosa podataka. Postoje različite metode za kompresiju slika, a najučinkoviti način je koristeći skupinu slika (GOP) kako bi smanjili prostornu i vremensku redundanciju. Prostorna redundancija se smanjuje registracijom razlika između dijelova jednog kadra, a još se naziva i intraframe i usko je povezana sa kompresijom slike. Vremenska redundancija se može smanjiti tako da se registriraju razlike između okvira, a ova tehnika se još zove i interframe kompresija. Zadatak GOP-a je da odredi u kojem redoslijedu su uređeni intraframe i interframe. Najpoznatiji koder koji koristi ovaj način komprimiranja je h264.

Video streaming

Multimedijski sadržaj kod kojeg korisnik prima i reproducira podatke istovremeno, dok kod klasičnih metoda korisnik prvo preuzima video u cijelosti i tek onda može izvršiti reprodukciju. Streaming servisi se najviše koriste za gledanje sadržaja uživo putem internetskih servisa kao što su Netflix, Youtube, Twitch itd. Početkom 20. stoljeća povećao se pristup računalnim mrežama, posebice internetu. Ta tehnološka poboljšanja omogućila su prijenos audio i video sadržaja korisnicima računala. Povećanjem pristupa internetu došlo je do sve većeg korištenja standardnih protokola i formata od kojih su najpopularniji HTTP , TCP/IP, HTML. Širokopojasne brzine od 2 Mbit/s ili više preporučuju se za prijenos videozapisa standardne rezolucije bez doživljavanja preskakanja videa, pogotovo za prijenos uživo. 5 Mbit/s je preporučeno za sadržaj visoke rezolucije, a 9 Mbit/s za sadržaj ultra visoke rezolucije. Veličina koja je potrebna za pohranu medija izračunava se pomoću širine pojasa i duljine medija. Video se komprimira pomoću formata video kodiranja kako bi se smanjila veličina datoteke. Formati videokodiranja uključuju VP8 ili VP9, HEVC, i H.264. Kodirani audio i video streamovi sastavljeni su u bitstream

spremniku, kao što su MP4, FLV, WebM i ostali. Bitstream se isporučuje sa servera prema klijentu koristeći transportni protokol. Kao što je RTMP ili RTP.

3.4 H.264

H.264 je industrijski standard za kompresiju videozapisa. Proces pretvaranja digitalnog videozapisa u format koji zauzima manje kapaciteta kada se pohranjuje ili prenosi [6]. Video kompresija (kodiranje) bitna je tehnologija za aplikacije kao što su digitalna televizija, DVD-video, mobilni TV, videokonferencija i prijenos video signala. H.264 video koder provodi procese predviđanja, transformacije i kodiranja kako bi se proizveo komprimirani h.264 bitstream. Važne novosti koje je uveo ovaj format:

- korištenje do maksimalno šesnaest sličica za pronalaženje iduće sličice.

- prošli formati su imali P-frame, I-frame i B-frame sličice, dok h.264 ima P-blok, I-blok ili B-blok. Ta tri dijela se ponašaju kao P-frame, I-frame i B-frame dio sličice.

- filtriranje blockinga prilikom gradnje aproksimacije iduće sličice.

- makroblokovi mogu biti različitih dimenzija od 4x4 do 16x16 i sličice mogu sadržavati makroblokove sa različitim dimenzijama.

3.5 Flv

Flash video je format datoteke kontejnera koji se upotrebljava za isporuku digitalnog video sadržaja. Flash video je dugo vremena bio standardni video format koji je korišten za prijenos video signala na internetu preko RTMP-a, uključujući videozapise na youtubeu i mnogim drugim mjestima. Trenutno se najviše koristi HTML5 (engl. Hypertext Markup Language) dok je podrška za flash video sve lošija.

3.6 RTMP

RTMP (engl. Real Time Messaging Protocol) je izvorno razvila tvrtka Macromedia (sada Adobe) za prijenos zvuka, videozapisa i podataka putem interneta između flash playera i poslužitelja [7]. Prema zadanim postavkama RTMP koristi TCP (engl. Transmission Control Protocol) port 1935. RTMP je dostupan kao otvorena specifikacija za stvaranje proizvoda i tehnologije koja omogućuje isporuku videozapisa, zvuka i podataka u otvorenim formatima AMF, SWF, FLV I F4V koji su kompatibilni s Adobe Flash Playerom.

RTMP je implementiran u tri faze:

- video enkoder za prijenos uživo

- poslužitelj za prijenos uživo i na zahtjev

- klijent za prijenos uživo i na zahtjev

Najveća značajka RTMP-a je prijenos uživo uz podršku enkodera kao što je flash media enkoder, a korisnik može uživati u video produkciji uživo na RTMP poslužitelju i putem kompatibilnog online video playera koji se reproducira na uređajima korisnika.

3.7 HLS

HLS (engl. HTTP Live Streaming) je baziran na temelju MPEG2-TS, a razvio ga je Apple. To je adaptivni protokol za prijenos. U početku je korišten samo kod Apple uređaja, ali se podrška proširila i sada ga podržavaju i drugi uređaji poput pametnih telefona sa Android sustavom. HLS video se isporučuje putem HTML5 web komunikacijskog protokola, najnovijeg online standarda za web sadržaj. HTML5 podržava online video isporuku putem različitih protokola. To uključuje HLS i MPEG-DASH. Protokoli za prijenos videa mogu se gledati kao metode koje definiraju koji je format videa, kako je video komprimiran, veličinu dijeljenog podatka i ostalo. HLS je razvijen kao alternativa flash videozapisu. Tehnički gledano HLS koristi h.264 kodek za kompresiju videozapisa, AAC ili MP3 za audio kompresiju i prenosi signal pomoću MPEG-TS kontejnerskog formata. Prijenos video signala putem HLS-a funkcionira tako da se MP4 video "sjecka" u kratke video dijelove u trajanju od 10 sekundi. Prijenos je opisan korištenjem M3U8 popisa za reprodukciju koji je stvoren od strane HTTP servera. Ovaj popis za reprodukciju, koji se još naziva i manifest datoteka, indeksira dijelove videozapisa. Popis se šalje klijentu I na osnovu njega prikazuje video.

Usporedba RTMP I HLS:

- RTMP je starija tehnologija koja je rasprostranjena i jednostavna je za korištenje.

- Niska latencija u odnosu na HTTP tehnologije

- HLS pruža visoku kvalitetu slike zahvaljujući bitrateu koji se prilagođava ovisno o brzini internet veze.

- HLS je podržan na gotovo svim uređajima za razliku od RTMP-a, ovo se posebno odnosi na mobilne uređaje.

3.8 Nginx

Nginx je besplatan, open-source HTTP poslužitelj visokih performansi i reverse proxy, kao i IMAP /POP3 proxy poslužitelj. Poznat je po svojim visokim performansama, stabilnosti, jednostavnoj konfiguraciji i maloj potrošnji resursa. Kreirao ga je Igor Sysoev i prvi je puta predstavljen 2004. godine. Kompanija sa istim imenom pokrenuta je 2011. godine kako bi pružila podršku I plaćeni Nginx plus softver. Za razliku od tradicionalnih poslužitelja, Nginx se ne oslanja na niti za obradu zahtjeva, umjesto toga rabi asinkronu arhitekturu događaja. Ova arhitektura koristi male, ali predvidljive količine memorije pod opterećenjem. Čak i ako ne očekujete da obrađujete tisuće zahtjeva istovremeno, i dalje se može iskoristiti Nginx-ov visoki učinak i mali memorijski otisak. Nginx se prostire u svim smjerovima, od malih VPS-a pa sve do velikih klaster poslužitelja.

4. ANDROID APLIKACIJA

Aplikacija je zamišljena za upotrebu na globalnoj računalnoj mreži. Nakon pokretanja aplikacije na android uređaju, korisniku se otvara početni aktiviti u kojem su dostupne informacije o stranici na kojoj se video signal može vidjeti i odabir imena prijenosa. Pritiskom na gumb otvara se aktiviti kamere, u kojem je naveden RTMP kanal na koji se video prijenos šalje i gumb s kojim se započinje slanje video signala.

4.1 Kamera

Android framework uključuje podršku za različite kamere i značajke kamere dostupnih na uređajima, što omogućuje snimanje slika i videozapisa u aplikacijama. Klasa kamera koristi se za postavljanje snimanja slike, start/stop pregleda, snimanje slika i preuzimanje okvira za kodiranje videozapisa.

4.2 Manifest deklaracija

Prije početka razvoja programa pomoću API-ja za kameru, treba provjeriti ima li manifest odgovarajuće deklaracije kako bi se omogućila upotreba hardvera fotoaparata i drugih srodnih značajki. U ovom slučaju osim deklaracije za kameru potrebna nam je i deklaracija za pristup internetu.

```
1.  <uses-permission android:name="android.permission.CAMERA" />
2.  <uses-feature android:name="android.hardware.camera" />
3.  <uses-feature android:name="android.hardware.camera.autofocus" />
```

Dozvola za kameru (engl. Camera Permission) – android.permission.CAMERA aplikacija mora zatražiti dopuštenje za korištenje kamere uređaja. Ako se kamera koristi pozivajući se na postojeću aplikaciju onda ovo dopuštenje ne treba tražiti.

Značajke kamere (engl. Camera Features) – aplikacija također mora sadržavati izjavu o korištenju značajki fotoaparata. Dodavanje značajki kamere u manifest uzrokuje da Google Play spriječi instaliranje aplikacije na uređaje koji ne sadrže kameru ili ne podržavaju navedene značajke kamere. Android.hardware.camera - aplikacija upotrebljava kameru na stražnjoj strani uređaja. Uređaji koji imaju kameru samo na prednjoj strani uređaja ne navode tu značajku. Android.hardware.camera.autofocus - aplikacija koristi značajku automatskog fokusiranja koju kamera podržava.

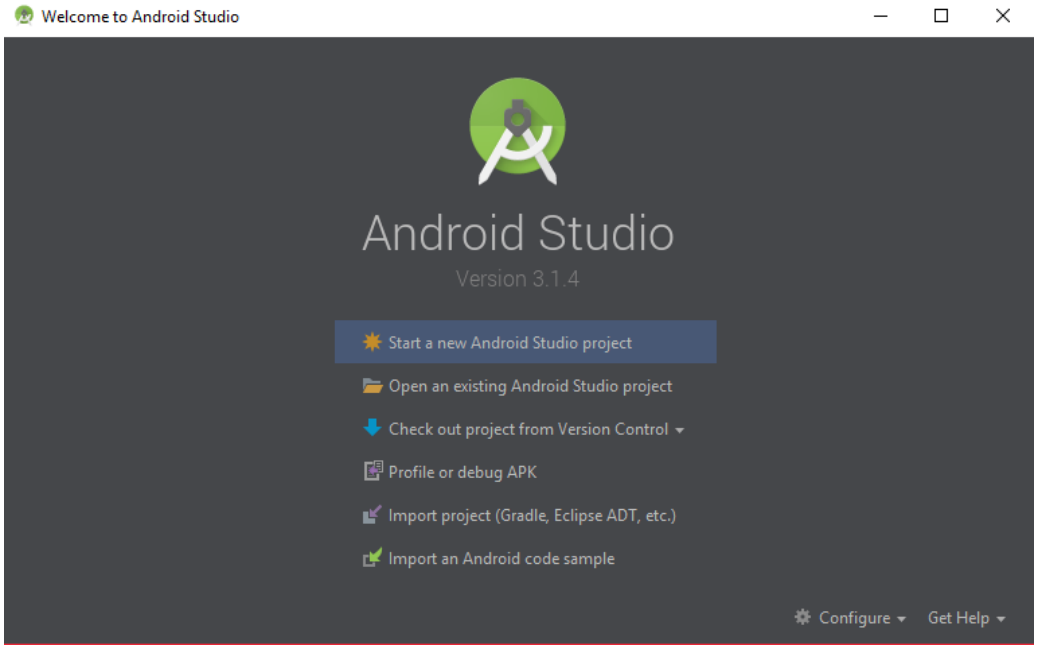
```
1.  <uses-permission android:name="android.permission.INTERNET" />
```

Da bi se moglo pristupiti internetu i izvršiti mrežne operacije, manifest mora sadržavati internet dopuštenje.

4.3 Priprema razvojnog okruženja

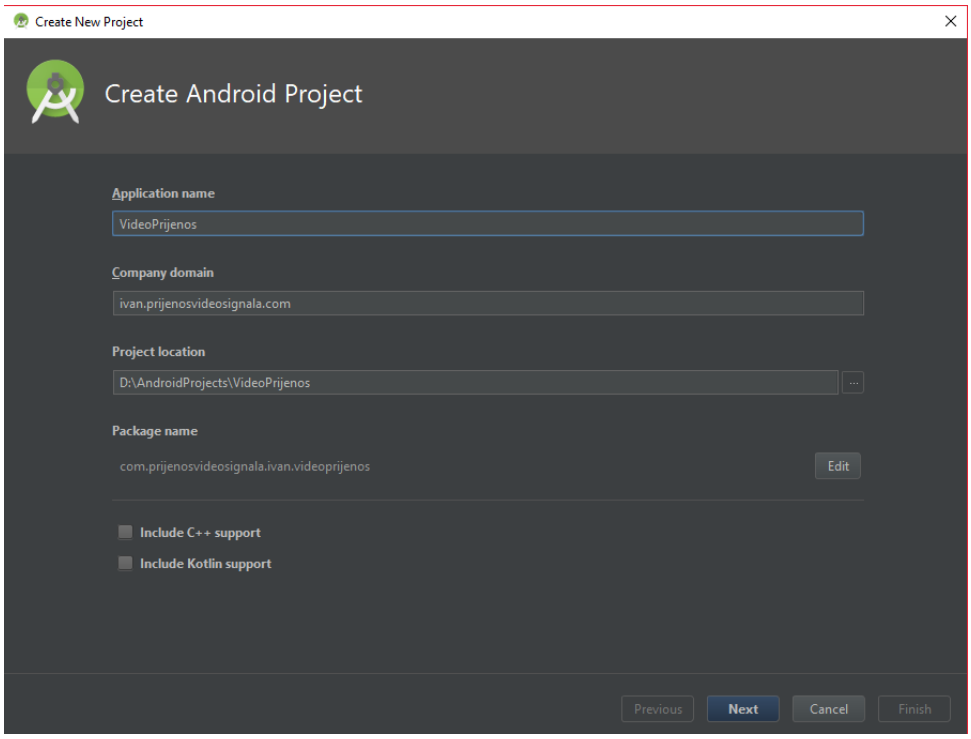
Za razvoj aplikacije potreban je Android Studio koji već sadrži Android SDK alate. Testiranje aplikacije nije moguće izvesti na emulatoru nego jedino na stvarnom uređaju, zbog toga što je potreban pristup kameri. Na Android uređaju potrebno je omogućiti debugging opciju kako bi se moglo instalirati i testirati aplikaciju. Spajanje uređaja sa računalom vrši se uz pomoć USB kabla. Testiranje je izvršeno sa mobitelom (api 26), tabletom (api 24) i tabletom (api 21).

Nakon otvaranja Android Studia odabire se ponuđena opcija za kreiranje novog projekta prema slici 4.1.



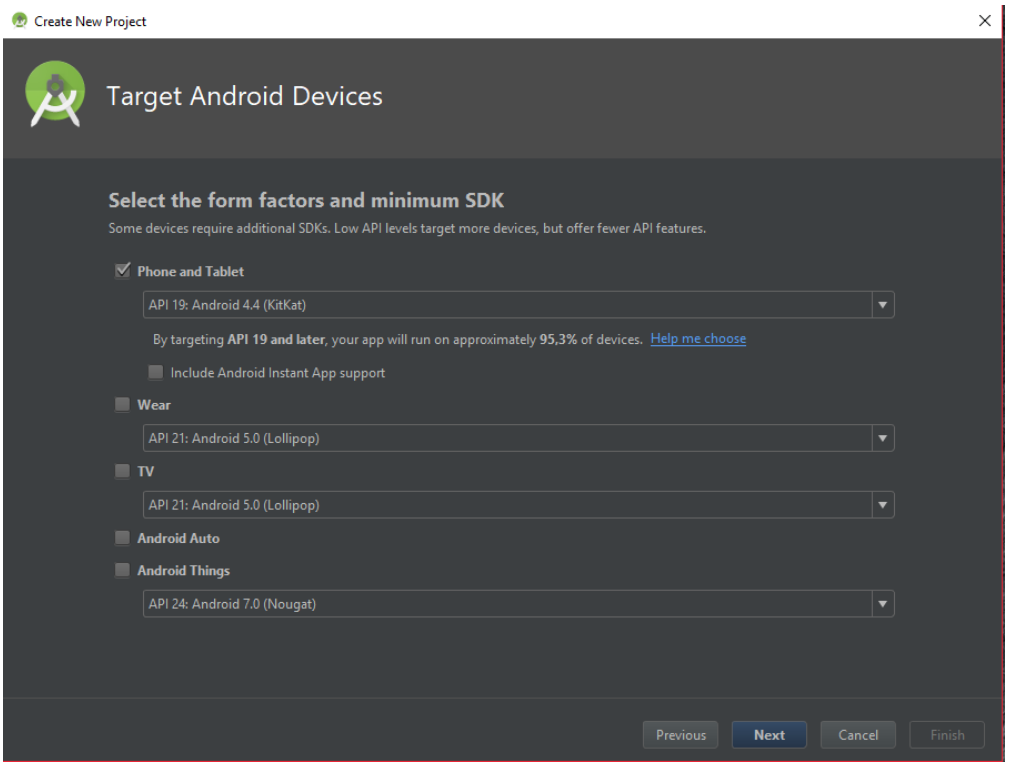
Slika 4.1. Android Studio

Za kreiranje novog projekta potrebno je odrediti njegovo ime (koje mora početi s velikim slovom) i domenu kompanije, koja je jedinstvena kod svakog projekta koji se nalazi na Google Play-u. Još je potrebno odabrati lokaciju na koju će projekt biti spremljen (Sl 4.2.). Ime aplikacije je VideoPrijenos, a ime domene ivan.prijenosvideosignala.com.



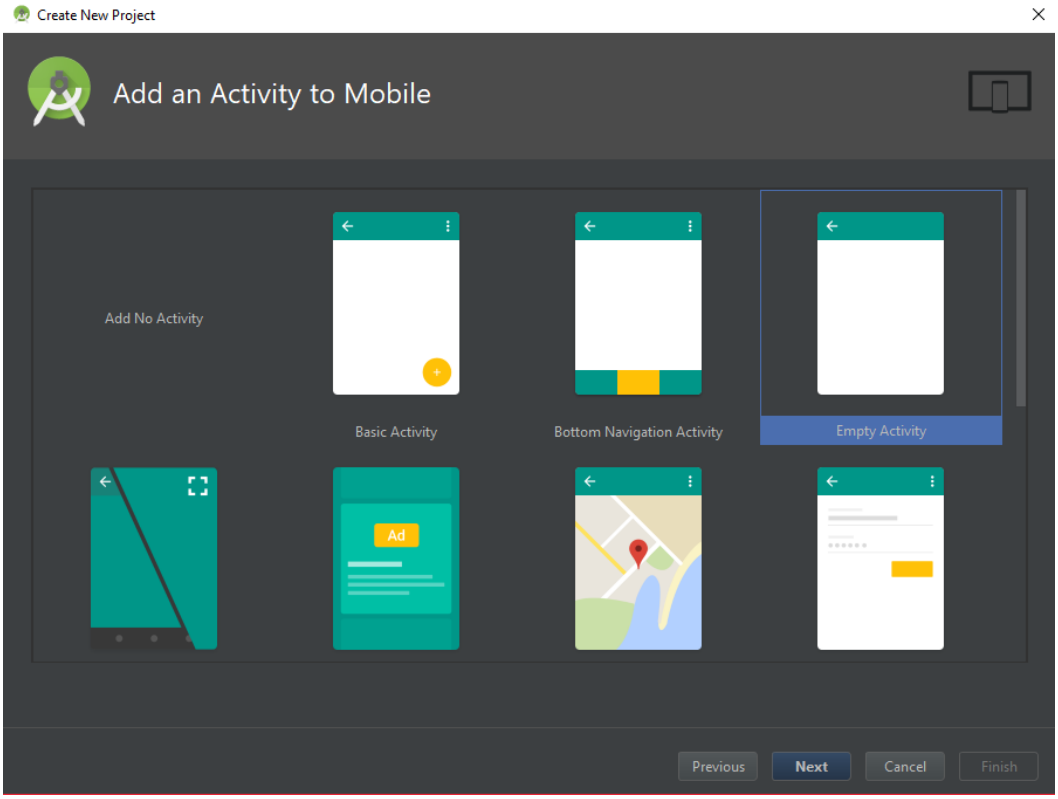
Slika 4.2. Konfiguracija imena, domene i lokacije

Zatim je potrebno odabrati minimalni SDK level na kojem će aplikacija raditi(Sl 4.3.). U ovom slučaju odabran je Android 4.4 (API level 19) i to pruža pristup 95,3% svih uređaja.



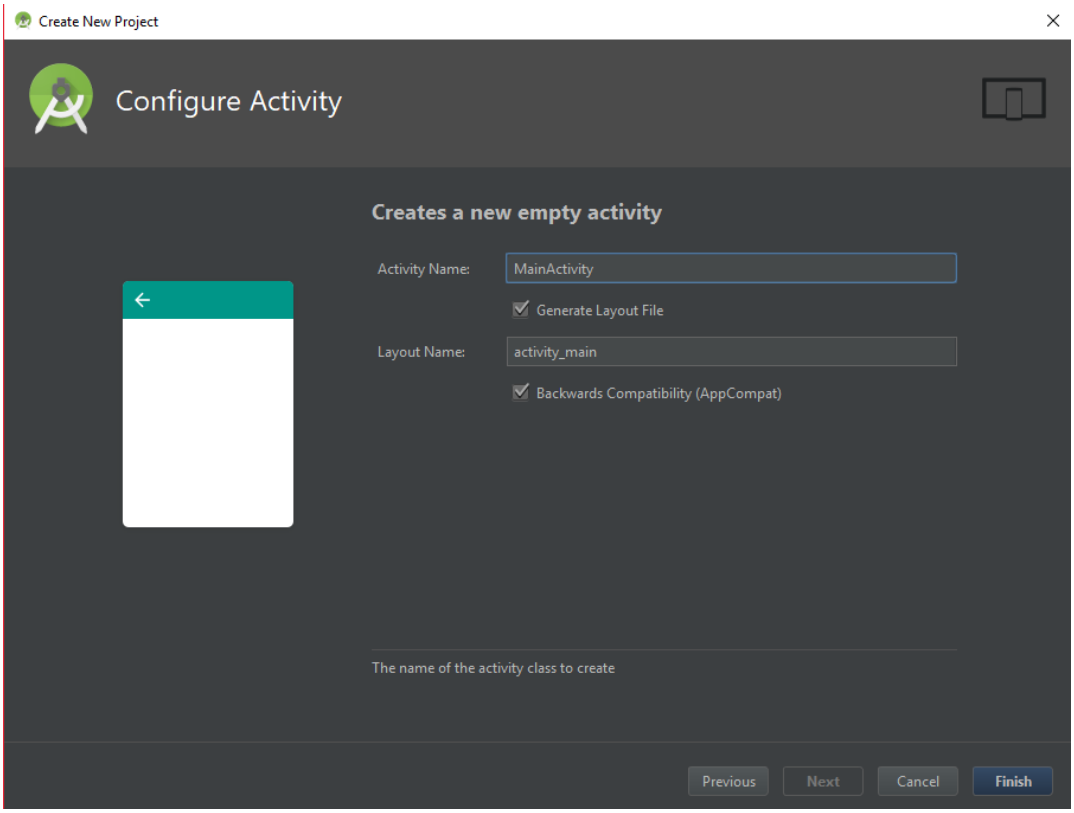
Slika 4.3. Odabir minimalnog SDK

Nakon toga potrebno je odabrati neku od već gotovih aktivnosti (engl. Activity) (Sl 4.4.) ili jednostavno otvoriti projekt bez klase aktivnosti i naknadno ju dodati.



Slika 4.4. Odabir activity-a

U slučaju da je aktivnost odabrana, a ovdje je odabrana prazna aktivnost (engl. Empty Activity), otvara se prozor(Sl 4.5.) u kojem se definira ime klase i ime layouta. Ime klase je MainActivity, a ime layouta activity_main.



Slika 4.5. Odabir imena activity klase i layouta

4.4 Razvoj aplikacije

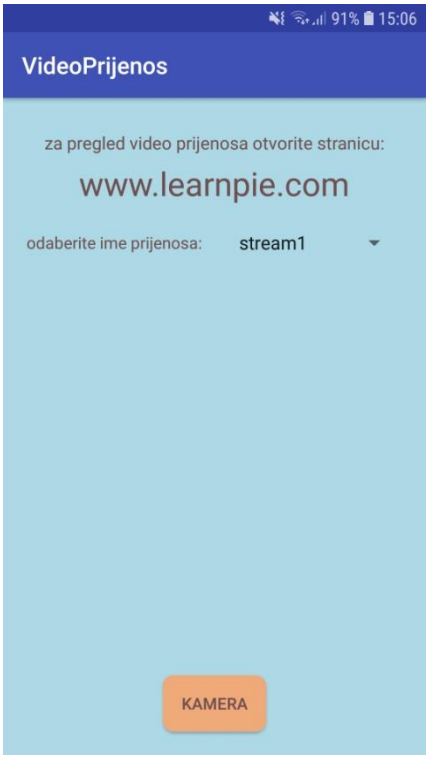
Nakon što je projekt kreiran, dodaju se već gore navedene deklaracije za kameru i internet u manifest aplikacije, a zatim se dodaje biblioteka (engl. Library) koja omogućuje prijenos video signala. U build.gradle treba dodati sljedeći kod:

```
1.  allprojects {
2.    repositories {
3.      maven { url 'https://jitpack.io' }
4.    }
5.  }
6.  dependencies {
7.    implementation 'com.github.pedroSG94.rtmp-rtsp-stream-client-java:rtplibrary:1.4.1'
8.  }
```

Rtmp-rtsp-stream-client-java je android biblioteka za prijenos video i audio signala prema medijskom poslužitelju preko RTMP ili RTSP protokola. Kodira AAC i H264 i šalje ih u RTMP ili RTSP modul za prijenos. U RTMP modulu kreira RTP pakete koji se sastoje od audia i videa, sprema ih u flv i šalje na poslužitelj.

Nakon što su sva potrebna dopuštenja i biblioteke uključene u projekt kreira se još jedna aktivnost u koju će biti spremljen kod za pokretanje prijenosa videa. Ime te nove klase je VideoActivity, a ime njenog layouta je activity_video.

Sljedeći korak je izrada dizajna, najprije za početnu aktivnost, MainActivity(Sl 4.6). Dizajn se izrađuje u jeziku za označavanje podataka XML (engl. eXtensible Markup Language).



Slika 4.6. Početni zaslon

U activity_main.xml (Sl 4.7) je definirano kako će aplikacija izgledati, točnije kako će izgledati prva aktivnost nakon što se aplikacija pokrene. Koristi se relativni layout koji omogućuje pozicioniranje elemenata relativno jedni prema drugima. U ovom layoutu su korištene četiri vrste elemenata:

TextView – element korisničkog sučelja koji korisniku prikazuje tekst.

Spinner – način odabira jedne vrijednosti iz skupa (polje elemenata). Sastoji se od padajućeg izbornika sa svim dostupnim vrijednostima, od kojih korisnik može odabrati jednu.

Button – element korisničkog sučelja koji korisnik može kliknuti za izvođenje određene radnje.

LinearLayout – prikaz u kojem su elementi poravnati horizontalno ili vodoravno

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/light_blue"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="28dp"
        android:textColor="@color/wenge"
        android:textSize="15sp"
        android:text="@string/src_tv_info" />

    <TextView
        android:id="@+id/tv_page"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="54dp"
        android:textColor="@color/wenge"
        android:textSize="28sp"
        android:text="@string/src_tv_page" />

    <LinearLayout
        android:layout_below="@+id/tv_page"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_margin="20dp"
        android:gravity="center">

        <TextView
            android:id="@+id/tv_ime"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/src_tv_ime"
            android:textColor="@color/wenge" />

        <Spinner
            android:id="@+id/sp_ime"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textColor="@color/wenge"/>
    </LinearLayout>

    <Button
        android:id="@+id/btn_kamera"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/button"
        android:text="@string/src_btn_kamera"
        android:layout_marginBottom="20dp"
        android:textColor="@color/wenge" />
</RelativeLayout>
```

Slika 4.7. activity_main.xml

Nakon što je layout za MainActivity izrađen, sljedeći na redu je layout za VideoActivity koji je prikazan na slici 4.8.

Slika 4.8. Aktivnost za prikaz kamere

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_example_rtmp"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <SurfaceView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/surfaceView"
        />

    <TextView
        android:textColor="@color/wenge"
        android:textColorHint="@color/wenge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:background="@color/middle_yellow_red"
        android:textSize="18dp"
        android:layout_margin="20dp"
        android:id="@+id/tv_url"
        />

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:background="@drawable/button"
        android:text="@string/src_btn_start"
        android:layout_marginBottom="20dp"
        android:textColor="@color/wenge" />
</RelativeLayout>
```

Slika 4.9. activity_video.xml

Activity_video.xml (Sl. 4.9) se sastoji od tri elementa od kojih je onaj najbitniji prikaz površine (engl. SurfaceView) koji služi za prikaz kamere. Nakon što je dizajn gotov započinje rad na MainActivity-u. Ovdje definiramo zahtjev za kamerom bez kojega kamera ne radi na sustavima iznad api-a 23(Marshmallow). Sljedeći kod je funkcija koja uspoređuje api mobitela na kojem je aplikacija pokrenuta sa api-em 23.

```
1.  private boolean hasPermissions(Context context, String... permissions) {
2.      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M && context != null && permissions != null) {
3.          for (String permission : permissions) {
4.              if (ActivityCompat.checkSelfPermission(context, permission)
5.                      != PackageManager.PERMISSION_GRANTED) {
6.                  return false;
7.              }
8.          }
9.      }
10.     return true;
11. }
```

Nakon što je provjera api-ja uspješna, odnosno dopuštenja za kameru su omogućena, uz pomoć gumba btnKamera otvara se VideoActivity(Sl 4.7).

```
1.  btnKamera.setOnClickListener(new View.OnClickListener() {
2.      @Override
3.      public void onClick(View v) {
4.          if (hasPermissions(MainActivity.this,PERMISSIONS)){
5.              startCamera();
6.          }
7.      }
8.  });
```

Uz pomoć spinnera koji je definiran u MainActivity odabire se ime za prijenos. Moguće je birati između 3 imena: stream1,stream2 i stream3. To ime se zatim prenosi u VideoActivity i tamo se dodaje na link rtmp://138.197.185.193/hls/{ime}.

Još preostaje da se napiše kod za ActivityVideo. Najprije je potrebno implementirati ConnectCheckerRtmp koji je dio RTMP biblioteke i SurfaceHolder.Callback koji pruža informacije o promjenama na površini(eng. Surface).

Implementacijom SurfaceHolder.Callback kreiraju se sljedeće funkcije:

```
1.  @Override
2.   public void surfaceCreated(SurfaceHolder holder) {
3.
4.   }
5.
6.   @Override
7.   public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
8.
9.   }
10.
11.  @Override
12.  public void surfaceDestroyed(SurfaceHolder holder) {
13.
14.  }
```

Funkcije koje implementacijom kreira ConnectCheckerRtmp:

```
@Override
public void onConnectionSuccessRtmp() {

}

@Override
public void onConnectionFailedRtmp(String s) {

}

@Override
public void onDisconnectRtmp() {

}

@Override
public void onAuthErrorRtmp() {

}

@Override
public void onAuthSuccessRtmp() {

}
```

Nakon što su potrebne funkcije implementirane potrebno je pozvati RtmpCameru1 koja je sastavni dio RTMP biblioteke.

```
private RtmpCamera1 rtmpCamera1;
```

Korištenjem rtmpCamera1 pozivaju se funkcije potrebne za pokretanje i prekidanje video prijenosa. Pritiskom na btnStart if naredba provjerava jel postoji trenutni prijenos, ako ne postoji pokreće video prijenos prema poslužitelju i postavlja vrijednost btnStart na btnStop, odnosno prekida prijenos, ako već postoji.

```
btnStart.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!rtmpCamera1.isStreaming()) {
            if (rtmpCamera1.prepareVideo()) {
                btnStart.setText(R.string.src_btn_stop);
                rtmpCamera1.startStream(tvUrl.getText().toString());

            } else {
                Toast.makeText(VideoActivity.this, "Video prijenos nije podrzan na ovom uredaju", Toast.
LENGTH_SHORT).show();
            }
        } else {
            btnStart.setText(R.string.src_btn_start);
            rtmpCamera1.stopStream();
        }
    }
});
```

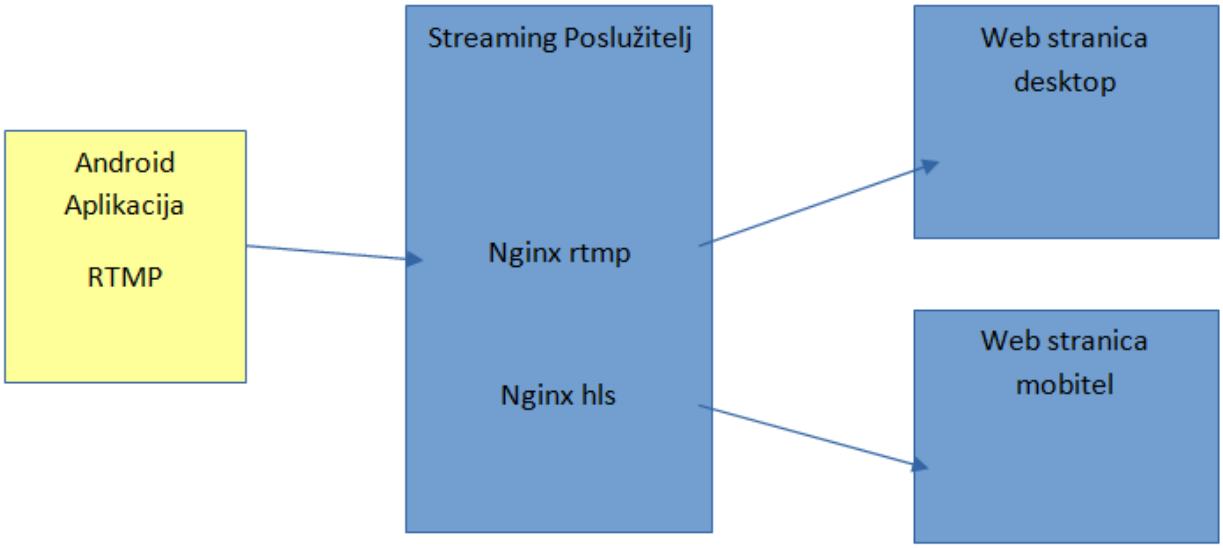
Ako je spajanje uspješno prikazat će se poruka „spajanje uspješno“, a u slučaju da spajanje nije uspješno poruka „spajanje neuspješno“.

```
@Override
public void onConnectionSuccessRtmp() {
    runOnUiThread(new Runnable() {
        @Override
```

```
5.          public void run() {
6.              Toast.makeText(VideoActivity.this, "spajanje uspješno", Toast.LENGTH_SHORT).show();
7.          }
8.      });
9.  }
10.
11. @Override
12. public void onConnectionFailedRtmp(final String reason) {
13.     runOnUiThread(new Runnable() {
14.         @Override
15.         public void run() {
16.             Toast.makeText(VideoActivity.this, "spajanje neuspješno" + reason, Toast.LENGTH_SHORT).show(
    );
17.             rtmpCamera1.stopStream();
18.             btnStart.setText(R.string.src_btn_start);
19.         }
20.     });
21. }
```

5. SERVERSKA APLIKACIJA

Nakon što je android aplikacija gotova i spremna za slanje signala potrebno je kreirati server koji će taj signal primiti. Aplikacija šalje RTMP signal prema Nginx serveru sa RTMP modulom koji taj signal šalje prema web stranici u dva oblika(sl 5.1), ovisno o platformi na kojoj je stranica otvorena, prihvatit će RTMP ili HLS.



Slika 5.1. arhitektura servera

5.1 Kreiranje RTMP servera

Za kreiranje poslužiteljske aplikacije korišten je DigitalOcean VPS (engl. Virtual Private Server). Na stranici DigitalOcean VPS se još zove i droplet. Pri kreaciji dropleta odabire se operacijski sustav koji će biti instaliran na njemu. U ovom slučaju korišten je Ubuntu 16.04. Nakon što je napravljen droplet, korištenjem Linux baziranog sustava otvara se terminal i upisuju se podatci potrebni za spajanje(ip adresa i šifra), a ako se koristi windows sustav onda je potrebno koristiti putty. Budući da se za prijenos video signala koristi RTMP, potrebno je napraviti RTMP poslužitelj. Najbolja opcija za to je korištenje Nginx poslužitelja uz Nginx-rtmp modul koji je napisan posebno za Nginx. Zbog korištenja dodatnog modula, Nginx nije moguće instalirati uz naredbu sudo apt-get install nginx, nego ga je potrebno kompajlirati iz izvornog koda zajedno sa rtmp modulom.

Najprije se instaliraju alati potrebni za kompajliranje:

```
~$ sudo apt-get install build-essential libpcre3 libpcre3-dev libssl-dev
```

U idućem koraku preuzimaju se datoteke Nginx i Nginx-RTMP modula:

```
~$ wget http://nginx.org/download/nginx-1.15.3.tar.gz

~$ wget https://github.com/arut/nginx-rtmp-module/archive/master.zip
```

Nakon toga treba ih raspakirati i postaviti se u mapu nginx-a:

```
~$ tar -zxvf nginx-1.15.3.tar.gz

~$ unzip master.zip

~$ cd nginx-1.15.3
```

Sada se dodaje Nginx-RTMP modul, a zatim kompajlira i instalira Nginx:

```
~$ ./configure –with-http_ssl_module –add-module=../nginx-rtmp-module-master

~$ make

~$ sudo make install
```

U sljedećem koraku potrebno je instalirati nginx init skriptu:

```
~$ sudo wget https://raw.github.com/JasonGiedymin/nginx-init-ubuntu/master/nginx -0/etc/init.d/nginx

~$ sudo chmod +x /etc/init.d/nginx

~$ sudo update-rc.d nginx defaults
```

Prije nastavka treba napraviti update:

```
~$ sudo apt-get update
```

Da bi video prijenos bio moguć na mobilnim uređajima potrebno je konfigurirati HLS. Najprije se kreira struktura direktorija koji će sadržavati manifest i video fragmente.

```
~$ sudo mkdir /HLS

~$ sudo mkdir /HLS/live

~$ sudo mkdir /HLS/mobile

~$ sudo mkdir /video_recordings

~$ sudo chmod -R 777 /video_recordings
```

Preporučeno je da se uključi vatrozid i omogući promet prema portovima koje koristi Nginx i HLS:

```
~$ sudo ufw limit ssh

~$ sudo ufw allow 80
```

```
~$ sudo ufw allow 1935

~$ sudo ufw enable
```

Nakon što je vatrozid uključem i portovi konfigurirani, otvara se nginx konfiguracijska datoteka:

```
~$ sudo nano /usr/local/nginx/conf/nginx.conf
```

Kod koji se nalazi unutar konfiguracijske datoteke mijenja se sljedećim kodom:

```
#user  nobody;

worker_processes  1;
error_log  logs/rtmp_error.log debug;
pid        logs/nginx.pid;

events {
    worker_connections  1024;
}


http {

    server {
        listen       80;
        server_name  localhost;

        location /hls {
            # Serve HLS fragments
            # CORS setup
            add_header 'Access-Control-Allow-Origin' '*' always;
            add_header 'Access-Control-Expose-Headers' 'Content-Length';
            # allow CORS preflight requests

            if ($request_method = 'OPTIONS') {
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Max-Age' 1728000;
                add_header 'Content-Type' 'text/plain charset=UTF-8';
                add_header 'Content-Length' 0;
                return 204;
            }


            types {
                application/vnd.apple.mpegurl m3u8;
                video/mp2t ts;
            }

            root /tmp;
            add_header Cache-Control no-cache;
        }
    }
}


rtmp {

        server {
                listen 1935;
                chunk_size 8192;

                application hls {
                        live on;
                        meta copy;
                        hls on;
                        hls_path /tmp/hls;
        }
    }
}
```

Nakon što je nginx konfiguriran, treba se pobrinuti oko ograničenja domena.

```
<?xml version="1.0"?>

<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">

<cross-domain-policy>

<allow-access-from domain="*"/>

</cross-domain-policy>
```

Poslužitelj je spreman za primanje video signala. Da bi video signal bio isporučen prema poslužitelju potrebno ga je napisati u formatu: rtmp://ip-adresa/hls/ime-prijenosa.

5.2 Kreiranje web stranice

Web stanica je također kreirana upotrebom digitalocean vps-a. Prilikom kreiranja odabrana je One-click apps i tu se nudi izbor već gotovih aplikacija, u ovom je slučaju odabran LAMP (linux,apache,mysql,php) poslužitelj. Da bi stranica sa lokalnog računala bila prebačena na web korišten je FTP klijent FileZilla. U toj aplikaciji potrebno je upisati ip adresu dropleta odnosno LAMP servera, ime i šifru nakon čega se dobije pristup datotekama koje su instalirane na poslužitelju. Navigacijom do /var/www/html ulazi se u mapu gdje se nalazi web stranica. Sa računala se zatim prebacuje stranica u tu mapu i nakon što se upiše ip adresa u pregledniku, ona postaje vidljiva(sl 5.2). Da pristup stranici ne bi bio putem ip adresu, poslužitelju se dodjeli domena i stranica je spremna za preglede. Video na stranici prikazuje se korištenjem videojs playera koji ima podršku za html5 i flash video.

U <video> su definirani elementi:

width i height – koji označuju visinu i širinu,

poster – koji umjesto crne pozadine playera prikazuje sliku

video-js i vjs-big-play-centered – klasa koja definira videojs i klasa koja play gumb postavlja na sredinu okvira videa.

data-setup – ovdje su navedene vrste videa koji su podržani na stranici

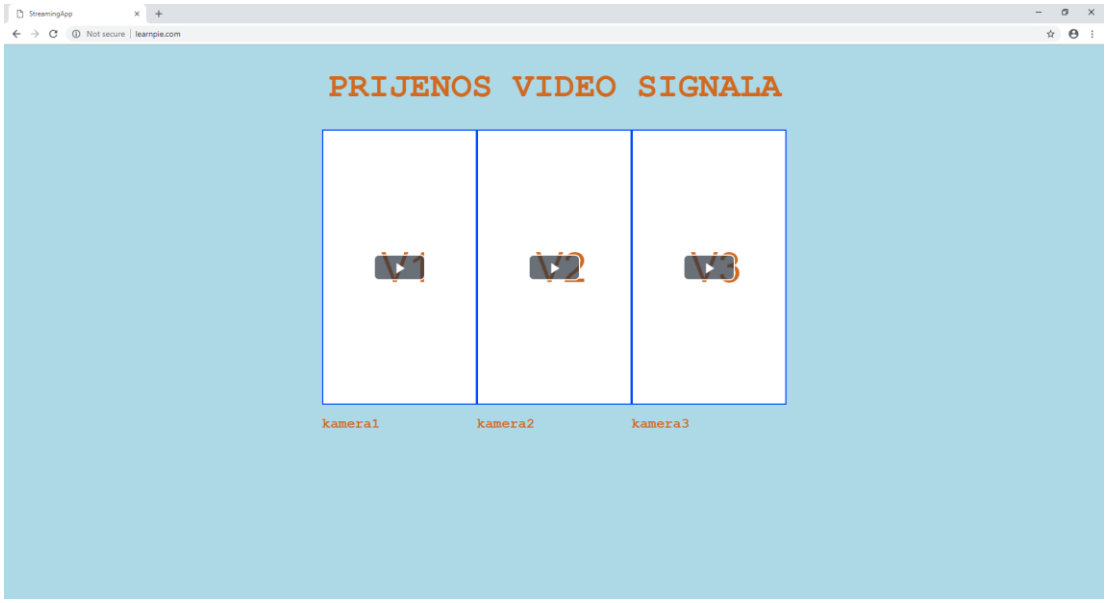
U kodu su navedena dva src filea, jedan je definiran za rtmp, a drugi za hls.

```
<div class="container-1">
    <div class="container-1-box">

        <video class="video-js vjs-big-play-
centered" controls preload="auto" width="270" height="480" poster="stream1.png" data-
setup='{"techorder" : ["flash","html5] }'>
            <source src="rtmp://138.197.185.193/hls/stream1" type='rtmp/mp4'>
```

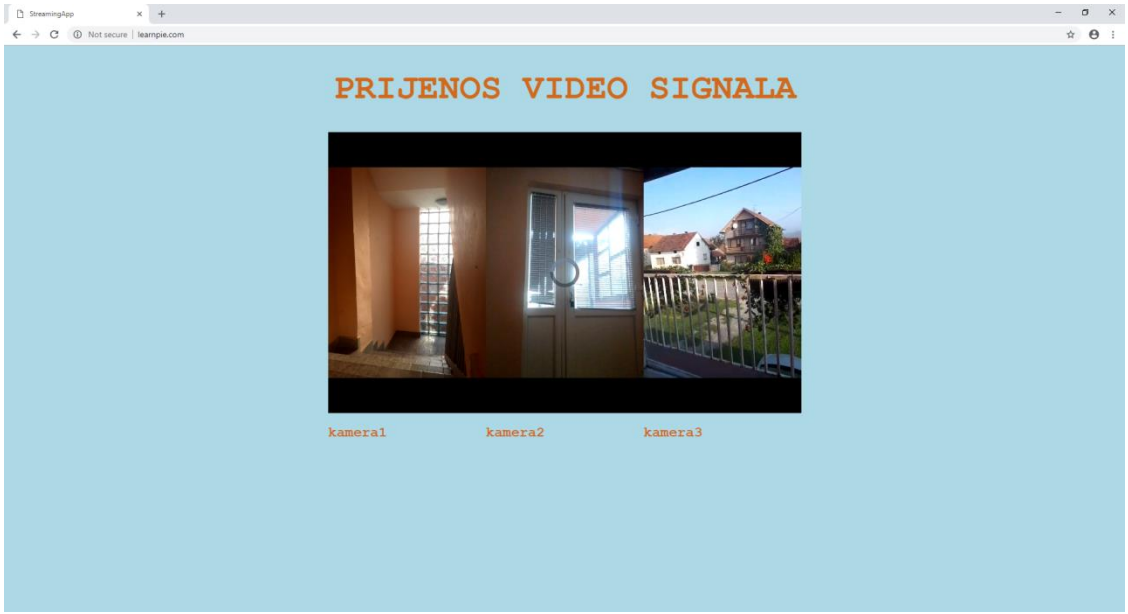
```
            <source src="http://138.197.185.193/hls/stream1.m3u8" type="application/x-mpegURL">
            <p class="vjs-no-js">
                Da bi vidjeli ovaj video omogućite JavaScript, i razmislite o nadogradnji preglednika koji
                <a href="https://videojs.com/html5-video-support/" target="_blank">podržava HTML5 video</a>
            </p>
        </video>

        <h2>stream1</h2>
    </div>
```



Slika 5.2. web stranica

Nakon što se pristupi stranici i pošalje video signal sa tri različita uređaja imamo kompletan video nadzor(sl 5.3)



Slika 5.3. Prikaz videa na stranici

6. ZAKLJUČAK

U ovom diplomskom radu opisan je proces izrade klijentske aplikacije za Android operativni sustav korištenjem alata Android studio i poslužiteljske aplikacije korištenjem Nginx poslužitelja sa RTMP modulom. Pametni mobilni telefoni postali su ljudska svakodnevica i dostupni su svima. Android je vodeći operativni sustav već nekoliko godina, a činjenica da je open-source operativni sustav čini ga idealnim za razvoj aplikacija. Video nadzor uz pomoć android telefona omogućuje nam jeftin nadzor. U ovom radu korišten je osnovni VPS poslužitelj i osnovne funkcije kamere, ali i sa tim opcijama opet se dobije moćna aplikacija koja bi se uz još neke dodatke mogla bez problema koristiti u svrhu video nadzora.

LITERATURA

[1] Wikipedia(Android operacijski sustav), s interneta, https://hr.wikipedia.org/wiki/Android_(operacijski_sustav), 15. rujan 2018.

[2] Wikipedia(Dosadašnje inačice sustava Android), s interneta, https://hr.wikipedia.org/wiki/Dosada%C5%A1nje_ina%C4%8Dice_sustava_Androida,

15. rujan 2018.

[3] Android Developers, s interneta, https://developer.android.com/guide/platform/index.html, 15.rujan 2018.

[4] Wikipedia(Android software development), s interneta, https://en.wikipedia.org/wiki/Android_software_development, 15. rujan 2018.

[5] Wikipedia(Video), s interneta, https://en.m.wikipedia.org/wiki/Video, 20. rujan 2018.

[6] Vcodex, s interneta, www.vcodex.com/an-overview-of-h264-advanced-video-coding/, 20. rujan 2018.

[7] Muvi, s interneta, www.muvi.com/wiki/real-time-messaging-protocol-rtmp.html, 20. rujan 2018.

SAŽETAK

Ovaj rad predstavlja proces izrade Android aplikacije za prijenos video signala i izradu poslužiteljske aplikacije i web stranice na kojoj se taj video signal prikazuje. Koriste se postojeće biblioteke koje olakšavaju upotrebu formata, kodera i ostalih stvari vezanih uz video. Negativna stvar vezana za video prijenos je povremeno loš signal pa tu dolazi do zastoja, pogotovo ako se koristi HLS. Potrebna je kupnja poslužitelja koji prenosi signal i kupnja web stranice na kojoj se taj signal prikazuje. Pozitivna stvar je što je realizacija takvog projekta sve jednostavnija zbog raširenosti video prijenosa.

Ključne riječi: Android aplikacija, video prijenos, RTMP, HLS, h.264, Nginx

ABSTRACT

ANDROID APPLICATION FOR TRANSFERRING VIDEO SIGNAL

This work represents the process of creating an Android application for video signal transfer, and creating a server application and the site where that video signal is displayed. Existing libraries are used to facilitate the use of formats, encoders, and other video related stuff. The negative thing about video transmission is occasionally bad signal, so there is a delay, especially if HLS is used. Purchase of a server that transmits a signal and the purchase of a web page on which that signal is displayed is required. The positive thing is that the realization of such a project is getting easier because of the widespread use of video transmission

Keywords: Android application, streaming, RTMP, HLS, h.264, Nginx.

ŽIVOTOPIS

Ivan Benke rođen je u Vukovaru 8.11.1991. godine, a odrastao je i živi u Pitomači sa roditeljima i bratom. Osnovnu školu započeo je 1998. godine u osnovnoj školi Petra Preradovića Pitomača. 2006. godine upisuje Tehničku školu u Virovitici, smjer računalni tehničar za strojarstvo, gdje je bio vrlo dobar učenik. Nakon srednje škole upisao je stručni studij informatike na Elektrotehničkom fakultetu u Osijeku, koji je završio 2013. godine sa temom završnog rada „Virtualni sat“. Nakon toga odlučuje se na nastavak studija i upisuje diplomski studij računarstva, ali je prije toga morao završiti jednu godinu razlikovnog studija.