



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ



Срђан Живанић

**Веб апликација за дистрибуцију
мултимедијалног тока података у
реалном времену**

ДИПЛОМСКИ РАД
- Основне академске студије -

Нови Сад, 2020



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Срђан Живанић		
Ментор, МН:	проф. др Илија Башичевић		
Наслов рада, НР:	Веб апликација за дистрибуцију мултимедијалног тока података у реалном времену		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2020		
Издавач, ИЗ:	Ауторски репрингт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/ слика/графика/прилога)	7/24/0/0/7/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кјучне речи, ПО:	Рачунарске мреже, Пренос податка преко видео тока, ИП камера		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	<p>Имплементација апликативног решења која омогућава масовно опслуживање корисника са видео и аудио садржајем. Циљ је уштеда ресурса приликом директног повезивања на ток података.</p> <p>Апликација успоставља конекцију са извором и омогућава да се сигнал преноси на сервере већих капацитета или да ради као властити сервис масовног услугивања.</p>		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	«име председника комисије»	
	Члан:	«име члана комисије»	Потпис ментора
	Члан, ментор:	проф. др Илија Башичевић	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Srđan Živanić	
Mentor, MN:	Ilijा Bašičević, PhD	
Title, TI:	Web application for real-time multimedia data distribution	
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:	2020	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/24/0/0/7/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	Computer networks, Data transmission via video stream, IP cameras	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	Implementation of an application solution that enables mass customer service with video and audio content. The goal is to save resources when connecting directly to a data stream. The application establishes a connection with the source and allows the signal to be transmitted to higher capacity servers or to operate self service.	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President: Member: Member, Mentor:	<ime predsednika komisije> <ime člana komisije> Ilijा Bašičević, PhD
		Menthor's sign

Zahvalnost

Zahvaljujem se mentoru prof. dr Iliji Bašičeviću i asistentu Milošu Pilipoviću na korisnim savetima, strpljenju i pomoći pri izradi ovog rada.

Zahvaljujem se porodici i prijateljima koji su mi pružali podršku tokom studiranja.

SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove	3
2.1	Digitalna televizija	3
2.2	Protokoli za realizaciju rada.....	4
2.2.1	HTTP/HTTPS protokol	4
2.2.2	TCP/IP protokol.....	4
2.2.3	RTMP protokol.....	4
2.2.4	RTP/RTSP protokol.....	5
2.2.5	HLS protokol	5
2.3	Docker	5
2.4	NGINX	5
2.5	Node.js.....	6
2.6	Clapper player	6
2.7	FFmpeg.....	6
3.	Koncept rešenja.....	7
3.1	Opis platforme.....	7
3.2	Opis rešenja	7
4.	Programsko rešenje.....	10
4.1	Klase aplikacije:	10
4.2	LoginControl scena	13
4.3	Main scena.....	13
4.4	Pokretanje aplikacije	15
4.5	HeaderControll segment.....	16

4.6 Footer segment	17
5. Ispitivanje i verifikacija	18
6. Zaključak	23
7.Literatura	24

SPISAK SLIKA

Slika 3.1 Koncept rada aplikacije.....	8
Slika 5.1 Prikaz uspešnog kompajliranja aplikacije na Docker Hub platformi	18
Slika 5.2 Login scena	19
Slika 5.3 Login scena sa prosleđenim pogrešnim podacima za pristup.....	19
Slika 5.4 Main scena sa izabranim parametrima kodovanja i prenosa	20
Slika 5.5 Main scena sa uspešnim povezivanjem ka izvoru i spoljnom server.....	21
Slika 5.6 Scena sa prikazom signala unutar Clapper player-a	22

SPISAK TABELA

SKRAĆENICE

IP	- <i>Internet protocol</i> , Internet protokol
TCP	- <i>Transmision control protocol</i> , Transmisioni kontrolni protokol
UDP	- <i>User Datagram Protocol</i> , Korisnički datagram protokol
HTTP	- <i>Hypertext Transfer Protocol</i> , Protokol za prenos hiperteksta
HTTPS	- <i>Hypertext Transfer Protocol Secure</i> , Sigurni protokol za prenos hiperteksta
RTP	- <i>Real-time Transport Protocol</i> , Protokol za prenos u realnom vremenu
RTSP	- <i>Real Time Streaming Protocol</i> , Protokol za prenos striminga u realnom vremenu
RTMP	- <i>Real Time Messaging Protocol</i> , Protokol za prenos poruka u realnom vremenu
JS	- <i>Java Script</i> , Java skript datoteka
HTML	- <i>Hyper Text Markup Language</i> , Jezik za označavanje hiperteksta
CSS	- <i>Cascading Style Sheets</i> , Kaskadni stilovni listovi
VPS	- <i>Virtual private server</i> , Virtualni privatni serveri
CPU	- <i>Central Processing Unit</i> , Centralna procesorska jedinica
RAM	- <i>Random-access memory</i> , radna memorija
GUI	- <i>Graphical User Interface</i> , Grafički korisnički pristup

1. Uvod

Prenos multimedijalnog sadržaja u realnom vremenu putem interneta od samog izvora sve do krajnjeg korisnika, tema je ovog rada. Ova vrsta prenosa zajedno sa digitalnom televizijom svoju revoluciju doživela je početkom 21. veka, razvijanjem interneta i sve većom dostupnošću interneta u domaćinstvima.

Za realizaciju prenosa putem interneta koriste se dva najpoznatija protokola, IP i TCP protokol. U ovom radu biće prezentovani i drugi protokoli koji su neophodni za realizaciju aplikacije HTTP/HTTPS, RTSP/RTP, HLS i RTMP protokoli.

Cilj aplikacije je da u realnom vremenu omogući masovno usluživanje većeg broja korisnika. U slučaju da veći broj korisnika direktno pristupa samom izvoru, u slučaju ove aplikacije adresi IP kamere, došlo bi do zagušenja saobraćaja, naročito ako je izvor multimedijalnog toka podataka povezan na kućni internet ograničenog protoka, pa bi određeni broj korisnika bio uskraćen za sadržaj.

Aplikacija rešava upravo ovaj problem, jer ona vrši samo jednu konekciju ka izvoru i omogućava da se signal prenosi na servere sa većim resursima. U slučaju da se aplikacija izvršava na serveru većih resursa moguće je režim rada u kom aplikacija opslužuje korisnike.

Programsko rešenje realizovano je pomoću JS-a, HTML-a, CSS-a, Node.js-a, FFmpeg-a, Clapper-a i Docker-a.

Ovaj rad sačinjen je od sedam poglavlja.

Prvo poglavlje sadrži kratak uvod o radu i aplikaciji.

Druge poglavlje sadrži teorijske osnove koje su neophodne za razumevanje načina funkcionisanja prenosa multimedijalnog toka podataka putem interneta, internet protokola, Docker-a, FFmpeg-a, odnosno delove od kojih je sama aplikacija konstruisana.

Treće poglavlje sadrži idejno rešenje problema i opis arhitekture u koju je rešenje integrисано.

Četvrto poglavlje opisuje programsku implementaciju i integraciju.

U petom poglavlju opisano je testiranje i verifikacija programske aplikacije, koje je izvršeno na Docker Hub-u a zatim pokrenuto na VPS-u.

U šestom poglavlju napravljen je rezime onoga što je urađeno sa naznakom šta bi se moglo implementirati u nekoj bližoj budućnosti.

Poslednje sedmo poglavlje sadrži spisak literature koja je korišćena pri izradi ovog rada.

2. Teorijske osnove

Ovo poglavlje sadrži teorijske osnove o digitalnoj televiziji, internet protokolima i ostalim platformama i modulima koji su neophodni za realizaciju ovog rada.

2.1 Digitalna televizija

Digitalna televizija predstavlja prenos multimedijalnih podataka, kao i drugih informacija zapisanih u digitalnom formatu. Prednosti digitalne televizije u odnosu na analognu televiziju, su pre svega kvalitetniji zvuk i slika, dodatne mogućnosti i opcije poput gledanja propuštenog sadržaja, pristup internet pretraživačima, igranje video igara, video klub i slične mugućnosti.

Trenutno postoje četiri vrste prenosa podataka:

- Zemaljski
- Kablovske
- Satelitski
- Preko IP mreže – putem Interneta.

U televiziji zasnovanoj na IP protokolu, televizijski signal prenosi se tehnologijama koje se koriste u računarskim mrežama.

U osnovi ovih tehnologija je IP protokol. Sam digitalni signal pakuje na transportnom, ili sesionom ISO OSI sloju, pomoću UDP, TCP ili HTTP protokola. Za fizički prenos signala najčešće se koristi IP veza i Ethernet protokol.

2.2 Protokoli za realizaciju rada

Osnovni rad aplikacije zasnovan je na IP protokolu, odnosno HTTP/S protokolu. Prenos signala i komunikacija između aplikacije i izvora, odnosno spoljnog servera, zasnovana je na RTMP, RTSP i HLS protokolima.

2.2.1 HTTP/HTTPS protokol

HTTP (*HyperText Transfer Protocol*) je osnovni protokol za distribuciju sadržaja na internetu. Osnovna funkcija ovog protokola je prenos zahteva za hipertekstualnim dokumentima od klijenta ka serveru i od servera ka klijentu. HTTP protokol spada u protokole aplikativnog nivoa. Kao transportni protokol najčešće se koristi TCP, s tim da je u teoriji moguće korišćenje i nekih drugih protokola koji obezbeđuju pouzdan prenos podataka. Podrazumevani port HTTP protokola je 80, s tim da se mogu koristiti i ostali dostupni portovi. HTTP protokol se ne smatra zaštićenim, jer je zbog svojih propusta podložan zloupotrebljama.

HTTPS predstavlja sigurnu verziju HTTP protokola. Nastao je kombinovanjem HTTP i SSL/TLS protokola kako bi se obezbedila enkripcija i sigurna identifikacija. HTTPS najčešće koristi port 443.

2.2.2 TCP/IP protokol

TCP/IP je slojevit protokol, čiji je stek sačinjen od četiri sloja: aplikativni, transportni, mrežni i protokol veze podataka. Aplikativni sloj obezbeđuje zajedničku spregu preko koje korisničke aplikacije komuniciraju sa nižim slojevima, tj. obezbeđuje vezu između aplikacije i mreže. Transportni sloj zadužen je za kontrolu toka između dva krajnja računara, pored toga on informaciju iz aplikativnog sloja predaje mrežnom sloju. Mrežni sloj odgovoran je za usmeravanje paketa kroz mrežu. Protokol veze podataka zadužen je za vezu operativnog sistema sa mrežnom karticom unutar računara.

2.2.3 RTMP protokol

RTMP (Real Time Messaging Protocol) je protokol razvijen od strane Macromedia-e, a sada se nalazi u vlasništvu kompanije Adobe i uglavnom se koristi za dostavljanje multimedijalnih sadržaja. RTMP je pouzdan za prenos signala preko javnog interneta. Prednosti su mu nisko kašnjenje i za realizaciju njegovog rada nije potreban veliki bafer. Vrlo često se upotrebljava u digitalnoj televiziji za aplikacije poput "Video na zahtev".

2.2.4 RTP/RTSP protokol

RTSP (Real-Time Streaming Protocol) je protokol na aplikativnom nivou koji se koristi za kontrolu nad dostavom podataka u realnom vremenu. Vrlo često se koristi zajedno sa RTP (Real-Time Protocol) protokolom. Sprega RTP/RTSP realizovana je mešavinom TCP i UDP protokola, jer RTSP radi preko TCP-a, dok RTP se šalje preko UDP-a. Prednost RTSP protokola je što audio i video format šalje odvojeno, pa u zavisnosti od potreba može se koristiti samo jedan format, dok se sinhronizacija realizuje u RTP zaglavljima.

2.2.5 HLS protokol

HLS (HTTP Live Streaming) je protokol koji je razvijan od strane Apple-a prvobitno samo za IOS uređaje, dok kasnije pronađe primenu i na ostalim platformama. Karakteriše ga vremensko kašnjenje od nekoliko sekundi, ali i veoma stabilan prenos podataka bez prekida. Baziran je na TCP protokolu.

2.3 Docker

Docker je alat dizajniran tako da olakšava razvoj, implementaciju i pokretanje aplikacija koristeći kontejnere. Softver koji sadrži kontejnere naziva se Docker Engine. Prvi put se pojavio 2013. godine i razvila ga je kompanija Docker Inc. Kontejneri su izolovani jedan od drugog i sadrže sopstveni softver, biblioteke i konfiguracione fajlove. Komunikacija između kontejnera odvija se putem tačno definisanih kanala. Svi kontejneri su upravljeni pod kernelom operativnog sistema i stoga koriste manje resursa od virtualnih mašina.

2.4 NGINX

NGINX je veoma moćna veb server aplikacija koja pruža mnoge mogućnosti. Dizajniran je za malu potrošnju memorije i veliki broj konekcija. Umesto da kreira nove procese za svaki veb zahtev, Nginx koristi asinhroni *event-driven* pristup, gdje se zahtevi obrađuju u jednom *thread-u*. Nginx zajednica otvorenog koda razvila je mnogo modula i proširenja, a u ovom projektu koristiće se NGINX-RTMP modul za kontrolu video i audio toka.

2.5 Node.js

Node.js je serverska JavaScript platforma koja radi na V8 JavaScript engine-u, što znači da je veoma brza u izvršavanju. Omogućava korišćenje jedinstvenog jezika između front-end-a i back-end-a. Zbog svoje brzine izvršavanja veoma je pogodan za realizaciju aplikacija u realnom vremenu.

2.6 Clapper player

Clapper player je besplatna multimedijalna aplikacija koja omogućava prikaz audio i video formata. Clapper koristi HTMLVideoElement i iz tog razloga je pogodan za korišćenje na velikom broju različitih platformi, jer skoro sve imaju podršku za HTMLVideoElement. Veoma je prilagodljiv za razne upotrebe kada je u pitanju prikaz video i audio formata.

2.7 FFmpeg

FFmpeg je besplatan programski paket namenjen za emitovanje, obradu i slanje multimedijalnog sadržaja. Podržava širok spektar različitih formata zapisa i iz tog razloga je primenu našao u većini danas dostupnih alata i aplikacija. Sadrži alate koju se mogu koristiti za filtersku obradu audio i video formata, njihovo konvertovanje u druge formate, prijem i slanje audio i video toka podataka, kao i mnoge druge operacije.

3. Koncept rešenja

Ovo poglavlje sadrži opis platforme i koncept rešenja.

3.1 Opis platforme

Pri izradi ovog rada korišćen je virtuelni privatni server na kome je programski kod generisan i testiran.

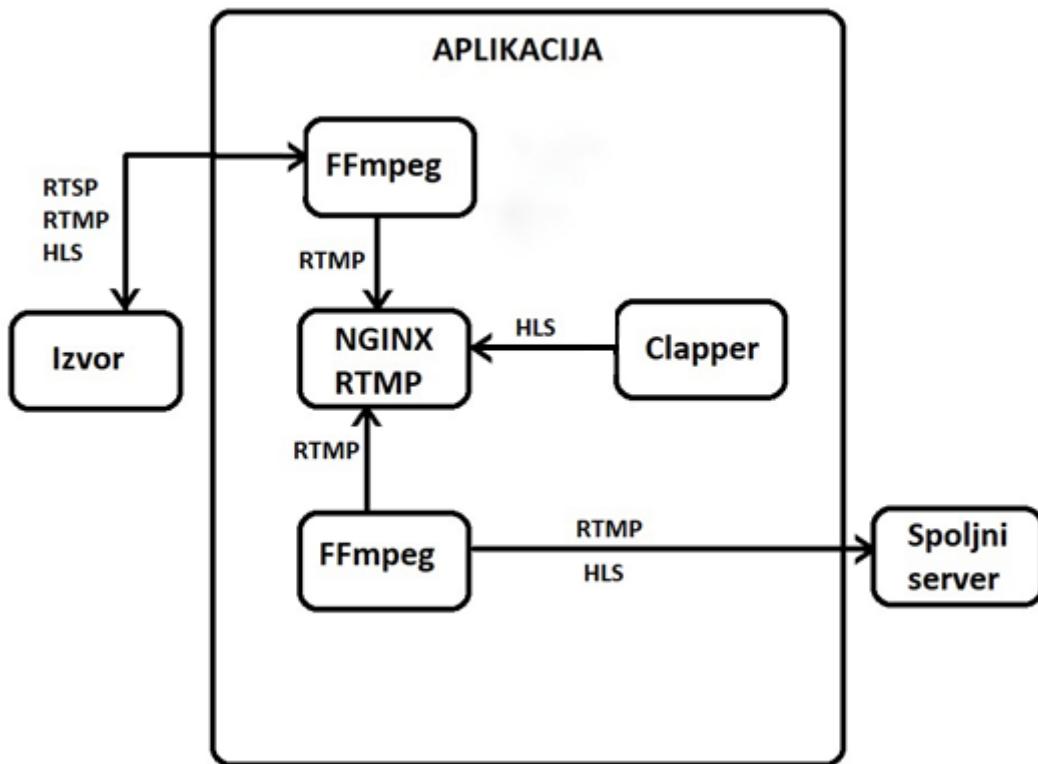
Specifikacija servera:

- CPU: 2 jezgra
- RAM: 4 GB
- SSD: 20 GB
- Mesečni protok podataka: 50 000 GB (Bandwidth)

3.2 Opis rešenja

Cilj ovog rada je da omogući video i audio prenos ka većem broju korisnika u realnom vremenu. Radi dobijanja realnih rezultata u odzivu aplikacije i kontrole multimedijalnog toga podataka, od izvora do aplikacije i od aplikacije do krajnjeg izvora, aplikacija kao i sam izvor povezani su na globalnu računarsku mrežu. Na samom početku je neophodno kreirati GitHub nalog, zatim kreirati repozitorijum aplikacije. Nakon dodavanja sadržaja GitHub repozitorijumu potrebno je kreirati nalog na Docker Hub-u. Nakon kreiranja naloga neophodno je napraviti repozitorijum na Docker Hub-u, te isti povezati sa GitHub nalogom i repozitorijumom koji je napravljen na GitHub-u. Kada su repozitorijumi povezani, na Docker Hub-u je potrebno pokrenuti bildovanje aplikacije. Radi lakšeg rada moguće je uključiti opciju automatskog sinhronizovanja, tako da u slučaju promene sadržaja na GitHub repozitorijumu, Docker Hub će automatski pokrenuti bildovanje aplikacije.

Koncept rada same aplikacije prikazan je na sledećoj slici.



Slika 3.1 Koncept rada aplikacije

Korisnički interfejs aplikacije realizuje se na HTTP/HTTPS adresi servera koji koristi NGINX. U zavisnosti od izbora HTTP ili HTTPS koriste se različite konfiguracione datoteke. Korišćenje HTTPS protokola zahteva validan SSL sertifikat. Prijavom u korisnički interfejs omogućava se pristup izvoru signala, kao i dodatnim opcijama o načinu kodovanja signala. Aplikacija pomoću FFmpeg-a pristupa izvoru signala upotrebom RTSP, RTMP ili HLS protokola u zavisnosti od izlaza koji isporučuje sam izvor. Nakon realizovane uspešne konekcije ka izvoru na raspolaganju su mogućnosti prosleđivanja signala ka spoljnem serveru, kao i prikaz signala unutar same aplikacije. Dobijeni signal FFmpeg prosleđuje NGINX-RTMP modulu preko RTMP protokola.

Clapper se povezuje na NGINX-RTMP modul i preko HLS protokola dobija signal koji je moguće prikazati unutar same aplikacije. Pomoću iFrame koda dobijeni signal moguće je ugraditi i na neku drugu veb lokaciju .

Opciona mogućnost je da FFmpeg od NGINX-RTMP modula dobije signal i isti prosledi spoljnem serveru pomoću RTMP ili HLS protokola.

Pokretanje aplikacije vrši se pomoću Docker komande:

```
docker run -d --restart always \
--name camappzs \
-e "RS_USERNAME=admin" -e "RS_PASSWORD=password" \
-p 8080:8080 -v /mnt/camappzs/db:/camappzs/db \
z1le/camappzs:latest
```

Pristupanje se vrši korisničkim imenom “admin” i lozinkom “password”.

-d - označava “Detach the container” da je kontejner nezavisan od drugih aplikacija
--name camappzs - postavlja camappzs kao ime kontejnera
--restart always - u slučaju da dođe do prestanka rada aplikacije, Docker će ponovo pokrenuti kontejner
--e "RS_USERNAME=..." -e "RS_PASSWORD=..." - postavlja pristupne podatke
-p 8080:8080 – postavlja port aplikacije na 8080, aplikacija će biti pokrenuta na adresi: <http://IP-adresa:8080>
-v /mnt/ camappzs /db:/ camappzs /db - koristi direktorijum u kome se čuva trenutno stanje
z1le/camappzs:latest – označava adresu poslednjeg kompajliranog Docker kontejnera ove aplikacije

U slučaju korišćenja HTTPS protokola potrebno je promeniti port aplikacije i dodati parameter HTTPS=true:

-p 443:8181 -e RS_HTTPS=true

Datoteke sertifikata i ključa sačuvati unutar cert.pem i key.pem datoteka. Datoteke je potrebno sačuvati unutar ssl foldera koji se nalazi na putanji /mnt/camappzs/db/ssl
Drugi način korišćenja HTTPS protokola je pomoću *NGINX Reverse Proxy*-a. Aplikacija se pokreće na HTTP protokolu ali koristi *Reverse Proxy* kako bi se prikazivala preko HTTPS protokola.

4. Programsко rešenje

U ovom poglavlju opisano je programsko rešenje aplikacije.

Na samom početku kreiran je Dockerfile koji sadrži sve module neophodne za funkcionisanje aplikacije.

Spisak modula:

- Nasm
- x264
- libmp3lame
- ffmpeg
- nginx
- nginx-rtmp
- node.js

Korišćen je i modul docker-compose koji omogućava pokretanje aplikacije sa unapred definisanim konfiguracijom i dodatnim kontejnerima koji su neophodni za rad.

4.1 Klase aplikacije:

- Camappzs.js - glavna klasa u kojoj su realizovane sve funkcionalnosti aplikacije
- CamappzsData.js - klasa koja proverava upisane podatke u bazu
- EnvVar.js - klasa koja postavlja početne vrednosti
- Logger.js – klasa za kreiranje izveštaja
- Nginxrtmp.js – klasa za praćenje i kontrolu NGINX-RTMP mudula
- WebsocketsController.js – klasa za kontrolu veb utičnica

Funkcije koje su realizovane unutar klase Camappzs.js:

static getRTMPStreamUrl() - generiše RTMP adresu toka podataka
static getSnapshotPath() - generiše izlazni snimak iz konfiguracione datoteke
static fetchSnapshot () - generiše snimku koristeći prvi kadar ponovljenog video zapisa
static addStreamOptions(command, name, replace) - postavlja FFmpeg opcije za prenos
static getSnapshotInterval() - interval nakon koga se učitava nova slika
static stopStream(streamType) - zaustavlja prenos signala
static restoreProcesses() - ponovo pokreće zadani FFmpeg process iz baze ako je aplikacija
nasilno zatvorena
static writeToDB () - upisuje podatke u JSON bazu podataka
static writeToPlayerConfig () - upisuje konfiguraciju prikaza u konfiguracioni dokument
static updateStreamDataOnGui () - šalje websocket-u signal da GUI ažurira stanje o
multimedijalnom toku podataka
static updateProgressOnGui () - šalje websocket-u signal da GUI ažurira stanje o napretku
static probeStream(rtmpUrl, streamType) - postavlja FFmpeg opcije konfiguracijske
datoteke na izlaz
static updateState(streamType, state, message) - ažurira stanje o multimedijalnom toku
static setState(streamType, state, message) - postavlja novo stanje na tok podataka
static getState(streamType) - vraća stanje o multimedijalnom toku podataka
static getUserAction(streamType) - vraća trenutnu korisničku akciju
static setUserAction(streamType, action) - postavlja trenutnu korisničku akciju (stop/start)
static updateUserAction(streamType, action) - ažurira poslednju korisničku akciju
static updatePlayerOptions(player) - ažurira podatke o prozoru za prikaz
static updateOptions (options) - ažurira opcije
static startStream(streamUrl, streamType, force) - pokreće proces toka podataka
static setTimeout(streamType, target, func, delay) - postavlja vremensko ograničenje
static bindWebSocketEvents() - postavlja websocket događaj pri startovanju aplikacije
static extractDataOfStreams() - stvara objekat o dostupnim multimedijalnim podacima
static dataForJsonDb() - stvara objekat o dostupnim multimedijalnim podacima za JsonDB
static getPublicIp() - vraća javnu IP adresu na kojoj se aplikacija izvršava

Funkcije koje su realizovane unutar klase CamappzsData.js:

static checkJSONDb() - funkcija koja proverava podatke koji se nalaze u JSON bazi podataka, kao što su podaci o podešavanjima audio i video toka.

Funkcije koje su realizovane unutar klase EnvVar.js:

constructor() - konstruktor

log(message, level) - upisuje podatke o greškama

init(config) - prikazuje i podešava početno stanje

reset() - restartuje vrednosti

Funkcije koje su realizovane unutar klase Logger.js:

static isMuted () - proverava da li je opcija generisanja izveštaja o greškama uključena
constructor (context) - konstruktor

logline(message, context, type) - kreira liniju izveštaja

stdout (message, context, type) - ispis poruke na standardni izlaz

file (message, context, type) - ispis poruke u datoteku

info (message, context, alertGui) - funkcija koja ispisuje informativnu poruku

warn (message, context, alertGui) - funkcija koja ispisuje poruku upozorenja

debug (message, context, alertGui) - funkcija koja ispisuje debug poruku

error (message, context, alertGui) - funkcija koja ispisuje poruku o grešci

Funkcije koje su realizovane unutar klase Nginxrtmp.js:

constructor(config) - konstruktor

sync start(useSSL) - funkcija koja pokreće lokalni NGINX server

async isRunning(delay) - funkcija koja vraća trenutno stanje NGINX server

Funkcije koje su realizovane unutar klase WebsocketsController.js

static emit (event, data) - emituje događaj na Websocket-u

static setConnectCallback (callback) - dodaje povratni poziv na Websocket vezu

Upotreboom funkcija koje su realizovane u klasama omogućeno je upravljanje aplikacijom. Rad aplikacije realizovan je iz nekoliko scena i delova.

4.2 LoginControl scena

Ova scena se prikazuje prilikom svakog pristupa aplikaciji. Unutar prozora za pristup aplikaciji neophodno je uneti parametre, korisničko ime i lozinku. Ovi parametri zadaju se prilikom startovanja Docker kontejnera aplikacije `--e "RS_USERNAME=admin" -e "RS_PASSWORD=password"`, u ovom slučaju korisničko ime je “admin”, a lozinka “password”. Nakon unošenja ispravnih podataka i pritiskom na dugme “Prijava” moguće je pristupiti glavnoj sceni aplikacije. Ukoliko su unešeni podaci neispravni i ne podudaraju se sa zadatim podacima, pojaviće se poruka o grešci.

LoginController.js realizovan je na sledeći način:

```
window.angular.module('Login').controller('loginController',  
 ['$scope', '$http', '$rootScope', function loginController ($scope, $http, $rootScope) {  
   $scope.submit = function submit () {  
     $http.post('login', {'user': $scope.user, 'pass': $scope.pass}).then((response) => {  
       $scope.message = response.data.message;  
       $rootScope.loggedIn = response.data.success;  
     });  
   };  
 }]  
);
```

4.3 Main scena

Ovo je scena koja se pojavljuje nakon uspešnog pristupa aplikaciji. Neophodno je proslediti ulaznu adresu, odnosno adresu koju generiše multimedijalni izvor. Omogućen je rad sa ulaznim RTMP/RTSP i HLS adresama. Nakon postavljene adrese, na raspolaganju su dodatne opcije kodovanja multimedijalnog toka podataka.

Za audio kodovanje omogućen je odabir brzine uzorkovanja : 44100 Hz, 22050 Hz, 11025 Hz, 8000 Hz.

Za video kodovanje dostupne su opcije “copy” i “H.264”. Opcija “copy” će samo prosleđivati dobijeni signal sa ulaza ka izlazu bez dodatne obrade, dok “H.264” koduje tok podataka u skladu sa svojim standardom, odnosno prema postavljenim dodatnim parametrima kao što su brzina prenosa, broj slika u sekundi, profil i filteri. Kada su svi parametri postavljeni, pritiskom na dugme “Pokreni” aplikacija pristupa ulaznom toku podataka i obrađuje ga prema postavljenim parametrima. Ukoliko adresa ulaznog toka podataka nije ispravna aplikacija će nas obavestiti o grešci.

Dostupna je opcija prosleđivanja ulaznog signala ka spoljnem serveru pomoću RTMP ili HLS protokola, kao i opcija prikaza signala u okviru aplikacije.

Za prikaz signala unutar aplikacije koristi se *Clapper player*. Omogućeno je postavljanje dodatnih parametara samog prozora za prikaz kao što su:

- Automatska reprodukcija kada se otvorí prozor za prikaz
- Isključen zvuk
- Boja prozora za prikaz
- Adresa na kojoj se nalazi sličica koja predstavlja zaštitni znak, pozicija i adresa koja se otvara klikom na sličicu

Otvaranjem prozora za prikaz unutar aplikacije vidljive su promene koje su postavljene pomoću dodatnih parametara. Ispod samog prozora za prikaz dostupne su opcije za ugradnju iFrame okvira na neku drugu veb lokaciju. Moguće je ugraditi kod za video prikaz signala i za trenutne fotografije.

Unutar main scene prilikom uspešne konekcije ka izvoru moguće je prikazati i podatke o ulaznom toku kao što su broj slika u sekundi i brzina prenosa.

```
<!-- Stream is connected -->
<div class="jumbotron progress-bar-success" ng-if="connected()">
  {{'process_success' | translate}}
  <div ng-if="fps()">
    <span class="ffmpeg-progress fps">{{fps()}} FPS</span>
    /
    <span class="ffmpeg-progress kbps">{{kbps()}} Kb/s</span>
  </div>
</div>
```

4.4 Pokretanje aplikacije

Pokretanje aplikacije realizovano je unutar start.js i app.js datoteka.

Metoda nginxrtmp.start će na samom početku proveriti da li se u bazi nalaze već ranije upisani parametri, a ako ih pronađe koristiće ih za trenutne postavke. Ovo je realizovano u slučaju da dođe do naglog prekida u radu aplikacije ili prilikom ponovnog pokretanja aplikacije.

Zatim će aplikacija preuzeti IP adresu na kojoj će se izvršavati, obavestiti Websocket i pokrenuti funkciju startWebserver.

```
// start the app
nginxrtmp.start(process.env.RS_HTTPS == "true")
  .then(() => {
    return CamappzsData.checkJSONDb();
  })
  .then(() => {
    Camappzs.getPublicIp();
    Camappzs.bindWebSocketEvents();
    return camappzsApp.startWebserver();
  })
  .then(() => {
    return Q.fcall(Camappzs.restoreProcesses());
  })
  .catch((error) => {
    logger.error('Error starting webserver and nginx for application: ' + error);
  });
});
```

Funkcija za pokretanje veb servera, izbor porta i IP adresu na kojoj će se aplikacija izvršavati.

```
startWebserver () {
  var deferred = Q.defer();
  var server = null;

  logger.info('Starting ...');
  this.app.set('port', process.env.RS_NODEJS_PORT);
```

```

server = this.app.listen(this.app.get('port'), ()=> {
    this.app.set('io', require('socket.io')(server, {path: '/socket.io'}));
    this.secureSockets();
    this.app.set('server', server.address());

    // promise to avoid ws binding before the webserver has been started
    this.app.get('websocketsReady').resolve(this.app.get('io'));
    logger.info('Running on port ' + process.env.RS_NODEJS_PORT);
    deferred.resolve(server.address().port);
});

return deferred.promise;
}

```

4.5 HeaderControll segment

Ovaj segment aplikacije prikazuje se na samom vrhu iznad main scene. Sadrži naslov aplikacije i mogućnost izbora jezika koji će se koristit pri radu aplikacije. Trenutno aplikacija podržava srpski i engleski jezik.

Heder.html

```

<div>
    <p class="pull-right">
        <a ng-click="switchLanguage('en_US')"
            class="locales"
            ng-class="{'active': langIs('en_US')}>EN</a>
        /
        <a ng-click="switchLanguage('rs_SRB')"
            class="locales"
            ng-class="{'active': langIs('rs_SRB')}>SRB</a>
    </p>
    <h1>CamAppZS</h1>
</div>

```

Primer prevoda aplikacije na srpskom jeziku:

```

"login_username": "Korisničko ime",
"login_password": "Lozinka",

```

Unutar prvih navodnika nalazi se pojam, odnosno ključ kojeg je moguće prikazati bilo gde unutar aplikacije. Unutar drugih navodnika isписан је текст poruke koji se prikazuje na mestu gdje je postavljen ključ. Prevod je posebno definisan unutar JSON datoteke.

4.6 Footer segment

Ovaj segment prikazuje se na dnu aplikacije i sadrži informaciju o verziji aplikacije i opciju za odjavu iz aplikacije. Pritisom na dugme “Odjava”, aplikacija se ponovo vraća na scenu za pristup aplikaciji.

Footer.html

```
<div class="footer">
    <span class="version">
        v{{ version }}
    </span>
    <p class="pull-right links">
        <a ng-if="loggedIn" ng-click="logout()">{{ 'logout' | translate }}</a>
    </p>
</div>
```

5. Ispitivanje i verifikacija

U ovom poglavlju prikazano je testiranje i verifikacija rada aplikacije.

U okviru rada izvršena su ručna testiranja aplikacije. Izvršena je provera da li se aplikacija uspešno kompajlira na Docker Hub nalogu, rezultate je bio pozitivan.

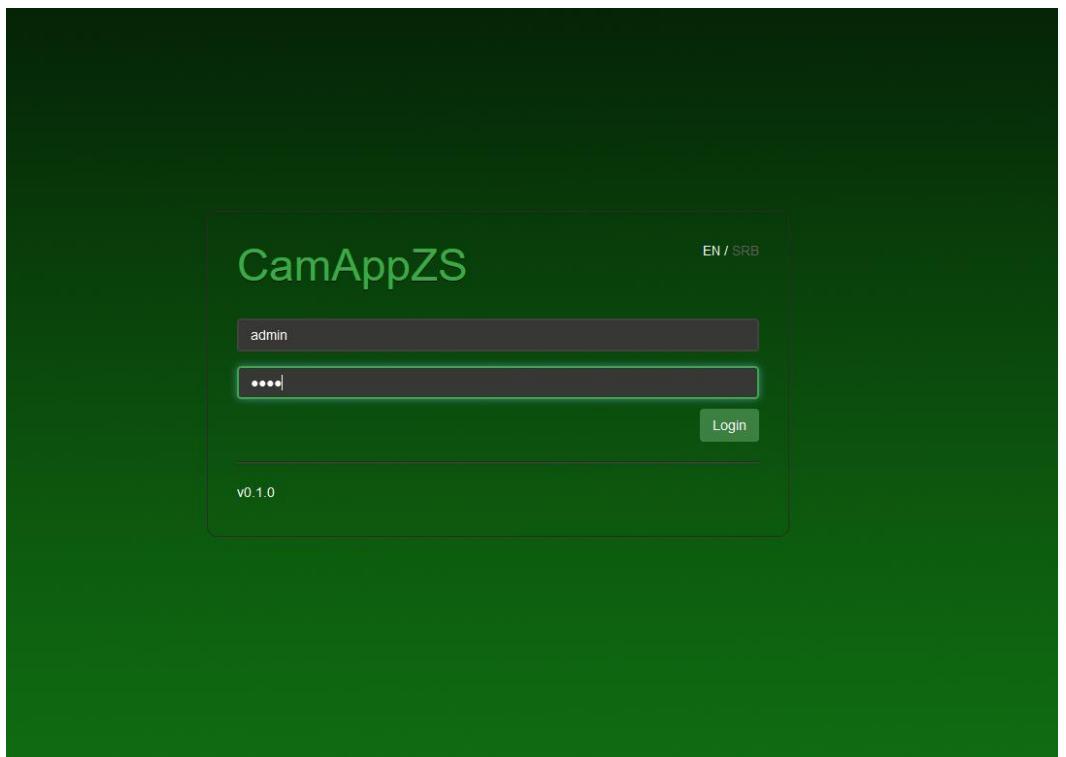
The screenshot shows the build history for the Docker Hub repository `ztile/camappzs`. It includes sections for 'Automated Builds' and 'Recent Builds'. In the 'Automated Builds' section, there is a table with columns: Docker Tag, Source, Latest Build Status, Autobuild, and Build caching. The table shows one entry with Docker Tag 'latest', Source 'master', Latest Build Status 'SUCCESS', Autobuild checked, and Build caching checked. A 'Trigger ▶' button is also present. In the 'Recent Builds' section, there are three entries, each with a green checkmark, indicating success. The details for these builds are: Build in 'master' (999c6a2a), Build in 'master' (184edbe9), and Build in 'master' (73a0c39c). Each entry includes the Docker tag (latest), commit hash, and time ago (15 hours ago or 5 days ago).

Docker Tag	Source	Latest Build Status	Autobuild	Build caching
latest	master	SUCCESS	✓	✓

Build	Docker Tag	Commit Hash	Time Ago
Build in 'master' (999c6a2a)	latest	999c6a2	15 hours ago
Build in 'master' (184edbe9)	latest	184edbe	5 days ago
Build in 'master' (73a0c39c)	latest	73a0c39	5 days ago

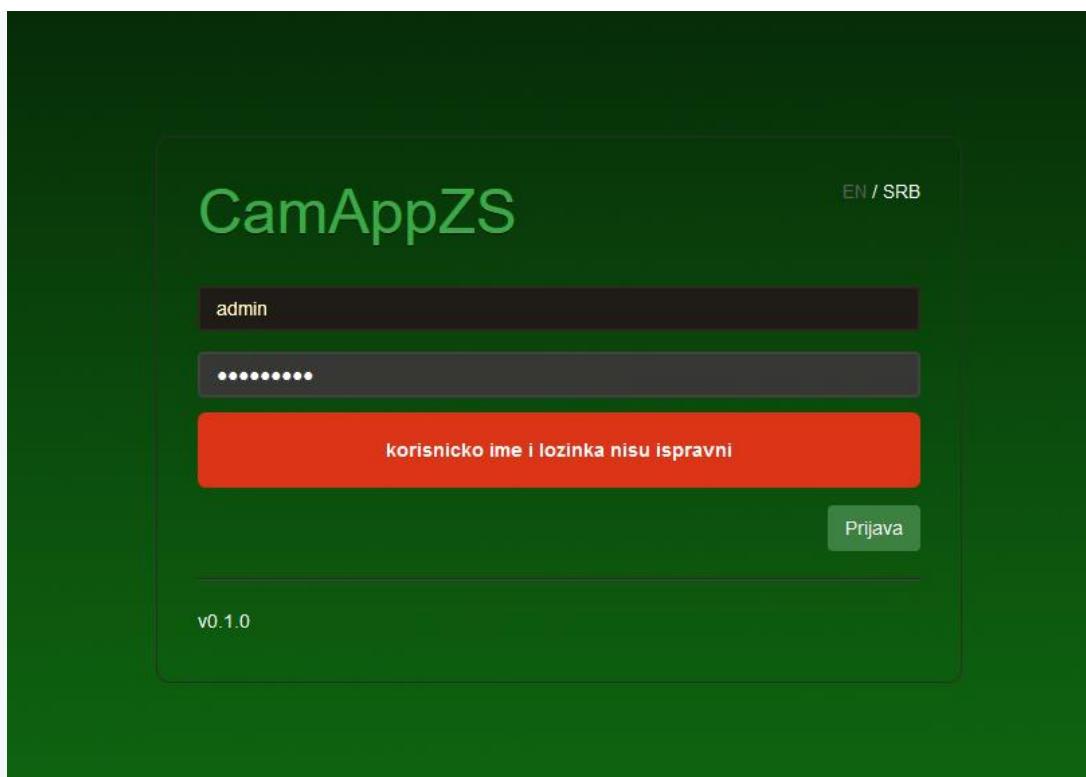
Slika 5.1 Prikaz uspešnog kompajliranja aplikacije na Docker Hub platformi

Nakon uspešnog kompajliranja na VPS-u testirane su i ostale funkcionalnosti aplikacije. Ispitana je funkcionalnost Login scene. Nakon unošenja korisničkog imena i lozinke pritiskom na dugme “Login” scena se uspešno izvršila.



Slika 5.2 Login scena

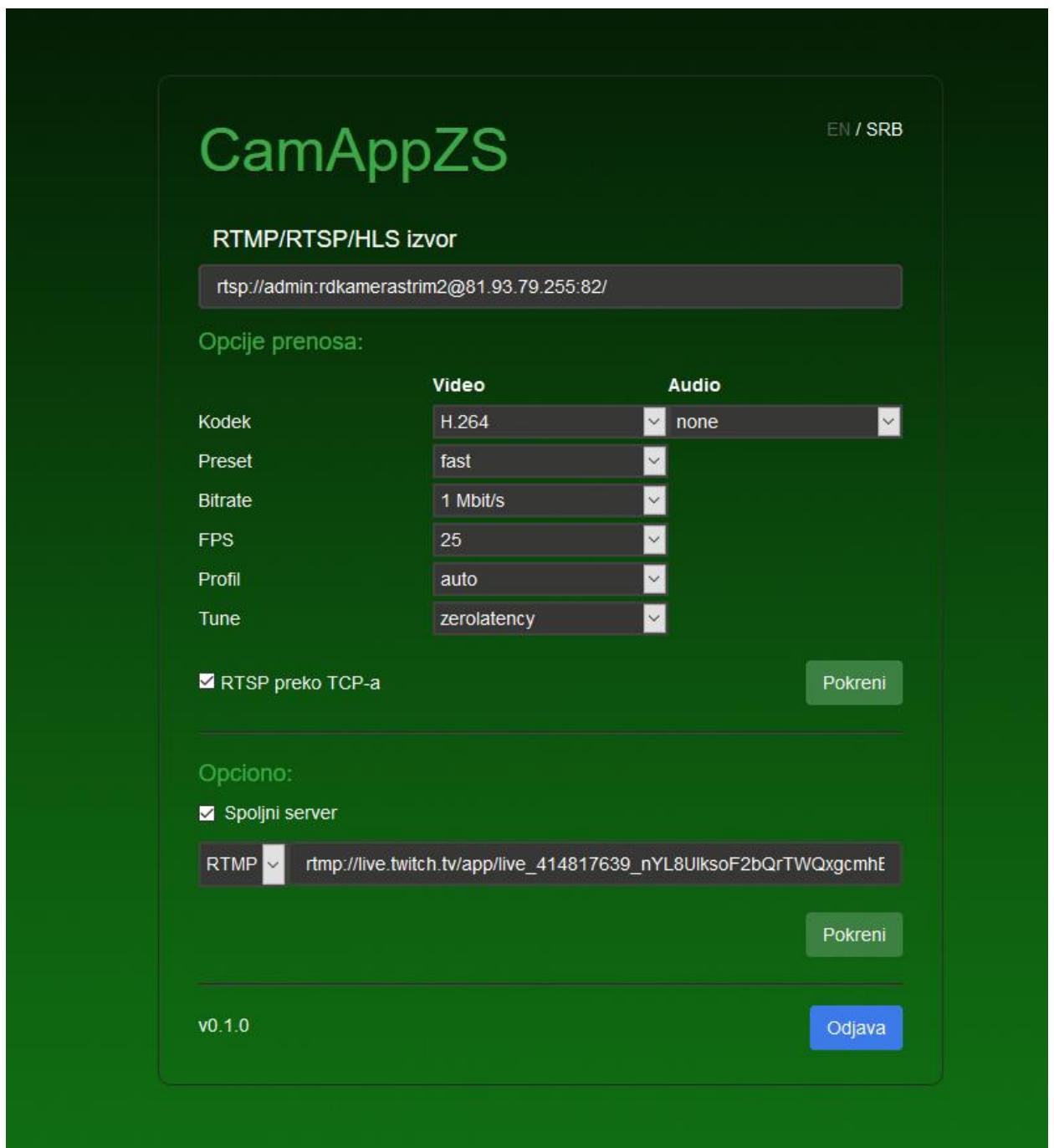
Takođe testirana je i scena kada se proslede netačni podaci za pristup. Rezultat je bio očekivan, pojavila se greška.



Slika 5.3 Login scena sa prosleđenim pogrešnim podacima za pristup

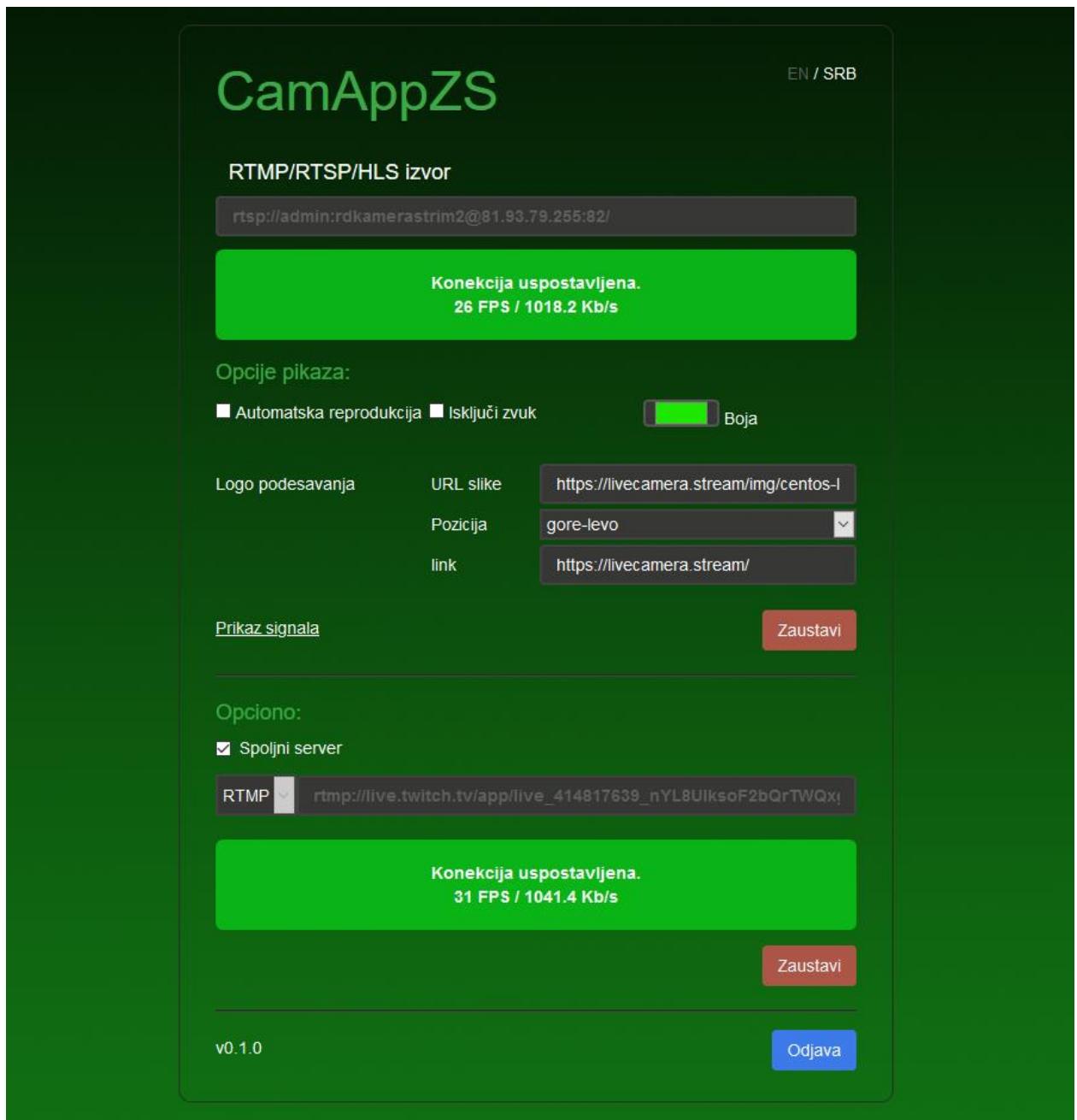
Prijavom u aplikaciju uspešno je testirana i Main scena.

Prosleđeni su parametri za pristup javnoj IP kameri pomoću RTSP protokola, zatim su podešene opcije kodovanja i pritiskom na dugme “Pokreni” pokrenut je proces povezivanja.



Slika 5.4 Main scena sa izabranim parametrima kodovanja i prenosa

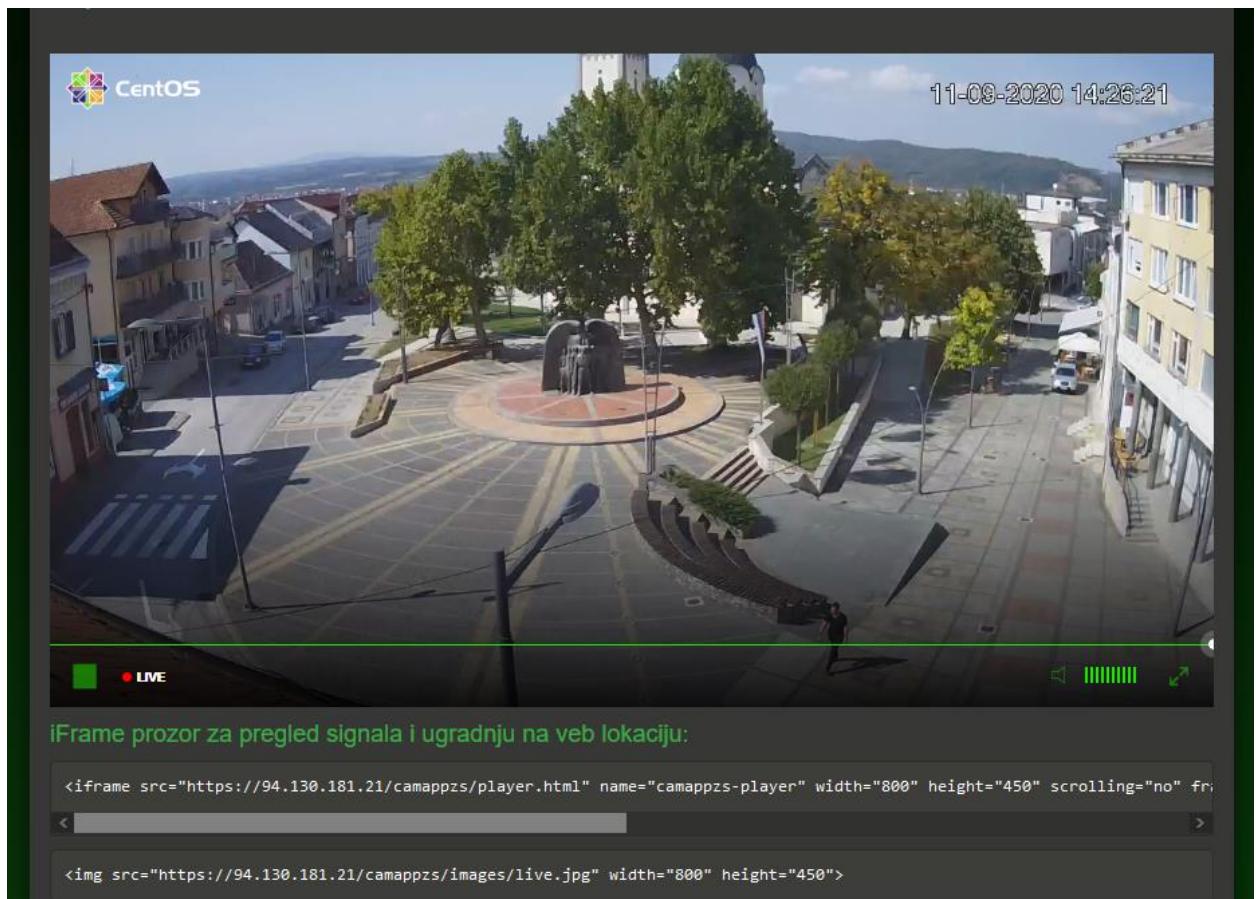
Nakon uspešnog povezivanja sa izvorom dobijena je poruka da je multimedijalni tok uspešno pokrenut.



Slika 5.5 Main scena sa uspešnim povezivanjem ka izvoru i spoljnog serveru

Testirana je mogućnost prosleđivanja signala ka spoljnem serveru, za ovaj primer korišćen je RTMP protokol, a kao spoljni server platforma Twitch. Testiranje je uspešno izvršeno.

Testiran je i prikaz multimedijalnog toka podataka unutar same aplikacije koju vrši Clapper. Testiranje je takođe bilo uspešno.



Slika 5.6 Scena sa prikazom signala unutar Clapper player-a

6. Zaključak

U ovom radu uspešno je realizovan proces prenosa multimedijalnog toka podataka u realnom vremenu. Rešen je početni problem, da se omogući prikaz sadržaja većem broju korisnika. O konkretnim brojevima teško je govoriti jer taj podatak zavisi od nekoliko faktora, a najbitniji su protok podataka na uređaju gde je aplikacija pokrenuta i brzina prenosa signala.

Jedan gledalac za prikaz signala u trajanju od 60 minuta, pri brzini prenosa od 2 Mbps potroši 0,86 GB serverskog protoka.

Ako se ukupan mesečni serverski protok od 50 000 GB podeli sa 0.86 GB, dobija se broj 58 139, što predstavlja ukupan broj gledalaca koje je moguće opslužiti pri brzini prenosa od 2 Mbps za period od jednog sata.

U slučaju da aplikacija prosleđuje signal ka serverima većeg kapaciteta, poput YouTube-a, broj gledalaca je neograničen, jer YouTube trenutno nema ograničenja u broju gledalaca kada su u pitanju prenosi u realnom vremenu.

Aplikacija se uspešno izvršava na svim uređajima zahvaljujući Docker kontejneru i može se koristiti kako na lokalnoj mreži tako i na globalnoj.

Dalji rad na ovoj aplikaciji mogao bi se usmeriti na dodavanju funkcionalnosti poput:

- Mogućnosti da se signal istovremeno prosleđuje na više od jednog spoljnog servera
- Podrška za H.265 standard
- Dodavanje Google AdSense video reklama pri samom pokretanju prozora za prikaz signala

7.Literatura

- [1] Dr Ilija Bašičević, Dr Miroslav Popović, Dr Vladimir Kovačević: Osnove računarskih mreža 1
- [2] Dr Milan Z. Bjelica, dr Nikola Teslić, mr Velibor Mihić, Softver u televiziji i obradi slike 1
- [3] FFmpeg wiki: <https://trac.ffmpeg.org/> , septembar 2020.
- [4] Docker docs: <https://docs.docker.com/> , septembar 2020.
- [5] NGINX docs: <https://docs.nginx.com/> , septembar 2020.
- [6] Clapper docs: <https://clappr.github.io/> , septembar 2020.
- [7] Node.js docs: <https://nodejs.org/en/docs/> , septembar 2020.