УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Кнежевић Милан

Реализација екосистема послужиоца за подршку персонализованим интерактивним апликацијама дигиталне телевизије

МАСТЕР РАД

Нови Сад, 2015

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	проф. др Никола Јорговановић

Студент:	Милан Кнежевић	Број индекса:	Е2 36/2014
Област:	Програмска подршка у телевизији и обради слике 2		
Ментор:	доц. др Милан Бјелица		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Реализација екосистема послужиоца за подршку персонализованим интерактивним апликацијама дигиталне телевизије

ТЕКСТ ЗАДАТКА:

Апликације дигиталне телевизије постају све комплексније и у могућности да прикажу све разноврсније садржаје прилагођене укусу корисника. Поставља се проблем како дате садржаје доставити апликацијама. Потребно је реализовати екосистем послужиоца који нуди решење овог проблема у сфери испоруке разноврсног мултимедијалног садржаја у циљу подршке персонализованим интерактивним апликацијама дигиталне телевизије. Решење треба да буде реализовано тако да нуди подршку за прикупљање метаподатака и развој компонети за прикупљање акција корисника, прикупљање садржаја и препоруку садржаја. Процену квалитета решења извршити испитивањем времена одзива послужиоца.

Руководилац студијског програма:	Ментор рада:

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**:		**Милан Кнежевић**	
Ментор, **МН**:		**доц. др Милан Бјелица**	
Наслов рада, **НР**:		**Реализација екосистема послужиоца за подршку персонализованим интерактивним апликацијама дигиталне телевизије**	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2015**	
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)		**7/51/0/9/13/0/0**	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		**Дигитална телевизија, послужилац, OTT, телевизија сладеће генерације, апликативни послужилац, интернет телевизија**	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		**Апликације дигиталне телевизије постају све комплексније и у могућности да прикажу све разноврсније садржаје прилагођене укусу корисника. Поставља се проблем како дате садржаје обезбедити апликацијама. У овом раду је изложена реализација екосистема послужиоца који нуди решење овог проблема у сфери испоруке разноврсног мултимедијалног садржаја преко интернета (OTT, енгл. *Over the top*) у циљу подршке персонализованим интерактивним апликацијама дигиталне телевизије. Решење је реализовано у виду радног оквира за развој послужица који нуди подршку за прикупљање метаподатака и развој компонети за прикупљање акција корисника, прикупљање садржаја и препоруку садржаја.**	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:		29/06/2015	
Чланови комисије, **КО**:	Председник:	**проф. др Милан Видаковић**	
	Члан:	**доц. др Данијела Лалић**	Потпис ментора
	Члан, ментор:	**доц. др Милан Бјелица**	

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**:		**Milan Knežević**	
Mentor, **MN**:		**PhD doc. Milan Bjelica**	
Title, **TI**:		**Implementation of server ecosystem to support personalized and interactive digital television applications**	
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**:		**2015**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/51/0/9/13/0/0**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**Digital television, OTT**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		**Digital television applications are increasingly becoming complex and able to present more diverse content tailored to the taste of users. This raises the problem of how to provide various content to those applications. This master thesis presents implementation of server ecosystem which offers a solution to this problem in the domain of the delivery of various multimedia content over the Internet (OTT, *Over the top*) to support personalized interactive digital television applications. Solution is implemented as a framework for the development of server side software, offering support for metadata acquisition and development of the components for collecting user actions, content aggregation and content recommendation.**	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		29/06/2015	
Defended Board, **DB**:	President:	**Milan Vidaković Ph. D.**	
	Member:	**Danijela Lalić, Ph. D.**	Menthor's sign
	Member, Mentor:	**Milan Bjelica, Ph. D.**	

Zahvalnost

Ovom prilikom bih želeo da se zahvalim mentoru, profesorima sa Fakulteta Tehničkih nauka i instituta RT-RK i kolegama iz *c-more* tima.

Veliku podršku u školovanju su mi pružili roditelji, prijatelji i svi bližnji i na tome sam im veoma zahvalan.

SADRŽAJ

1. Uvod....2
2. Teorijske osnove....6
2.1 Digitalna Televizija....6
2.1.1 Pojam digitalne televizije....6
2.1.2 OTT u televiziji....7
2.1.3 Internet televizija....9
2.2 Pojam poslužioca....9
2.2.1 Arhitektura korisnik - poslužilac....9
2.2.2 Aplikativni poslužilac....10
2.2.3 REST....11
2.2.4 Sistemi za skladištenje podataka....11
2.2.5 Cloud sistemi....12
3. Analiza zahteva....13
3.1 Analiza postojećih rešenja....13
3.1.1 Nangu.tv....13
3.1.2 Ericcsson MediaRoom Reach....14
3.1.3 Nagra MediaLive....14
3.1.4 AltiPlex....15
3.2 Zaključak Analize zahteva....15
3.3 Tehnologije za realizaciju zahteva....16
3.3.1 Java EE platforma....17
3.3.1.1 JAX-RS specifikacija....18
3.3.1.2 JPA specifikacija....18

3.3.2 JBoss aplikativni poslužilac 19
3.3.3 PostgreSQL 19
4. Koncept rešenja 20
4.1 Pregled elementarnih klasa 20
4.2 Predlog arhitekture i infrastrukture 22
4.3 Opis Radnog okvira poslužioca 23
4.3.1 Menadžer Resursa 24
4.3.2 Magistrala Informacija 26
4.3.3 Obogaćivač sadržaja 27
4.3.4 Komponenta tekstualne analize 28
4.3.5 Baza znanja 28
4.3.6 Menadžer sistemima za preporuku 29
4.3.7 Menadžer upita za sadržajem 30
4.3.8 Skladište podataka 30
4.3.9 Menadžer korisničkih naloga 31
4.3.10 Menadžer sistema 32
4.3.11 Sakupljači sadržaja 33
4.3.12 Sakupljači korisničkih akcija 34
4.3.13 Sistemi za preporuku sadržaja 34
4.3.14 Usmerivač sadržaja 35
4.3.15 Sprega za mrežni pristup (REST API) 35
4.3.15.1 Mrežni pristup sadržajima 36
4.3.15.2 Mrežni pristup izvorima sadžaja 38
4.3.16 Mrežna aplikacija 38
5. Procena kvaliteta rešenja 40
5.1 Tehnike procene 40
5.1.1 Funkcionalno poređenje sa konkurentnim rešenjima 40
5.1.2 Ispitno okruženje 41
5.1.3 Apache JMeter 41
5.1.4 Ispitni skup 42
5.1.4.1 Ispitni slučaj vremena odziva 43
5.1.4.2 Ispitni slučaj vremena odziva na promenljiv broj korisnika 43
5.1.4.3 Ispitni slučaj simulacije stvarnog slučaja korišćenja 43
5.1.5 Mogućnost proširivanja rešenja 43
5.1.6 Opšti parametri stabilnosti 43

5.1.7 Ispitni slučaj vremena odziva sa različitih lokacija 44
5.2 Rezultati procene 44
5.2.1 Ispitni slučaj vremena odziva 44
5.2.2 Ispitni slučaj vremena odziva na promenljiv broj korisnika 44
5.2.3 Ispitni slučaj simulacije stvarnog slučaja korišćenja 46
5.2.4 Stabilnost sistema 46
5.2.5 Odziv iz referentne aplikacije 47
5.2.6 Fleksibinost rešenja 47
5.2.7 Vreme odziva sa različitih lokacija 48
6. Zaključak 49
7. Literatura 50

SPISAK SLIKA

Slika 1.1 Izgled koncepta televizije sledeće generacije 3
Slika 2.1 Primena OTT poslužioca u digitalnoj televiziji 9
Slika 2.2 Primer arhitekture korisnik - poslužilac 10
Slika 3.1 *MediaRoom Reach* arhitektura 14
Slika 3.2 Arhitektura *Alticast Altiplex* 15
Slika 3.3 Prikaz tipične višeslojne aplikacije 18
Slika 4.1 Prikaz UML dijagrama elementarnih klasa sadržaja 21
Slika 4.2 UML dijagram elementarnih klasa korisnika 22
Slika 4.3 Arhitektura i infrastruktura poslužioca 23
Slika 4.4 UML dijagram sekvence ciklusa obogaćivanja 27
Slika 4.5 UML dijagram klase koja opisuje korisničku akciju 34
Slika 4.6 Snimak ekrana mrežne aplikacije 39
Slika 5.1 Prikaz stabilnosti rešenja 46

SPISAK TABELA

Tabela 5.1 Poređenje funkcionalnosti OTT poslužilaca 41
Tabela 5.2 Vreme odziva poslužica na poziv metoda.....44
Tabela 5.3 Vreme odziva metode sadržaja na zahtev za promenjiv broj korisnika.....45
Tabela 5.4 Vreme odziva zahteva za TV vodičem za promenjiv broj korisnika 45
Tabela 5.5 Vreme odziva zahteva za preporukama za promenjiv broj korisnika 45
Tabela 5.6 Vreme odziva na zahtev za listom kanala za promenjiv broj korisnika.....45
Tabela 5.7 Vremenski odziv metoda u stvarnom slučaju korišćenja 46
Tabela 5.8 Udeo poslužioca u utrošku vremena isporuke sadržaja do korisnika.....47
Tabela 5.9 Udeo u vremenu odziva sa različitih lokacija 48

SKRAĆENICE

API - *Application Programming Interface*, opis aplikativne programske sprege

CDN - *Content Delivery Network*, mreža sa dostavljanje sadržaja

DVR - *Digital video recorder*, digitalni video snimač

EJB - *Enterprise Java Bean,* komponente koje opisuju poslovnu logiku kod JEE aplikacija

HbbTV - *Hybrid Broadcast Broadband TV*, standard za hibridni prenos televizijskog sadržaja

HTTP - *HyperText Transfer Protocol*, protokol za prenos *hypertext*-a

IP - *Internet Protocol*, internet protokol

IPTV - *Internet Protocol Television*, televizija preko IP protokola

JAX-RS - *Java API for RESTful Web Services*, Java API za razvoj usluga zasnovanih na REST arhitekturi

JEE - *Java Enterprise Edition*, java za razvoj poslovnih aplikacija

JPA - *Java Persistance API*, Java API za skladištenje podataka

JSON - *JavaScript Object Notation*, JavaScript objektna notacija

OTT - *Over the Top*, Sadržaj koji se prenosi internet vezom

PVR - *Personal video recorder*, lični video snimač

REST - *Representational State Transfer*, prenos reprezentacija stanja

STB - *Set-top Box*, Set-top boks uređaj

TV - *Television*, Televizija

URL - *Uniform Resource Locator*, jedinstvena adresa resursa

URI - *Uniform Resource Identifier*, jedinstveni identifikator resursa

XML - *EXtensible Markup Language*, proširivi jezik za označavanje

1. Uvod

Televizija je svojim pojavljivanjem napravila revoluciju u sferama informisanja, kulture i razonode. Kako je čovek biće zavisno o napredku i razvoju, taj napredak je delom zahvatio i televiziju i sva stanovišta televizije.

Nekada je televizija predstavljala medijum za prenos zvuka i slike do udaljenih korisnika, dok je danas skup usluga koje obezbeđuje televizija značajno povećan. Danas televizijski prijemnik može da zameni veliki broj svakodnevnih uređaja (video rekorder, telefon, igračke konzole...) i usluga (video na zahtev, vodič kroz program...).

Najveća prekretnica u razvoju televizije svakako je prelazak sa analognog prenosa signala na digitalni prenos. Pojavljivanjem digitalne televizije probijaju se granice koje su bile nedostižne korišćenjem analognih tehnologija. U televizijski signal je, osim video i audio sadržaja koji je daleko većeg kvaliteta, omogućeno ugrađivanje informacija o programskoj šemi, višejezičnog teleteksta i višejezičnih prevoda, omogućena je roditeljska kontrola i pojavio se elementarni tip interaktivnosti televizije sa korisnikom.

Digitalna televizija zahteva dodatne blokove fizičke arhtiekture koji mogu, ali ne moraju biti deo televizijskog prijemnika, odnosno mogu biti deo uređaja koji se naziva STB (engl. *set-top box*). Osnovna uloga STB uređaja je da iz televizijskog signala preuzme potrebne informacije i da ih prikaže na ekranu. Kako se razvoj digitalne televizije i računarske tehnike nastavljaju, STB uređaji poseduju sve veću količinu raspoloživih procesorskih I memorijskih resursa. Zahvaljujući povećanju dostupnosti ovih resursa danas se realizuju sve kompleksnije grafičke korisničke sprege I obezbeđuje sve veći kvalitet sadržaja koji se prikazuju.

Na STB uređajima sve češće možemo videti, pored priključka za prenos televizijskog signala, i priključak koji omogućava vezu sa internetom. Potreba za internet vezom je u početku proisticala iz potrebe za interakcijom gledaoca sa određenim televizijskim emisijama, dok je danas njena upotreba sve šira. Internet vezom se otvaraju vrata ka izuzetno velikoj količini

sadržaja koja je dostupna na Internetu. Teži se tome da se neobavezno „surfovanje“ internetom izmesti iz neudobnog računarskog okruženja u prostor dnevne sobe gde se kao centralni uređaj nalazi televizijski prijemnik.

Ubrzan razvoj digitalne televizije donosi nove skupove funkcionalnosti, a kako televizijski prijemnici nisu podložni čestoj nabavci novih od strane korisnika, STB uređaj mora da obezbedi fleksibilnost tako što će da poseduje svu potrebnu fizičku arhitekturu i programsku podršku potrebnu za digitalnu televiziju. Iz ovoga možemo zaključiti da STB uređaj koji poseduje vezu sa internetom (hibridni STB uređaj) postavlja kao motor pokretač ideje televizije sledeće generacije. Jedan koncept aplikacija digitalne je prikazan na Slici 1.1.

Slika 1.1 Izgled koncepta televizije sledeće generacije

Kako bi se aplikacije digitalne televizije, koje se izvršavaju na hibridnim STB uređajima, snadbele traženim sadržajima potrebno je da postoji poslužilac koji će biti u mogućnosti da podrži interaktivnost i personalizaciju datih aplikacija i da obezbedi raznorodne sadržaje koji su u skladu sa ukusom korisnika. Sa druge strane poslužilac mora da obezbedi veliki stepen fleksibilnosti i prilagodljivosti novim standardima koje diktira razvoj digitalne televizije. U ovoj tački možemo zaključiti da ukoliko je hibridni STB uređaj motor pokretač televizije sledeće generacije onda je ovakav poslužilac, u najmanju ruku, upravljač televizije sledeće generacije.

Analizom tržišta došli smo do zaključka da su zahtevi TV aplikacija sledeće generacije:

- Bogata kolekcija sadržaja na zahtev;

- Bogat izbor pratećih metapodataka koji uključuju probrane OTT sadržaje;
- Podrška za socijalne interakcije;
- Podrška za različite ciljne uređaje / ekrane;
- Elekronski vodič kroz program i video na zahtev;
- preporučivanje sadržaja u skladu sa profilom korisnika.

U ovom radu je predstavljeno jedno od rešenja za ispunjenje ovih zahteva. Kao rešenje se predlaže OTT (engl. *Over the top*) poslužilac. OTT predstavlja pružanje usluga preko Internet veze, u našem slučaju su to usluge podrške personalizovanim interaktivnim aplikacijama digitalne televizije.

Kao dodatak ovom skupu zahteva realizovani poslužilac pronalazi prostor za unapređivanje u sferi povezivanja fiksnih metapodataka i sadržaja koji su dostupni na internetu, i njihova povezivanje sa katalozima video sadržaja na zahtev i elektronskim programskim vodičima.

Iz gore navedenih zahteva možemo zaključiti da poslužilac mora da ima sledeće komponente: sistem za skladištenje podataka, sistem za izvršavanje poslovne logike, sistem za preporuku i sistem koji će omogućiti mrežni pristup.

Ideja je da poslužilac, koji je tema rada, ne isporučuje konkretan sadržaj, već se pod pojmom „isporuka sadržaja“ misli na isporučivanje metapodataka koji opisuju sadržaj i ukazivača na konkretan sadržaj. Ako tako postavimo stvari, poslužilac treba da omogući isporuku sadržaja, što preporučenog, što filtriranog prema određenim uslovima, zajedno sa metapodacima vezanim za dati sadržaj. Takođe treba da omogući rukovanje korisničkim nalozima i sakupljanje korisničkih akcija.

Ovaj rad je struktuiran u sedam poglavlja.

U drugom poglavlju su navedene teorijske osnove digitalne televizije koje uključuju pojam digitalne telvizije, upotrebu OTT u televiziji i upoznavanje sa pojmom internet televizije, a zatim i sa pojmovima poslužioca, aplikativnog poslužioca, sistema za skladištenje podataka i *Cloud* sistema.

Treće poglavlje sadrži analizu već postojećih rešenja i pregled referentnih tehnologija koje su korišćene pri realizaciji ekosistema poslužioca.

U četvrtom poglavlju je izložen koncept rešenja poslužioca za podršku aplikacijama digitalne televizije.

Peto poglavlje je posvećeno proceni kvaliteta realizovanog rešenja. Prikazani su ispitni slučajevi i okruženje u kome ispitivanje vršeno, rezultate ispitivanja i komentare rezultata ispitivanja. Na kraju je napravljena diskusija procene kvaliteta.

U šestom poglavlju je dat zaključak i predlog mogućnosti proširenja i unapređenja rada u budućnosti.

Sedmo poglavlje sadrži spisak korišćene literature tokom izrade rada.

2. Teorijske osnove

U ovom poglevlju su navedene teorijske osnove za upoznavanje sa teorijskim i konceptualnim pojmovima korištenim u radu. Ovo poglavlje sadrži:

- kratak osvrt na principe digitalne televizije;
- OTT koncept i njegova uloga u digitalnoj televiziji;
- Internet televizija, kao jedna od primena OTT koncepta u digitalnoj televiziji;
- Objašnjenje pojma poslužioca i pojma aplikativnog poslužioca;
- objašnjenje REST smernica;
- upoznavanje sa sistemima za skladištenje podataka i sistemima „u oblaku“ (engl. *cloud*) sistemima.

2.1 Digitalna Televizija

2.1.1 Pojam digitalne televizije

Digitalna televizija (DTV)[1] podrazumeva prenos zvuka, slike i dodatnih informacija putem digitalno obrađenog i multipleksiranog signala. Što se tiče prenosa digitalnog signala, digitalna televizija je donela mogućnost prenosa više televizijskih programa preko jednog kanala u spektru, mogućnost prenosa slike koja je daleko većeg kvaliteta, prenos koji je otporniji na interferenciju i kvalitet prenosa koji je istog kvaliteta nezavisno od udaljenosti predajnika.

Pojavom digitalne televizije omogućene su nove specifične usluge koje obuhvataju: elektronski vodič kroz program sa podsetnicima; izbor audio jezika i jezika za prevod; roditeljske kontrole; interaktivne i multimedijalne sadržaje.

Prijemnike digitalne televizije možemo podeliti na integrisane televizijske prijemnike i STB uređaje. Integrisani televizijski prijemnici, pored ekrana, poseduju svu potrebnu fizičku arhitekturu i programsku podršku potrebnu za obradu i prikaz televizijskog signala. Drugi slučaj

podrazumeva da je celokupna potrebna fizička arhitektura i programska podrška sadržana u posebnom uređaju koji se naziva STB uređaj. STB uređaji su češće rešenje za prijem televizijskog signala jer nude fleksibinost, koja je usled brzog razvoja novih funkcionalnosti digitalne televizije itekako potrebna kako bi korisnici na lakši i pristupačniji način bili u mogućnosti da dođu do novih funkcionalnosti. STB uređaji imaju sve više resursa i omogućavaju pristup sve većem broju usluga i sadržaja, pa su samim tim podložniji zameni od strane korisnika ili operatera.

Prema tipu prenosa sadržaja digitalne televizije možemo razlikovati zemaljski, satelitski i kablovski prenos kao emiterski pristup i IPTV (engl. *Internet Protocol Television*) kao širokopojasni pristup. Sve veći je broj STB uređaja koji mogu da podrže i emiterski i širokopojasni pristup, i to su takozvani hibridni STB uređaji.

Prenos digitalnog signala je standardizovan i postoji više standarda po kojima se digitalni signal prenosi. Najzastupljeniji standardi jesu DVB grupa standarda, ATSC standardi i ISDB grupa standarda. U Evropi se koristi DVB grupa standarda, koja je primenjena i u Srbiji. Treba naglasiti da prenos televizijskog sadržaja putem IPTV i dalje nema formalno usvojene i primenjene standardizacije.

Digitalna televizija jeste donela mnogo novina, ali postoje zahtevi koje nije moguće ostvariti upotrebom digitalne televizije. Televizija sledeće generacije ima velike zahteve u sferi personalizacije i visokog nivoa interakcije, kao i sadržaja na zahtev koji ne mogu biti ispunjeni upotrebom nekog od standarda digitalne televizije. Internet kao dodatak digitalnoj televiziji ili Internet televizija se nameću kao rešenje ovog problema.

Internet u digitalnoj televiziji je prisutan od pojave koncepta povezane televizije (engl. *Connected TV*) koji je podrazumevao da televizijski prijemnik ili STB uređaj ima vezu sa internetom i da ima ugrađen Internet pretraživač. HbbTV (engl. *Hybrid Broadcast Broadband TV*) predstavlja industrijski standard kojim je definisan način dostavljanja sadržaja krajnjim korisnicima koristeći koncept povezane televizije. Usluge koje se prenose kroz HbbTV uključuju poboljšan teletext, video na zahtev, elektronski vodič kroz program, interaktivne akcije, personalizaciju i ostale multimdijalne aplikacije.

2.1.2 OTT u televiziji

OTT (engl. *Over-the-top*) je konceptualan pojam koji se sve češće sreće u oblasti digitalne televizije. OTT predstavlja prenos televizijskog sadržaja i/ili pružanje drugih usluga u sferi digitalne televizije preko internet veze, uz to da provajder usluga ili sadržaja (OTT provajder) ne mora biti ni u kakvoj vezi sa provajderom date internet usluge.

OTT se često greškom dovodi u vezu sa IPTV sistemom iako su oni potpuna suprotnost. Glavna razlika je u tome što da bi operater pružao usluge IPTV mora da ima svoju zatvorenu mrežnu infrastrukturu kojoj je moguće pristupiti samo plaćanjem pretplate operateru, dok OTT provajderi svoje usluge nude preko Internet mreže. Takođe, kod IPTV sistema usluge su optimizovane i prilagođene ciljnim uređajima koje obezbeđuje operater, dok OTT nije platformski zavisan.

OTT provajder je provajder usluga koji nudi usuge (ili deo usluga) operatera digitalne televizije, ali ne poseduje mrežnu infrastruktru operatera niti je iznajmljuje od operatera, već se oslanja na Internet kao medijum za prenos. OTT provajder nudi OTT usluge koje, kao rezultat isporučuju OTT sadržaje.

Neki od predstavnika OTT provajdera su: *Nangu.tv* [2], *Ericcsson MediaRoom* [3], *Nagra MediaLive* [4], *Alticast Altiplex* [5], *Netflix* [6], *YouTube* [7] itd. Usluge koje nude su video na zahtev, Catch-up TV, interaktivne aplikacije, „televizija svuda“ itd, a sadržaji koje nude mogu biti audio, video, televizijski sadržaji, različiti tipovi metapodataka itd.

Zanimljivo je da udeo saobraćaja koji se ostvari za prenos OTT sadržaja iznosi oko 67% u pojedinim vremenskim periodima [8].

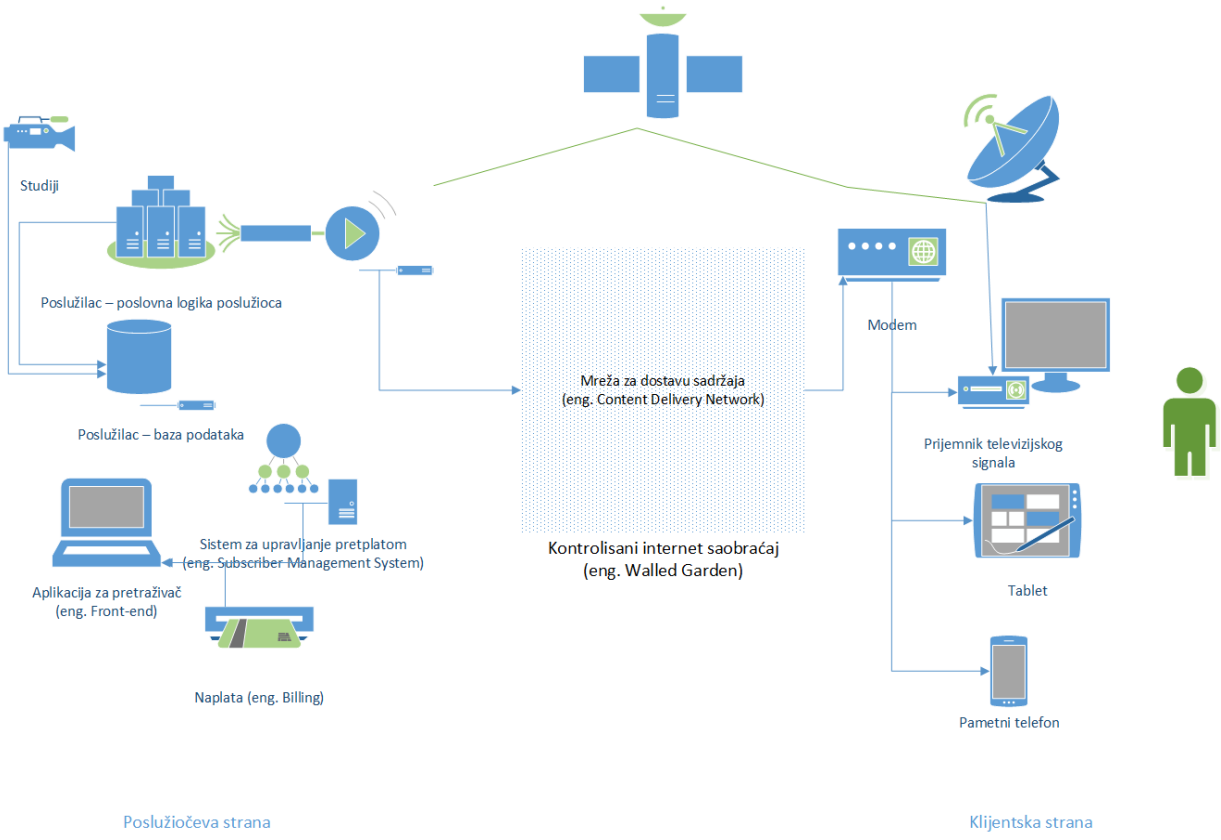
Prednosti OTT-a:

- provajder ne mora da poseduje nikakvu mrežnu infrastrukturu već se koristi postojeća koju pruža provajder internet usluge, tako da OTT provajder nema potrebe za ulaganja u infrastrukturu;
- OTT provajder može da pruža usluge korisnicima preko više provajdera internet usluga istovremeno;
- korisnicima koji poseduju više uređaja, gde dati uređaji ne koriste istog provajdera internet usluga, može biti serviran isti sadržaj. Samim tim korisniku se omogućava veći izbor uređaja;
- OTT omogućava korisnicima pristup drugim sadržajima sa interneta koje ne nudi izabrani OTT provajder;
- OTT omogućava direktnu veza između tvoraca sadržaja i krajnjih korisnika, bez potrebe za operaterom kao posrednikom.

Mane OTT:

- Kvalitet usluge ne može biti garanotovan zbog prenosa preko internet veze.

Na Slici 2.1 prikazan je koncept poslužioca OTT sadržaja u digitalnoj televiziji koji svoje funkcionalnosti korisnicima nudi preko kontrolisane internet mreže sa dostavljanje sadržaja (engl. *Content Delivery Network*) i preko konvencionalnih prenosa digitalne televizije.



Slika 2.1 Primena OTT poslužioca u digitalnoj televiziji

2.1.3 Internet televizija

Internet televizija, kao tip OTT usluge, podrazumeva digitalni prenos televizijskog sadržaja putem Interneta. Internet televizija predstavlja suprotan koncept od tradicionalnih tipova televizije (zemaljska, satelitska i kablovska) jer za njen prenos nije potrebna dodatna infrastruktura već je dovoljna Internet veza.

Gledalac koji koristi Internet televiziju može da konzumira sadržaje od neograničenog broja provajdera sadržaja, gde nije mandatorno da provajderi moraju da budu iz iste zemlje kao i gledalac. Sadržaji mogu biti prenošeni preko toka podataka i istovremeno gledani ili mogu biti preneti na uređaj korisnika pa gledani nakon prenosa. Gledalac može da poseduje neograničan broj uređaja različitog tipa i sa svih njih istovremeno da pristupa sadržaju.

2.2 Pojam poslužioca

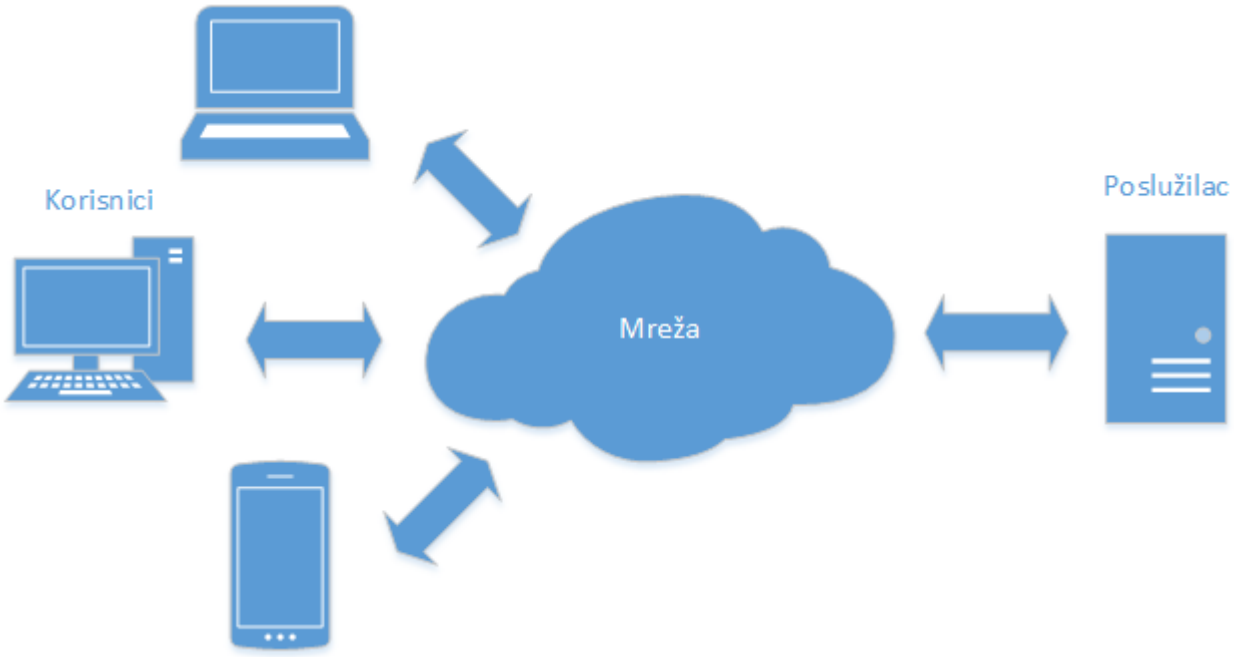
2.2.1 Arhitektura korisnik - poslužilac

Najčešće korišćena arhitektura po kojoj se postavljaju mreže i mrežno bazirane usluge u današnjem internetu jeste „korisnik - poslužilac“[9]. Uloga poslužioca je da pruža neku funkcionalnost ili uslugu jednom ili više korisnika koji zahtevaju datu uslugu. Veza između

korisnika i poslužioca se najčešće uspostavlja preko računarske mreže između dva računara. Vezu započinje korisnik dok poslužilac čeka na zahtev.

Poslužioci se klasifikuju prema tipu usluga koje nude ili prema protokolu koji realizuju. Tipični primeri poslužilaca su mrežni poslužilac koji nudi usluge dostavljanja mrežne stranice, poslužilac datoteka koji dostavlja datoteke, aplikativni poslužilac koji izvršava poslovnu logiku i vraća njen rezultat i drugi.

Na Slici 2.2 je prikazan klasičan primer arhitekture korisnik poslužilac.



Slika 2.2 Primer arhitekture korisnik - poslužilac

2.2.2 Aplikativni poslužilac

Aplikativni poslužilac [10] (engl. *Application server*) je programski okvir za razvoj koji obezbeđuje podršku za pravljenje mrežnih aplikacija, kao i okruženje za pokretanje i izvršavanje istih. Čini ga skup komponenti kojima programer pristupa kroz programsku spregu koja je definisana platformom aplikativnog poslužioca. Uloga aplikativnog poslužioca jeste da bude sloj apstrakcije između platforme i razvijane aplikacije, tako da obezbeđuje prenosivost aplikaciji i omogućava programeru da se fokusira na razvoj poslovne logike. Usluge koje aplikativni poslužilac apstrahuje mogu biti proširivost sistema (engl. *scalability*), oporavak od grešaka, raspoređivanja opterećenja, rukovanje transakcijama i dr.

Poslovna logika određuje transformacije i/ili proračune nad podacima i sprovodenje putanja i metoda kojima se rukuje podacima.

Aplikativni poslužilac se ponaša kao virtuelna mašina koja omogućava pokretanje aplikacija, transparentno rukovanje korisničkim vezama i vezama ka sistemima za skladištenje podataka i ostalim spoljnim zavisnostima koje su potrebne za izvršavanje aplikacije.

Većina vodećih rešenja aplikativnih poslužilaca je realizovana u skladu sa *Java Enterprise Edition* [11] ili *Microsoft .NET* [12] specifikacijama.

2.2.3 REST

REST (engl. *Representational State Transfe*r) [13] je programski šablon koji se sastoji od skupa pravila i dobrih praksi za projektovanje proširivog pristupa mrežnim poslužiocima. Sistemi koji koriste REST su korisnik-poslužilac arhitekture koje uglavnom komuniciraju preko HTTP protokola i koriste isti skup metoda kao i HTTP protokol.

Ukoliko se pri realizacije web usluge ispoštuju pravila koja nalaže REST može se očekivati da web usluga ima dobre performanse, laku proširivost i laku izmenljivost.

REST šablon ne pravi razliku između podataka i funkcionalnosti, već i jedne i druge posmatra kao resurse kojima je moguće pristupiti preko jedinstvenog identifikatora (URI, engl. *Uniform Resource Identifier*).

Elementarne metode koje REST koristi su dobavi (engl. *GET*), postavi (engl. POST), umetni (engl. PUT), ukloni (engl. DELETE). Metoda „umetni“ je namenjena za pravljenje novog resursa. Metoda „dobavi“ dobavlja trenutno stanje resusa u nekoj reprezentaciji. Metoda „postavi“ postavlja zadati resurs u novo stanje, a metoda „ukloni“ briše zadati resurs.

2.2.4 Sistemi za skladištenje podataka

Sistem za skladištenje podataka (engl. *Data Storage*) [14] je programska podrška koja obezbeđuje skladište podataka koje je postavljeno u skladu sa određenim modelom podataka, kao i operacije za skladištenje i rukovanje podacima.

Model podataka je skup pojmova i pravila koja opisuju podatke, njihove karakteristike i veze između njih. Neki od važnijih modela podataka su relacioni model i objektno relacioni model, koji su specifični za baze podataka, modeli ključ-vrednost, uređeni ključ-vrednost (engl. *ordered key-value*), modeli podataka zasnovani na dokumentu i graf modeli podataka koji su specifični za NoSQL [15] sisteme za skladištenje podataka.

Osnovni skup operacija za skladištenje i rukovanje podacima čine operacije za:

- skladištenje, koja dodaje novi unos u sistem za skladištenje podataka;
- čitanje, kojom se vrši dobavljanje, pretraga i pregled postojećih unosa u sistemu za skladištenje podataka;
- izmenu, koja omogućuje izmenu postojećih unosa u sistemu za skladištenje podataka;

- brisanje, kojom se postojeći unosi brišu iz sistema za skladištenje podataka.

Sisteme za skladištenje podataka možemo podeliti na dva načina:

Prema organizaciji arhitekture sistema mogu se podeliti na centralizovane i distribuirane sisteme za skladištenje podataka.

Prema podržavanju SQL standarda mogu se podeliti na baze podataka, koje podržavaju SQL standard i NoSQL baze podataka koje uglavnom nepotpuno podržavaju SQL standard jer im on nije primarni način pristupa.

2.2.5 Cloud sistemi

Cloud sistemi [16] se zasnivaju na korisnik-poslužilac arhitekturi. Fizički se sastoje od jednog ili više računarskih jedinica (čvorova) poslužilaca koji su povezani i ostvaruju komunikaciju preko računarske mreže koja ima veliki protok i malo kašnjenje. Svoj rad zasnivaju na virtuelizaciji koju obezbeđuju programskom podrškom koja predstavljaja sloj apstrakcije između usluga i resursa koje nude i fizičke infrastrukture računara poslužilaca na koju se oslanjaju. Resursi fizičkih poslužilaca su podeljeni u virtuelne poslužioce koji su nezavisni jedan od drugog i na lak način se mogu nadograđivati i sa njima se lako može rukovati, čak i u toku rada. Virtuelni poslužioci se dodeljuju administrativnim korisnicima prema njihovim zahtevima u svrhu pružanja usluga.

Modeli usluga koje cloud sistemi nude poređani prema kompleksnosti su:

- Infrastruktura kao usluga (IaaS, engl. *Infrastrukture as a Service*) – obezbeđuje virtuelne poslužioce, virtuelne računare, prostor za skladištenje podataka, raspoređivače opterećenja
- Platforma kao usluga (PaaS, engl. *Platform as a Service*) – obezbeđuje operativni sistem, bazu podataka, okruženje za razvoj pogramske podrške u određenom programskom jeziku, platformu za mrežne poslužioce.
- Programska podrška kao usluga (SaaS, engl. *Software as a Service*) – obezbeđuje određenu programsku podršku kojoj korisnici mogu da pristupaju i da je koriste na zahtev.

3. Analiza zahteva

U ovom poglavlju je napravljena analiza već postojećih rešenja koja delom ili u potpunosti ispunjavaju zahteve realizacije OTT poslužioca. Nakon toga je napravljen zaključak i urađen pregled referentnih tehnologija koje su korišćene pri realizaciji ekosistema poslužioca za podršku aplikacijama digitalne televizije.

3.1 Analiza postojećih rešenja

Postojeća rešenja koja će biti uzeta u razmatranje su jedna od vodećih rešenja na tržištu poslužitelja OTT sadržaja. Analizirana rešenja su Nangu.tv, Ericcsson MediaRoom, Nagra MediaLive i AltiPlex. Data rešenja nude usluge u sferi IPTV i OTT, ali kako je tema rada vezana za OTT sadržaje analiziraćemo ispunjene zahteve koji su vezani za isporuku OTT sadržaja.

3.1.1 Nangu.tv

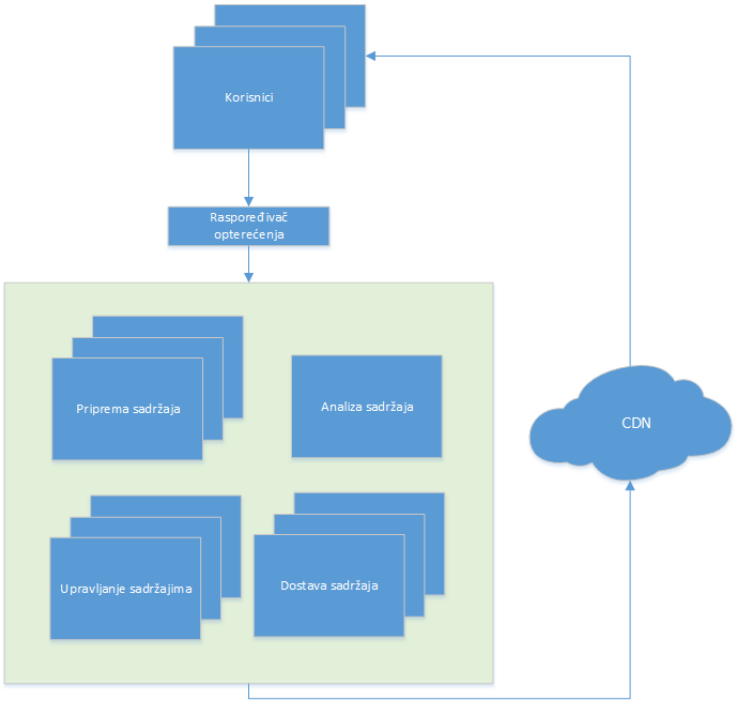
Nangu.tv [2] je platforma za dostavu IPTV i OTT sadržaja koju je na tržište plasirala istoimena firma bazirana u Češkoj. Od OTT usluga, Nangu.tv platforma nudi isporuku videa na zahtev, podršku za socijalne interakcije i reklamiranje, mrežni PVR, isporuku liste kanala i elektronskog vodiča kroz program. Obezbeđuje podršku za različite tipove uređaja i različite veličine ekrana. Sistemom je moguće upravljati kroz administratorsku mrežnu aplikaciju ili je moguće napraviti aplikaciju koja se oslanja na nangu.tv administratorsku spregu. Postoji i podrška za proširivanje i za povezivanje sistema sa postojećim sistemima za naplatu.

Arhitektura *nangu.tv* rešenja se sastoji od komponenata koje obezbeđuju preporuku, isporuku sadžraja i metapodataka, upravljanje korisnicima.

Operateri koji su izabrali Nangu.tv kao isporučioca OTT sadržaja su, između ostalih, *T-Mobile*, *Swisscom Broadcast*, *O2 TV* i *Orange*.

3.1.2 Ericcsson MediaRoom Reach

Ericsson MediaRoom Reach [3] je deo *Ericsson MediaRoom* platforme koja je platforma za televiziju sledeće generacije kompanije Ericsson. Od OTT usluga *MediaRoom Reach* nudi personalizaciju aplikacije digitalne televizije, video na zahtev, isporuku liste kanala, elektronski vodič kroz program, pretragu sadržaja, označavanje sadržaja, ocenjivanje sadržaja i preporučivanje sadržaja. Na Slici 3.1 prikazana je pojednostavljena arhitektura koja se sastoji samo od komponenata koji upravljaju sadržajem a to su komponente za pripremu, analizu, upravljanje i dostavu sadržaja.



Slika 3.1 *MediaRoom Reach* arhitektura

3.1.3 Nagra MediaLive

MediaLive [4] je IPTV/OTT platforma Švajcarske kompanije *Nagra*.

Skup Usluga koje nudi *MediaLive* je veoma bogat i to su personalizacija TV aplikacija, liste kanala, elektronski vodič kroz program, reklamiranje i prodaja proizvoda, video na zahtev, pretraga dostupnih sadržaja, preporuka dostupnih sadržaja, podrška za ocenjivanje sadržaja i označavanje sadržaja kao omiljenih.

Medialive pruža SDK (engl. *Software development kit*) kojim je omogućeno provajderima usluga da lako izmene postojeće usluge ili dodaju nove koje će biti ugrađene u poslužilac.

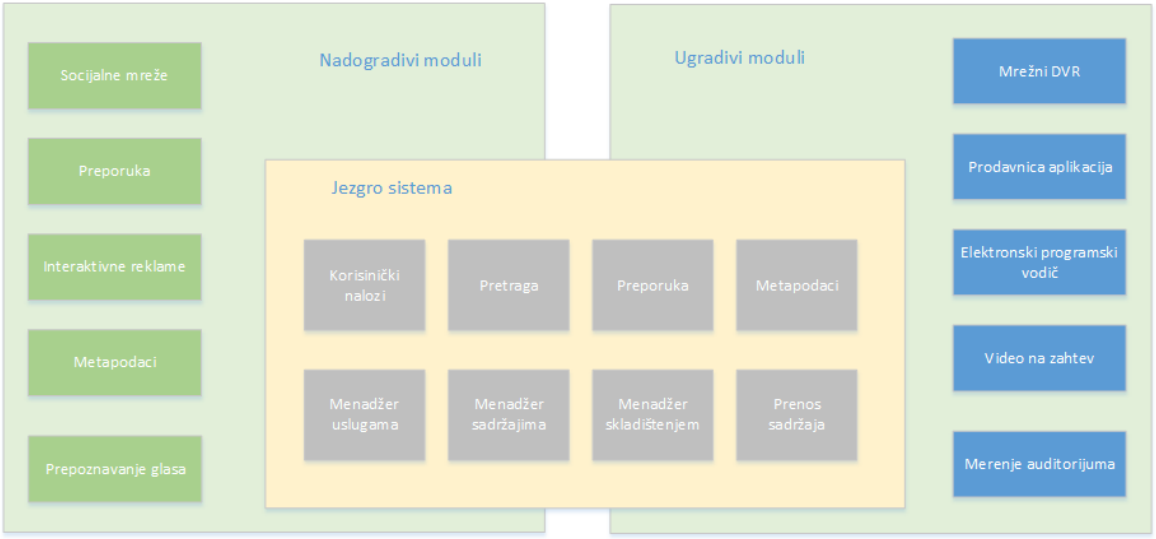
Blokovi aritekture Nagra MediaLive se sastoje od blokova koji čine sisteme za preporuku, blokova koju obezbeđuju isporuku sadržaja, blokovi koji upravljaju pretplatom i naplatom sadržaja.

3.1.4 AltiPlex

Alticast-ov *Alitplex* [5] je poslužilac za podšku TV aplikacijama koji nudi usluge: video na zahtev, mrežni DVR, preporučivanje sadržaja, sisteme za naplatu, personalizaciju, elektronski vodič kroz program i pretragu sadržaja, isporuku metapodataka.

Altiplex poslužilac takođe nudi API za ugrađiivanje postojećih rešenja kao što su sistemi za preporuku i prodavnice aplikacija.

Altiplex-om je moguće upravljati iz administratorske konzolne aplikacije koja nudi pristup izmeni video na zahtev kataloga, grupisanju kanala, izmeni reklamnih sadržaja. Konzola takođe omogućava pristup logovima poslužioca, upravljanje korisničkim nalozima i upravljanje funkocinalnostima poslužioca. Na slici 3.2 je prikazana arhitektura *Alticast Altiplex* rešenja.



Slika 3.2 Arhitektura *Alticast Altiplex*

3.2 Zaključak Analize zahteva

Sumiranjem analize datih rešenja dolazimo do skupa zahteva koje treba da ima poslužilac za podršku aplikacijama digitalne televizije.

A zahtevi su:

- Bogata kolekcija sadržaja na zahtev;
- Bogat izbor pratećih metapodataka koji uključuju probrane OTT sadržaje;
- povezivanja metapodataka i sadržaja koji su dostupni na internetu;
- Podrška za socijalne interakcije;
- Podrška za različite ciljne uređaje / ekrane;
- Elektronski vodič kroz program i video na zahtev;
- preporučivanje sadržaja u skladu sa profilom korisnika ;
- mrežni PVR/DVR;

- podrška za integraciju sa sistemima za naplatu.

Analizom arhitektura došli smo i do skupa komponenti arhitekture koja sva navedenih rešenja poseduju. Osonovni skupovi komponenti arhitekture koje poseduju sva navedenih rešenja jesu:

- komponente za upravljanje i isporuku sadržaja i metapodataka;
- komponente za preporuku sadržaja;
- komponente koje manipulišu korisničkim nalozima.

Razmatranjem liste komponenti možemo uvideti nedostatak komponenti koje sakupljaju sadržaje i metapodatke.

3.3 Tehnologije za realizaciju zahteva

Za ispunjenje zahteva iz analize zahteva je potrebno da poslužilac, kao osnovne gradivne komponente, poseduje sistem za izvršavanje poslovne logike, sistem za skladištenje podataka, sistem za preporuku i sistem koji će omogućiti mrežni pristup.

Navedene zahteve je moguće ispuniti realizacije aplikacije nad aplikativnim poslužiocem gde bi aplikativni poslužilac obezbedio izvršavanje poslovne logike sistema. Kao aplikativni poslužilac dobar izbor predstavlja JBoss aplikativni poslužilac koji je realizacija JEE specifikacije i ima otvoren kod i veliku zajednicu koja ga održava. Kao još jedna od prednosti JEE specifijacije jeste to što nudi širok spektar podspecifikacija koje prave apstrakciju prema kompleksnim operacijama koje sistemi poslužioci koriste. Neke od operacija su rukovanje transakcijama, rukovanje komunikacijom sa skladištem podataka, rukovanje pristupom sa mreže i dr.

Sistem za skladištenje podataka bi služio za čuvanje podataka vezanih za korisnike i podataka koji predstavljaju sadržaje. Kao referentni sistem za skladištenje podataka koristi se *PostgreSQL* baza podataka otvorenog koda sa velikom zajednicom koja je održava. Za pristup skladištu podaka JEE nudi JPA specifikaciju koja je podržana od strane JBoss aplikativnog poslužioca.

Sistem za preporuku ima ulogu da, u zavisnosti od profila korisnika, dostavlja sadržaje koji su najpogodniji i najviše bi zainteresovali korisnika. Kao sistem za preporuku uzeta je već postojeća realizacija koja je zasnovana na *Apache Mahout* projektu. *Apache Mahout* je platforma za razvoj proširivih aplikacija za mašinsko učenje.

Za mrežni pristup predložen je API koji je u skladu sa REST arhitekturom. Za realizaciju API-ja u skladu sa REST arhitekturom JEE nudi JAX-RS specifikaciju koja je podržana od strane Jboss aplikativnog poslužioca.

3.3.1 Java EE platforma

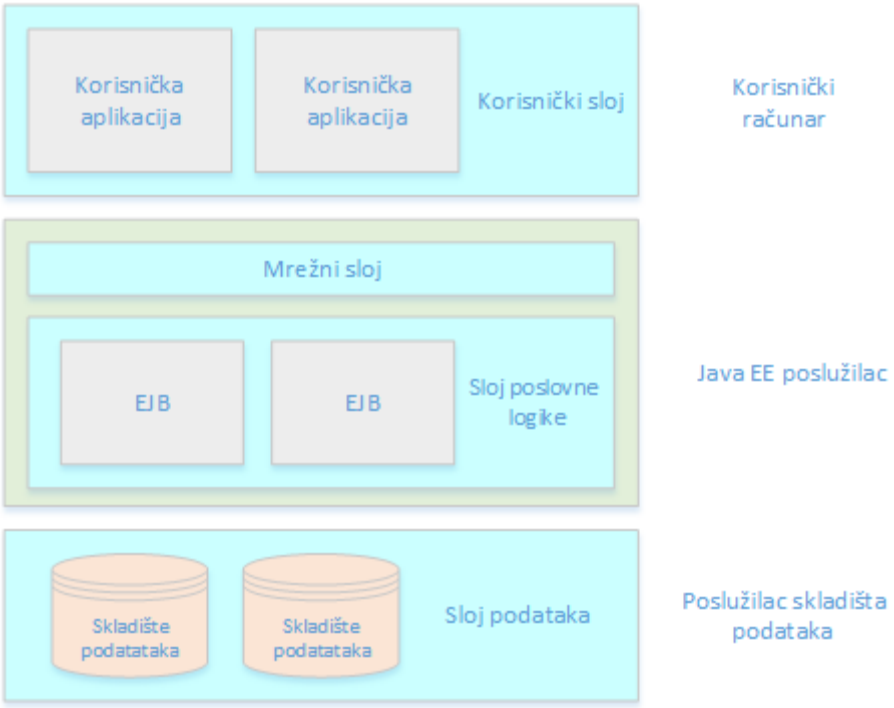
Java EE (engl. *Java Enterprise Edition*) [11] platforma je osmišljena kao olakšanje programerima pri projektovanju i realizaciji poslovnih aplikacija koje su robustne, proširive, pouzdane i sigurne. Date karakteristike čine da poslovna aplikacija bude kompleksna i razvoj takvih aplikacija zahteva velika ulaganja. Java EE platforma smanjuje nivo kompleksnosti tih aplikacija pružanjem već gotovih modela, API-ja i okruženja za izvršavanje i tako programerima dozvoljava da se koncentrišu na razvoj funkcionalnosti. Osim što se smanjuje kompleksnost aplikacija, smanjuje se i vreme potrebno za razvoj i poboljšavaju se performanse.

Poslovna aplikacija je aplikacija koja realizuje model nekog realnog poslovnog rešenja.

Java EE platforma je definisana JEE specifikacijom čiji se razvoj i održavanje obavlja kroz proces Java zajednice (engl. *Java Community Process*) na koji članovi zajednice mogu da postave zahtev (engl. *Java Specification Request*) za nekom funkcionalnosti u specifikaciji ili da prijave neki nedostatak ili nepravilnost. Programske podrške koje realizuju JEE specifikaciju jesu aplikativni poslužioci. Poslovne aplikacije koje koriste aplikativne programske sprege JEE platforme postaju prenosive tj. mogu biti izvršavanje nad bilo kojim aplikativnim poslužiocem koji je u skladu sa JEE specifikacijom .

Java EE koristi pojednostavljeni način programiranja, programiranje naznakama (engl. *annottations*). Naznake se unose u kod i naznačavaju Java EE poslužiocu kako da izvršava datu komponentu.

Java EE se fokusira na podršku razvoju srenjeg sloja kod višeslojnih aplikacija. Termin „višeslojna aplikacija“ podrazumeva aplikaciju koja je podeljena u više izolovanih funkcionalnih oblasti koje se nazivaju slojevi. Tipičnu višeslojnu aplikaciju možemo podeliti na korisnički sloj (engl. *client tier*), srednji sloj (engl. *middle tier*) i sloj podataka (engl. *data tier*). Korisnički sloj pravi zahteve srednjem sloju. Srednji sloj, koji je podeljen u mrežni sloj (engl. *web tier*) i sloj poslovne logike (engl. *business tier*), prihvata zahtev, obrađuje ga i smešta ga u skladište podataka u sloju podataka. Prikaz tipične višeslojne aplikacije je prikazan na Slici 3.3.



Slika 3.3 Prikaz tipične višeslojne aplikacije

Uloga mrežnog sloja jeste da pruži podršku za razvoj sprega prema korisniku u vidu mrežnih aplikacija. Osnovne komponente sloja poslovne logike jesu EJB (engl. *Enterprise Java Bean*) komponente koje se realizuju po EJB specifikaciji. EJB specifikacija je podskup JEE specifikacije kojom se definiše način realizacije funkcionalnosti u sloju poslovne logike u poslovnim aplikacijama. EJB-ovi predstavljaju komponente poslužioca koje enkapsuliraju logičke celine poslovne logike i izlažu tu poslovnu logiku mrežnom sloju i dalje korisnicima.

Ranije pomenute JPA i JAX-RS su podspecifikacije koje pripadaju sloju poslovne logike.

3.3.1.1 JAX-RS specifikacija

JAX-RS [11] je aplikativna programska sprega za programski jezik Java koji je osmišljen da pojednostavi razvoj aplikacija koje koriste REST arhitekturu.

JAX-RS aplikativna programska sprega koristi Java naznake da definiše resurse i akcije koje mogu biti izvršene na tim resursima. Naznakama, koje nudi JAX—RS specifikacija, je moguće zadati putanju do resursa, tip REST metode, tip opisnog jezika koji prihvata metoda, tip opisnog jezika odgovora, tip parametara, mapiranje parametara itd.

3.3.1.2 JPA specifikacija

JPA (engl. Java Persistance API) [11] je specifikacija koja ima za cilj da definiše aplikativnu programsku spregu za skladištenje podataka koja je u najvećem broju slučajeva baza podataka. Jedna od glavnih funkcionalnosti JPA specifikacije jeste objektno-relaciono preslikavanje koje je lako za korišćenje. Pojam „Objektno-relaciono preslikavanje“ označava

tehniku za mapiranje podataka između objektno orijentisanih i objektno neorijetisanih programskih jezika.

Korišćenje realizacija JPA specifikacije se svodi na dopunjavanje naznakama prostih Java objekata, koje želimo da skladištimo, na osnovu kojih aplikativni poslužilac generiše model podataka koji se koristi pri generisanju šeme baze podataka ili strukture nekog drugog skladišta podataka.

Za pravljenje upita skladištima podataka JPA specifikacija nudi korišćenje JPQL (engl. *Java persistance query language*). JPQL je platformski nezavisan upitni jezik koji sintaksno podseća na SQL upitni jezik.

Upotrebom rešenja koje je razvijeno po JPA specifikaciji postižemo platformsku nezavisnost od skladišta podataka.

3.3.2 JBoss aplikativni poslužilac

JavaBeans Open Source Software Application Server ili JBoss AS [17] je aplikativni poslužilac otvorenog koda koji realizuje JEE specifikaciju. JBoss AS (engl. *Application server*) je ključna komponenta srednjeg sloja JBoss platforme za razvoj poslovnih aplikacija (engl. *JBoss Enterprise Middleware System*) koja obezbeđuje ugrađen i ispitan skup programskih podrški koje realizuju određene delove JEE specifikacije. Taj skup programskih podrški čine *Hibernate*, *Apache Tomcat, Jboss Cache, Jboss Eclipse IDE* i mnoge druge.

Treba istaći da se JBoss AS odlikuje velikim brojem ugrađenih funkcionalnosti koje uključuju podršku za proširivost aplikativnog poslužioca, podrška za skladištenje i privremeno skladištenje podataka, podršku za raspoređivanje opterećenja, podršku za rukovanje transakcijama i drugih.

3.3.3 PostgreSQL

PostgreSQL [18] je objektno-relaciona baza podataka otvorenog koda koja podržava veći deo SQL standarda. Po osobinama uporediva je za mnogim komercijalnim bazama podataka. Poseduje podršku za transakcije i potpuno je ACID kompatibilna. ACID (engl. Atomicity, Consistency, Isolation, Durability) je akronim koji predstavlja skup svojstava koja pouzdano garantuju da će transakcija biti izvršena. Podaci koje skladišti mogu biti neki od osnovih tipova, ali podržava i definisanje složenih tipova od strane korisnika.

Od verzije 9.0 uvedena je podrška za binarno umnožavanje podataka na više poslužilaca koja je zasnovana na asinhronom distribuiranju razlika. Ovaj tip umnožavanja podataka se je realizovan metodom vodeći-prateći (engl *master-slave*). Umnožavanje podataka tipa više vodećih je moguće ostvariti upotrebom dodatne programske podrške, ali ona nije deo PostgreSQL paketa programske podrške.

4. Koncept rešenja

U ovom poglavlju je naveden koncept rešenja poslužioca za podršku aplikacijama digitalne televizije. Na početku poglavlja je napravljen pregled komponeneata arhitekture i infrastrukture, zatim je u nastavku detaljno opisana svaka komponenta sa API-jem i objašnjenjima.

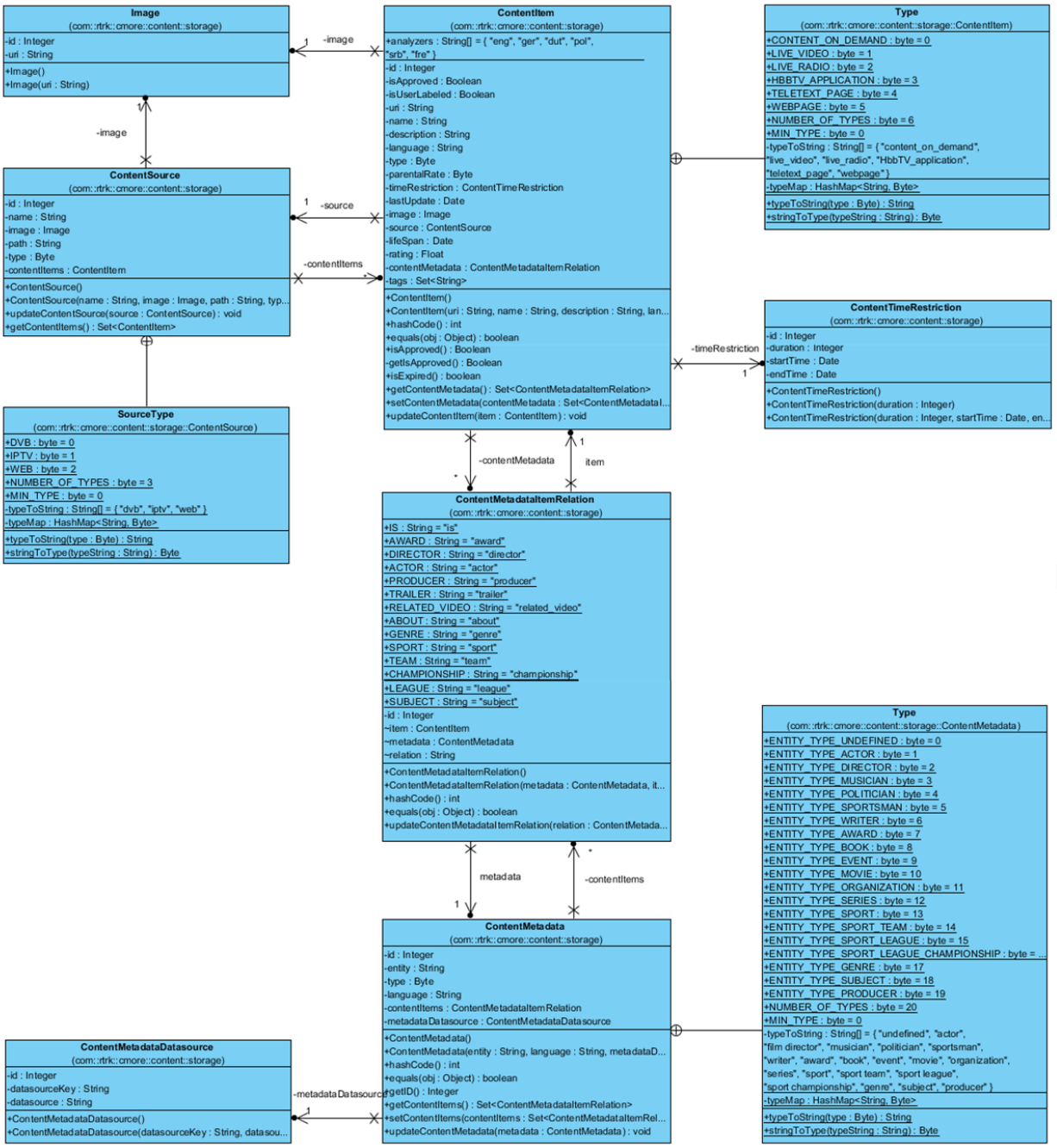
Programska podrška koje je rezultat ovog rada je ekosistem poslužioca u vidu radnog okvira za razvoj. Razvijeni radni okvir ima za cilj da olakša razvoj poslužioca za podršku aplikacijama digitalne televizije.

Rešenje je razvijano sa ciljem da se izvršava u cloud okruženju i da nudi programsku podršku kao uslugu (*SaaS*) ili platformu kao uslugu (*PaaS*) gde bi okvir za razvoj bio platforma za razvoj ugradivih komponenti sistema.

4.1 Pregled elementarnih klasa

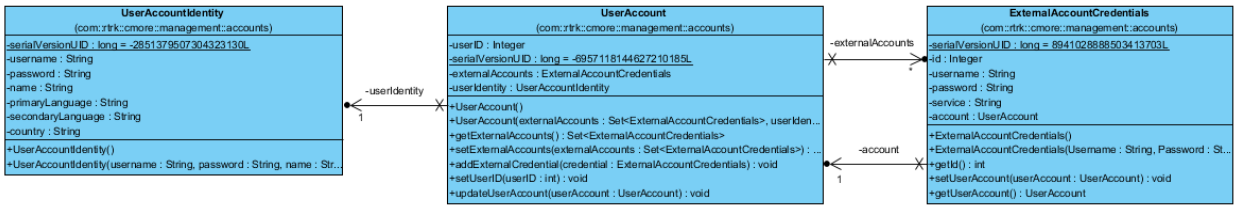
Predstava sadržaja u ovom rešenju ne podrazumeva konkretan sadržaj (audio,video) već njegove tehničke karakteristike, metapodatke vezane za dati sadržaj i ukazivač na konkretan sadržaj. Audio sadržaj, video sadržaj i ostale elementarne komponente (teletekst, prevodi i dr) obezbeđuju se korišćenjem dodatnih, spoljnih poslužioca koji ne pripadaju arhitekturi predloženoj i realizovanoj u ovom radu. Osnovna predstava sadržaja je predstavljena klasom *ContentItem* koja je ključna klasa u radnom okviru. Pored klase koja opisuje sadržaj tu su klase *ContentSource* kojom je opisan izvor sadržaja, klasa *Image* koja opisuje sliku, klasa *ContentTimeRestriciton* koja opisuje vremenska ograničenja sadržaja, ukoliko ona postoje. Pored klase *ContentItem* osnovu elementarnih klasa čine i klasa *ContentMetadata* kojom su opisani metapodaci i klasa *ContentMetadataItemRelation* koja predstavlja tačku spajanja sadržaja i metapodatka tako što opisuje njihovu vezu. I na kraju je tu klasa *ContentMetadataDatasource* koja povezuje pojam metapodatka sa pojmom u bazi znanja.

Na Slici 4.1 je UML dijagramom prikazana struktuara elementarnih klasa poslužioca koje opisuju sadržaje.



Slika 4.1 Prikaz UML dijagrama elementarnih klasa sadržaja

Podaci o korisničkim nalozima su opisani klasom *UserAccount* koja sadrži podklasu *UserAccountIdentity* koja opisuje korisničke podatke, dok klasa *ExternalAccountCredentials* opisuje korisničke naloge spoljnih servisa koje je moguće povezati sa sistemom. Slika 4.2 prikazuje UML dijagrame navedenih klasa.



Slika 4.2 UML dijagram elementarnih klasa korisnika

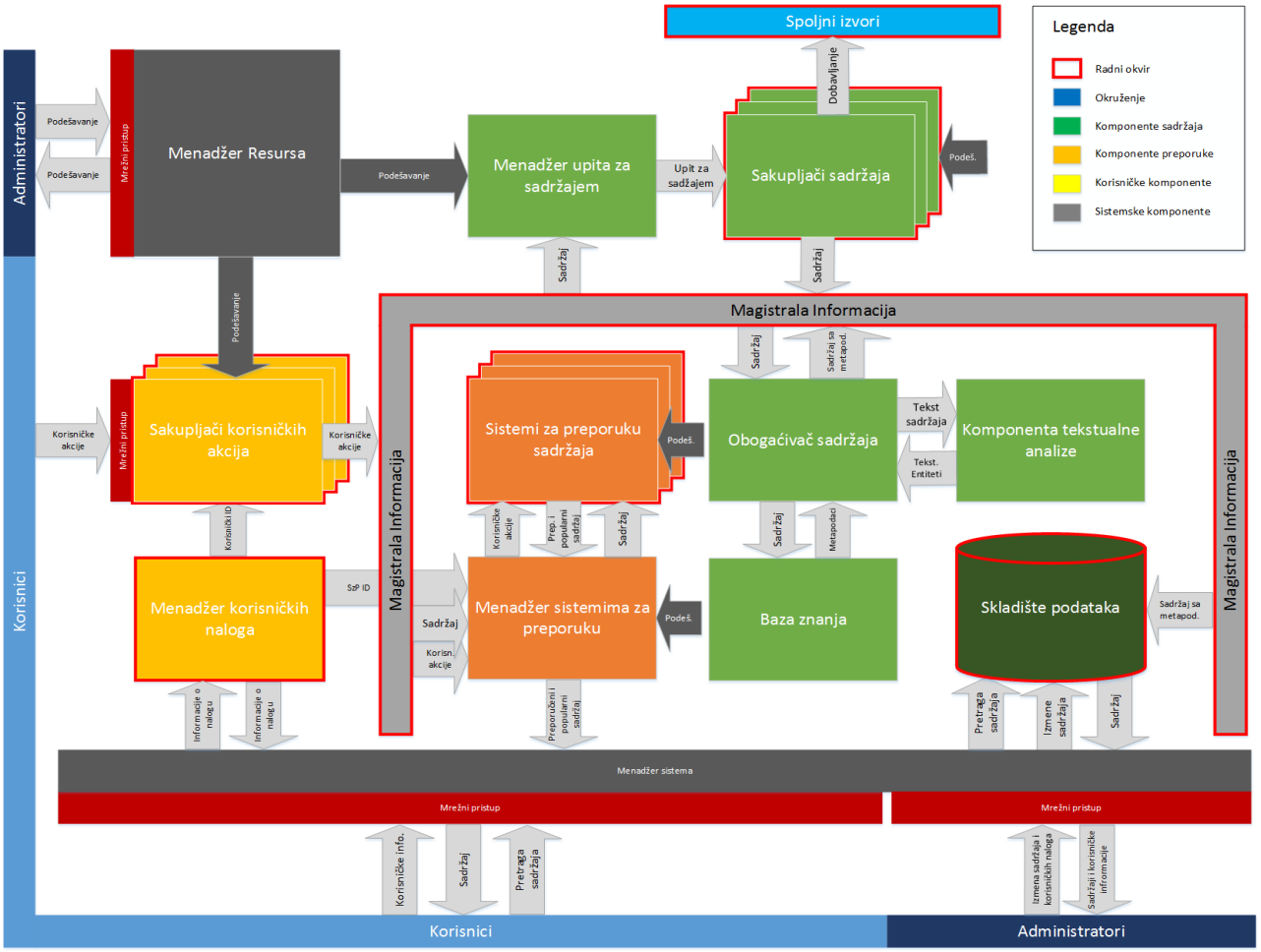
4.2 Predlog arhitekture i infrastrukture

Razmatranjem zahteva do kojih se došlo u Poglavlju III dolazimo do skupa komponenti poslužioca potrebnih za realizaciju zahteva. Predlog potrebnih komponenti za realizaciju zahteva:

- Da bi se odgovorilo na zahtev za bogatom kolekcijom sadržaja na zahtev, elektronskim vodičem i video sadržajem na zahtev potrebne su komponente koje će omogućiti unos, skladištiti date sadržaje i na zahtev ih isporučivati.
- Za zahtev koji se tiče bogatog izbora pratećih metapodataka koji uključuju probrane OTT sadržaje će biti potrebne komponente za analizu unetih sadržaja, potragu za metapodacima.
- Da bi se obezbedilo povezivanje metapodataka i sadržaja koji su dostupni na internetu, potrebne su komponente koje vrše potragu za sadržajima koji su povezani sa pronađenim metapodacima.
- Za podršku socijalnim interakcijama i profilisanju korisnika potrebna je komponenta koja će rukovati korisničkim nalozima i manipulisati njima.
- Preporuku sadržaja omogućava komponenta za preporučivanje sadržaja.
- Podrška za različite ciljne uređaje / ekrane je postignuta samim tim što je rešenje bazirano na OTT konceptu.

Podrška za konkrentan prenos audio i video sadržaja , povezivanje sa sistemima za naplatu i pretplatu neće biti razmatrana u ovoj verziju poslužioca, ali svakako da će biti ostavljen prostor za razvoj ili ugrađivanje programskih podrški ovog tipa. Realizaciju ovoga možemo uzeti kao prostor za unapređenje.

Analizom potrebnih komponenti dolazimo do predloga arhitekture i infrastrukture koja je prikazana na Slici 4.3.



Slika 4.3 Arhitektura i infrastruktura poslužioca

4.3 Opis Radnog okvira poslužioca

Radni okvir za razvoj poslužioca možemo podeliti na dve grupe komponenata. Grupu koja se sastoji od komponenata koje čine infrastrukturu poslužioca i grupu koja je sačinjena od komponenata koje korisnik radnog okvira može razvijati i unapređivati (ugradive komponente poslužioca).

Infrastrukturu poslužioca čine:

- Menadžer resursa
- Magistrala informacija
- Obogaćivač sadržaja
- Komponenta tekstualne analize
- Baza znanja
- Menadžer sistemima za preporuku
- Menadžer upita za sadržajem
- Skladište podataka

- Menadžer korisničkih naloga
- Menadžer sistema
- Sprega za mrežni pristup

Ugradive komponente su:

- Sakupljači sadržaja
- Sakupljači korisničkih akcija
- Sistemi za preporuku sadržaja

Radni okvir je realizovan sa idejom da se omogući lako dodavanje i isključivanje i podešavanje komponenti sistema u toku izvršavanja.

U nastavku će detaljno biti opisane komponente sistema i biće prikazan njihov API.

4.3.1 Menadžer Resursa

Menadžer Resursa je ključna komponenta sistema koja manipuliše ostalim komponentama. Zadatak menadžera resursa je da vrši otkrivanje, podešavanje i upravljanje aktivnošću ostalih podesivih komponenata sistema. Podešavanja sistema je moguće promeniti u toku izvršavanja od strane samih komponenti ili adinistrativnih korisnika kroz mrežni pristup.

Menadžer resursa u podešenom intervalu vrši skeniranje konteksta imenovanja (engl. *naming context*) u cilju pronalženja novih ugradivih komponenti. Kao rezulatat skeniranja konteksta se dobija lista *JNDI* imena *EJB*-ova koji realizaciju programske sprege(engl. *interfaces*) komponenata. Nakon toga menadžer resursa radi pretragu konteksta (engl. *lookup*) za konkretnim realizacijama EJB-ova i kroz neke od dole navedenih metoda omogućava da oni budu dostupni korisnicima.

Svaka komponenta sistema se odlikuje tipom, stanjem i identifikatorom i to se postiže implementiranjem programske sprege *SystemComponent*. Tip komponente je definisan programskom spregom koju komponente implementira, stanje komponente se dodeljuje komponenti u toku izvršavanja i može biti „aktvna“ ili „neaktivna“, a identifikator definiše programer koji razvija datu komponetu.

Aplikativnu programsku spregu mendžera resursa čine sledeće metode:

```
void enableComponent(ComponentType componentType, componentID);
```

enableComponent je metoda koja se koristi za aktiviranje zadate komponente. *componentType* predstavlja tip komponente koja se aktivira, a *componentID* identifikator komponente koja se aktivira.

```
void disableComponent(ComponentType componentType, componentID);
```

DisbleComponent je metoda koja se koristi za aktiviranje zadate komponente. *componentType* predstavlja tip komponente koja se deaktivira, a *componentID* identifikator komponente koja se deaktivira.

```
List<RecommendationEngine> listRecommendationEngines(
      ComponentStatus componentStatus);
```

Metoda `listRecommendationEngines` se koristi za dobavljanje liste pronađenih sistema za preporuku koje zadovoljavaju uslov da se komponenta nalazi u stanju koje je zadato parametrom *componentStatus.*

```
List<OnDemandContentAggregator> listContentAggregators(
      ComponentStatus componentStatus);
```

listContentAggregators je metoda za izlistavanje sakupljača sadržaja koji se nalaze u *componentStatus*-om zadatom stanju.

```
List<UserActionAggregator> listUserActionAggregators(
      ComponentStatus componentStatus);
```

Metoda koja vraća listu sakupljača korisničkih akcija u stanju zadatom parametrom *componentStatus* je `listUserActionAggregators.`

```
ComponentStatus checkComponentState(ComponentType componentType,
      String componentID);
```

Metoda koja se koristi za proveru stanja u kom se nalazi komponenta koja se identifikuje sa *componentType* `i` *componentID*.

```
void registerComponentStatusListener(ComponentStatusListener
      componentStatusListener);
```

Metoda *registerComponentStatusListener* je metoda kojom se prijavljuje osluškivač na stanje komponenti sistema koji se prosledi kao *componentStatusListener*. U slučaju da se stanje određene komponente promeni, sve ostale komponente koje imaju prijavljen osluškivač će biti obaveštene metodom povratne sprege (engl. *callback*).

```
void unregisterComponentStatusListener(   ComponentStatusListener
      componentStatusListener);
```

Metoda *unregisterComponentStatusListener* služi za odjavljivanje *componentStatusListener* osluškivača na promenu stanja komponeni.

```
void signalComponentStatusChange(ComponentType componentType,
      ComponentStatus componentStatus, String componentID);
```

Signaliranje menadžeru resursa da je neka komponenta promenila stanje se obavlja upotrebom metode *signalComponentStatusChange* koja kao parametre prihvata identifikatore komponente i novo stanje zadate komponente.

```
public List<EventListener> getEventListeners();
```

Metodom *getEventListeners* se dobavlja lista komponenata koje su u mogućnosti da komuniciraju preko magistrale informacija.

4.3.2 Magistrala Informacija

Magistrala informacija je magistrala za komunikaciju zasnovanu na događajima. Osnovna ideja magistrale infromacija je da se omogući asinhrona komunikacija na nivou događaja unutar sistema između ugradivih komponenata i ostatka sistema. Da bi ugradive komponente komunicirale sa ostatkom sistema dovoljna je magistrala informacija, dok ostatak sistema za njih ostaje nevidljiv.

Da bi komponenta bila povezana na magistralu informacija njena programska sprega mora da nasledi programsku spregu *EventListener*. *EventListener* poseduje dve metode koje se moraju implementirati u svim komponentama koje ga implementiraju.

```
boolean containsType(int type);
void callback(Event event);
```

Metoda *containsType* služi za proveru da li je komponenta prijavljena na parametorm prosleđivani tip događaja *type.* Dok je metoda *callback* metoda povratne sprege koja biva pozvana ukoliko je zadata komponenta prijavljena na dati događaj.

Komponente sistema koje su povežane na magistralu informacija su:

- Sakupljači sadržaja na zahtev;
- Sakupljači korisničkih akcija;
- Menadžer sistemima za preporuku;
- Skladište podataka;
- Obogaćivač sadržaja;
- Menadžer upita za sadržajem.

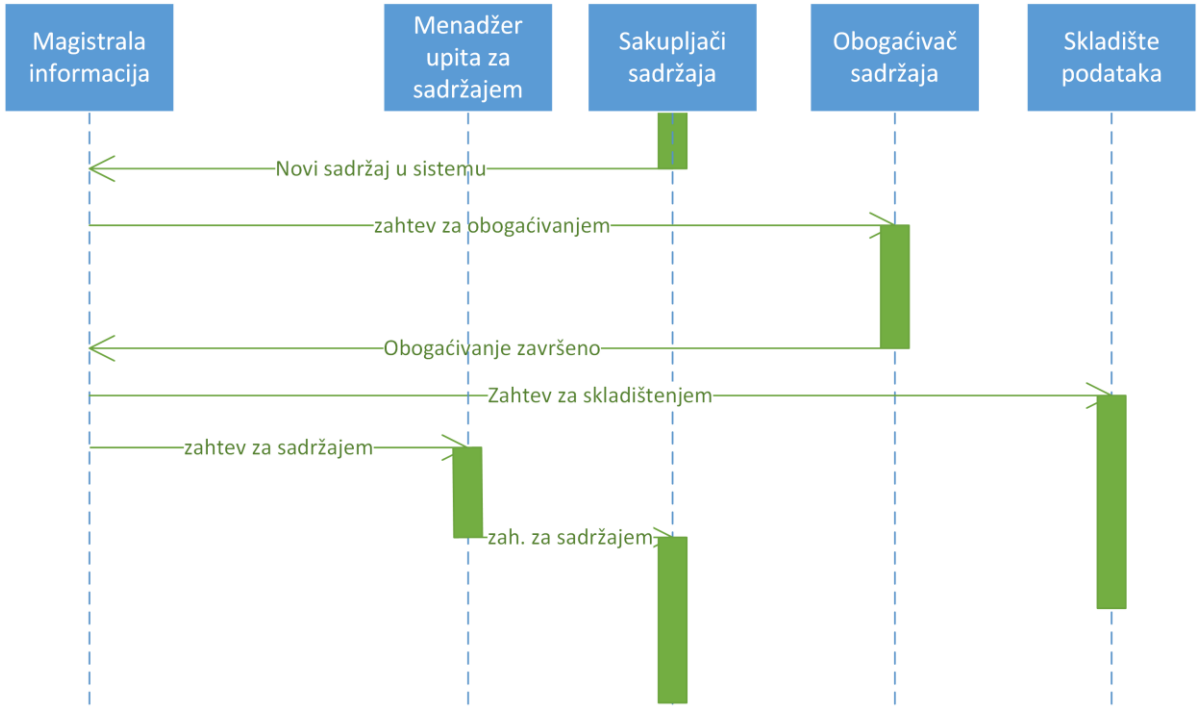
Magistrala informacija ima samo jednu metodu i to:

```
public void submitEvent(Event event);
```

Metoda *submitEvent* se koristi za postavljanje događaja *event* na magistralu događaja.

Klasa *Event* se identifikuje jedinstvenim identifikatorom na nivou sistema i tipom događaja. Klasu *Event* nema veliku korist, sama po sebi, njome je moguće samo obavestiti da se desio neki događaj u sistemu, međutim nasleđivanjem i proširivanjem klase *Event* je moguće i proširiti njenu funkcionalnost i time omogućiti da se uz obaveštenje o događaju prenose i neke dodatne informacije vezane za dati događaj.

Na Slici 4.4 je, u vidu UML dijagrama sekvence, prikazan jedan od sučajeva korišćenja magistrale infromacija. U pitanju je ciklus obogaćivanja sadržaja metapodacima i sakupljanja dodatnih sadržaja za koji su povezani sa mezapodacima.



Slika 4.4 UML dijagram sekvence ciklusa obogaćivanja

4.3.3 Obogaćivač sadržaja

Obogaćivač sadržaja je komponenta koja na osnovu imena i opisa sadržaja pronalazi tehničke podatke i metapodatke vezane za dati sadržaj, u zavisnosti od putanje kojom se sadržaj kreće kroz sistem

Obogaćivanje sadržaja se sastoji pronalaženja nedostajućih tehničkih podataka i pronalaženja metapodataka vezanih za sadržaj. Započinje se slanjem zahteva komponenti tekstualne analize od koje se traži analiza imena i opisa sadržaja u cilju preuzimanja povezanih tekstualnih entiteta iz tekstualnih podataka sadržaja. Nakon toga se bazi znanja šalje zahtev za identifikacijom tekstualnih entiteta sa pojmom iz baze znanja i iz baze znanja se dobavljaju informacije o datim pojmovima. Dobavljene informacije iz baze znanja se dalje koriste ta dopunu nedostajećih podataka i pravljenje metapodataka čime je proces obogaćivanja sadržaja završen.

Obogaćivač sadržaja ima jednostavanu programsku spregu sačinjenu od jedne metode:

```
Future<ContentItem> annotateContent(ContentItem item);
```

annotateContent je metoda kojom se od obogaćivača sadržaja zahteva da pokrene proces dopunjavanja nedostajućih tehničkih informacija i prikupljanja metapodataka vezanih za parametrom prosleđeni sadržaj *item.*

4.3.4 Komponenta tekstualne analize

Komponenta teksutalne analize analizira tekstualne podatke koji joj se proslede i pokušava iz njih da izvuče semantička značenja. Kao rezultat analize dobijaju se tekstualni entiteti koji se koriste za kasniju pretragu baze znanja radi identifikacije tih entiteta i dobavljanja metapodataka.

Rad komponente tekstualne analize se zasniva na *Apache UIMA* (engl. *Unstructured Information Management Architecture*) projektu. UIMA je skup programskih sprega čije realizacije omogućavaju analizu nestruktuiranih podataka. Realizacija UIMA-e koja je korištena jeste OpenNLP (engl. *Natural language processing*) koja podržava pronalaženje rečenica, parčanje rečenica na reči i sintagme, otkrivanje osoba, lokacija i organizacija.

Programska sprega komponente tekstualne analize se sastoji od jedne metode:

```
List<String> extractEntities(String text);
```

Metoda *extractEntities* prihvata teksualne podatke kroz parametar *text* iz kojih tekstualnom analizom preuzima entitete i vraća ih kao rezultat. Entiteti su tekstualne celine koje mogu biti reči ili sintagme.

4.3.5 Baza znanja

Baza znanja je sistem koji sadrži semantički povezane podatke o pojmovima sa kojima se sistem može susresti u procesu obogaćivanja. Baza znanja predstavlja programsku spregu iza koje može da se krije neka baza znanja kojoj se pristupa preko interneta ili baza znanja koja se nalazi lokalno. Baza znanja je, u ovoj verziji poslužioca, spoljni resurs i sistem nema mogućnost da je dopunjuje ili menja. Takođe, treba napomenuti da postoji podrška za istovremeno postojanje više implementacija baze znanja koje mogu uporedo da funkcionišu, ovo omogućava da postoji više baza znanja koje su tematski podeljene.

Metode programske sprege Baze znanja:

```
Future<List<Result>> searchEntity(KnowledgeBaseSearchQuery searchQuery);
```

Metoda za pretragu baze znanja *searchEntity* presleđivanjem upita *searchQuery* kojim se zadaje pojam za pretragu, jezik zadatog pojma i informacija koja baza znanja treba da se pretražuje. Kako povratna vrednost metode se dobija lista rezultata u kojoj svaki rezultat sadrži rezultni pojam i jedinstveni identifikator pomoću kojeg je moguće dobiti više informacija o datom rezultantnom pojmu.

```
List<Entity> requestEntity(KnowledgeBaseEntityRequestQuery
      entityRequestQuery);
```

Metodom *requestEntity* se od baze znanja zahteva zahtevaju kompletni podaci o rezultatu koji se dobije kao rezultat metode *searchEntity*. Zahtev *entityRequestQuery* koji se šalje sadrži

jedinstveni identifikator rezultata u bazi znanja, identifikator baze znanja i jezik na kome treba da budu informamcije.

Ukoliko se bazi znanja pristupa sa inerneta, postoji podrška da se rezultati gore navedenih metoda privremeno skladište.

4.3.6 Menadžer sistemima za preporuku

Uloga menadžera sistemima za preporuku je da podrži više sistema za preporuku koji rade istovremeno. On prosleđuje korisničke akcije, nove sadržaje i korisničke zahteve sistemima za preporuku. Pošto sistemi za preporuku obrade zahteve i vrate preporučene sadržaje zadatak menadžera je da sabere sadržaje od svih sistema za preporuku i prosledi ih dalje prema korisniku.

Metode programske sprege menadžera sistemima za preporuku izgleda ovako:

```
List<ContentItem> resolveRequestAndCollectData(UserAccount userAccount,
       ContentCategory contentCategory,ContentFilter contentFilter);
```

Metoda *resolveRequestAndCollectData* je osnovna metoda menadžera sistema za preprouku. Kao parametre prihvata korisnički nalog *userAccount* za koji je potrebno preporučiti sadržaje, kategoriju *contentCategory* preporuke i *contentFilter* za filtriranje sadržaja. Rezultat metode jesu sumirani sadržaji svih sistema za preporuku.

```
void consumeUserAction(int userID, UserAction userAction);
```

Metodom *consumeUserAction* prosleđujemo sistemima za preporuku novu korisničku akciju *userAction* korisnika čiji je identifikator *userID*, koju oni dalje uključuju u proračun za preporuku.

```
void consumeContentItems(List<ContentItem> contentItems);
```

Metodom *consumeContentItems* obaveštavamo sisteme za preporuku o novim sadržajima u sistemu koje oni treba da uzmu u razmatranje za preporuku. Lista novih sadržaja se prosleđuje kao parametar *contentItems*.

```
void updateContentItems(List<ContentItem> contentItems);
```

Metodu *updateContentItems* koristimo da obavestimo sisteme za preporuku da su prosleđeni sadržaji *contentItems* ažurirani i da treba ažurirati i proračune za dati sadržaj.

```
void deleteContentItems(List<ContentItem> contentItems);
```

Metoda *deleteContentItems* služi za prosleđivanje informacije o brisanju sadržaja sistemima za preporuku. Sistemi za preporuku treba da izuzmu date sadržaje iz daljeg preporučivanja.

```
void addUserAccount(UserAccount userAccount);
```

Metoda *addUserAccount* služi za dodavanje novih profila korisnika u sisteme za preporuku koji se vezuju za sistemski profil *userAccount* koji je prosleđen kao parametar.

```
public void deleteUserAccount(UserAccount userAccount);
```

U slučaju brisanja sistemskog korisničkog naloga potrebno je ukloniti i korisničke profile kod svih sistema za preporuku, ova operacija se obavlja pozivom metode *deleteUserAccount* i prosleđivanjem sistemskog naloga kao parametra.

```
public List<ContentItem> labelLikedContentItems(
      List<ContentItem> contentItems, UserAccount userAccount);
```

Sistemi za preporuku sadržaja čuvaju istoriju korisničkih akcija, među kojima je i korisnička akcija „dopada“. S tim u vezi metoda *labelLikedContentItems* se koristi za označavanje sadržaja, koje su korisnici označili da im se dopadaju, u listi prosleđenih sadržaja *contantItems*.

4.3.7 Menadžer upita za sadržajem

Menadžer upita je komponenta povezana na magistralu informacija i odgovara na događaj izazvan od strane obogaćivača sadržaja koji govori da je završeno obogaćivanje sadržaja. Menadžer upita za sadržajem ima zadatak da analizira metapodatke prosleđenog sadržaja i za svaki metapodatak da napravi zahteve za sadržajem odgovarajućim sakupljačima sadržaja na zahtev koji su u stanju da odgovore na dati upit u zavisnosti od tipa metapodatka ili relacije. Metode menadžera upita za sadržajem jesu:

```
void queryAggregators(ContentItem item);
```

Metoda *queryAggregators* vrši analizu metapodataka prosleđenog *item*-a na osnovu koje pravi upite i prosleđuje ih sakupljčima sadržaja na zahtev.

4.3.8 Skladište podataka

Skladište podataka predstavlja omotač oko JPA baziranog pristupa sistemu za skladištenje podataka. Skladište podataka ima ulogu da sadržaje dobijenje sa magistrale informacija skladišti u bazu podataka. Osim programske sprege za skladištenje podataka sadržaja, skladište podataka, nudi i podršku za skladištenje pojedinačnih metapodataka, relacija između sadržaja i metapodataka i izvora sadržaja. Skladište podataka ima podršku za skladištenje, dobavljanje, uklanjanje i pretragu skladištenih entiteta.

Automatizovani proces uklanjanja sadržaja je realizovan tako da se poziva od strane skladišta podataka u unapred zadatom intervalu. Proces uklanjanja se sastoji od proveravanja da li u skladištu podataka postoji sadržaj kojem je *lifespan* (polje u klasi *contentItem*) istekao i ukoliko jeste sadržaj se uklanja.

Skup metoda programske sprege skladišta podataka se sastoji od:

```
void submitContentItems(List<ContentItem> contentItems);
```

Metoda *submitContentItems* se koristi za upis sadržaja u skladište podataka. Ova metoda prihvata listu sadržaja *contentItems* i zatim za svaki pojedini sadržaj, po jedinstevnom *uri* identifikatoru, ispituje da li takav sadržaj već postoji skladišten. Ukoliko razmatrani sadržaj postoji on biva ažurirиran koristeći podatke prosleđenog, a ukoliko ne postoji prosleđeni sadržaj se upisuje.

```
Future<List<ContentItem>> retrieveContentItems(ContentFilter filter);
```

Metodom *retrieveContentItems* se dobavljaju sadržaji. Sadržaji koji čine povratnu vrednost su filtrirani na osnovu kriterijuma filtriranja iz parametra *filter*. Filter sadrži bogat skup kriterijuma koji se mogu kombinovati i po kojima se sadržaji filtriraju u cilju dobavljanja željenih sadržaja.

```
List<ContentItem> searchContentItems(String textQuery, Short maxResults);
```

Metoda *searchContentItems* omogućava pretragu skladištenih sadržaja. Pretraživanje, koje je opisano u [19], se vrši na osnovu parametra *textQuery* i pretraga je bazirana na pretrazi indeksa koji se pravi po upisivanju sadržaja upotrebom *Apache Lucene* biblioteke. *Apache Lucene* je biblioteka koja nudi analizu teksta sa aspekta pretrage. U našem slučaju pretraga se vrši prema imenu i opisu sadržaja. Parametrom *maxResults* se zadaje maksimalni željeni broj rezultata.

```
void deleteContentItems(List<ContentItem> contentItems);
```

deleteContentItems je metoda koja služi za uklanjane sadržaja iz skladišta podataka koja kao parametar *contentItems* očekuje listu sadržaja koje je potrebno obrisati.

Ostale metode skladišta podataka neće biti navedene zbog poklapanja funkcionalnosti sa gore prikazanim metodama, one uključuju isti skup operacija kao i za manipulaciju sadržajima samo što su akteri metoda metapodaci, relacije i izvori sadržaja.

4.3.9 Menadžer korisničkih naloga

Menadžer korisničkih naloga je komponenta koja skladišti informacije i manipuliše informacijama o korisničkim nalozima. Menadžer korisničkih naloga nudi podršku za povizivanje korisničkih naloga sa spoljnim servisima u cilju interakcije sa datim servisom, primer za ovaj slučaj korišćenja bi mogla biti interakcija sa socijalnim mrežama.

Skup metoda menadžera korisničkih naloga je naveden ispod.

```
int submitUserAccount(UserAccount userAccount)
```

Metodom *submitUserAccount* je moguće napraviti novi ili ažurirati postojeći korisnički nalog. Kao parametar, metoda očekuje korisnički nalog *userAccount* a nakon izvršene operacije vraća sistemski identifikator korisnika.

```
void deleteUserAccount(UserAccount userAccount)
```

deleteUserAccount je metoda koja se koristi za uklanjanje korisničkog naloga iz sistema. Korisnički nalog koji je potrebno ukloniti se prosleđuje kao parametar *userAccount.*

```
UserAccount getUserAccountByID(Integer userID)
```

Metoda *getUserAccountByID* se koristi za dobavljanje korisničkog naloga čiji sistemski identifikator korisnika *userID.*

```
UserAccount getUserAccountByIdentity(UserAccountIdentity identity)
```

Korišćenjem metode *getUserAccountByIdentity* je moguće dobaviti korisnički nalog uz pomoć osnovnih podataka o korisničkom nalogu kao što su korisničko ime i lozinka.

```
void addUserAccountExternalCredential(
      ExternalAccountCredentials externalAccount)
```

addUserAccountExternalCredential je metoda koja služi za dodavanje podataka za povezivanje sa spoljnim servisima koji se zadaju kao parametar *externalAccount*.

```
void updateUserAccountExternalCredential(
      ExternalAccountCredentials externalCredential)
```

Metodom *updateUserAccountExternalCredential* je omogućeno ažuriranje naloga spoljnih servisa. Kao parametar, ova metoda prihvata ažurirane podatke *externalCredential.*

```
void deleteUserAccountExternalCredential(
      ExternalAccountCredentials credential)
```

Metoda *deleteUserAccountExternalCredential* se koristi za uklanjanje podataka o nalozima spoljnih servisa iz sistema. c*redential* je parametar kojim je određen željeni nalog.

```
Future<List<ExternalAccountCredentials>> retrieveExternalCredentials(
      UserAccount account)
```

retrieveExternalCredentials je metoda kojom je moguće dobaviti listu podatka o nalozima spoljnih servisa za zadati korisnički naloga *account.*

4.3.10 Menadžer sistema

Menadžer sistema je komponenta koja objedinjuje usluge menadžera korisničkih naloga, menažera sistema za preporuku i sistema za skladištenje, s razlikom što uvodi ograničenje da njegove metode mogu pozivati samo korisnici sa ispravnim korisničkim imenom i lozinkom. Njegova uloga je da prihvata zahteve od sprege za mrežni pristup, proveri ispravnost unetih korisničkih podatka i, ukoliko su podaci ispravni, prosledi ih odgovarajućim komponentama.

Kako je ovo samo omotač oko programskih sprega komponenti čije programske sprege su već opisane u ovom poglavlju, s jedinom razlikom što metode menadžera sistema kao parametre prihvataju i korisničko ime i lozinku, obmotane metode programskih spega datih komponenta neće biti opisivane.

Menažer sistema poseduje samo jednu metodu, pored brojnih metoda koje obmotava, i to je metoda za proveru tačnosti unetih korisničkih podataka koja je preduslov za pozivanje bilo koje druge metode menadžera sistema. Njen prototip izgleda ovako:

```
UserAccount validateUserAccount(String username, String password);
```

Parametri metode su korisničko ime i lozinka. Ukoliko se provera ispravnosti uspešno izvrši povratna vrednost metode je korisnički nalog.

4.3.11 Sakupljači sadržaja

Sakupljači sadržaja [20] su ugradive komponente koje sakupljaju sadržaje iz raznovrsnih izvora na internetu i to rade na način koji je transparentan ostatku sistema. Nakon sakupljanja sadržaji se postavljaju na magistralu informacija odakle idu na dalju obradu i skladištenje. Sakupljači sadržaja se dele na nezavisne i sakupljače na zahtev.

Nezavisni sakupljači sadržaj sakupljaju nezavisno od ostatka sistema, tj ne očekuju nikakvu pobudu od strane sistema. Oni sadržaj sakupljaju u određenom vremenskom intervalu ili čekaju na zahtev koji dolazi preko mreže od strane spoljnih faktora koji mogu biti korisnici ili drugi spoljni sakupljači sadržaja. Programska sprega nezavisnih skupljača sadržaja nema ni jednu svoju metodu tako da nezavisni skupljači sadržaja poseduju samo sistemske metode koje su dobili nasleđivanjem programske sprege *SystemComponent*.

Sakupljači na zahtev sadržaje sakupljaju kao odgovor na zahtev menadžera upita za sadržajem. Svaki sakupljač sadržaja na zahtev ima skup tipova i tipova relacija za koje je u stanju da sakuplja sadržaje.

Programska sprega sakupljača sadržaja na zahtev:

```
boolean isRelationSupported(String relation);
```

Menadžer upita za sadržajem pozivom metode *isRelationSupported* utvrđuje da li dati sakupljač sadržaja podržava sakupljanje sadržaja za relaciju prosleđenu parametrom *relation*.

```
void getContent(ContentQuery query);
```

Metoda *getContent* prima kao parametar zahtev za sadržajem koji nosi informacije bitne za pretragu a to su ime sadržaja, jezik na kome je sadržaj, tip sadržaja, informacije o izvoru sa koga dolazi sadržaj, naznake i relacije sa metapodacima. Na osnovu navedenih informacija se sakupljaju sadržaji koji se nakon sakupljanja postavljaju na magistrali informacija.

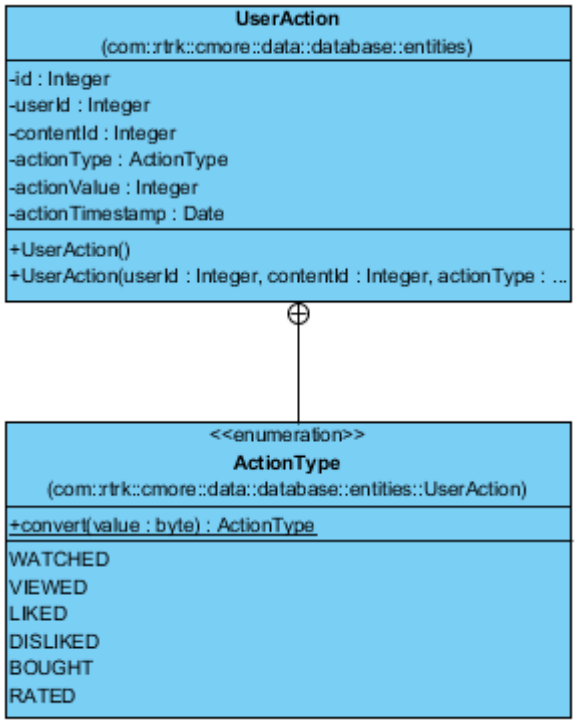
4.3.12 Sakupljači korisničkih akcija

Sakupljači korisničkih akcija imaju zadatak da prihvataju akcije korisnika od strane korisničkih aplikacija, kroz spregu za mrežni pristup, i da ih prosleđuju sistemima za preporuku upotrebom magistrale podataka. Jedna implementacija sakupljača korisničkih akcija je prikazana u [21].

```
void submitUserAction(String username, String password,
      String contentItemURI, byte type, int value, String timestamp)
```

submitUserAction je metoda koja prihvata korisničku akciju od strane korisnika sa korisničkim imenom *username* i lozinkom *password* koja je načinjena nad sadržajem čiji je jedinstveni identifikator *contentItemURI*. Parametrima *type, value* i *timestamp* se definiše tip, vrednost i vreme kad je akcija načinjena, respektivno. Navedena metoda, nakon formiranja korisničke akcije, akciju prosleđuje dalje kroz sistem preko magistrale podataka.

UML dijagram klase koja opisuje korisničku akciju je prikazan na Slici 4.5.



Slika 4.5 UML dijagram klase koja opisuje korisničku akciju

4.3.13 Sistemi za preporuku sadržaja

Sistemi za preporuku sadržaja imaju ulogu, da na zahtev korisnika isporučuju sadržaje koji su preporučeni za datog korisnika, sadržaje koji su popularni među svim korisnicima na nivou sistema ili sadržaje koji su po karakteristikama srodni zadatom sadržaju.

Sistemi za preporuku sadržaja na osnovu dostavljenih korisničkih akcija prave profil korisnika. Na osnovu profila korisnika treba da iz celokupnog skupa sadržaja izdvoje sadržaje koji će biti zanimljivi korisniku.

Popularnost sadržaja se određuje na osnovu broja korisničkih akcija i tipa načinjenih akcija nad određenim sadržajem. Što sadržaj ima više pozitivnih akcija to njegova popularnost raste. UML dijagram klase koja opisuje korisničku akciju je predstavljen na Slici 4.4.

Svaka korisnička akcija je prosleđena svakom sistemu za skladištenje podataka tako da svaki sistem za preporuku može da generiše svoj profil korisnika i svaki sadržaj je dostupan svakom sistemu za preporuku tako da svaki sadržaj može da bude preporučen od strane svakog sistema za preporuku.

Programska sprega sistema za preporuku sadržaja je identična programskoj sprezi menadžera sistemima za preporuku čija je programska sprega detaljno objašnjena u poglavlju 4.3.6.

4.3.14 Usmerivač sadržaja

Usmerivač sadržaja je komponenta koja je proširenje magistrale informacija. Njegova uloga je da u zavisnosti od zadate putanje, zadate pri ulasku sadržaja u sistem, usmeri i isprati kretanje sadržaja kroz sistem.

Postoje tri unapred definisane putanje:

1. Putanja skladištenja – sadržaj će biti prosleđen skladištu podataka;
2. Putanja obogaćivanja – sadržaj će biti obogaćen i zajedno sa metapodacima skladišten;
3. Puna putanja – sadržaj će proći putu putanju kroz sistem koja uključuje obogaćivanje, dobavljanje srodnih sadržaja za dati sadržaj i njegove metapodatke i njihovo skladištenje.

Pored unapred definisanih putanja moguće je i definisati nove putanje.

4.3.15 Sprega za mrežni pristup (REST API)

Aplikativna programska sprega poslužioca [22] je realizovana kao spregom za mrežni pristup koja poštuje REST arhitekturu kroz koju korisnici pristupaju uslugama poslužioca. Odgovor metoda mrežne sprege može biti u obliku XML-a (engl. *EXtensible Markup Language*) ili JSON-a (engl. *JavaScript Object Notation*). Sve metode mrežnog pristupa zahtevaju identifikaciju korisnika korisničkim imenom i lozinkom.

Građenje putanje (URL) do resursa se postiže po prikazanom modelu:

```
http://server_address:port/ CMoreRESTApi/rest/resource_path?{query_parameters}
```

server_address i *port* označavaju adresu i port za pristup poslužiocu, *resource_path* je identifikator resursa kome se pristupa, a *query_parameters* predstavlja parametre koji se prosleđuju datom resursu.

Spregu za mrežni pristup možemo podeliti na tri tipa resursa:

- Sadržaji;
- Izvori sadržaja;
- Resurski korisničkih naloga, koji neće biti razmatran jer je on direktna apstrakcija metoda Menadžera korisničkim nalozima.

4.3.15.1 Mrežni pristup sadržajima

Postoje dve metode za pristup sadržajima, a to su *contentItems* i *searchItems.*Metoda *contentItems* je osnovna metoda mrežnog pristupa i gradi se ovako:

```
GET http://serverIP:port/CMoreRESTApi/rest/contentItems?{query_parameters}
```

Podržani parametri upita *query_parameters* jesu:

- *uri*: Zadaje jedinstveni identifikator resursa koji ima svaki sadržaj. Odgovor na ovaj *zahtev* će sadržati samo jedan sadržaj (ako takav postoji);
- *name*: Zadaje ime sadržaja i odgovor će sadržati sve sadržaje zadatog imena;
- *language*: Jezik sadržaja je zadat ovim parametrom (pr. eng, ger, fra);
- *sourceName*: Parametar koji označava ime izvora sadržaja (pr. HBO, RTL);
- *sourceType*: Zadaje tip izvora sadržaja (pr. web, iptv, dvb)
- *sourcePath*: Postavlja jedinstveni identifikator resursa koji je pandan polju *uri* kod sadržaja
- *containsEntities*: Ovim parametrom se zahtevaju sadržaji koji su povezani sa entitetima metapodatka koji su zadati kroz ovaj parametar (pr. Ako je Leonardo DiCaprio parametar upita, biće vraćeni svi filmovi iz u kojima je on glumio, kao i dodatni sadržaji vezani za ovaj entitet). Moguće je zadati više entiteta koji su odvojeni zarezom i rezultat će biti unija pojedinačnih skupova rezultata za svaki entitet
- *tags*: Ovim parametrom se zadaju svi sadržaji koji su poseduju oznake prosleđene kao parametar. Više oznaka se odvaja zarezom;
- *types*: parametar za zadavanje tipa željenih sadržaja. Postoji šest definisanih tipova saržaja a to su: *content_on_demand*, *live_video, live_radio, HbbTV_application*, *teletext_page*, *webpage*. Moguće je zadati više tipova koji se odvajaju zarezom;

- *relation*: parametar koji zahteva da sve navedene relacije budu sadržane u rezultnim sadržajima (pr. *trailer* će vratiti samo saržaje koji imaju trejlere), moguće je zadati više relacije i odvojiti ih zarezom;
- *minRating*: zahteva sve sadržaje koji imaju ocenu višu od prosleđene
- *minParentalRate*: zahteva sadržaje koji imaju roditeljsku kontrolu veću od zadate
- *maxParentalRate*: zahteva sadržaje koji imaju roditeljsku kontrolu manju od zadate
- *minDuration*: parametar koji zadaje minimalno trajanje sadržaja
- *maxDuration*: zahtevaju se sadržaji koji ne traju duže od navedenog
- *startsAfter*: zadaje se uslov da sadržaji počinju nakon zadatog vremena
- *startsBefore*: potrebno je da sadržaji počinju pre zadatog vremena
- *endsAfter*: parametar koji zahteva da se dostavljeni sadržaji završavaju pre zadatog vremena
- *endsBefore*: parametar kojim se zadaje da se sadržaki završe pre zadatog vremena
- *maxResults*: ovaj parametar postavlja gornju granicu broja dobavljenih sadržaja
- *category*: parametar kojim se zahteva sadržaj od sistema za preporuku, ukoliko se ovaj parametar ne postavi sadržaji će biti dobavljeni direktno iz skladipta podataka. Kategorija može biti *recommended* ili *popular*. Rezultati *recommended* kategorije su bazirani preporuci za korisnički profil sa kog je zahtev poslat, a za *popular* skup rezultata se sastoji od najpopularnijih sadržaja u sistemu.
- *sortTypes*: parametar kojim se zadaje da li će i kako rezultati bi uređeni, predstavljeno je numerički i vrednosti su:
- 0 – ne sortiraj (podrazumevano)
    - 1 – sortiraj po imenu
    - 2 – sortiraj po jeziku
    - 3 – sortiraj po oceni
    - 4 – sortiraj po vremenu početka
    - 5 – sortiraj po vremenu završetka
    - 6 – sortiraj po dužini
    - 7 – sortiraj po roditeljskoj kontroli
    - 8 – sortiraj po tipu sadržaja

```
GET http://serverIP:port/CMoreRESTApi/rest/search?{query_parameters}
```

Podržani parametri su:

- *searchQuery*: tekst za pretragu
- *maxResults*: maksimalni zahtevani broj rezultata

4.3.15.2 Mrežni pristup izvorima sadžaja

Mrežni pristup izvorim sadržaja se obavlja korišćenjem samo jedne metode:

```
GET http://serverIP:port/CMoreRESTApi/rest/contentSources?{query_parameters}
```

Parametri metode *contentSources* su:

- *name*: parameter kojim se zahteva izvor sa odrđenim imenom
- *path*: zadaje se jedinstveni identifikator izvora (pr. dvb://onid/tsid/sid, http://site.com)
- *type*: zadaje tip izvora koji treba da budu sadržani u rezultatu.Vrednosti mogu biti 0, 1 i 2 koje predstavljaju *dvb*, *iptv* i *web* respektivno.

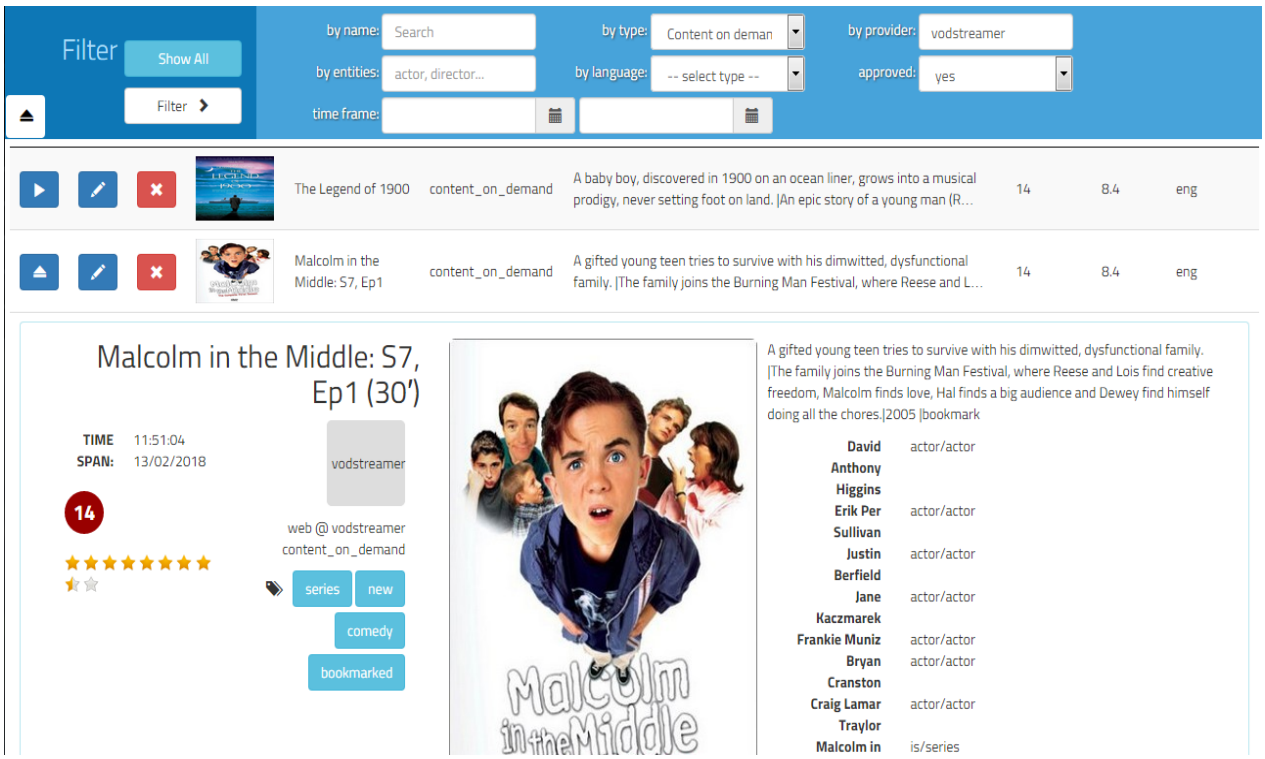
4.3.16 Mrežna aplikacija

Mrežna aplikacija je referentna aplikacija koja koristi spregu za mrežni pristup posužiocu i nudi administratorskim korisnicima grafičko okruženje za interakciju sa poslužiocem. Operacije koje nudi mrežna aplikacija su prijavljivanje na poslužilac, prikaz sadržaja pretragom ili filtriranjem, dodavanje novih sadržaja, odobravanje sadržaja, ažuriranje sadržaja i njihovo uklanjanje.

Dodavanje sadžaja je realizovano realizacijom nezavisnog saklupljača sadržaja kome se pristupa preko sopstvene mrežne sprege i kome se u XML obliku prosleđuju novi ili ažiurirani sadržaji. Uklanjanje se svodi na ažuriranje sadržaja gde se sadržaju *lifespan* polje postavi na trenutno vreme, čime on od tog trenutka više neće biti isporučivan korisnicima, a biće uklonjen tek u narednom procesu čišćenja koje se obavlja u zadatom intervalu. Ovo, takođe, omogućava da obrisani sadržaj bude restauriran.

Koncept odobravanja sadržaja podrazumeva da administrski korisnici moraju da donesu odluku da li su podaci o sadržaju, kao i sam sadržaj, prihvatljivi da budu prikazani korisnicima. Potreba za konceptom odobravanja proizilazi iz automatskog prikupljanja sadržaja od strane komponenti sakupljača sadržaja koji ne može adekvatno biti proveren bez ljudskog faktora.

Snimak ekrana mrežne aplikacije je prikazan na Slici 4.6.



Slika 4.6 Snimak ekrana mrežne aplikacije

5. Procena kvaliteta rešenja

5.1 Tehnike procene

Ovo je poglavlje koje je posvećeno proceni kvaliteta realizovanog rešenja, napravljeno je funkcionalno poređenje sa vodećim rešenjima na tržištu, navedeni su alati koji su korišćeni pri ispitivanju i opisan je ispitni skup sa komentarisanjem pojedinačnih ispitnih slučajeva. Nakon toga su navedeni rezultati ispitivanja sa diskusijom kvaliteta rešenja na kraju.

5.1.1 Funkcionalno poređenje sa konkurentnim rešenjima

U Tabeli 5.1 je napravljeno poređenje funkcionalnosti realizovanog rešenja sa funkcionalnostima vodećih rešenja na tržištu.

Funkcionalnosti/rešenja	Realizovano rešenje	Nangu.tv	MediaRoom Reach	MediaLive	Altiplex
Sadržaj na zahtev	✓	✓	✓	✓	✓
Sadržaji praćeni metapodacima	✓	✓	✓	✓	✓
Metapodaci praćeni sadržajima	✓				
Podrška za socijalne interakcije	✓	✓	✓	✓	✓
Preporuka sadržaja	✓	✓	✓	✓	✓
Ocenjivanje sadržaja	✓	✓	✓	✓	✓
Označavanje sadržaja	✓	✓	✓	✓	✓
Isporuka listi kanala	✓	✓	✓	✓	✓
TV vodič	✓	✓	✓	✓	✓
Podrška za različite ciljne uređaje	✓	✓	✓	✓	✓
Administratorska aplikacija	✓	✓			✓
Reklamiranje i prodaja		✓			✓

Ugrađivanje sistema za naplatu		✓			
Mrežni PVR/DVR		✓			✓
Prodavnice aplikacija					✓

Tabela 5.1 Poređenje funkcionalnosti OTT poslužilaca

U Tabeli 5.1 možemo videti da sva rešenja imaju podršku za standardne usluge digitalne televizije, manipulacije sadržajima i personalizaciju što postaje standardni skup usluga OTT posližilaca za podršku aplikacijama digitalne televizije. Sve češće podržavane usluge su i reklamiranje i prodaja, ugrađivanje sistema za naplatu, mrežni PVR/DVR i prodavnice aplikacija.

Realizovano rešenje se ističe po „zatvaranju kruga“ sadržaj-metapodatak-sadržaj, gde uz se uz metapodatke o sadržaju isporučuju i dodatni OTT sadržaji vezani za pomenute metapodatke.

5.1.2 Ispitno okruženje

Računarsko okruženje na kome je vršeno ispitivanje je bazirano na operativnom sistemu Ubuntu Linux OS v12.04, a fizičku arhitekturu čini 4GB RAM, 200GB HDD, single core CPU 2.3GHz.

Za ispitivanje performansi poslužioca je korišćen alat Apache JMeter, kao jedan od standardnih alata za ispitavanje performansi poslužica.

5.1.3 Apache JMeter

Apache JMeter [23] je alat otvorenog programskog koda napisan u Java programskom jeziku koji je, u ovom radu, korišćen za ispitivanje performansi sisema za podršku aplikacijama digitalne televizije, i to verzija 2.11. Predviđen je za simulaciju velikog opterećenja u cilju ispitivanja, testiranja i analize performansi sistema zasnovanih na arhitekturi poslužilac-korisnik iz vrlo intuitivnog grafičkog okruženja.

Apache JMeter omogućava ispitivanje opterećenja i performansi baza podataka posredstvom JDBC rukovaoca, Web programske podrške upotrebom HTTP/ HTTPS protokola, kao i mnogih drugih poslužilaca ili protokola.

Upotrebom određenih skriptnih komponenti koje podržavaju sintaksu Java programskog jezika moguće je izvršavanje koda napisanog Java programskim jezikom kao i uključivanje Java biblioteka.

Omogućava pokretanje testnih slučajeva iz velikog broja programskih niti čime se simulira veliki broj korisnika i postiže konkurentnost. Poseduje veliki broj podesivih strategija za vremensko raspoređivanje slanja zahteva u toku izvršavanja testnih slučajeva.

Nudi podršku i za ispitivanje očekivanog izvršavanja gde je moguće zadati skup predviđenih događaja koji ako ne bude zadovoljen određeni testni slučaj biva signaliziran kao neuspešno izvršen.

Rezultate i analize ispitivanja je moguće predstaviti u vidu grafa, stabla, tabele ili ih je moguće zapisati u datoteku.

5.1.4 Ispitni skup

Ispitni skup se sastoji od tri ispitna slučaja koji su zanovani na merenju vremena potrebnog poslužiocu da odgovori na zahtev korisnika. Svaki ispitni slučaj je sačinjen od kombinovanih poziva četiri metode koje nude standarden usluge OTT poslužioca digitalne televizije.

Pozivane metode su upit za sadržajima na zahtev koji uključuju metapodatke, upit koji zahteva elektronski vodič kroz program za jedan dan za jedan televizijski kanal, upit za preporučenim sadržajima sa svih televizijskih kanala koji uključuju sadržaje koji su muzički progam ili dokumentarni film i upit za listom kanala.

Poziv metode za zahtev za sadržajima na zahtev koji uključuju metapodatke izgleda ovako:

```
/rest/contentItems?types=content_on_demand&sourceName=vodstreamer
```

Parametri ove metode na govore da ona zahteva sadržaje na zahtev i da te sadržaje nudi provajder sadržaja čije je ime navedeno.

Zahtev za elektronskim vodičem kroz program izgleda ovako:

```
/rest/contentItems?sourceName=CAN%2BFR&endsAfter=10%3a15%3a00+22%2f06%2f2015
        &endsBefore=10%3a15%3a00+23%2f06%2f2015&types=live_video&sortTypes=4
```

Analizom parametara možemo zaključiti da se traži sadržaj uživo čiji je izvor zadati televizijski kanal, u zadatom intervalu od jednog dana koji je sortiran po vremenu početka televizijskog programa.

Zahtev za preporučenim sadržajima koji se emituju uživo i koji su muzički ili dokumentarni program izgleda ovako:

```
/rest/contentItems?category=recommended&tags=atmsphere&types=live_video
```

Upit za listom kanala izgleda ovako:

```
/rest/contentSources?type=0
```

U zahtevu za listom kanala pozivamo metodu *contentSources* REST API-ja kojoj prosleđujemo kao parametar tip televizijskog kanala.

Ispitni slučajevi su:

- Vreme odziva
- Vreme odziva na promenljiv broj korisnika

- Simulacija stvarnog slučaja korišćenja
- Odziv iz referentne aplikacije
- vreme odziva sa različitih lokacija

Ispitivano vreme odziva uključuje vreme koje je potrebno poslužiocu da odgovori na zahtev korisnika od trenutka prijema zahteva do trenutka kada je odgovor spreman tj. prikazani rezultati ne uključuju vreme mrežnog kašnjenja, ukoliko drugačije nije rečeno.

5.1.4.1 Ispitni slučaj vremena odziva

Ovaj ispitni slučaj se sastoji od poziva četiri, gore navedene, metode od strane jednog korisnika. Napravljeno je po hiljadu zahteva za svaku ispitivanu metodu i beležena su vremena odziva za svaki zahtev. Cilj ovog ispitnog slučaja je da se ispita vreme odziva datih metoda u uslovima niskog opterećenja.

5.1.4.2 Ispitni slučaj vremena odziva na promenljiv broj korisnika

Ovaj ispitni slučaj je proširenje prethodnog dodavanjem promenljivog broja korisnika. U ispitnom slučaju se meri vreme odziva datih metoda od strane većeg broja korsinika istovremeno. Broj korisnika ispitivanih metoda se menja za 10, 50 i 100 korisnika. Svaki od korisnika šalje po 1000 zahteva jedan za drugim tako da poslužialac u svakom trenutku opslužuje isti broj zahteva koliko ima i korisnika. Cilj ispitnog slučaja je da ispita koliko zahteva može poslužilac da obrađuje istovremeno.

5.1.4.3 Ispitni slučaj simulacije stvarnog slučaja korišćenja

Simulacija stvanog slučaja korišćenja je zamišljena kao ispitivanje vremena odziva metoda u realnom slučaju korišćenja. Uzet je skup od 1000 korisnika koji šalju zahteve prema Gausovoj raspodeli sa konstantim razmakom od jedne sekunde i rasipanjem od dve sekunde.

5.1.5 Mogućnost proširivanja rešenja

Nakon povećavanja broja korisnika do određene količine, realizovano rešenje ne uspeva da u adekvatnom vremenskom roku odgovori zahteve ili ne uspeva uoopšte da odgovori na zahteve. Rešenje za ovaj problem treba tražiti u sferi proširivanja sistema.

Realizovano rešenje podržava proširivanje, a proširivost rešenja obezbeđuje JBoss aplikativni poslužilac kao i JPA specifikacija, prema kojoj je realizovana komponeta skladište podataka, koja podžava prenosivost tako da omogućava laku zamenu skladišta podataka sa nekim proširivim rešenjem.

5.1.6 Opšti parametri stabilnosti

Opšti parametri stablinosti su praćeni upotrebom operacije *top* operativnog sistema *linux*.

Operacija je pratila stanje zauzetosti procesora i zauzetosti memorije od strane procesa koji predstavlja ispitivano rešenje i skladištila uzorke u datoteku. Uzorci su uzimani na svake tri sekunde u vremenskom periodu od šest časova.

Mereni parametri stabilnosti su:

- stabilnost utroška memorije tokom dužeg rada
- zauzeća procesora tokom dužeg rada

5.1.7 Ispitni slučaj vremena odziva sa različitih lokacija

Ovaj ispitni slučaj je osmišljen u cilju merenja uticaja mrežnog kašnjenja na opšti utisak o odzivu poslužioca. Uzete su obzir tri lokacije sa područja Evrope. Kao referentne vrednosti odziva biće korišćeni rezultati isitnog slučaja vremena odziva, gde dati rezultati ne uključuju vreme mrežnog kašnjenja. Ispitivanje je vršeno korišćenjem usluga sajta *web pagetest* [24].

5.2 Rezultati procene

U ovom poglavlju su prikazani i komentarisani rezultati ispitivanja iz prethodnog poglavlja. Početo je sa rezultatima ispitnih slučajeva, a nakon toga je razmatrana stabilnost rešenja, udeo u odzivu iz referentne aplikacije i fleksibilnost rešenja.

5.2.1 Ispitni slučaj vremena odziva

Tabela 5.2 sadrži prosečna, minimalna i maksimalna vremena odziva za koja je poslužilac isporučio odgovor na pozvanu metodu. Ovi rezultati se koriste kao referentne vrednosti pri daljim ispitivanjima.

Metoda	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
Sadržaj na zahtev	1000	598	581	624	251737
TV vodič	1000	288	286	298	53453
Preporučeni sadržaj	1000	521	518	542	18012
Lista kanala	1000	17	16	20	9850

Tabela 5.2 Vreme odziva poslužica na poziv metoda

5.2.2 Ispitni slučaj vremena odziva na promenljiv broj korisnika

Prikaz rezultata ispitivanja vremena odziva, na promenljiv broj korsničkih zahteva, metode za isporuku sadržaja na zahtev se može videti u Tabeli 5.3.

Broj korisnika	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
10	10000	894	723	1543	251737
50	50000	1220	706	1849	251737
100	100000	25420	25902	26354	251737

Tabela 5.3 Vreme odziva metode sadržaja na zahtev za promenjiv broj korisnika

U Tabeli 5.4 se mogu videti rezultati ispitivanja metode za isporuku TV vodiča pri promenljivom broju korisnika.

Broj korisnika	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
10	10000	356	411	489	53453
50	50000	421	357	1417	53453
100	100000	11502	11801	12354	53453

Tabela 5.4 Vreme odziva zahteva za TV vodičem za promenjiv broj korisnika

Tabela 5.5 prikazuje odzive poslužioca na zahteve promenljivog broja korisnika za preporučenim sadržajem.

Broj korisnika	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
10	10000	630	712	987	18012
50	50000	894	578	1417	18012
100	100000	15084	14598	15321	18012

Tabela 5.5 Vreme odziva zahteva za preporukama za promenjiv broj korisnika

Odziv metode za dobavljanje liste kanala na promenljiv broj korisnika je prikazan u Tabeli 5.6.

Broj korisnika	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
10	10000	26	18	34	9850
50	50000	127	62	202	9850
100	100000	1301	223	2422	9850

Tabela 5.6 Vreme odziva na zahtev za listom kanala za promenjiv broj korisnika

Rezultati ovog ispitnog slučaja nam govore o zavisnosti vremena izvršavanja zahteva od broja korisnika. Možemo zaključiti da se vreme odziva za 10 i 50 korisnika razlikuje u razumnoj meri i da je to očekivano. Iz gore navedenih rezultata, takođe, možemo zaključiti da poslužilac postavljen na računar sa ovom količinom resursa ne može da podrži obradu 100 aktivnih korisničkih zahteva koji se izvršavaju istovremeno, stoga je potrebno posegnuti za vertikalnim ili horizontalnim proširivanjem računarskog sistema na kojem je postavljen poslužilac.

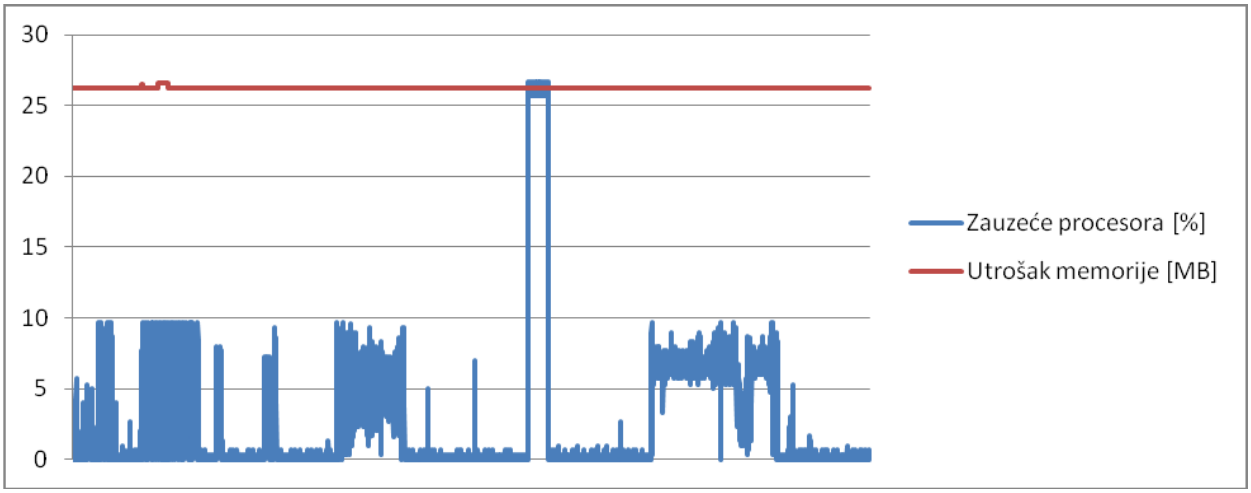
5.2.3 Ispitni slučaj simulacije stvarnog slučaja korišćenja

Iz ovog testa možemo da zaključimo da referentna realizacija poslužioca upotrebom radnog okvira koja je postavljena na spomenutu fizičku arhitekturu može da podrži aktivnost do 1000 korisnika u standardnom slučaju korišćenja. U Tabeli 5.7 se nalaze rezultati merenja za pojedinačne metode poslužioca u ispitnom slučaju simulacije stvarnog slučaja korišćenja.

Metoda	Broj uzoraka	Prosečno [ms]	Minimalno [ms]	Maksimalno [ms]	Količina [Bajt]
Sadržaj na zahtev	25219	1635	596	3157	251737
TV vodič	26151	1015	306	1920	53453
Preporučeni sadržaj	23942	1556	531	2570	18012
Lista kanala	24688	35	16	125	9850

Tabela 5.7 Vremenski odziv metoda u stvarnom slučaju korišćenja

5.2.4 Stabilnost sistema



Slika 5.1 Prikaz stabilnosti rešenja

Na grafiku na Slici 5.1 je prikazana potrošnja radne memorije i zauzeće procesora u periodu od šest časova. Na osnovu grafika možemo zaključiti da je potrošnja radne memorije stabilna i da nema curenja memorije. Što se tiče zauzeća procesora, vrh koji dostiže oko 26

procenata zauzeća procesora je dobijen obradom većeg broja korisničkih zahteva koji su stigli istovremeno i obrađivani istovremeno. Ovakvo ponašanje je rezultat jednog od ispitnih slučajeva vremena odziva na promenljiv broj korisnika. Na osnovu grafika možemo zaključiti da je zauzeće procesorskog vremena bilo stabilno u razmatranom periodu.

5.2.5 Odziv iz referentne aplikacije

Ideja ovog poređenja jeste da se prikaže udeo utroška vremena od strane poslužioca u utrošku vremena koje je potrebno da sadržaj dođe do korisnika. Ostatak koji ne spada u udeo poslužioca se izgubi u mrežnim gubicima, parsiranju rezultata i učitavanje potrebnih resursa u aplikaciji. Rezultati analize su prikazani u Tabeli 5.8. Rezultati odziva poslužioca su preuzeti iz ispitnog slučaja vremena odziva, i oni ne uključuju vreme koje se unosi mrežnim kašnjenjem. Vremena odziva iz aplikacije su merena pri minimalnom uticaju mrežnog kašnjenja jer su aplikacija i poslužilac postavljeni u istoj lokalnoj mreži.

Metoda	Odziv iz aplikacije [ms]	Odziv poslužioca [ms]	Udeo poslužioca [%]
Sadržaj na zahtev	1482	598	40,35
TV vodič	1385	288	20,79
Preporučeni sadržaj	1104	521	47,19
Lista kanala	1254	17+288	28,84

Tabela 5.8 Udeo poslužioca u utrošku vremena isporuke sadržaja do korisnika

Kod prikaza odziva poslužaoca za metodu koja isporučuje listu kanala sabrani su rezultati metoda koja isporučuju TV vodič i listu kanala jer se u referentnoj aplikaciji kod učitavanja kombinuju ovi upiti.

Na osnovu ove analize možemo da zaključimo da vreme odziva posližioca u očekivanoj i podnošljivoj meri utiče na aktivnost aplikacije kada izuzmemo vremensko kašnjenje.

5.2.6 Fleksibinost rešenja

Kao jednu od glavnih odlika realizovanog rešenja treba istaći veliku fleksibinost koju nudi. Realizovani radni okvir se ističe svojom podrškom za ugradive komponente zahvaljujući kojoj se olakšava razvoj i proširivanje realizovanih poslužilaca.

Potrebno vreme za razvoj komponente je procenjeno na osnovu razvoja referentnog rešenja i iznosi 10 dana po komponenti. Ovo vreme uključuje realizaciju programske sprege komponente, razvoj funkcionalnosti i preuzimanje sadržaja iz spoljnih izvora. Može doći do odstupanja od procenjenog vremema srazmerno sa nivoom kompleksnosti pri razvoju

funkcionalnosti. Brojka od 10 dana deluje zanemarljivo malo na period realizacije celokupnog ekosistema poslužioca. Ovim se ostvaruje značajna ušteda kod "ponovnog projektovanja" (engl, *reengineering*) koje nije neobično s obzirom na dinamičnost razvoja API-ja poslužioca, njihovu dinamičnost dostupnosti, izbora izvora metapodataka i sprega sa okruženjem na internetu.

Stoga možemo zaključiti da je jedna od glavnih osobina rešenja njegova fleksibilnost prema daljem razoju i proširivanju.

5.2.7 Vreme odziva sa različitih lokacija

U Tabeli 5.9 je prikazan udeo vremena obrade zahteva od strane posližioca u vremenu odziva koji se postiže sa različitih mrežnih lokacija.

		London		Pariz		Moskva	
Metoda	Odziv poslužioca	Vreme odziva [ms]	Udeo poslužioca [%]	Vreme odziva [ms]	Udeo poslžioca [%]	Vreme odziva [ms]	Udeo poslužioca [%]
Sadržaj na zahtev	598	1491	40,1	1045	57,22	2041	29,3
TV vodič	288	811	35,5	514	56	1145	25,15
Preporuka	521	1274	40,89	911	57,18	1854	28
Lista kanala	17	251	6,77	116	14,46	311	5,46

Tabela 5.9 Udeo u vremenu odziva sa različitih lokacija

Iz rezultata ovog ispitnog slučaja možemo zaključiti da je vreme mrežne komunikacije kritično utiče na odziv poslužioca. Ovde se manifestuje mana OTT koncepta koja kaže da nije moguće garantovati kvalitet usluge.

6. Zaključak

U ovom radu su prikazani detalji realizacije radnog okvira koji predstavlja ekosistem poslužioca za podršku personalizovanim i interaktivnim aplikacijama digitalne televizije koji se zasniva na isporuci OTT sadržaja.

Upotreba OTT koncepta je sve više zastupljena u digitalnoj televiziji i doprinosti tome da televizijski prijemnik sve više ponovo postaje centar zbivanja u kućnom okruženju. Glavni konkurenti na tom polju mu predstavljaju prenosivi uređaji koji su pristupačniji i lakše njima rukovati, mada digitalna televizija teži tome da uključi i prenosive uređaje kroz koncept drugog ekrana (engl. *second screen*).

Radni okvir koji je predstavljen kao rešenje u ovom radu doprinosi razvoju digitalne telvizije na taj način što nudi fleksibilan odgovor na nove zahteve, u stanju je da obezbedi fuziju informacija koje korisnici inače dobijaju posredstvom više različitih uređaja i doprinosi cilju da TV prijemnik postaje dovoljan, centralni uređaj u kući.

Rad na daljem unapređivanju bi mogao da bude ugrađivanje podrške za mrežni PVR/DVR, podrška za sisteme za naplaćivanje, proširivanje sistema tako da se omogući skladištenje konkretnih sadržaja, proširivanja koncepta baze znanja tako da postoje sakupljači podataka koji će dopunjavati bazu znanja koja se skladišti lokalno.

Veliki značaj u daljem radu nosi potreba da se realizuje i ispita horizontalna proširivost sistema da bi on mogao da podrži broj korisnika koji se zahteva od komercijalnih rešenja.

Apstrakcija pojma sadržaja na viši nivo i pretvaranje ostalih komponenti u ugradive bi doprinelo tome da bi radni okvir mogao biti upotrebljavan i u svrhe koje nisu u vezi sa digitalnom televizijom.

7. Literatura

[1] Projektovanje namenskih računarskih struktura 2, http://www.rt-rk.uns.ac.rs/studijski-program-2009/viii-2009/pnrs2/634-pnrs2-predavanja, učitano 18.06.2015

[2] Pregled karakteristika nangu.tv platforme, https://api.nangu.tv/current/index.html, učitano 18.06.2015

[3] Pregled karakteristika Ericsson Mediaroom Reach platforme, http://www.ericsson.com/us/ourportfolio/products/mediaroom-reach, učitano 18.06.2015

[4] Pregled karakteristika Nagra MediaLive platforme, http://dtv.nagra.com/, učitano 18.06.2015

[5] Pregled karakteristika Alticast Altiplex platforme, http://www.alticast.com/tmp.2014.06.data/AltiPlex_OneSheet_final.pdf, učitano 18.06.2015

[6] Netflix, https://www.netflix.com/rs/, učitano 25.06.2015

[7] YouTube, https://www.youtube.com, učitano 25.06.2015

[8] Caching over-the-top services, the Netflix case, S. A. Jensen, M. Jensen, J.M. Gutierrez, Computing, Networking and Communications (ICNC), 2015 International Conference

[9] Client - server architecture, Berson, Alex, New York, NY : McGraw-Hill, 1992. - 452 p.

[10] A Smart Hill-Climbing Algorithm for Application Server Configuration, B. Xi, Z. Liu, M. Raghavachari, C.H. Xia, L. Zhang, Proceedings of the 13th international conference on World Wide Web, 2004

[11] Java EE, https://docs.oracle.com/javaee/6/tutorial/doc/bnaaw.html, učitano 18.06.2015

[12] Microsoft .NET http://www.microsoft.com/net, učitano 25.06.2015

[13] REST šablon, https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html, učitano 18.06.2015

[14] Liu, Ling, Özsu, M. Tamer (Eds.): Encyclopedia of Database System, Springer, 2009

[15] Highly Scalable Blog, http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/, učitano 17.06.2015

[16] Cloud Computing Competence Center, http://www.cloud-competence-center.com/, učitano 17.06.2015

[17] JBOSS AS, JavaBeans Open Source Software Application Server, http://www.jboss.org/overview/, učitano 19.06. 2015

[18] Korry Douglas, Susan Douglas: *PostgreSQL, Second Edition*

[19] One implementation of search feature in an Internet based EPG provider, Aleksandar Beserminji, Milan Knežević, Goran Stupar, Vukota Peković, ICCE Berlin 2014

[20] Cloud-based framework for collecting DTV data related information from the Internet, Nenad Jovanović, Violeta Vukobrat, Stefan Pijetlović, Velibor Mihić, ICCE Berlin 2014

[21] One solution of STB users cloud based profiling, Goran Stupar, Dejan Nađ, Aleksandar Beserminji, Istvan Papp, ICCE Berlin 2014

[22] One solution of a RESTful API for a cloud based DTV content provider, Stefan Pijetlović, Nevena Jovanov, Violeta Vukobrat, Ilija Bašićević, ICCE Berlin 2014

[23] Apache JMeter Wiki, http://wiki.apache.org/jmeter/, učitano 21.06.2015

[24] Web Page Test, http://www.webpagetest.org/, učitano 26.06.2015