УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Дејан Нађ

Методе интеграције графичког корисничког интерфејса са софтвером хибридног средњег слоја за дигиталне телевизијске уређаје на примеру Андроид ТВ платформе

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:	Дејан Нађ	Број индекса:	E2 4/2014
Област:	Програмска подршка у телевизији и обради слике 2		
Ментор:	проф. 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:

Методе интеграције графичког корисничког интерфејса са софтвером хибридног средњег слоја за дигиталне телевизијске уређаје на примеру Андроид ТВ платформе

ТЕКСТ ЗАДАТКА:

Хетерогеност градивних компоненти софтвера дигиталног телевизијског уређаја, представља један од кључних проблема приликом развоја модерних графичких корисничких интерфејса за ове уређаје. Неопходна је компонента која ће све повезати у једну складну и пре свега функционалну целину. Реализовати софтверско решење које ће омогућити интеграцију графичког корисничког интерфејса са софтвером хибридног средњег слоја за дигиталне телевизијске уређаје. Решење треба да обезбеди комуникацију модерне Андроид ТВ апликације са средњим слојем уређаја и серверским средњим слојем. Методе за преузимање садржаја са удаљеног сервера морају бити прилагођене графичкој апликацији уз минимално кашњење, како би се обезбедио хитар одзив и бољи кориснички угођај. Резултате рада и функционалност решења проверити испитним случајевима који ће показати да је реализовано решење у складу са захтевима модерне корисничке апликације.

Руководилац студијског програма:	Ментор рада:

Примерак за: O - Студента; O - Ментора

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**:		Дејан Нађ	
Ментор, **МН**:		доц. др Милан Бјелица	
Наслов рада, **НР**:		Методе интеграције графичког корисничког интефејса са софтвером хибридног средњег слоја за дигиталне телевизијске уређаје на примеру Андроид ТВ платформе	
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)		7/70/43/13/34/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		Графички кориснички интерфејс, хибридни средњи слој, Андроид ТВ, интеграција	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		У овом раду су представљене различите методе интеграције графичког корисничког интерфејса са софтвером хибридног средњег слоја за дигиталне телевизијске уређаје. Хетерогеност градивних компоненти софтвера дигиталног телевизијског уређаја, представља један од кључних проблема приликом развоја модерних графичких корисничких интерфејса. Неопходна је компонента која ће све повезати у једну складну и пре свега функционалну целину. Рад садржи опис конкретног софтверског решења потребног за реализацију графичког корисничког интерфејса и комуникацију са софтвером хибридног средњег слоја дигиталног телевизијског пријемника на примеру Андроид ТВ платформе. Испитивање је показало да је имплементирано решење функционално и у скаду са постављеним захтевима.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:		29/06/2015	
Чланови комисије, **КО**:	Председник:	проф. др Милан Видаковић	
	Члан:	доц. др Данијала 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**:		Dejan Nađ	
Mentor, **MN**:		Milan Bjelica, PhD	
Title, **TI**:		Methods for integrating graphical user interface with hybrid DTV device middleware based on Android TV platform	
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)		7/70/43/13/34/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Graphical user interface, hybrid middleware, Android TV, integration	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper presents various methods for integrating graphical user interface with hybrid DTV device's middleware. The heterogeneity of DTV device's software components, is one of the main problems during during modern graphical user interface's development. The component that will connect everything into one functional system is needed. This paper describes a particular Android TV based software solution used for the implementation of graphical user interface and communication with hybrid middleware. Verification has showed that the given solution is fully functional and in accordance with requirements.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:		29/06/2015	
Defended Board, **DB**:	President:	Milan Vidaković, PhD	
	Member:	Danijela Lalić, PhD	Menthor's sign
	Member, Mentor:	Milan Bjelica, PhD	

Zahvalnost

Zahvaljujem se svom mentoru doc. dr Milanu Bjelici na stručnoj pomoći, savetima i podršci tokom izrade ovog rada.

Posebno se zahvaljujem rukovodstvu Instituta RT-RK na ukazanoj prilici da se bolje upoznam sa načinom rada u inženjerskom okruženju i budem uključen u proces razvoja novih programskih rešenja.

Zahvaljujem se svim članovima svoje porodice na pruženoj podršci u toku čitavog školovanja.

Na kraju zahvaljujem se svima onima koji su na bilo koji način doprineli izradi ovog rada.

SADRŽAJ

1. Uvod....1
2. Teorijske osnove....4
2.1 Metode realizacije grafičkih korisničkih interfejsa za STB i TV uređaje i problemi integracije sa softverom srednjeg sloja....4
2.1.1 Softver STB i TV uređaja....4
2.1.2 Nativno realizovan GUI....7
2.1.3 GUI zasnovan na pregledaču....8
2.1.4 GUI zasnovan na virtuelnoj mašini....10
2.1.5 Hibridni GUI zasnovan na virtuelnoj mašini....11
2.1.6 Predloženi mehanizmi integracionog sloja....12
2.2 Korišćene tehnologije....12
2.2.1 Android operativni sistem....12
2.2.1.1 Arhitektura Androida....13
2.2.1.2 Linux jezgro....13
2.2.1.3 Sistemske biblioteke....14
2.2.1.4 Izvršno radno okruženje....14
2.2.1.5 Okruženje za razvoj aplikacija....15
2.2.1.6 Sloj aplikacija....15
2.2.1.7 Integracija realizovanog rešenja u Android....15
2.2.2 Android TV platforma....15
2.2.2.1 Primer moderne Android TV aplikacije....16
2.2.3 Serverski srednji sloj....18
2.2.4 REST....19

2.2.5 HTTP 20
2.2.6 XML 22
2.2.7 cURL biblioteka 24
3. Koncept rešenja 25
3.1 Analiza zahteva 25
3.2 Arhitektura sistema 26
3.3 Arhitektura klijentskog rešenja 27
3.4 CMore klijent biblioteka 29
3.4.1 Višeklijentski pristup 30
3.4.2 Formiranje zahteva 31
3.4.3 Rukovanje korisničkim profilima 33
3.4.4 Formiranje akcija korisnika 34
4. Programsko rešenje 37
4.1 Konfiguraciona datoteka 37
4.2 Format podataka 38
4.3 Strukture podataka 39
4.4 Funkcije 43
4.4.1 Klijentska biblioteka 43
4.4.1.1 Komunikacija sa poslužiocem 43
4.4.1.2 Rukovanje korisničkim profilima i akcijama 49
4.4.2 JNI prilagodni sloj 55
4.5 Java klase i interfejsi 57
4.5.1 Klase i interfejsi za komunikaciju sa serverskim srednjim slojem 57
4.5.2 Klase i interfejsi za komunikaciju sa srednjim slojem uređaja 58
5. Ispitivanje i rezultati 60
5.1 Ispitivanje brzine izvršavanja 60
5.2 Ispitivanje zauzeća radne memorije 64
5.3 Ispitivanje u okviru TV aplikacije 65
6. Zaključak 67
7. Literatura 68

SPISAK SLIKA

Slika 2.1 Blok dijagram softverske arhitekture STB/TV uređaja 5
Slika 2.2 Blok dijagram softver arhitekture nativno realizovanog GUI 7
Slika 2.3 Blok dijagram softver arhitekture GUI zasnovanog na pregledaču.....9
Slika 2.4 Blok dijagram softver arhitekture GUI zasnovanog na virtuelnoj mašini. 11
Slika 2.5 Arhitektura Android operativnog sistema.....13
Slika 2.6 Android TV interfejs.....16
Slika 2.7 Prikaz rasporeda scena C-More aplikacije u 3D prostoru 17
Slika 2.8 Primer HTTP zahteva od strane klijenta.....21
Slika 2.9 Primer odgovora poslužioca 21
Sika 2.10 Primer XML dokumenta 23
Slika 3.1 Arhitektura sistema 26
Slika 3.2 Arhitektura klijentskog rešenja 28
Slika 3.4 Arhitektura CMore klijent biblioteke.....29
Slika 3.5 Dijagram toka komunikacije između klijenta i CMore klijent biblioteke 30
Slika 3.6 Prikaz formiranja zahteva 31
Slika 3.7. Dijagram toka komunikacije između korisničke aplikacije i serverskog srednjeg sloja.....32
Slika 3.8 Dijagram toka komunikacije prilikom slanja akcija korisnika i zahteva za preporučenim sadržajem.....36
Slika 4.1 Izgled konfiguracione datoteke.....37
Slika 4.2 Prikaz dokumenta koji sadrži podatke dostavljene od poslužioca.....38
Slika 4.3 Prikaz dokumenta sa podacima o servisima 39
Slika 4.4 Struktura podataka koja opisuje jedan servis.....39

Slika 4.5 Struktura podataka koja opisuje jedan podatak 40
Slika 4.6 Struktura podataka koja opisuje jednu oznaku podatka.....40
Slika 4.7 Struktura podataka koja opisuje jedan metapodatak.....40
Slika 4.8 Struktura podataka koja opisuje filter za podatke.....41
Slika 4.9 Struktura podataka koja opisuje filter za servise 41
Slika 4.10 Struktura koja opisueje jedan korisnički profil.....41
Slika 4.11 Struktura koja opisuje jedan nalog korisnika na drugom Internet servisu.....42
Slika 4.12 Izled enumeracije sa kodovima grešaka 42
Slika 4.13 Dijagram klasa za posredstvom kojih se obavlja komunikaciju sa serverskim srednjim slojem 58
Slika 4.14 Dijagram klasa i interfejsa posredstvom kojih se obavlja komunikacija sa srednjim slojem uređaja.....59
Slika 5.1 Grafik zauzeća radne memorije prilikom rada biblioteke.....64
Slika 5.2 CMore aplikacija – prikaz pristiglih EPG i VoD podataka 65
Slika 5.3 CMore aplikacija – prikaz dodatnog sadržaja.....66

SPISAK TABELA

Tabela 2.1 Spisak usluga serverskog srednjeg sloja 18
Tabela 3.1 Skup zahteva moderne GUI aplikacije za STB/TV 25
Tabela 4.1 Prototip funkcije za slanje obaveštenja o zahtevima klijentu 42
Tabela 4.2 Prototip funkcije za slanje obaveštenja o zahtevima nad korisničkim profilima 43
Tabela 4.3 Pregled funkcija JNI prilagodnog sloja.....57
Tabela 5.1 Merenja vremena izvršavanja zahtreva za jednim podatkom 61
Tabela 5.2 Merenja vremena izvršavanja zahteva za 10 podataka 61
Tabela 5.3 Merenja vremena izvršavanja zahteva za 100 podataka 61
Tabela 5.4 Merenja vremena izvršavanja zahteva za 500 podataka 61
Tabela 5.5 Merenje vremena dobavaljanja jednog podatka.....62
Tabela 5.6 Merenje vremena dobavljanja 10 podataka.....63
Tabela 5.7 Merenje vremena dobavljanja 100 podataka.....63
prosek.....63
Tabela 5.8 Merenje vremena dobavljanja 500 podataka.....63

SKRAĆENICE

API - *Application Programming Interface*, Aplikativna programska sprega

DTV - *Digital Television*, Digitalna televizija

EPG - *Electronic Program Guide*, Elektronski vodič kroz program

GUI - *Graphical User Inteface*, Grafička korisnička sprega

HTTP - *HyperText Transfer Protocol*, Mrežni protokol za prenos podataka

JNI - *Java Native Interface*, Java mehanizam za povezivanje koda pisanog u nativnim programskim jezicima (C/C++) sa Java programskim jezikom

OTT - *Over the Top*, Internet servis za isporuku raznovrsnog multimedijalnog sadržaja

STB - *Set Top Box*, Prijemnik digitalnog televizijskog signala

VoD - *Video on Demand*, Video na zahtev

XML - *Extensible Markup Language*, Proširivi jezik za opis podataka

1. Uvod

U ovom radu su predstavljene različite metode integracije grafičkog korisničkog interfejsa sa softverom hibridnog srednjeg sloja za DTV (engl. *Digital Television*) uređaje. Dat je opis konkretnog softverskog rešenja potrebnog za realizaciju grafičkog korisnilkog interfejsa i komunikaciju sa softverom hibridnog srednjeg sloja digitalnog TV prijemnika na primeru *Android TV* platforme. Realizovano rešenje je implementirano u programskom jeziku C kako bi se omogućila laka prenosivost koda i maksimalno iskoristile mogućnosti hardverske platforme.

Poslednjih godina svedoci smo velike revolucije u oblasti televizije. Digitalna televizija [1] je u potpunosti zamenila tradicionalni analogni način prenosa i emitovanja televizijskog signala [2]. Osim prenosa slike i zvuka u digitalnom obliku, što doprinosi boljem i kvalitetnijem prenosu televizijskog sadržaja, digitalna televizija omogućuje prenos dodatnih informacija do krajnjih korisnika kao i povezivanje sa svetskom globalnom mrežom - Internetom. Prelazak na digitalnu televiziju omogućio je korisnicima pogled na svet televizije iz sasvim drugačijeg ugla i pružio im mnoštvo novih mogućnosti kao što su: elektronski vodič kroz program, video na zahtev, veza sa internetom i prikaz internet sadržaja.

Uporedo sa razvojem televizije došlo je velikog napretka u oblasti nauke i tehnologije što je dovelo do pojave novih tehnoloških rešenja koja ljudski život čine lakšim i zanimljivijim. Usled takvog razvoja podstaknut je i ubrzan razvoj namenskih uređaja koji mogu da prime, obrade i prikažu signal digitalne televizije, kao što su televizori, prijemnici digitalnog TV signala (engl. *Set-Top Box* - STB), a poslednjih godina i tablet računari i mobilni telefoni. Kako bi se korisnicima omogućilo da na što jednostavniji i funkcionalniji način rukuju ovim uređajima potreban je kvalitetan softver i grafički korisnički interfejs.

Grafički korisnički interfejs (engl. *Graphical User Interface* - GUI) predstavlja spregu korisnika sa uređajem koja mu omogućuje da kroz manipulaciju grafičkim elementima pristupi svim njegovim funkcionalnostima. U prošlosti STB uređaji su posedovali ograničen i siromašan skup hardverskih resursa, što je bio ključni razlog zbog kojeg grafičke korisničke sprege nisu bile atraktivne za korisnike i nisu mogle na pravi način da im dočaraju svet digitalne televizije. Napredak modernih tehnologija i načina proizvodnje omogućio je razvoj STB uređaja sa mnogo boljim hardverskim karakteristikama i raspoloživim resursima, što je pružilo mnogo više slobode za razvoj bogatijih i boljih rešenja grafičke korisničke sprege za ove uređaje. Grafička korisnička sprega na TV ekranu je danas, za razliku od predhodnih godina, od ključnog značaja. Ona omogućava korisnicima da pristupaju velikom broju funkcija na lak i prijemčiv način, s jedne strane, a sa druge strane, ona je ključna u ponudi operatora s obzirom da omogućava popularnost brenda i privlačenje novih korisnika. Moderne grafičke korisničke sprege pružaju mnoštvo novih mogućnosti, kao što su pregled metapodataka o sadržajima koji se gledaju, prikaz internet sadržaja, aplikacije široke namene i veza sa uređajima u okruženju (npr. dodatnim ekranom na pametnom telefonu ili tablet računaru). Ovakav razvoj zahteva i slučajeva korišćenja nesumnjivo prati i usložnjavanje softvera STB i TV uređaja, kao i sve većih problema integracije. Pre svega potrebno je izvršiti integraciju sa softverom srednjeg sloja (engl. *client-side middleware*), u cilju upravljanja namenskim hardverskim blokovima, s jedne strane, i sa druge strane potrebno je izvršiti integraciju sa softverom serverskog srednjeg sloja (engl. *over-the-top middleware*) [3] koji se nalazi na udaljenom računaru.

Dostupnost raznovrsnih GUI rešenja, kao i raznovrsna softverska rešenja realizacije srednjeg sloja za DTV uređaje, daje mogućnost proizvođačima da biraju gradivne komponente softvera svog proizvoda. Heterogenost koja se ogleda u odabiru raznorodnih gradivnih softverskih elemenata, implementiranih pomoću različitih tehnologija i metoda razvoja, je sve prisutnija u ovim uređajima. Problem integracije sa softverom hibridnog srednjeg sloja predstavlja jedan od najvećih izazova pri realizaciji modernih grafičkih korisničkih interfejsa. Upravo rešavanje ovog problema je bio glavni motiv prilikom realizacije programskog rešenja koje je tema ovog rada.

Realizovano programsko rešenje je razvijeno i testirano u sklopu grafičke korisničke sprege koja je zahtevala integraciju sa softverom hibridnog srednjeg sloja. Bilo je potrebno izvršiti integraciju sa klijentskim srednjem slojem i sa udaljenim serverskim srednjim slojem. Iako je sama grafička korisnička aplikacija pisana u programskom jeziku *Java*, implementirana softverska komponenta je pisana u programskom okruženju C, a sve u cilju približavanju granicama mogućnostima hardverske infrastrukture i formiranju univerzalnog rešenja lako prenosivog i na druge platforme. Komunikacija sa serverskim srednjim slojem, zasnovanom na

REST (engl. *Representational State Transfer*) [4] aplikativnoj programskoj sprezi, je realizovana korišćenjem *libCurl* biblioteke [5] i HTTP (engl. *Hypertext Transfer Protocol*) protokola [6], dok je komunikacija sa grafičkom korisničkom aplikacijom izvršena preko JNI (engl. J*ava Native Inteface*) programske sprege [7]. Funkcionalnost realizovanog sistema čine:

- Funkcionalnosti digitalne televizije (pretraga kanala, pregled liste kanala, elektronski vodič kroz program, teletekst informacije, itd.),
- Prikupljanje podataka sa udaljenog poslužioca putem aktivne Internet konekcije,
- Podrška za video na zahtev usluge (engl. Video on Demand - VoD),
- Podrška za korisničke profile,
- Preporuka dodatnog sadržaja na osnovu navika korisnika.

Dato rešenje je omogućilo integraciju *CMore* [8] grafičke korisničke aplikacije softverom hibridnog srednjeg sloja i obezbedilo metode preuzimanja sadržaja koje izazivaju minimalno kašnjenje. Rezultati rada su provereni korišćenjem realizovane korisničke aplikacije. Skupovima ispitnih merenja je utvrđeno da je realizovano rešenje funkcionalno i ispravno i da zadovoljava sve potrebne zahteve.

Rad se sastoji iz sledećih poglavlja:

- Uvod – sadrži osnovne podatke o radu i opis rada.
- Teorijske osnove – sadrži pregled različitih metoda realizacije grafičkog korisničkog interfejsa i problema integracija sa softverom hibridnog srednjeg sloja, opis *Android TV* platforme i tehnologija korišćenih prilikom implementacije rešenja.
- Koncept rešenja – analiza zahteva i opis ciljne platforme, prikaz arhitekture čitavog sistema, pregled slojeva programske podrške uz akcenat na realizovano rešenje.
- Programsko rešenje – detaljan opis programskog rešenja i funkcionalnih modula potrebnih za integraciju grafičkog korisničkog interfejsa i hibridnog srednjeg sloja.
- Ispitivanje i rezultati – opis ispitnih slučajeva koji su sprovedeni u cilju provere funkcionalnosti rešenja, pregled i analiza dobijenih rezultata.
- Zaključak – kratak pregled onog što je urađeno u radu, mogućih unapređenja i daljih pravaca razvoja.
- Literatura – spisak korišćene literature tokom izrade ovog rada.

2. Teorijske osnove

U ovom poglavlju dat je pregled različitih ekosistema integracije grafičkog korisničkog interfejsa sa softverom hibridnog srednjeg kao i problema integracije za svaki od njih. Takođe, u nastavku je dat kratak opis tehnologija na kojim se zasniva realizovano rešenje integracionog sloja.

2.1 Metode realizacije grafičkih korisničkih interfejsa za STB i TV uređaje i problemi integracije sa softverom srednjeg sloja

Dostupnost velikog broja raspoloživih tehnologija i alata za razvoj, pruža mogućnost izbora prilikom realizacije GUI rešenja za STB i TV uređaje. Kao posledica toga došlo je do pojave različitih metoda i pristupa prilikom realizacije grafičkih korisničkih interfejsa za ove uređaje [9]. Pre nego što se počne sa pregledom različitih metoda realizacije GUI-ja za STB i TV uređaje kao i problema integracije za svaki od njih, neophodno je dati pregled osnovnih gradivniih elemenata ovih uređaja. U nastavku će kroz podnaslove biti predstavljene različite metodologije razvoja GUI-ja i načina integracije sa softverom hibridnog srednjeg sloja, kao i pregled prednosti i nedostataka za svaku od njih.

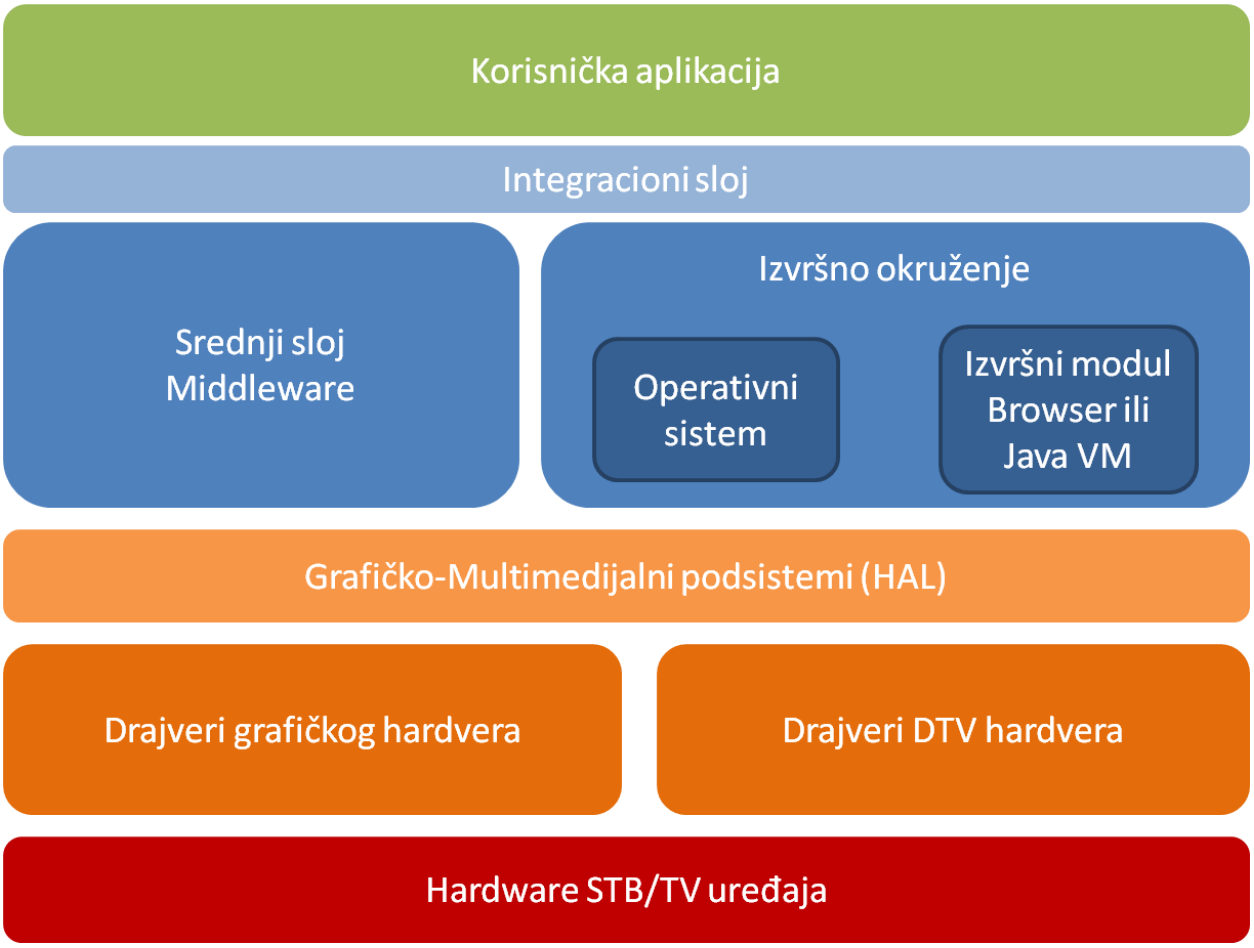
2.1.1 Softver STB i TV uređaja

Na najvišem nivou mogu se izdvojiti četiri ključne komponente koje čine tipičnu softversku arhitekrutu STB i TV uređaja nezavisno od same instance arhitekture. To su:

- Izvršno okruženje (*Runtime environment*),
- Srednji sloj (*Middleware*),

- Grafičko-Multimedijalni podsistem (*Graphics and Multimedia*),
- Korisnička aplikacija (*User application*).
- Integracioni sloj

Na Slici 2.1 prikazan je tipičan blok dijagram softverske arhitekture STB i TV uređaja.



Slika 2.1 Blok dijagram softverske arhitekture STB/TV uređaja

Izvršno okruženje omogućava izvršavanje korisničkih aplikacija pisanih u programskom jeziku najvišeg nivoa apstrakcije za to okruženje. Njegovo jezgro čini operativni sistem (npr. *Real Time Operating System* – RTOS, *Linux*, *Android*), uz dodatne izvršne aplikacije, poput virtuelne mašine (npr. *Java Virtual Machine* – VM na *Linux*, *Dalvik* i *ART* u *Android* OS [10]), internet pregledača (npr. *webkit*) ili skrip interpretera (*Perl*, *Python*, *Lua* i sl.) Osnovne funkcije ovog modula su: snimanje i čitanje podataka, prijem komandi sa ulaznih uređaja kao što je daljinski upravljač, povezivanje sa intranetom i Internetom i sl.

Srednji sloj (*middleware*) obezbeđuje potrebne programske sprege ka višim softverskim nivoima i na taj način apstrakuje funkcionalnost TV uređaja. Posredstvom ovog sloja se obavlja komunikacija sa nižim softverskim komponentama i upravlja hardverskim komponentama

uređaja. Zahvaljujući ovom sloju, softverske komponente sa višeg nivoa ne moraju da poznaju specifičnosti hardvera uređaja, već rukuju njime na jedinstven način.

Grafičko-Multimedijalni podsistemi omogućavaja korisničkoj aplikaciji da, posredstvom softvera srednjeg sloja, daje naredbe namenskim hardverskim blokovima, koji, na primer upravljaju digitilnim prenosnim tokom (*transport stream* - TS), dekoduju audio i video signal, upravljaju blokovima za dekripciju i deskremblovanje sistema za uslovni pristup (*Conditional Access* - CA), rukuju grafičkim hardverom i sl. Funkcije ovog modula su dostupne aplikaciji preko aplikativne programske podrške (*Application Programming Interface* - API) koja se isporučuje u obliku biblioteka funkcija, SDK (*Software Development Kit*) paketa, HAL (*Hardware Abstraction Layer*) nivoa, pozadinskih servisa i sl.

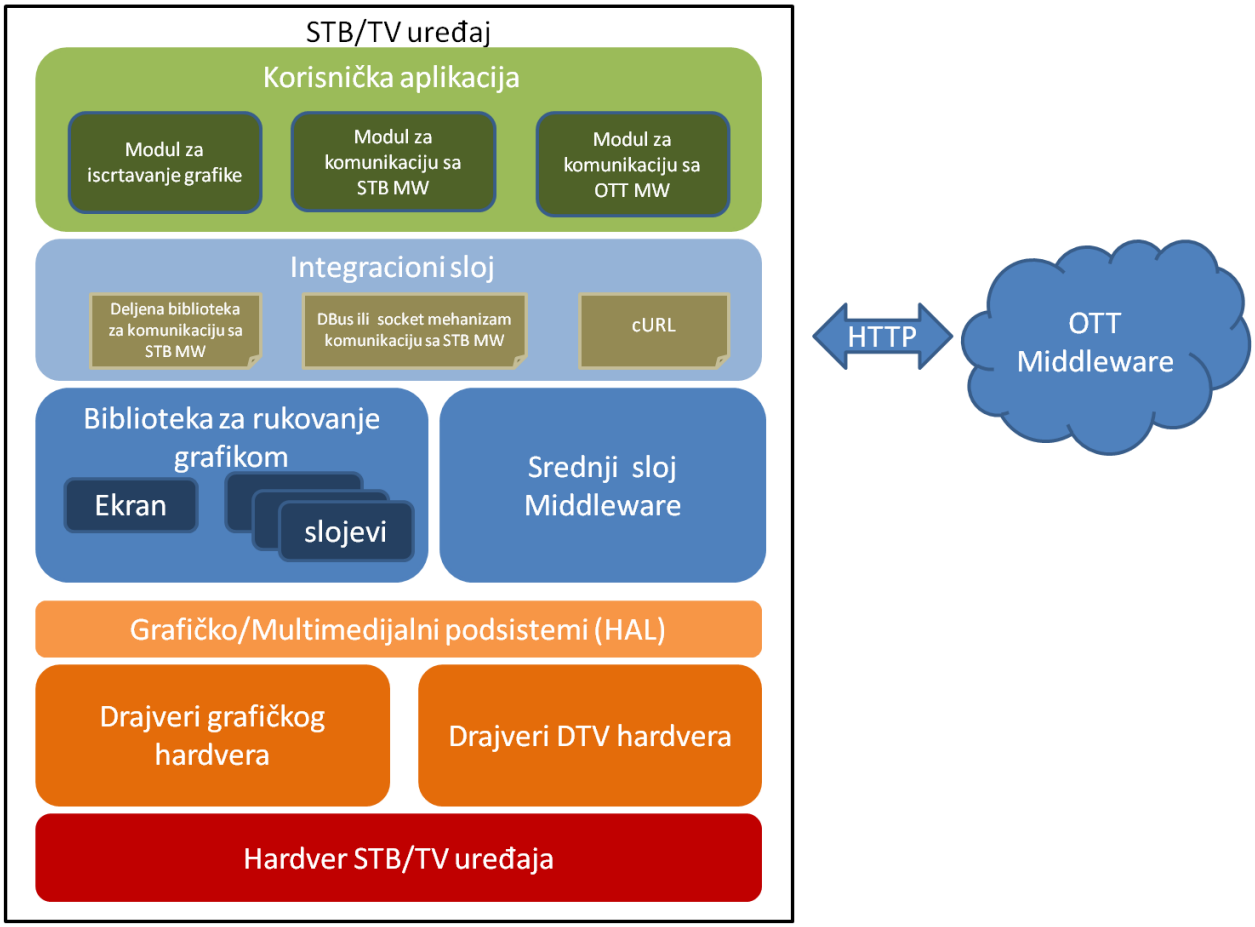
Korisnička aplikacija predstavlja konkretnu realizaciju sprege korisnika sa samim uređajem i u velikoj meri oblikuje korisničko iskustvo i doživljaj (engl. *User Experience* - UX) prilikom rukovanja uređajem. Najčešće ova aplikacija se realizuje korišćenjem programskog jezika najvišeg nivoa apstrakcije za izvršno okruženje u kome se izvršava, uz mogućnost realizacije pojedinih specifičnih delova u nižim nivoima. Na primer, izvršno okruženje koje je isključivo zasnovano na *Linux OS* zahteva realizaciju korisničke aplikacije u *C/C++* jezicima. Okruženje koje poseduje virtuelnu mašinu (*Android* ili *Linux* sa *Java VM*), omogućava realizaciju u *Java* jeziku, uz mogućnost realizacije specifičnih delova, koji utiču na brzinu i na performanse aplikacije, u jezicima C i C++. Povezivanje delova realizovanih u Javi i delova realizovanih u C/C++ se vrši preko *Java Native Interface* (JNI) sprege. Izvršno okruženje sa internet pregledaćem omogućuje realizaciju aplikacije u skript jeziku *Javascript*, uz kreiranje grafičke strukture u HTML5, odnosno CSS notaciji. Konačno izvršno okruženje sa proizvoljnim skript interpreterom, omogućava realizaciju aplikacija u skript jezicima, kao što su *Lua*, *Python* ili *Perl*.

Integracioni sloj predstavlja spregu između korisničke aplikacije i softvera hibridnog srednjeg sloja. Posredstvom ovog sloja korisnička aplikacija komunicira sa srednjim slojem uređaja i sa udaljenim serverskim srednjim slojem. Ovaj sloj omogućuje korisničkoj aplikaciji da na jedinstven način upravlja nižim softverskim komponentama i komunicira sa serverskim srednjim slojem u cilju korišćenja njegovih usluga i dobavljanje sadržaja neophodnih za prikaz. Inegracioni sloj je od izuzetne važnosti za funkcionalnost čitavog sistema, jer je on taj koji je zadužen da sve softverske komponente uređaja poveže u jednu skladnu celinu.

2.1.2 Nativno realizovan GUI

Nativno realzovan GUI podrazumeva implementaciju grafičke korisničke sprege u programskim jezicima C i C++, tzv. nativnom sloju (engl. *native*), što zahteva strukturno ili objektno-orijentisani pristup. Za iscrtavanje grafičkih elemenata u grafičkoj ravni na ekranu, aplikacija direktno iz programskog koda poziva funkcije grafičkih biblioteka nižeg nivoa (npr. *OpenGL ES* [11] ili *DirectFB* [12]). Takođe aplikacija direktno može da poziva funkcije srednjeg sloja, kako bi dobavila neophodni podatke ili upravljala odgovarajućim hardverskim blokovima.

Ovaj način realizacije GUI-ja je danas retko u upotrebi, koristi se samo za realizaciju jednostavnijih aplikacija na STB uređajima ograničenih performansi [13][14]. Realizacija grafičke korisničke sprege na ovaj način ne može da zadovolji sve zahteve modernog GUI za STB i TV uređaje. Pogodan je za uređaje minimalnih funkcionalnosti koje se danas sreću na tržištima zemalja u kojima digitalna televizija još uvek nije "zaživela".



Slika 2.2 Blok dijagram softver arhitekture nativno realizovanog GUI

Prednosti ovog metoda realizacije su:

- Brzina i robusnost što je rezultat nativne implementacije,

- Niska cena uređaja s obzirom na skromne hardverske resurse i mali broj softverskih licenci.

Nedostaci ovog metoda realizacije su:

- Otežana portabilnost, visoko vreme potrebno za prenošenje na nove ciljne platforme,
- Loša mogućnost proširivosti novim funkcijama,
- Skromne mogućnosti i funkcionalnost, loše odgovara modernim zahtevima i aktuelnim trendovima i pravcima razvoja.

Integracioni sloj kod ovog pristupa predstavljaju deljene biblioteke namenjene za komunikaciju sa nižim slojevima. Integracija sa softverom hibridnog srednjeg sloja, vrši se direktno iz programskog koda. Poziv udaljenih metoda srednjeg sloja, u cilju dobavljanja podataka za prikaz ili upravljanja resursima, vrši se direktno putem deljene biblioteke ili putem *DBus* ili *socket* mehanizma. Sa druge strane komunikacija sa serverskim srednjim slojem, korišćenje njegovih usluga i dobavljanje sadržaja, obavlja se putem REST poziva posredstvom određene komunikacione biblioteke (npr. *cURL*). Na Slici 2.2 prikazan je blok dijagram softver arhitekrure nativno realizovanog GUI.

2.1.3 GUI zasnovan na pregledaču

GUI zasnovan na pregledaču svoju funkcionalnost zasniva na internet pregledaču (engl. *browser*). Razvoj ovih aplikacija zasniva se na implementaciji odgovarajućih HTML/HTML5 stranica, sa grafičkim elementima opisanim posredstvom CSS stilova, dok je dinamika i interkcija sa korisnikom opisana upotrebom skript jezika *Javascript*. Svojim izgledom aplikacija podseća na realizacije tipične web aplikacije. U cilju obezbeđivanja potrebnih TV funkcionalnosti, aplikacija koristi dodatke (engl. *plug-ins*) za video reprodukciju i komunikaciju sa srednjim slojem, odnosno nativnim servisima operativnog sistema (npr. *Netscape Plugin API* – NPAPI [15]).

Vodeći svetski proizvođači STB i TV uređaja i softvera *NAGRA*, *OpenTV*, *Samsung*, *LG*, *Zappware*, *Accedo*, *Wyplay* i dr. zasnivaju svoja rešenja na ovom pristupu. Takođe, ova metodologija razvoja se naširoko primenjuje i u okviru akademskih i industrijskih istraživanja [16][17].

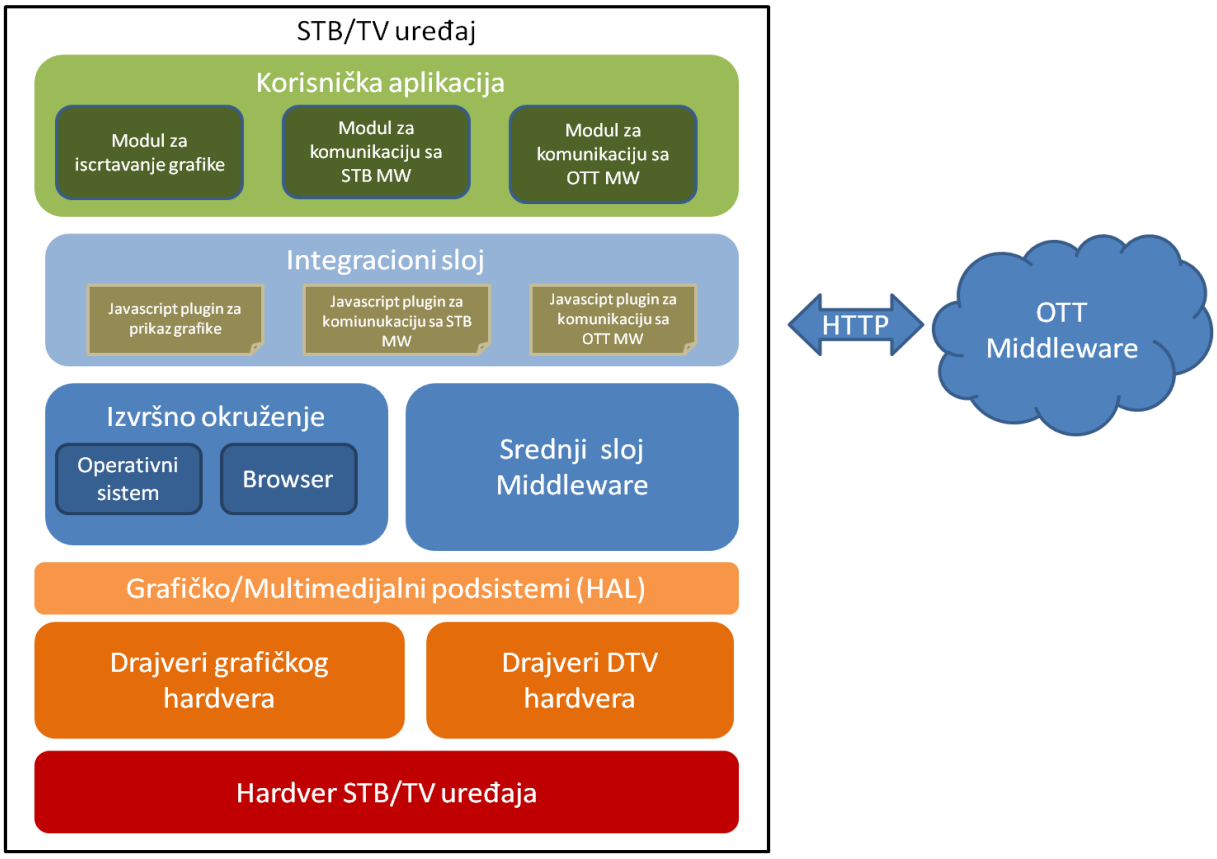
Prednosti ovog pristupa su:

- Visoka brzina odziva,
- Izuzetno velika baza programera upoznatih sa tehnologijama potrebnim za realizaciju rešenja,

- Jednostavna implementacija, za vrlo kratki period moguće je implementirati moderna GUI rešenja koja su atraktivna za korisnike,
- Laka prenosivost rešenja koja se praktično zasniva na prenosivosti i mogućnostima pregledača,
- Jednostavna integracija sa postojećim web sadržajem i web-zasnovanim aplikacijama.

Nedostaci ovog pristupa su:

- Nepostojanost pri izvršavanju - s obzirom da web pregledač nije primarno namenjen za izvršavanje složenih animacija i dinamika kretanja, kojima obiluju moderna GUI rešenja, vrlo često se javljaju problemi koji utiču na postojanost rada aplikacije.
- Velika zahtevnost u pogledu potrebnih resursa što dovodi do smanjivanja performansi i brzine odziva, što u velikoj meri narušava korisnički doživljaj.
- HTML tehnologija ne omogućava većinu grafičkih rešenja, tranzicija i animacija, koja moderna GUI rešenja zahtevaju, takođe ne postoji adekvatna podrška za 3D. Ovi problemi pokušavaju da se reše uvođenjem HTML5 i WebGL [18] tehnologija, ali njihova primena na namenskim platformama je i dalje retka.



Slika 2.3 Blok dijagram softver arhitekture GUI zasnovanog na pregledaču

Integracioni sloj kod ovog pristupa čine posebno realizovani dodaci. Komunikacija sa softverom srednjeg sloja se vrši posredstvom ovih dodataka pozivima iz *Javascript* programskog koda. Takođe na isti način, posredstvom *Javascript* dodatka i putem HTTP protokola, formiraju se odgovarajući zahtevi za dobavljaje sadržaja sa udaljenog OTT (engl. *Over-The-Top*) srednjeg sloja. Na slici 2.3 dat je blok dijagram softver arhitekture GUI zasnovanog na pregledaču.

2.1.4 GUI zasnovan na virtuelnoj mašini

Ovoj grupi korisničkih aplikacija pripadaju aplikacije implementirane u programskom jeziku *Java*. Za razvoj ovakvih aplikacija koristi se veliki broj biblioteka, alata za razvoj grafičkih komponenti koji su već dostupni za ovaj programski jezik. Zbog široke rasprostranjenosti i olakšane prenosivosti, većina modernih pristupa se zasniva na *Android* operativnom sistemu. Osnovu čini *Android* DTV aplikacija [19][20][21], dok se putem JNI sprege vrši sprezanje sa serverom, odnosno sa hardverskim komponentama za dekodovanje i reprodukciju audio i video sadržaja. Na Slici 2.4 prikazan je blok dijagram softver arhitekture GUI zasnovanog na virtuelnoj mašini.

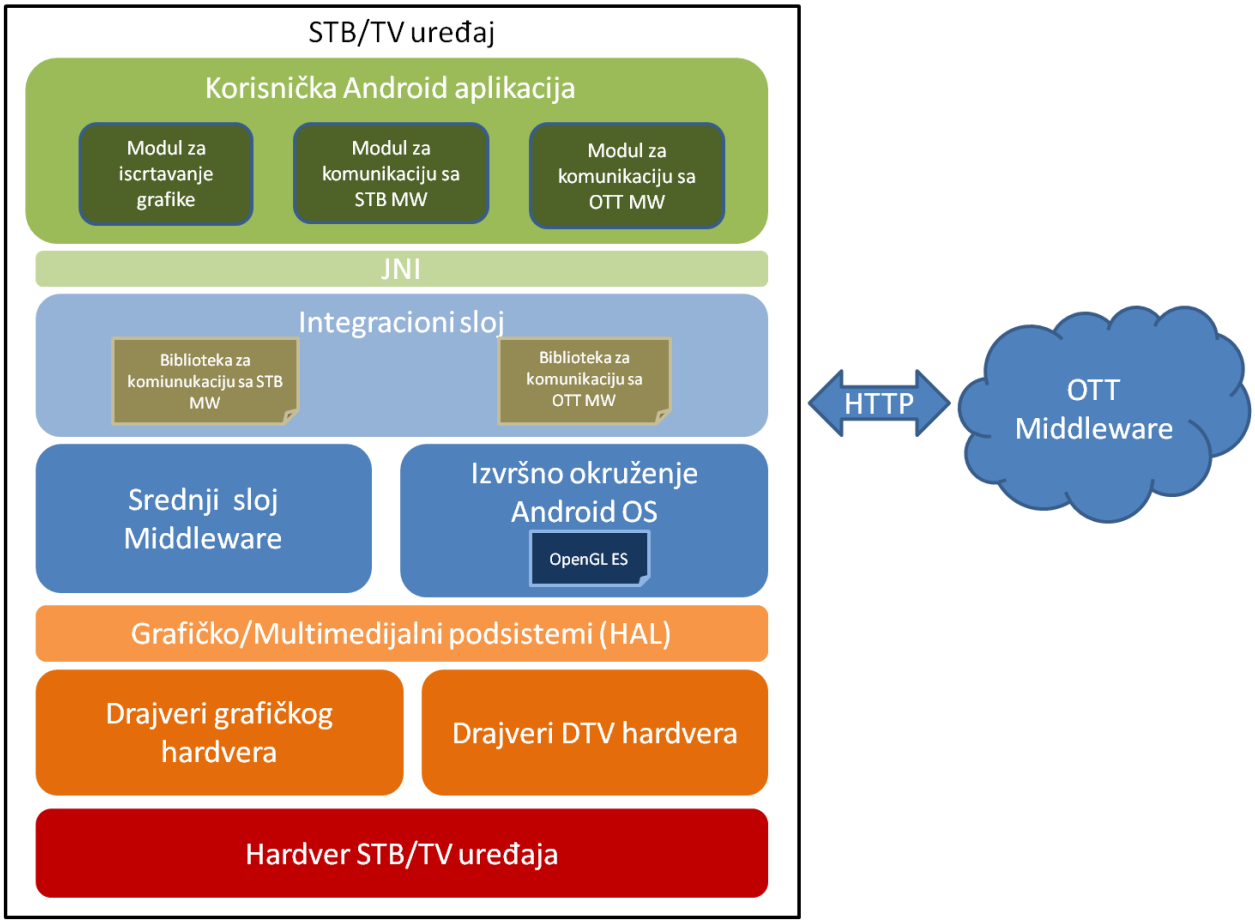
Prednosti ovakvog pristupa su:

- Velika fleksibilnost prilikom dizajniranja rešenja - dostupnost velikog broja tehnologija i alata omogućavaju brz, skalabilan i modularan razvoj,
- Laka prenosivost - široka rasprostranjenost *Android OS* olakšava prenosivost aplikacije na različite ciljne platforme koje izvršavaju *Android*,
- Veliki broj postojećih dostupnih aplikacija obogaćuje mogućnosti samog GUI rešenja. Pre svega to su aplikacije za pregled i pretragu različitih internet sadržaja, aplikacije za reprodukciju multimedijalnog sadržaja različitog formata, i sama prodavnica aplikacija sa koje se mogu preuzeti aplikacije različite namene.

Nedostaci ovog pristupa su:

- Visoka zahtevnost ciljne platforme u pogledu raspoložive procesorske snage i dostupne memorije.
- Neretko se dešava da zbog ograničenih resursa uređaja, dolazi do kašnjenja sofverskih događaja i neprevidivog odziva na korisničke komande.

Integracioni sloj kod ovog pristupa čine sistemske biblioteke namenjene za komunikaciju sa STB i serverskim srednjim slojem. Posredstvom JNI sprege korisnička aplikacija poziva funkcije sistemskih biblioteke i izdaje naredbe softverskim komponentama sa nižih slojeva.



Slika 2.4 Blok dijagram softver arhitekture GUI zasnovanog na virtuelnoj mašini.

2.1.5 Hibridni GUI zasnovan na virtuelnoj mašini

Hibridni GUI zasnovan na virtuelnoj mašini je najnoviji pristup realizacije modernih GUI rešenja. Osnovna ideja je da se upotrebom različitih tehnologija, alata i biblioteka (hibridni pristup), obezbedi jedinstveno korisničko iskustvo kroz realizovani grafički korisnički interfejs. Tipična realizacija se zasniva na upotrebu *Java* jezika i *Android OS*, sa svim prednostima i mogućnostima koje donose, uz odgovarajuća proširenja koja se odnose na uvažavanje specifičnosti okruženja i kreiranje nativnih, brzih komponenti kako bi se omogućila veća kontrola nad grafičkim podsistemom.

S obzirom da se zasniva na upotrebi *Java* jezika i *Android OS*-a, ovaj metod pruža mogućnosti lake prenosivosti GUI rešenja na razčičite ciljne platforme. Razvoj dodatnih biblioteka kao što je *C-More UI Development Framework* [8] omogućavaju kreiranje brzih 2D i 3D grafičkih komponenti i potpuno novog korisničkog iskustva. Kombinacija HTML i *Android* platforme omogućava iskorišćavanje postojećih HTML aplikacija, prodavnica aplikacija i sprega u okviru *Android*-a, što nudi maksimalan broj dostupnih funkcionalnosti korisniku.

Osnovni nedostatak ovog pristupa je u njegovog zahtevnosti u pogledu performansi i memorijskih resursa. U budućnosti se očekuje dodatno povećavanje performansi i snižavanje cena STB i TV uređaja, što će hibridni pristup primamljivijim pri razvoju modernih GUI rešenja.

2.1.6 Predloženi mehanizmi integracionog sloja

Grafička korisnička aplikacija, za čiju funkcionalnost je bilo potrebno izvršiti integraciju sa softverom hibridnog srednjeg sloja, razvijena je u okviru *C-More* istraživačkog-razvojnog projekta [8] instituta RT-RK u Novom Sadu. U pitanju je hibridni GUI zasnovan na virtuelnoj mašini. S obzirom da se radi o hibridnom pristupu, jedan od ključnih problema implementacije rešenja predstavalja integracija sa softverom hibridnog srednjeg sloja. Sa jedne strane neophodno je obezbediti osnovne DTV funkcionalnosti kroz pristup hardverskim komponentama posredstvom srednjeg sloja uređaja, dok je sa druge strane neophodno obezbediti odgovarajuće metode za preuzimanje sadržaja sa udaljenog serverskog srednjeg sloja uz minimalno kašnjenje. Realizovani softver predstavlja univerzalno, brzo i pre svega fleksibilno rešenje ovog problema. Nativna implementacija omogućuje maksimalno iskorišćenje mogućnosti hardverskih resusrsa i laku prenosivost koda. Posredstvom jedinstvenog, lako proširivog API-ja, obezbeđen je pristup svim potrebnim funcionalnostima prilikom razvoja korisničke aplikacije. Rešenje je realizovano u vidu biblioteke, koja posredstvom *libcurl* i REST aplikativne programske sprege, formira odgovarajuće HTTP zahteve za prihvat sadržaja sa serverskog srednjeg sloja. Obrada pristiglih podataka, koji su u XML formatu, se vrši posredstvom API-ja *libXML2* biblioteke. Rezultat obrade su podaci koji su prilagođeni korisničkoj aplikaciji.

2.2 Korišćene tehnologije

U ovom poglavlju dat je opis i namena *Android* operativnog sistema [22] i *Android TV* platforme [23], kao i opis funkcionisanja pojedinih delova koji su od interesa za rešavanje postavljenog problema. Pored toga, u okviru poglavlja će biti opisane tehnologije na kojima se zasniva softversko rešenje integracionog sloja koje je tema ovog rada.

2.2.1 Android operativni sistem

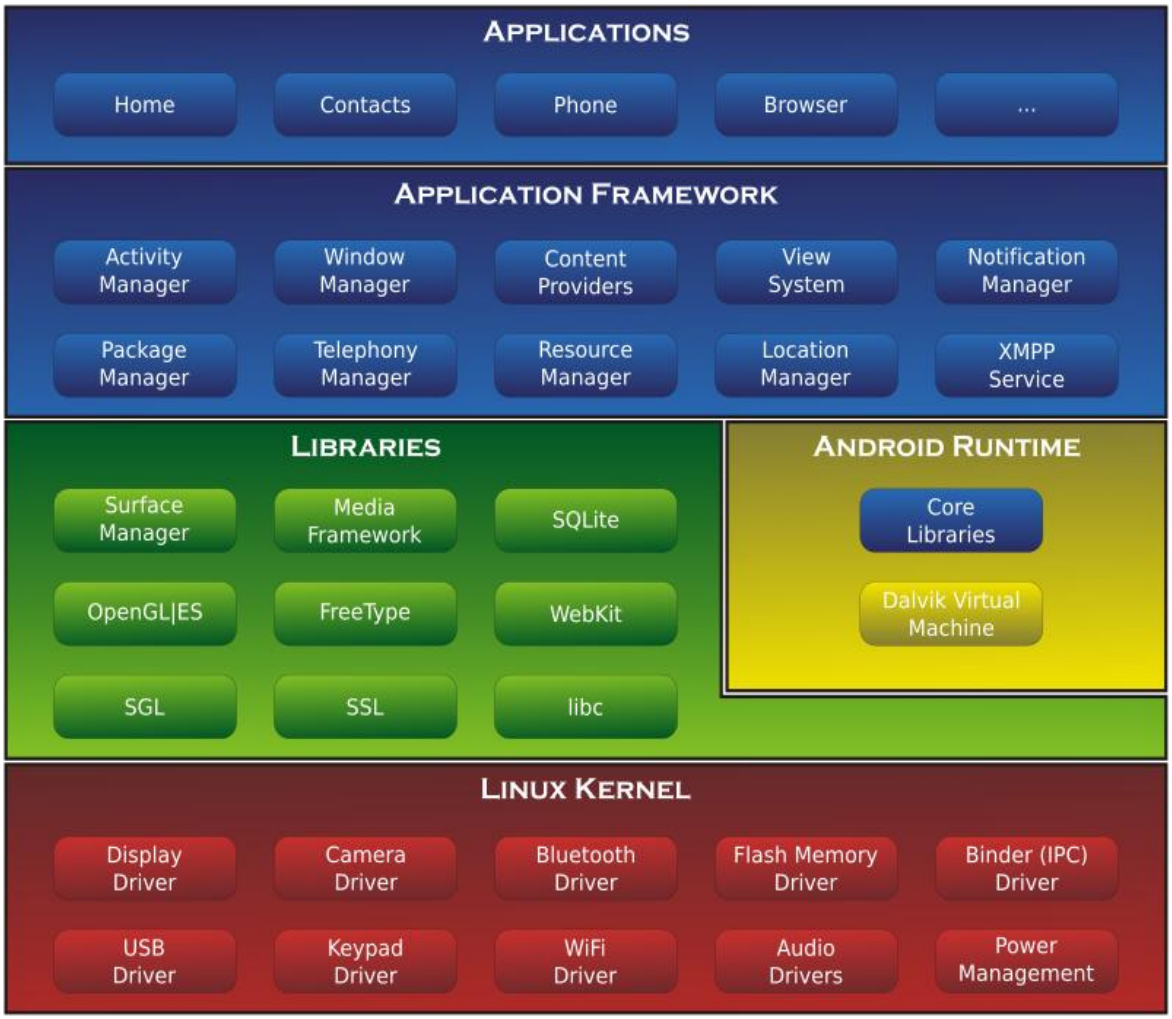
Android je trenutno jedan od najpopularnijih operativnih sistema za mobilne telefone i tablet računare, a poslednjih godina pronalazi svoju namenu i u pametnim satovima, televizorima, STB uređajima, igračkim konzolama i drugim namenskim uređajima. Osnovni razlog za njegovu veliku rasprostranjenost proizilazi iz činjenice da je *Android* operativni sistem otvorenog koda (engl. *open source*) [22] dostupan pod *Apache* licencom, koja dopušta slobodnu izmenu i distribuciju softvera od strane proizvođača uređaja.

2.2.1.1 Arhitektura Androida

Android se sastoji od sledećih slojeva programske podrške:

- Linuks jezgra (*Linux kerner*),
- Sistemskih biblioteka (*Libraries*),
- Izvršnog radnog okruženja (*Android Runtime*),
- Okruženja za razvoj aplikacija (*Application framework*),
- Aplikacije (*Applications*).

Prikaz arhitekture Android operativnog sistema [24] dat je na Slici 2.5.



Slika 2.5 Arhitektura Android operativnog sistema

2.2.1.2 Linux jezgro

Najniži sloj programske podrške čini *Linux* jezgro. To je jezgro na kome je *Android* zasnovan. *Linux* jezgro je iskorišćeno kao sloj koji omogućava apstrakciju hardvera i predstavlja

most između aplikacija i raspoloživih hardverskih resursa. Razlozi za korišćenje *Linux* jezgra kao centralnog dela *Android* operativnog sistema su: proveren skup modula za upravljanje fizičkom arhitekturom (engl. *driver*), upravljanje memorijom (engl. *memory management*), upravljanje procesima (engl. *process management*), rukovanje mrežnim komunikacijama kao i dobra infrastruktura koja je robusna, industrijski dokazana i koja se konstantno razvija.

2.2.1.3 Sistemske biblioteke

Iznad *Linux* jezgra nalazi se sloj sistemskih biblioteka koje su implementirane u programskom jeziku C i C++. Neke od najvažnijih su:

- Surface Manager – biblioteka za upravljanje grafičkom spregom. Ova biblioteka je ujedno zadužena za pravilno iscrtavanje različitih grafičkih komponenti u vremenu i prostoru,
- Media Framework – sadrži veliki broj audio i video kodeka, IMPEG4, H.264, MP3, AAC,itd.
- SQLite – pruža podršku za upravaljnje bazama podataka korisničkih aplikacija,
- WebKit – pruža podršku za pregledanje Internet sadržaja,
- Libc – skup standardnih C biblioteka koje su prilagođene za upotrebu u namenskim sistemima (optimizovana veličina koda i brzina izvršavanja).

2.2.1.4 Izvršno radno okruženje

Radno okruženje (engl. *Android Runtime*) se nalazi u okviru istog sloja kao i sistemske biblioteke i predstavlja skup osnovnih biblioteka koje omogućavaju razvoj *Android* aplikacija korišćenjem *Java* programskog jezika. Jezgro *Android* izvršnog okruženja je *Dalvik* ili *ART* virtuelna mašina [10] koja je zadužena za pokretanje aplikacija višeg nivoa napisanih u *Java* programskom jeziku. *Dalvik/ART* virtuelna mašina je specijalno dizajnirana za *Android* i optimizovana za uređaje sa baterijskim napajanjem, ograničenom memorijom i procesorskom snagom. Kod pisan u *Java* programskom jeziku se izvršava u virtuelnoj mašini. Kada se aplikacija prevodi, ona se inicijalno prevodi u *Java* bajt (engl. *byte*) kod, nakon čega se iz tog oblika dalje prevodi u takozvani .dex format (engl. *Dalvik Executable*). Na ovaj način je omogućeno da aplikacije izrađene za *Android* teorijski mogu da funkcionišu na bilo kojoj hardverskoj platformi koja poseduje *Android* operativni sistem.

2.2.1.5 Okruženje za razvoj aplikacija

Okruženje za razvoj aplikacija predstavlja skup biblioteka i sistemskih aplikacija koje su neophodne za pokretanje i korišćenje korisničkih aplikacija. Ovaj sloj programske podrške je pisan u *Java* programskom jeziku.

2.2.1.6 Sloj aplikacija

Sloj aplikacija je najviši sloj u arhitekturi *Android* operativnog sistema i čine ga korisničke aplikacije. Ovaj sloj je jedini vidljiv krajnjem korisniku. Posredstvom ovog sloja korisnik vrši interakciju sa nižim slojevima.

2.2.1.7 Integracija realizovanog rešenja u Android

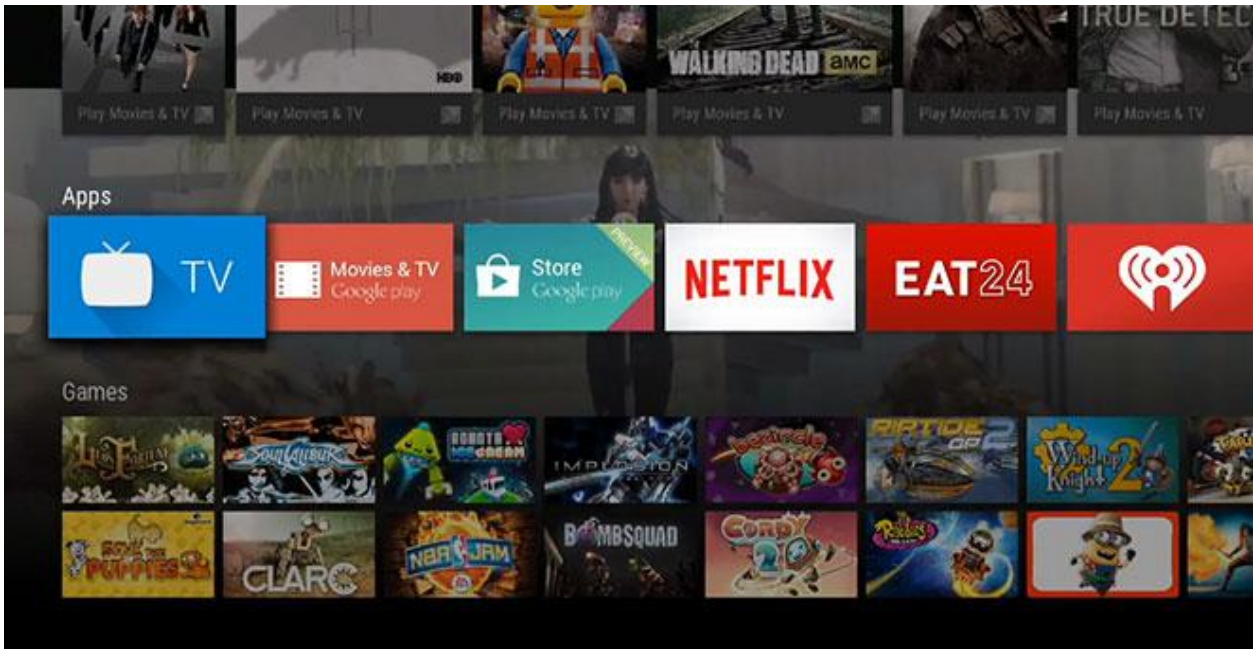
Rešenje koje je opisano ovim radom prestavlja deljenu sistemsku biblioteku. Ova biblioteka pripada sloju sistemskih biblioteka arhitekture *Android* operativnog sistema. Putem JNI sloja biblioteka kominicira sa grafičkim korisničkim interfejsom koji je pozicioniran na najvišem nivou *Android* arhitekture, sloju okruženja za razvoj aplikacija. Sa druge strane realizovana biblioteka posredstvom prilagodnog sloja (engl. *Hardver Abstraction Layer*) pristupa i upravlja raspoloživim hardverskim resursima.

2.2.2 Android TV platforma

Android TV [10] je pametna TV platforma razvijena od strane kompanije *Google*. Ova platfoma je objavljena prošle godine i trenutno predstavlja "vruću temu" u svetu digitalne televizije. Pokreće je najnovija verzija *Andorid* oprerativnog sistema (*Android 5.0 - Lollipop*), i omogućava interaktivno televizijsko iskustvo kroz takozvani "korisnički interfejs sa 10 stopa" (engl. *10-foor user interface*) [25].

Android TV može biti ugrađen u TV ali i u multimedijalne namenske uređaje kao što su STB uređaji, pametni telefoni, tablet uređaji, itd. Putem ove platforme korisnici imaju pristup *Google* prodavnici aplikacija (engl. *Google Play Store*) odakle mogu direktno da instaliraju Android aplikacije, takođe korisnicima je omogućen pristup različitim servisima za reprodukciju multimedijalnog sadržaja kao što su *Netflix* i *Hulu,* kao i igranje *Android* igrica. TV interfejs je horizontalno podeljen na tri sekcije: na vrhu se nalaze preporuke, formirane na osnovu praćenja korisničkih navika gledanja, u sredini se nalaze multimedijalne aplikacije, dok najniža sekcija pripada igricama. Navigacija kroz interfejs je je moguća korišćenjem specijalnog kontrolera za

igrice (engl. *game controller*), daljinskog upravljača ili *Android TV* aplikacije za pametne uređaje. Na Slici 2.6 je prikazan opisani izgled *Android TV* interfejsa.



Slika 2.6 Android TV interfejs

2.2.2.1 Primer moderne Android TV aplikacije

Tipičan primer moderne Android TV aplikacije je grafička korisnička aplikacija, za čiju funkcionalnost je bilo potrebno izvršiti integraciju sa softverom hibridnog srednjeg sloja. To je GUI rešenje za STB uređaje zasnovane na *Android OS*-u, unapređeno upotrebom biblioteka za podršku 2.5D i 3D grafičkih elemenata. U pitanju je hibridni GUI zasnovan na virtuelnoj mašini. Ceo koncept je osmišljen kako bi se poboljšao korisnički doživljaj i obogatilo korisničko iskustvo prilikom gledanja televizijskog sadržaja. GUI elementi su organizovani u vidu scena, koje su raspoređene na različitim pozicijama u 3D prostoru. Sam video sadržaj (engl. *live TV*) nalazi se na video ravni koja predstavlja jednu scenu. Dizajn je osmišljen tako da kretanje korisnika kroz GUI odgovara kretanju kamere od scene do scene, koje su unapred pozicionirane na odgovarajuća mesta u 3D svetu. Na Slici 2.7 prikazan je raspored scena sa GUI elementima u 3D prostoru.

Implementirana aplikacija je kao glavni cilj imala da na što bolji način korisniku dočara svet digitalne televizije kroz sasvim novu dimenziju, a da pri tome ni korisnički ugođaj ni funkcionalnost ne budu zapostavljeni na račun onog drugog. Percepcija 3D sveta je suptilna i nenapadna za korisnika. Tokom korišćenja aplikacije kamera je uglavnom ortogonalno postavljena u odnosu na video sadržaj, te je percepcija izgleda dvodimenzionalna, što je okruženje koje je najprijemčivije za korisnike STB i TV uređaja.

Slika 2.7 Prikaz rasporeda scena C-More aplikacije u 3D prostoru

2.2.3 Serverski srednji sloj

Tipičan STB i TV uređaj danas poseduje širok sprektar funkcionalnosti. Većina modernih grafičkih korisničkih interfejsa za ove uređaje, omogućava korisnicima pristup i manipulaciju nad raznovrsnim multimedijalnim sadržajem koji potiče od različitih izvora sa interneta. U cilju da se to omogući neophodan je jedinstven sistem koji će obezbediti pristup dodatnim uslugama i muitimedijalnim sadržajem različitog porekla. Serverski srednji sloj (engl. *OTT middleware*) predstavlja udaljeni server koji putem IP veze omogućava prenos televizijskog sadržaja i pružanje drugih usluga, uz napomenu da serverski srednji sloj ne mora biti ni u kakvoj vezi sa provajderom date internet usluge. Jedna od osnovnih prednosti serverskog srednjeg sloja je to što je platformski potpuno nezavisan. Koristeći IP vezu kao medijum za prenos podataka, serverski srednji sloj pruža svoje usluge koje ne moraju biti optimizovane i prilagođene krajnjim uređajima.

Neki od najpoznatijuh sistema ovog tipa su: *Nangu.tv, Ericsson MediaRoom, Nagra MediaLive, Netflix, YouTube*, itd. Sadržaj koji putem usluga koje pružaju nude može biti audio, video, televizijski sadržaj, itd. U Tabeli 2.1 prikazan je spisak usluga koje ovi sistemi pružaju.

Dostavljanje sadržaja nazahtev	TV vodič
Dostavljanje metapodaka	Reklamiranje i prodaja
Preporuka sadržaja	Televizija svuda
Ocenjivanje sadržaja	Mrežni snimanje (PVR)
Označavanje sadržaja	Prodavnice aplikacija
Isporuka listi kanala	Korisnički profili
Video na zahtev	Gledaj unazad

Tabela 2.1 Spisak usluga serverskog srednjeg sloja

Serverski srednji sloj, sa kojim je bilo potrebno izvršiti inregraciju, je sistem čija je osnovna funkcionalnost obogaćivanje DVB sadržaja dodatnim sadržajem iz oblaka (engl. *cloud*), koji je slične tematike kao i TV sadržaj koji korisnici trenutno ili najčešće gledaju. Osim OTT metapodataka ovaj sistem, pored standardnog elektronskog vodiča kroz program (engl. *Electronic Program Guide* - EPG), ima podršku za korisničke profile i reprodukciju video sadržaja na zahtev (engl. *Video on Demand* - VoD). Zasniva se na REST sofrverskoj arhitekturi i posredstvom REST aplikativne programske sprege prihvata klijentske zahteve, koje potom obrađuje i šalje odgovor klijentu sa traženim podacima. Sistem isporučuje dve vrste podataka:

podaci o DVB/IP servisima i podaci o emisijama, video na zahtev sadržaju uz metapodatke (dodatni podaci). Svi podaci se u XML formatu šalju u okviru HTTP odgovora, što zahteva njihovu obradu prilikom prijema. Sistem je razvijen u okviru istraživačko-razvojnog projekta *C-More* na Institutu RT-RK u Novom Sadu.

2.2.4 REST

REST (engl. *Representational State Transfer*) [4] je stil softverske arhitekture i pristup u mrežnim komunikacijama koji se najčešće koristi prilikom izrade web servisa (engl. *web services*). Predstavlja skup ograničenja ugrađenih u dizajn komponenti u distribuiranim sistemima za isporuku multimedijalnih podataka (grafika, audio, video, tekstualni fajlovi i hiperlinkovi). Osnovni razlog njegove široke rasprostranjenosti širom web-a, leži u tome što predstavlja mnogo jednostavniju alternativu za SOAP i WSDL zasnovane web servise. REST sistem najčešće, ne uvek, komunicira putem HTTP protokola sa istim HTTP akcijama (GET, DELETE, POST, PUT, itd.) koje se koriste i u internet pregledačima za prikaz internet stranica i za slanje podataka ka udaljenim serverima. Globalna svetska mreža (engl. *World Wide Web*) predstavlja najveći sistem čija se implementacija zasniva na REST-u.

Široko prihvaćena dizajn šema za REST interfejse podrazumeva uključivanje određenog broja resursa koji mogu biti prosleđeni putem standardnog HTTP zahteva. Klijent putem HTTP zahteva sa određenim brojem resursa specificira uslugu koju zahteva od servera. Na primer ako klijent želi da obriše osobu sa imenom Petar treba da formira sledeći HTTP zahtev "DELETE/osoba/Petar".

REST poseduje nekoliko ograničenja koja moraju biti zadovoljena prilikom projektovanja REST baziranih servisa. Neka od najvažnijih ograničenja su:

Klijent – server arhitektura – jedinstveni interfejs koji razdvaja klijenta od servera. Klijent nema uvid u skladište podataka, koje je smešteno interno unutar servera. Na ovaj način je omogućena jednostavna prenosivost klijentkog rešenja. Sa druge strane server nije upoznat sa korisničkim interfejsom ili sa stanjem korisnika, što doprinosi jednostavnosti i skalabilnosti servera.

Bez čuvanja stanja (stateless) – tokom komunikacije između klijenta i servera, server ne čuva konteks klijenta između zahteva. Svaki zahtev bilo kojeg klijenta sadrži sve potrebne informacije, dok se podaci o sesiji čuvaju na strani klijenta.

Pamćenje odgovora (cacheable) – klijent može da pamti odgovore. Odgovori moraju, implicitno ili eksplicitno, označeni za pamćenje, ili se ostavljaju neoznačeni kako bi se sprečila

ponovna upotreba nevažećih ili neispravnih podataka prilikom formiranja budućih zahteva od strane klijenta.

Slojevit sistem – klijent ne može znati da li je povezan sa krajnjim serverom ili sa nekim posrednikom na tom putu. Posrednici mogu značajno unaprediti skalabilnost sistema omogućujući ravnomernu raspodelu opterećenja i deljene memorijke resurse.

Jedinstven interfejs je od ključnog značaja za dizajn bilo kog REST servisa. Jedinstven interfejs pojednostavljuje i razdvaja arhitekturu, što omogućuje modularnost i razvoj nezavisnih celina. Postoje četiri ograničenja jedinstvenog interfejsa, a to su:

- Identifikacija resursa – traženi resursi se specificiraju unutar zahteva korištenjem URI-a. Sami resursi su potpuno nezavisni od načina prezentacije koji se vraća klijentu. Na primer, poslužilac može da šalje podatke iz svoje baze podataka u HTML, XML ili JSON formatu, ali nijedan od tih formata nije interno korišten u okviru servera iako se radi o istim podacima.
- Rukovanje resursima putem njihove reprezentacije – kada klijent prihvati podatke, uključujući metapodatke, poseduje dovoljno informacija pomoću kojih može da izmeni ili obriše resurs.
- Samoopisive poruke – svaka poruka sadrži dovoljno informacija koje opisuju način na koji poruka treba da bude procesirana.
- Upotreba hipermedije – pomoću hiperlinka ili hiperteksta server može da naglasi klijentu da je neki akcija dostupna za predhodno dobavljeni resurs.

Serverski srednji sloj sa kojim je potrebno izvršiti integraciju koristi REST aplikativnu programsku spregu. Realizovano programsko rešenje je moralo da zadovolji sva predhodno navedena ograničenja kako bi se uspostavila komunikacija sa poslužiocem i prihvatili neophodni podaci zahtevani od strane grafičkog korisničkog interfejsa.

2.2.5 HTTP

HTTP (engl. *HyperText Transfer Protocol*) [26] je mrežni protokol koji pripada sloju aplikacije ISO OSI referentnog modela, i predstavlja glavni i najčešći metod prenosa podataka na internetu. Osnovna namena ovog protokola je isporučivanje HTML dokumenata, tj. veb stranica. HTTP je samo jedan od internet protokola.

HTTP je protokol za komunikaciju između servera i klijenta, koji funkcioniše po principu zahtev/odgovor. HTTP klijent, koji je najčešće web pregledač, inicira prenos podataka što uspostavi TCP/IP [27] vezu s udaljenim web-serverom na određenom portu.

Poslužilac konstantno osluškuje zahteve na određenom mrežnom komunikacionom portu (tipično port 80), čekajući da se klijent poveže i pošalje svoj zahtev. Zahtev se sastoji od osnovne HTTP komande (čija je sintaksa propisana standardom i koja se sastoji od naziva komande, imena traženog dokumenta i verzije podržanog HTTP-a) i zaglavlja koje se sastoji od određenog broja redova teksta koji bliže određuje aspekte zahteva. Primer jednog HTTP zahteva za dokument je dat na Slici 2.8.

Zahtev klijenta se obrađuje na poslužiocu i, u zavisnosti od ispravnosti zahteva i mogućnosti zadovoljavanja istog, klijentu se šalje odgovor. Odgovor se sastoji od izveštaja o statusu zahteva (koji se sastoji od trocifrenog koda i kratkog deskriptivnog teksta statusa, npr. 200 OK) i od konkretnog odgovora, ukoliko je zahtev moguće zadovoljiti. Sadržaj odgovora čine zaglavlje, koje je iste sintakse kao i zaglavlje zahteva i daje osnovne podatke o prirodi odgovora, i eventualnog konkretnog sadržaja koji se tražio u zahtevu. Primer odgovora na HTTP zahtev je dat na Slici 2.9. U zavisnosti od verzije HTTP protokola kao i od zaglavlja zahteva, veza se može nakon toga prekinuti, a ista veza se može iskoristiti za slanje novog zahteva, u cilju uštede vremena i resursa.

```
GET /index.html HTTP/1.1
Host: www.example.com
```

Slika 2.8 Primer HTTP zahteva od strane klijenta

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (Unix)  (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html; charset=UTF-8
...
```

Slika 2.9 Primer odgovora poslužioca

HTTPS je verzija HTTP protokola koja koristi algoritam SSL/TLS [28] za zaštitu sadržaja zahteva i odgovora od eventualnog prisluškivanja. Ovaj protokol obično koristi port 443. Lokacija HTTP (i HTTPS) dokumenta se određuje pomoću URL adrese.

HTTP definiše metode koje predstavljaju akciju koju želimo da izvršimo nad identifikovanim resursom. HTTP/1.0 specifikacija je definisala tri metode. Te metode su GET, POST i HEAD. U HTTP/1.1 specifikaciji je definisano još pet metoda: OPTIONS, PUT, DELETE, TRACE i CONNECT.

GET – zahteva reprezentaciju specificiranog resursa. Zahtevi koji koriste GET metodu treba, isljučivo, dobavljaju podatke i da nemaju više nikakav efekat.

HEAD – zahteva odgovor identičan kao i GET, samo bez samih podataka. Služi za dobavljanje metapodataka koji se nalaze u zaglavlju.

POST - ova metoda šalje podatke na određeni URI i očekuje od resursa na tom URI-ju da zna šta da uradi sa njima.

PUT – ova metoda šalje podatke na određeni URI. Ukoliko na toj lokaciji već postoji neki resurs, on se modifikuje. Ukoliko ne postoji resurs, novi resurs će se napraviti.

DELETE – briše specificirani resurs.

TRACE – vraća zahtev nazad do korisnika.

OPTIONS – vraća listu metoda koje poslužilac podržava na specificiranom URL-u.

2.2.6 XML

XML (engl. *Extensible Markup Language*) [29] je jezik označavanja, koji definiše skup pravila za formiranje podataka u elektronskom formatu koji je čitljiv za čoveka, kao i za računar. XML je skraćenica za Extensible Markup Language, odnosno proširivi meta jezik za označavanje (engl. *markup*) tektualnih dokumenata. Ideja je bila da se stvori jezik koji će ljudi i računarski programi moći jednostavno da čitaju. XML definiše opštu sintaksu za označavanje podataka pomoću odgovarajućih etiketa (engl. *tags*) koje imaju poznato ili lako razumljivo značenje.

XML je prvobitno je dizajniran tako da naglasi jednostavnost, lakoću upotrebe, i upotrebljivost na internetu. Korišćenjem *Unicode* standarda obezbeđuje podršku za različite govorne jezike. Iako se njegov dizajn fokusira na dokumente, naširoko se koristi za prezentaciju različitih struktura podataka. Postoje na stotine formata dokumenata koji koriste sintaksu XML-a (npr. RSS [30], SOAP [31], XHTML [32], Atom [33]). XML je takođe prihvaćen kao osnovni jezik za neka komunikacione protokole, kao što je XMPP [34], a neke aplikacije ga koriste za definisanje konfiguracionih datoteka.

Stukturu XML dokumenta čine sledeći elementi:

- Karakter – po definiciji XML je niz karaktera, skoro svaki *Unicode* karakter se može pojaviti u XML dokumentu.

- Oznaka (engl. tag) – koristi se za označavanje jednog elementa. Postoje tri tipa oznaka:
    - Početna oznaka <section>
    - Završna oznaka </section>
    - Oznaka koja označava prazan element <section/>
- Element - jedna logička celina koja može da se sastoji od drugih elemenata. Element koji se nalazi unutar nekog drugog elementa predstavlja njegov potomak (engl. *child*). Svaki XML dokument sadrži samo jedan koreni element a svi ostali elementi su njegovi potomci. Ovo se naziva struktura stabla. Primer jednog elementa je <ime>Dejan</ime>,
- Atribut – podatak koji dodatno opisuje elemet. Ima svoj naziv i svoju vrednost koji su razdvojeni znakom jednakosti, a vrednost atributa je zatvorena zancima navoda. Primer atributa je <ime font=”Arial” style=”bold”>Dejan</ime>,
- Deklaracija – svaki XML dokumet bi trebalo da otpočinje deklaracijom, ona sadrži neke od osnovnih informacija vezanih za dokument. Na primer: <?xml version=“1.0“ encoding=“UTF-8“ standalone=“yes“?>.

Na Slici 2.10 je prikazan izgled jednostavnog XML dokumenta.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Root>
  - <Customers>
    - <Customer CustomerID="XYZ">
        <CompanyName>XYZ Market</CompanyName>
        <ContactName>John Doe</ContactName>
        <ContactTitle>Marketing Manager</ContactTitle>
        <Phone>(212) 555-1212</Phone>
        <Address>100 Park Avenue</Address>
        <City>New York</City>
        <State>NY</State>
      </Customer>
    - <Customer CustomerID="ABC">
        <CompanyName>ABC Market</CompanyName>
        <ContactName>Jane Doe</ContactName>
        <ContactTitle>VP Marketing</ContactTitle>
        <Phone>(201) 555-1213</Phone>
        <Address>102 Main Street</Address>
        <City>Newark</City>
        <State>NJ</State>
      </Customer>
    </Customers>
</Root>
```

Sika 2.10 Primer XML dokumenta

Osnovni razlog zbog kojeg je odlučeno da se u ovom rešenju za prenos podataka koristi XML je njegov čitljiv format, kao i činjenica da se dobro pokazao u sličnom scenariju korištenja (XMLTV [35]). Sa druge strane alternativa je bio JSON, ali s obzirom da se XML koristi u ostatku sistema, odlučeno je da se koristi XML.

2.2.7 cURL biblioteka

cURL je projekat pogramske podrške koja obezbeđuje klijentsku biblioteku (*libcurl*) i konzolnu aplikaciju (*curl*) za prenos podataka putem različitih protokola. Prvi put je objavljen 1997. Ime predstavlja rekurzivni akronim nastao od reči engleskog jezika Curl URL Request Library.

Libcurl [5] je besplatna klijentska biblioteka za prenos podataka putem URL-a. Pored velikog broja mrežnih komunikacionih protokola koje podržava, biblioteka ima podršku i za HTTP i HTTPS protokol. Putem HTTP metoda (GET, POST, PUT, DELETE) moguće je izvršiti prenos podataka sa udaljenim poslužiocem.

Jedan od ključnih razloga njene široke upotrebe je to što je lako prenosiva. Prevodi se i identično radi na različitim platformama: *Solaris, NetBSD, FreeBSD, OpenBSD, Linux, Windows, Symbian, Apple iOS, Android, DOS* i mnogim drugim. Prevođenje biblioteke je moguće na preko 40 programskih jezika uključujuću *C/C++, Java, PHP* i *Python.*

Pored svih predhodno opisanih prednosti ove biblioteke treba naglasiti i njenu brzinu. Upravo iz tih razloga je ova biblioteka izabrana za komunikaciju sa udaljenim serverskim srednjim slojem. Putem ove biblioteke i aplikativne programske sprege koju ona obezbeđuje izvršena je integracija sa serverskim srednjim slojem koji korisničku aplikaciju snabdeva potrebnim podacima. Korišćenjem HTTP protokola [6] i REST API-a [4] dobavljaju se traženi podaci i isporučuju na zahtev korisničkoj aplikaciji.

3. Koncept rešenja

U ovom poglavlju biće prikazana arhitektura čitavog sistema u okviru kojeg je implementirana softverska komponenta koja je tema ovog rada. Na početku poglavlja dat je kratak pregled zahteva koje bilo potrebno zadovoljiti prilikom implementacije, a koji od kojih u velikoj meri zavisi realizacija grafičke korisničke sprege i sam korisnički ugođaj prilikom upotrebe iste. Takođe, u nastvaku dat je detaljniji opis gradivnih blokova i modula od kojih se rešenje sastoji, kao i načina na koji su međusobno spregnuti, uz akcenat na klijentsku biblioteku koja je je rezultat ovog rada.

3.1 Analiza zahteva

Tipičan STB i TV uređaj danas poseduje širok spektar zahteva, nastalih kao rezultat velike količine dostupnih multimedijalnih sadržaja i potreba korisnika kao konzumenata tih sadržaja. U Tabeli 3.1 je dat pregled tipičnog skupa zahteva moderne GUI aplikacije za STB i TV uređaje.

TV uživo	Over-The-Top (OTT) sadržaj
TV vodič (EPG)	OTT metapodaci
Gledaj unazad	Korisnički profili
Video na zahtev (VoD)	Lista programa
Lokalno snimanje (PVR)	Omiljeni programi
Mrežno snimanje (nPVR)	Aplikacije
Digitalni teletext	Timeline

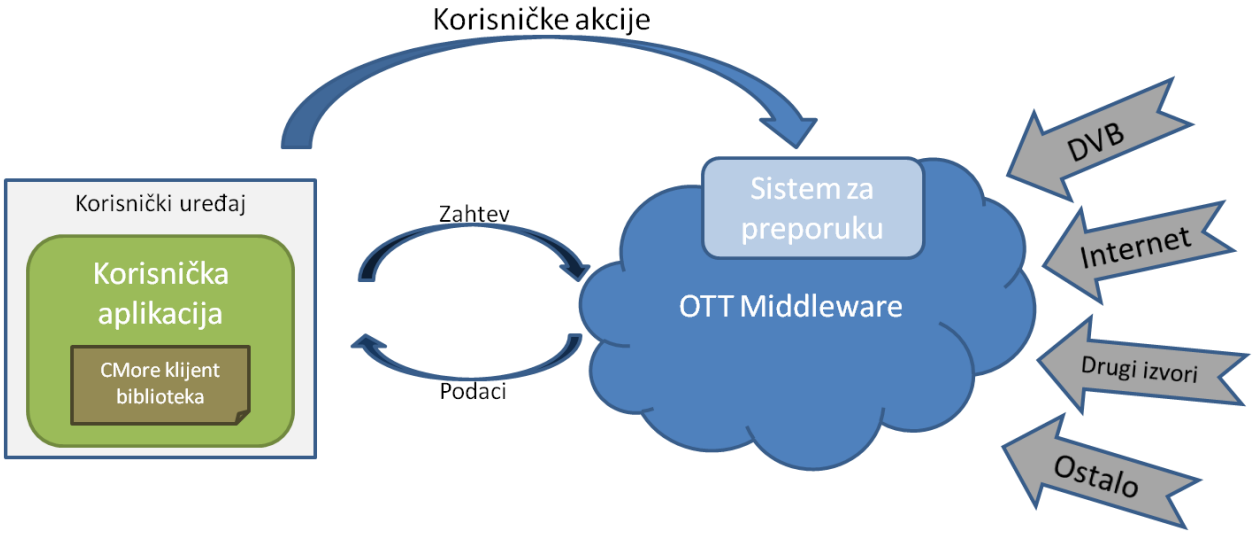
Tabela 3.1 Skup zahteva moderne GUI aplikacije za STB/TV

Većina zahteva iz tabele podrazumeva preuzimanje multimedijalnih podataka (slika, teksta, audio i video sadržaja) sa servera putem IP veze. IP veza je u opštem slučaju sklona promenjivosti protoka, te je jedan od zahteva GUI da obezbedi metode za preuzimanje sadržaja koje izazivaju minimalno kašnjenje. Pored toga, od velike važnosti je i asinhron pristup, kako GUI ne bi morao da čeka odgovor sa traženim podacima, što bi dovelo do zamrzavanja grafičkog prikaza pri čemu bi korisnički doživljaj bio značajno narušen. Takođe, količina dostupnih podataka zahteva njihovo filtriranje i isporuku na zahtev uz uvažavanje korisničkog profila.

U cilju zadovoljavanja predhodno nabrojanih zahteva, realizovana je programska podrška u vidu klijentske sistemske deljene biblioteke u native nivou (*CMore klijent biblioteka*). Rešenje je osmišljeno tako da bude univerzalno, brzo i lako prenosivo na različite platforme koje pokreću *Android OS*, a da pri tome bude primenjivo za različite metode realizacije GUI-ja. Biblioteka je lako proširiva i prenosiva za rad na drugim operativnim sistemima, asinhrona i ima podršku za pristup od strane više klijentata.

3.2 Arhitektura sistema

Pre nego što se počne sa opisom konkretnog rešenja, neophodno je dati prikaz sistema u okviru kojeg je razvijena softverka komponenta koja je tema ovog rada. Sistem se sastoji od klijenta (korisnički uređaj) i serverskog srednjeg sloja (OTT middleware). Na Slici 3.1 je prikazana arhitektura sistema.



Slika 3.1 Arhitektura sistema

Osnovna namena serverskog srednjeg sloja je da obogaćuje DVB sadržaj, podatke o servisima i emisijama, sa dodatnim sadržajem o njima. Dodatni sadržaj predstavlja sadržaj sa interneta, koji može biti tekstualni, audio, video sadržaj, internet stranica i slika. Takođe, poseban deo poslužioca predstavlja sistem za preporuku [36][37]. Na osnovu akcija korisnika, sistem za preporuku formira svoje interne profile, koje potom povezuje sa dostupnim sadržajem. Kada klijent zatraži preporučeni sadržaj, zahtev se prosleđuje sistemu za preporuku, koji potom iz skupa dostupnog sadržaja formira novi, prilagođen korisniku i njegovim navikama gledanja.

Realizovana biblioteka je deo korisničke aplikacije, koja je jedna od gradivnih komponenti softvera korisničkog uređaja i predstavlja grafičku spregu korisnika sa DTV uređajem. Osnovni zadatak realizovanog rešenja je da omogući integraciju sa hibridnim srednjim slojem i da obezbedi odgovarajuće metode preuzimanja sadržaja koje izazivaju minimalno kašnjenje.

Komunikacija sa serverskim srednjim slojem, slanje zahteva i akcija korisnika kao prijem podataka, obavlja se putem API-ja *CMore klijent biblioteke*, koja je posrednik u razmeni podataka između serverskog srednjeg sloja i korisničke aplikacije.

3.3 Arhitektura klijentskog rešenja

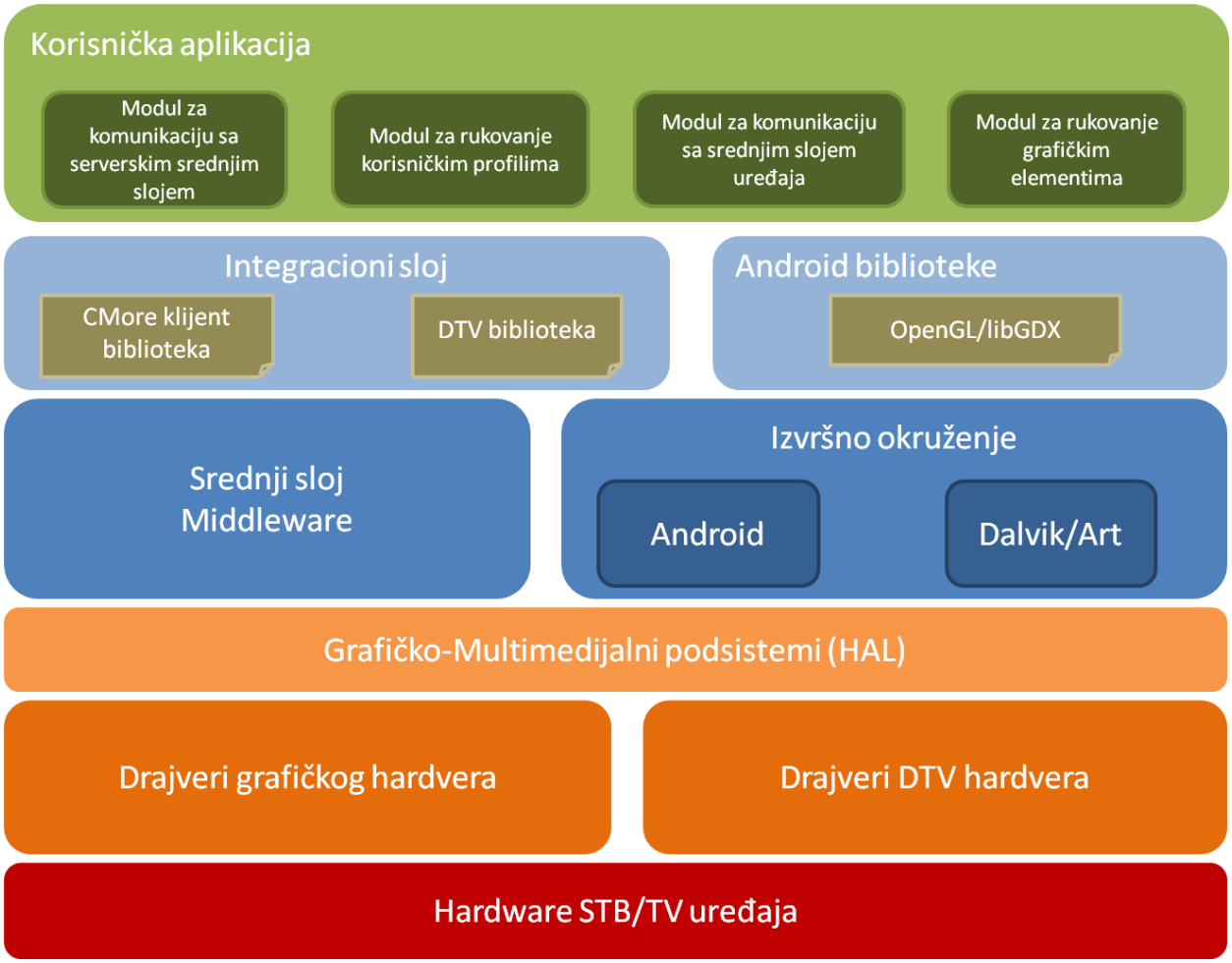
Klijentsko rešenje predstavlja programsku podršku koja se sastoji od sledećih modula:

- Modul za komunikaciju sa serverskim srednjim slojem.
- Modul za upravljanje korisničkim profilima i akcijama.
- Modul za komunikaciju sa STB srednjim slojem.
- Modul za rukovanje grafičkim elementima.

Arhitektura klijentskog rešenja je prikazana na Slici 3.2.

Modul za komunikaciju sa serverskim srednjim slojem je deo korisničke aplikacije čija je osnovna namena sprezanje aplikacije sa serverskim srednjim slojem. Putem ovog modula formiraju se zahtevi za potrebnim podacima ka serveskom srednjem sloju. Kada podaci budu spremni, korisnička aplikacija će sistemom povratne sprege (engl. *callback*) biti obaveštena o tome. Nakon toga aplikacija može da traži podatke. Ovaj modul putem JNI sprege koristi API *CMore klijent biblioteke*.

Modul za upravljanje korisničkim profilima je deo korisničke aplikacije čija je osnovna namena rukovanje korisničkim profilima uređaja. Ovaj modul omogućuje formiranje novog, izmenu i brisanje već postojećeg korisničkog profila, kao i čuvanje predhodno formiranih korisničkih profila. Takođe, pomoću ovog modula se vrši slanje akcija korisnika ka serveru, koji ih potom prosleđuje sistemu za preporuku. Kao i modul za komunikaciju sa serverskim srednjim, putem JNI sprege koristi API *CMore klijent biblioteke*.



Slika 3.2 Arhitektura klijentskog rešenja

Modul za komunikaciju sa srednjim slojem uređaja deo korisničke aplikacije putem kojeg se obavlja komunikacija sa srednjim slojem uređaja. Putem ovog modula aplikacija upravlja hardverskim komponentama. Komunikacija se obavlja putem softverkog rešenja [19] koje je predhodno razvijeno u okviru RT-RK instituta. Ovo rešenje predstavlja *Android* servis koji obezbeđuje kompletnu DTV funkcionalnost pri razvoju aplikacija. Rešenje se sastoji se od *Java* API-ja, koji aplikacijama omogućava pristup DTV funkcionalnostima (DTV biblioteka sa Slike 3.2) i od DTV sistemske biblioteke pomoću koje viši slojevi izdaju naredbe namenskim hardverskim blokovima.

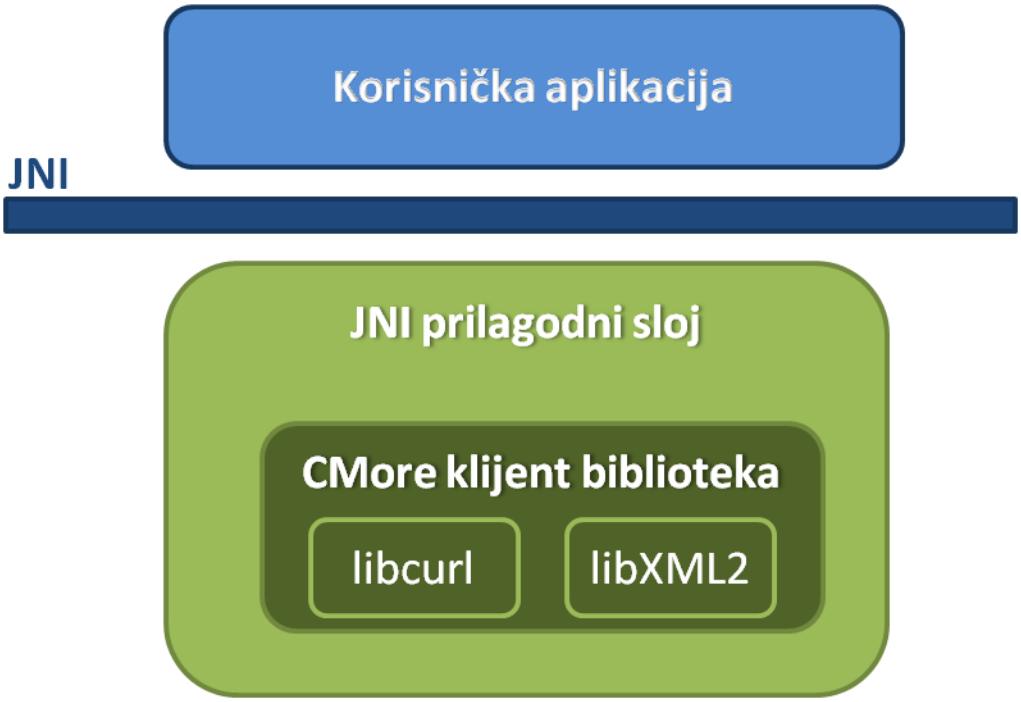
Modul za rukovanje grafičkim elementima je deo korisničke aplikacije koji je zadužen za iscrtavanje i prikaz grafičkih elemenata. Iako se za iscrtavanje i prikaz grafičkih elemenata koristi standardna Android biblioteka OpenGL (engl. Open Graphics Library), za dodatne napredne 3D efekte i animacije koristi se *libGDX* radno okruženje [38]. *LibGDX* biblioteka ima široku primenu u razvoju modernih igrica. Pisana je u programskom jeziku Java, uz pojedine delove koji su pisani u *C* i *C++*, radi boljih perhormansi i brzine izvršavanja. Pored *Android OS*,

biblioteka ima podršku i za *Windows*, *Linux*, *Mac OS X*, *iOS* i web pregledače koji imaju *WebGL* podršku.

Svi predhodno navedeni modulu predstavljaju osnovne gradivne elemente korisničke aplikacije, koja kroz grafički korisnički interfejs vrši interakciju sa korisnikom. Posredstvom *CMore klijent biblioteke* omogućena je integracija grafičkog korisničkog interfejsa sa serverskim srednjim slojem. *CMore klijent biblioteka* je softverska komponenta koja je tema ovog rada. Radi se o sistemskoj biblioteci pisanoj u C programskom jeziku, putem koje se obavlja komunikacija sa serverskim srednjim slojem i upravlja korisničkim profilima uređaja. Ovaj deo softvera će detaljnije biti opisan u nastavku.

3.4 CMore klijent biblioteka

CMore klijent biblioteka predstavlja osnovu datog softverskog proširenja. U njoj se realizuje sva komunikacija sa serverskim srednjim slojem. Sprega sa višim programskim nivoima se vrši putem JNI-a. Sama biblioteka poseduje JNI prilagodni sloj koji se sastoji od skupa funkcija koje grupišu sve funkcionalnosti *CMore klijent biblioteke* i omogućavaju potpunu kontrolu nad njom. Ovim prilagodnim slojem obezbeđena je dvosmerna komunikacija sa klijentskom aplikacijom. Na Slici 3.4 arhitektura *CMore klijent biblioteke* kao i sprega sa višim programskim slojem.

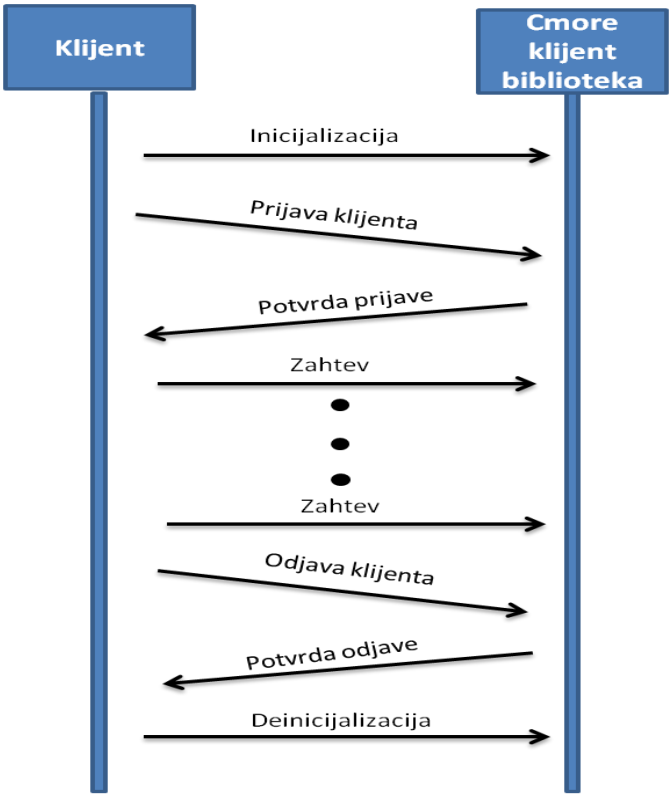


Slika 3.4 Arhitektura CMore klijent biblioteke

Biblioteka se oslanja na dve pomoćne biblioteke. Komunikacija sa udaljenim poslužiocem se obavlja putem *libcurl* biblioteke [5] i REST aplikativne programske podrške [4]. Pristigli podaci sa poslužioca su u XML formatu, pa ih je neophodno dodatno obraditi i od njih formirati podatke prilagođene aplikaciji. Za ovaj postupak se koristi API *libXML2* biblioteke [39].

3.4.1 Višeklijentski pristup

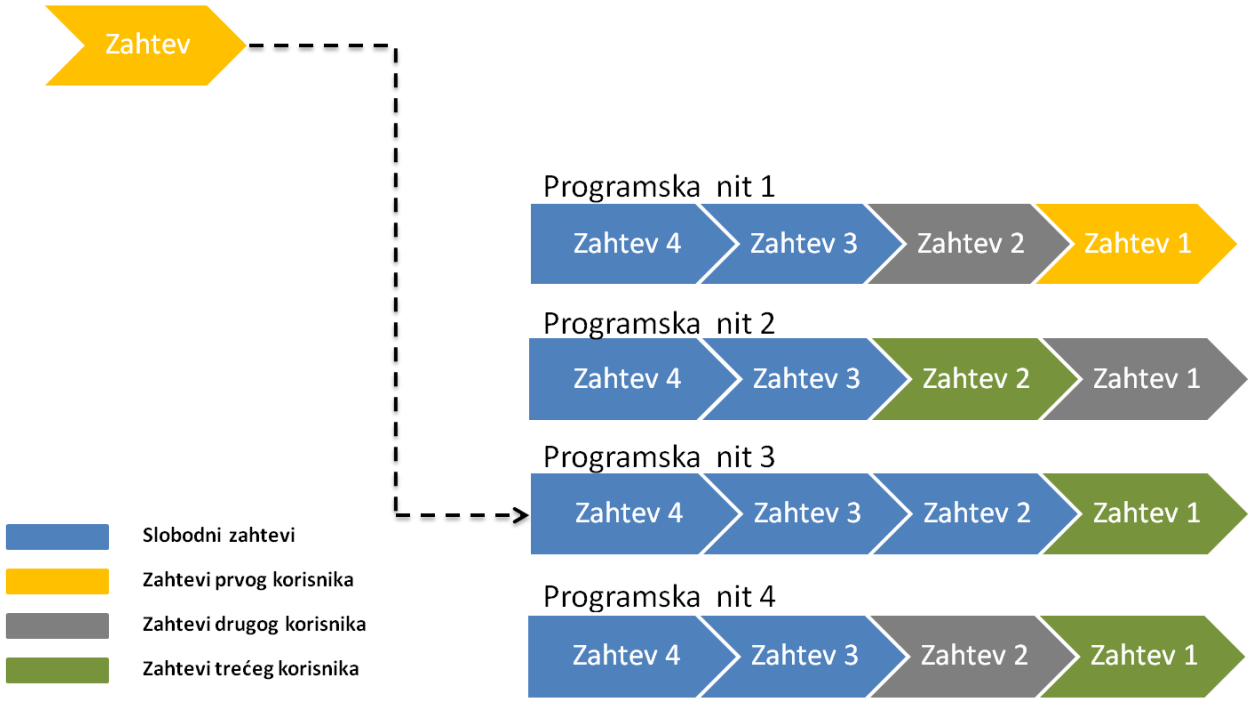
Prilikom implementacije programskog rešenja bilo neophodno zadovoljiti nekoliko zahteva. Pre svega neophodno je obezbediti višeklijentsiki pristup, kako bi se omogućila nesmetana komunikacija i svaki klijent imao svoju komunikacionu sesiju sa poslužiocem. Komunikacija sa bibliotekom započinje inicijalizacijom. Inicijalizacijom se zauzimaju resursi neophodni za dalju upotrebu biblioteke. Potom je neophodno da se klijent prijavi, kako bi mogao da koristi API biblioteke. Kao odgovor na uspešnu prijavu klijent dobija klijentski rukovalac (engl. *client handle*), koji predstavlja ulazni parametar pri pozivu ostalih metoda biblioteke. Na osnovu klijentskog rukovaoca biblioteka razlikuje klijente i povezuje njihove zahteve sa pristiglim podacima sa servera. Nakon prijave klijent može da formira proizvoljan broj zahteva za podacima, sve do trenutka odjave. Po završetku upotrebe biblioteke, neophodno je uraditi odjavu klijenta i deinicijalizaciju, kako bi se oslobodili svi predhodno zauzeti resursi. Na Slici 3.5 prikazan je dijagram toka komunikacije između klijenta i *CMore klijent biblioteke*.



Slika 3.5 Dijagram toka komunikacije između klijenta i CMore klijent biblioteke

3.4.2 Formiranje zahteva

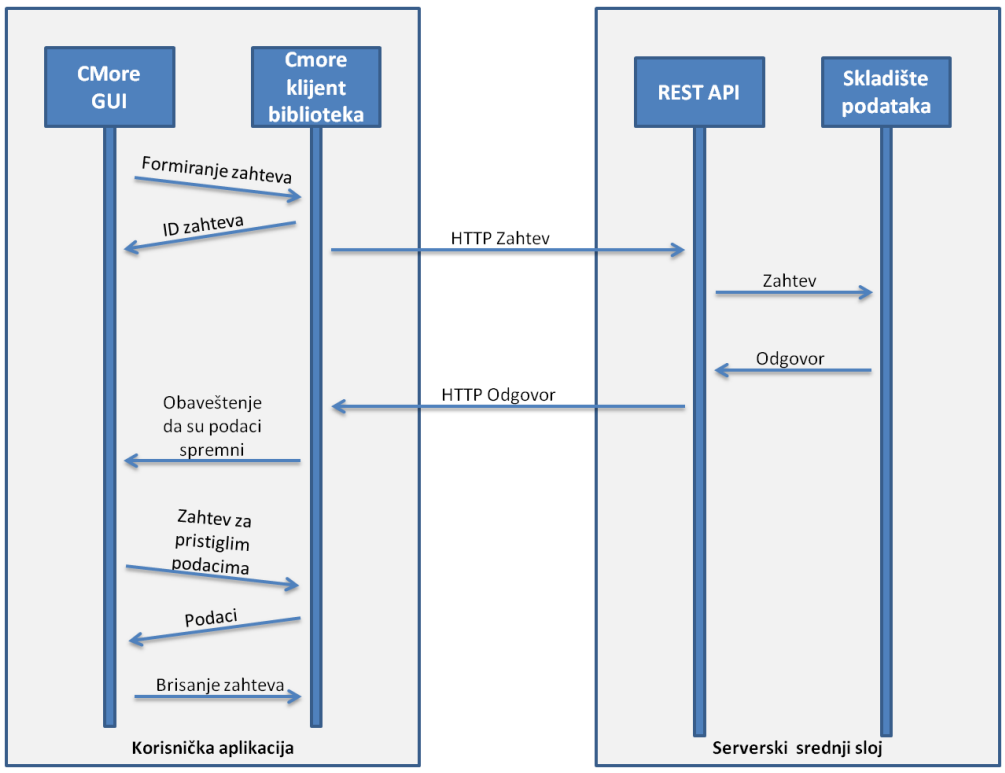
Sa aspekta korisničkog iskustva od izuzetne važnosti je da grafički korisnički interfejs poseduje brz odziv na korisničke komande. Kako komunikacija sa serverskim srednjim slojem može biti promenjivog trajanja, zbog promenjivog protoka IP veze i zbog samog odziva sistema, korisnička aplikacija neretko mora da čeka da traženi podaci pristignu. Ovaj problem je rešen tako što se komunikacija sa serverskim srednjim slojem posredstvom *CMore klijent biblioteke* odvija asinhrono. U skladu sa tim osmišljen je mehanizam kreiranja zahteva ka serveskom srednjem sloju. Svaki prijavljeni klijent ima svoj skup zahteva. Zahtevi svih klijenata su raspoređeni u četiri reda (engl. *queue* – struktura podataka). Svaki od redova inicijalno ima po četiri slobodna zahteva, kojima upravlja po jedna posebna programska nit. Svi formirani zahtevi se raspoređuju u jedan od raspoloživih redova i čekaju na svoje izvršavanje. Prilikom odabira reda bira se onaj koji je najmanje zauzet i u njega će biti dodat zahtev. Na Slici 3.6 je prikazan predhodno opisan koncept formiranja zahteva. Na slici su različitim bojama označeni zahtevi formirani od strane više različitih korisnika. Kao što se može videti, postoje tri klijenta koja koriste biblioteku, njihovi zahtevi su obojeni narandžastom, sivom i zelenom bojom, dok plava boja označava preostale slobodne zahteve. Takođe na slici je prikazan novi zahtev, formiran od strane klijenta čiji su zahtevi označeni narandžastom bojom, koji će biti smešten u red koji obslužuje programska nit 3, zato što je to red koji je najmanje zauzet.



Slika 3.6 Prikaz formiranja zahteva

Kao odgovor na uspešno formiranje zahteva klijent dobija identifikator formiranog zahteva. Identifikator zahteva se kasnije, nakon potvrde da su podaci pristigli i da su spremni za isporuku klijentu, koristi kao ulazni podatak prilikom zahtevanja podataka. Formirani zahtevi potom čekaju na izvršenje. Svaki zahtev sadrži filter pomoću kojeg se specificiraju podaci koji se zahtevaju od poslužioca. Filter je pre upotrebe potrebno inicijalizovati, kako bi se njegova polja popunila podrazumevanim vrednostima. Na osnovu polja filtera koja su promenjena nakon inicijalizacije, prilikom izvršavanja zahteva, formira se URI ka kojem će biti upućen HTTP zahtev za podacima. Kada se zahtev izvrši, serverki srednji sloj pošalje odgovor i podaci budu spremni, *CMore klijent biblioteka* će sistemom povratne sprege poslati obaveštenje klijentu o tome. Klijent se obaveštava o identifikatoru zahteva za koji su podaci pristigli, kao i o broju pristiglih podataka. Nakon toga klijent može da traži određenu količinu podataka.

Kada završi sa upotrebom podataka, podaci mu više nisu potrebni, klijent može da obriše predhodno formirani zahtev. Brisanjem zahteva oslobađaju se resursi za formiranje novih zahteva. Sve dok klijent ne zatraži brisanje zahteva, podaci koji su pristigli za taj zahtev se čuvaju i moguće ih je zatražiti pozivom odgovarajuće metode API-a biblioteke. Nakon brisanja zahteva, brišu se i podaci pristigli za taj zahtev. Predhodno opisan scenario korištenja *CMore klijent biblioteke* je prikazan na Slici 3.7.



Slika 3.7. Dijagram toka komunikacije između korisničke aplikacije i serverskog srednjeg sloja

Na ovaj način postignuto je korisnička aplikacija bude potpuno nezavisna i agnostična na tok komunikacije sa udaljenim serverskim srednjim slojem. Aplikacija može da formira zahteve za potrebnim podacima i da nastavi sa normalnom dinamikom prikaza grafičkih elemenata, bez da mora da čeka na isporuku traženih podataka. Kad predhodno formiran zahtev bude izvršen, i podaci budu spremni, aplikacija će o tome biti obaveštena. Nakon toga korisnička aplikacija može da traži pristigle podatke i da ih koristi prilikom formiranja grafičkog prikaza. Takođe, sve dok ne obriše predhodno formiran zahtev, aplikacija može da traži podatke pristigle za taj zahtev.

3.4.3 Rukovanje korisničkim profilima

Profilisanje korisnika, u proteklih par godina, postalo je vrlo popularno i prisutno u mnogim multimedijalnim servisima. Postojanje korisničkih profila omogućuje komforniju, personalizovaniju i efikasniju pretragu za raznim multimedijalnim sadržajem. Sve moderne grafičke korisničke sprege svoju funkcionalnost zasnivaju na personalizovanim podacima, prilagođenim korisniku i njegovim navikama gledanja. Jedan od zahteva korisničke aplikacije bio je prikaz personalizovanih podataka prilagođenim korisniku. Kako bi se zadovoljio ovaj zahtev, realizovana programsko rešenje ima podršku za koriničke profile.

Pored podataka o samom korisničkom profilu, za svaki korisnički profil se vezuje i lista naloga na drugim servisima za isporuku raznovrsnog multimedijalnog sadržaja (npr. Netflix, Facebook). Ovi podaci se skladište na serverskom srednjem sloju i mogu se dostaviti klijentu na zahtev. Klijent može da formira novi korisnički profil, obriše ili izmeni postojeći. Takođe, putem API-a *CMore klijent biblioteke*, klijent može da dobavi listu korisničkih naloga sa drugih servisa za nekog korisnika.

Implementacija unutar biblioteke je takva da svaki klijent, pored sklupa zahteva za dobavljanje podataka, prijavom dobija i skup zahteva za rukovanje korisničkim profilima. Sam mehanizam formiranja zahteva je isti kao i u slučaju predhodno opisanog formiranja zahteva za podacima. I u slučaju zahteva za rukovanjem korisničkim profilima, postoje četiri reda u koja se smeštaju zahtevi, svaki od njh obslužuje posebna programska nit. Ovi zahtevi su potpuno nezavsni od zahteva za podacima. U ovom slučaju redosled poziva funkcija između klijenta i bilioteke je nešto drugačiji. Nakon formiranja zahteva, klijent kao odgovor dobija identifikator zahteva, pomoću kojeg kasnije razlikuje za koji zahtev je pristiglo obaveštenje o uspešnosti. Ukoliko se radi zahtevu za korisničkim nalozima na drugim servisima, klijent se obaveštava i o broju pristiglih podataka koji predstavljaju jedan korisnički nalog, nakon čega može od

biblioteke da zatraži određenu broj pristiglih podataka. Na kraju klijent može da obriše predhodno formirani zahtev, čime oslobađa resurse za formiranje novih.

3.4.4 Formiranje akcija korisnika

Kombinovan DTV sadržaj sa sadržajem sa Interneta privlači publiku različite starosne dobi i interesovanja. U nameri da se pomogne korisnicima televizijskih prijemnika da pronađu svoj put u velikoj količini podataka različitog sadržaja, bolje razumevanje njihovih navika gledanja i njihovih interesovanja je neophodno.

Jedan od zahteva realizovane korisničke aplikacije je i dostavaljanje preporučenog sadržaja. U tu svrhu na strani serverskog srednjeg sloja, realizovan je poseban modul za preporuku sadržaja [36][37]. Ovaj modul zahteva akcije korisnika, kako bi formirao svoje interne profile korisnika i povezao ih sa dostupnim sadržajem, koji odgovara njihovim interesovanjima i navikama gledanja. Kako bi se omogućilo slanje akcije korisnika, realizovane grafičke korisničke sprege ka poslužiocu i sistemu za preporuku, programsko rešenje koje je tema ovog rada, poseduje i mehanizam slanja korisničkih akcija.

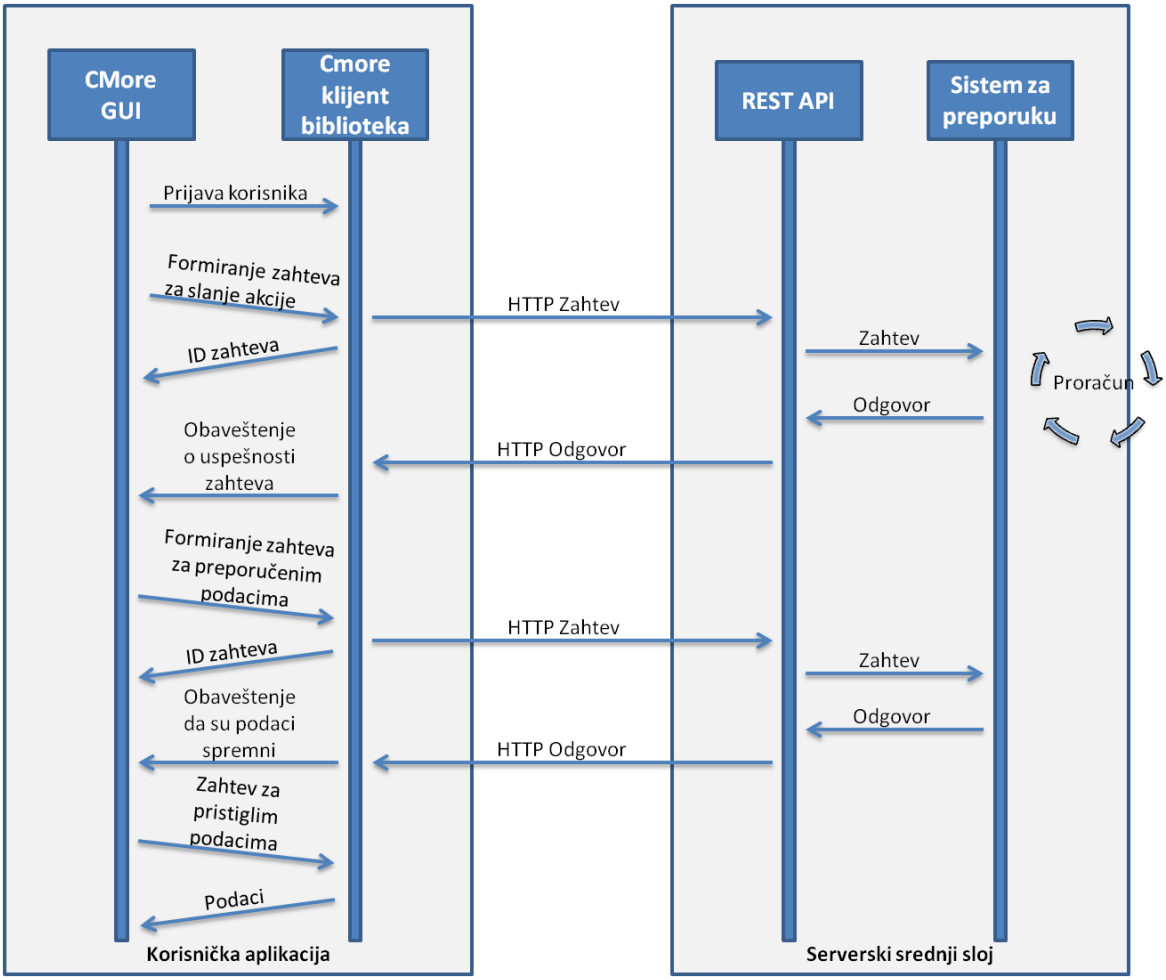
Sistem razlikuje nekoliko tipova korisničkih akcija. Svaka akcija ima svoj tip, identifikator sadržaja nad kojim je akcija načinjena, vrednost, kao i vremensku odrednicu kada je akcija načinjena. Prilikom slanja akcije, obavezna polja koja moraju biti popunjena su tip i identifikator sadržaja nad kojim je načinjena akcija kao i vremenska odrednica, dok vrednost akcije nije obavezna. Spisak akcija koje sistem prepoznaje uz kratak opis za svaku prikazan je u Tabeli 3.2.

Tip akcije	Opis
Pregledan (viewed)	Ova akcija se šalje ukoliko je korisnik pogledao neku internet stranicu koja mu je dostavljena.
Dopada se (liked)	U slučaju da je korisnik označio da mu se neki sadržaj dopada, šalje se akcija ovog tipa.
Ne dopada se (disliked)	Ukoliko korisnik označi da mu se dostavljeni sadržaj ne dopada, šalje se akcija ovog tipa.
Ocenjen (rated)	Korisnik ima mogućnost i da putem sistema oceni dostavljeni sadržaj, tada se sistemu za preporuku šalje akcija ocenjen. Uz ovu akciju, serverskom srednjem sloju se šalje i vrednost ocene (od 1 do 10).

Kupljen (bought)	Tip akcije kupljen, moguć je samo nad sadržajem koji predstavlja video na zahtev. Ova akcija se šalje ukoliko korisnik odluči da kupi neki od dostupnih videa na zahtev.
Pogledan (watched)	Tokom gledanja televizijskog sadržaja, aplikacija ka serverskom srednjem sloju šalje informaciju koju emisiju je korisnik gledao i koliko vremenski dugo. Slanje ovih informacija se vrši putem akcije pogledan (watched), za koju se u polju vrednosti šalje i koliko vremenski dugo je korisnik gledao emisiju izraženo u minutima.

Tabela 3.2 Korisničke akcije

Sam postupak slanja akcija, zasniva se na predhodno opisanom mehanizmu slanja zahteva nad korisničkim profilima. Akcija koja se prosleđuje ka serverskom srednjem sloju, vezuje se za korisnika koji je trenutno prijavljen u sistem. Ukoliko nijedan korisnik nije prijavljen, akcija se vezuje za anonimnog korisnika. Korisničko ime i šifra trenutno aktivog korisnika će, prilikom slanja akcije, biti dodati u zaglavlje HTTP zahteva, u okviru dela za autentifikaciju (engl. *basic access authentification*) [40]. Nakon što formira zahtev za slanje akcije korisnika klijent, kao povratnu vrednost dobija identifikator formiranog zahteva. Putem identifikatora klijent će kasnije biti obavešten o uspešnostu izvršenja akcije, da li je načinjena akcija prihvaćena od strane poslužioca i sistema za preporuku. Sve poslate akcije, prosleđuju se sistemu za preporuku koji ih obrađuje i uzima u obzir tokom budućih zahteva za preporučenim sadržajem. Ukoliko klijent želi da dobavi preporučeni sadržaj, neophodno je da to naglasi popunjavanjem određenog polja filtera. Na osnovu ovog polja, biblioteka će prilikom formiranja URI-a u okviru zahteva za podacima, dodati naznaku da se traži preporučeni sadržaj. Na slici ispod je prikazan dijagram toka prilikom slanja akcija i zahtevanja preporučenog sadržaja.



Slika 3.8 Dijagram toka komunikacije prilikom slanja akcija korisnika i zahteva za preporučenim sadržajem

4. Programsko rešenje

U ovom poglavlju dat je detaljan opis programskog rešenja za integraciju grafičkog korisničkog interfejsa sa hibridnim srednjim slojem. Kao što je već pomenuto, u cilju maksimalnog iskorišćenja hardverskih resursa i lake prenosivosti koda, odlučeno je da biblioteka bude realizovana u isključivo u C programskom jeziku, dok se sprezanje sa višim programskim nivoima vrši preko JNI prilagodne sprege.

4.1 Konfiguraciona datoteka

Konfiguraciona dataoteka je XML dokument koji sadrži podatke o poslužiocu, njegovu IP adresu i prolaz.

Adresa i prolaz koji se nalaze u konfiguracionoj datoteci se mogu programski menjati, putem API-a *CMore klijentske biblioteke*, pozivanjem metoda za izmenu adrese i prolaza. Isto tako, njih je moguće menjati ručno, otvaranjem datoteke u nekom od programa za uređivanje teksta (engl. *editor*) i promenom željenog parametra. Pri tome neophodno je obratiti pažnju da se ne naruši struktura konfiguracione datoteke, jer to može izazvati neispravan dalji rad biblioteke. Izgled konfiguracione datoteke je prikazan na Slici 4.1.

```
<config>
    <serverInfo>
        <serverIp>192.168.237.116</serverIp>
        <serverPort>8080</serverPort>
    </serverInfo>
</config>
```

Slika 4.1 Izgled konfiguracione datoteke

4.2 Format podataka

Podaci koji se dobavljaju od serverskog srednjeg sloja se nalaze u XML formatu čija je struktura strogo definisana. Kako bi biblioteka ispravno funkcionisala, neophodno je da podaci budu striktno struktuirani. Ukoliko dokument koji se dobavi od serverskog srednjeg sloja ne poštuje definisanu strukturu, podaci neće biti ispravno odbrađeni i doći će do greške.

Za svaku vrstu podataka koja se dobavlja postoji definisana struktura. Podaci se dobavljaju u listama, koje su popunjene odgovarajućim elementima (servisima, podacima o emisijama). Svaki od elemenata sadrži određen broj polja, koja mogu biti mandatorna ili opciona. Većina polja je opciona. Opciona polja mogu biti prazna ili ne moraju da postoje u dokumentu. U svakom slučaju, neće se uzeti u obzir. Ukoliko je mandatorno polje prazno ili ne postoji, smatraće se da je čitav dokument nevalidan i svi podaci će biti odbačeni. Struktura dokumenta sa podacima je prikazana na Slici 4.2 i 4.3.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<contentItems>
   <contentItem>
    <contentMetadataRelations/>
     <description>In a zoological park near Lyon, the adventures of two
     gorillas, Digit and Gincko among adoptive parents, Pierre and Eliane
     Thivillon, fleet owners and great lovers of gorillas.</description>
     <id>8794</id>
     <image>/epg/epg_images/vivre_avec_l_animal.jpg</image>
     <isApproved>true</isApproved>
     <language>fra</language>
     <lastUpdate>19:12:51 23/12/2014</lastUpdate>
     <lifeSpan>11:51:04 13/02/2018</lifeSpan>
     <name>Vivre avec l animal</name>
     <parentalRate>16</parentalRate>
     <contentSource>
       <image>http//www.servus.de/image.jpg</image>
       <name>Servus TV</name>
       <path>http://tv-f.akamaihd.net/i/servustvhd_1@51229/master.m3u8</path>
       <type>iptv</type>
     </contentSource>
     <tags>
       <tag>documentary</tag>
     </tags>
     <timeRestriction>
       <duration>2700</duration>
       <endTime>12:05:00 06/01/2015</endTime>
       <startTime>11:20:00 06/01/2015</startTime>
     </timeRestriction>
     <type>live_video</type>
     <uri>http://www.artehd.com/epg/54323</uri>
   </contentItem>
</contentItems>
```

Slika 4.2 Prikaz dokumenta koji sadrži podatke dostavljene od poslužioca

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<contentSources>
  <contentSource>
    <image>/epg/service_images/disney_channel.png</image>
    <name>Disney Channel</name>
    <path>fake://3</path>
    <type>web</type>
  </contentSource>
  <contentSource>
    <image>/epg/service_images/bbc.png</image>
    <name>BBC</name>
    <path>fake://24</path>
    <type>web</type>
  </contentSource>
  <contentSource>
    <image>http://192.168.237.116/epg/service_images/espn.png</image>
    <name>ESPN</name>
    <path>13</path>
    <type>iptv</type>
  </contentSource>
</contentSources>
```

Slika 4.3 Prikaz dokumenta sa podacima o servisima

4.3 Strukture podataka

Budući da je odgovor poslužioca u XML formatu, pre isporuke podatke je predhodno potrebno obraditi i formirati oblik koji je prihvatljiv aplikaciji. Strukture podataka biblioteke sadrže ista polja kao i XML dokumenti koji se vezuju za njih. S obzirom da XML ne mora da sadrži sva moguća polja, to znači da se ne moraju ni sva polja strukture popuniti. U implementaciji to je rešeno tako što se pre upisivanja podataka, strukture inicijalizuju na nevalidne vrednosi, tako da nepopunjena polja unutar dokumenta ostanu nevalidna unutar strukture. Na sledećim slikama prikazane su strukture podataka biblioteke.

```
typedef struct {
    char sourceName[CMORE_SOURCE_NAME_STRING_LENGTH];
    char sourceImageLink[CMORE_LINK_LENGTH];
    char sourcePath[CMORE_LINK_LENGTH];
    char sourceType[CMORE_SOURCE_TYPE_STRING_LENGTH];
} CMORE_Source;
```

Slika 4.4 Struktura podataka koja opisuje jedan servis

```
typedef struct {
     char uri[CMORE_LINK_LENGTH];
     char name[CMORE_CONTENT_NAME_STRING_LENGTH];
     char description[CMORE_CONTENT_DESCRIPTION_STRING_LENGTH];
     char language[CMORE_CONTENT_LANGUAGE_STRING_LENGTH];
     char type[CMORE_CONTENT_TYPE_STRING_LENGTH];
     int32_t contentId;
     uint8_t parentalRate;
     uint32_t duration;
     time_t startTime;
     time_t endTime;
     time_t lastUpdate
     time_t lifeSpan;
     float rating;
     float price;
     bool isUserLabeled;
     CMORE_Source source;
     char contentImageLink[CMORE_LINK_LENGTH];
     CMORE_Tags contentTags;
     CMORE_ContentMetadataRelations contentRelations;
} CMORE_ContentItem;
```

Slika 4.5 Struktura podataka koja opisuje jedan podatak

```
typedef struct {
     char tagsList[CMORE_TAG_MAX_TAGS][CMORE_TAG_MAX_TAG_LENGTH];
     uint8_t numberOfTags;
} CMORE_Tags;
```

Slika 4.6 Struktura podataka koja opisuje jednu oznaku podatka

```
typedef struct {
    char entity[CMORE_RELATION_MAX][CMORE_RELATION_ENTITY_STRING_LENGTH];
    char relation[CMORE_RELATION_MAX][CMORE_RELATION_NAME_STRING_LENGTH];
    char type[CMORE_RELATION_MAX][CMORE_RELATION_TYPE_STRING_LENGTH];
    uint8_t numberOfRelations;
} CMORE_ContentMetadataRelations;
```

Slika 4.7 Struktura podataka koja opisuje jedan metapodatak

Prilikom formiranja zahteva za podacima neophodno je proslediti predhodno popunjen filter. Filter sadrži polja na osnovu kojih se formira URI na koji se šalje zahev ka poslužiocu. Pre početka upotrebe, filter je potrebno inicijalizovati. Tako da se prilikom formiranja zahteva u postupku formiranja URI-a u obzir uzimaju samo polja filtera koja su promenjena nakon inicijalizacije. Izgled struktura podataka koje predstavaljaju filtere je prikazan na sledećim slikama.

```
typedef struct {
   char uri[CMORE_LINK_LENGTH];
   char uriSubPath[CMORE_LINK_LENGTH];
   char name[CMORE_CONTENT_NAME_STRING_LENGTH];
   char language[CMORE_CONTENT_LANGUAGE_STRING_LENGTH];
   char sourceName[CMORE_SOURCE_NAME_STRING_LENGTH];
   char sourceType[CMORE_SOURCE_TYPE_STRING_LENGTH];
   char sourcePath[CMORE_LINK_LENGTH];
   char relationEntities[CMORE_RELATION_MAX][CMORE_RELATION_ENTITY_STRING_LENGTH];
   char relations[CMORE_RELATION_MAX][CMORE_RELATION_NAME_STRING_LENGTH];
   char relationTypes[CMORE_RELATION_MAX][CMORE_RELATION_TYPE_STRING_LENGTH];
   int32_t contentId;
   float minRating;
   int8_t parentalRateMin;
   int8_t parentalRateMax;
   char types[CMORE_CONTENT_TYPES_MAX_NUMBER][CMORE_CONTENT_TYPE_STRING_LENGTH];
   char tags[CMORE_TAG_MAX_TAGS][CMORE_TAG_MAX_TAG_LENGTH];
   char category[CMORE_CATEGORY_STRING_LENGTH];
   int32_t maxDuration;
   int32_t minDuration;
   time_t startsBefore;
   time_t startsAfter;
   time_t endsBefore;
   time_t endsAfter;
   int32_t maxResults;
   int8_t sort[CMORE_SORT_TYPE_MAX];
} CMORE_ContentFilter;
```

Slika 4.8 Struktura podataka koja opisuje filter za podatke

```
typedef struct {
     char sourceName[CMORE_SOURCE_NAME_STRING_LENGTH];
     char sourcePath[CMORE_LINK_LENGTH];
     char sourceType[CMORE_SOURCE_TYPE_STRING_LENGTH];
     int16_t maxResults;
     char sourceImageLink[CMORE_LINK_LENGTH];
} CMORE_SourceFilter;
```

Slika 4.9 Struktura podataka koja opisuje filter za servise

```
typedef struct {
    char username[CMORE_USER_PROFILE_USERNAME_STRING_LENGTH];
    char password[CMORE_USER_PROFILE_PASSWORD_STRING_LENGTH];
    char name[CMORE_USER_PROFILE_ALIAS_STRING_LENGTH];
    char primaryLanguage[CMORE_USER_PROFILE_LANGUAGE_STRING_LENGTH];
    char secondaryLanguage[CMORE_USER_PROFILE_LANGUAGE_STRING_LENGTH];
    char country[CMORE_USER_PROFILE_COUNTRY_STRING_LENGTH];
} CMORE_UserProfile;
```

Slika 4.10 Struktura koja opisueje jedan korisnički profil

```
typedef struct {
     char username[CMORE_USER_PROFILE_USERNAME_STRING_LENGTH];
     char password[CMORE_USER_PROFILE_PASSWORD_STRING_LENGTH];
     char service[CMORE_USER_PROFILE_SERVICE_STRING_LENGTH];
} CMORE_ExternalCredential;
```

Slika 4.11 Struktura koja opisuje jedan nalog korisnika na drugom Internet servisu.

Ukoliko tokom rada dođe do greške biblioteka će o tome obavestiti klijenta, slanjem koda greške. Izgled enumeracije sa spiskom mogućih grešaka je prikazan na slici 4.10.

```
typedef enum {
     CMORE_NO_ERROR
     CMORE_ERROR_NO_DATA_RECEIVED
     CMORE_ERROR_BAD_ARGUMENT
     CMORE_ERROR_NOT_INITIALIZED
     CMORE_ERROR_ALREADY_INITIALIZED,
     CMORE_ERROR_UNINITIALISED_HANDLER
     CMORE_ERROR_BAD_CONTENT_FILTER
     CMORE_ERROR_TIME_CONVERSION
     CMORE_ERROR_FULL_QUEUE,
     CMORE_ERROR_AUTHENTICATION_FAILED,
     CMORE_ERROR_UNDEFINED
} CMORE_ErrorCode;
```

Slika 4.12 Izled enumeracije sa kodovima grešaka

Prilikom prijave klijenta, potrebno je proslediti pokazivač na funkciju koja će biti pozvana kada podaci budu spremni. Putem ove funkcije klijent će biti obavešten da li su podaci uspešno dobavljeni, o broju pristiglih podataka i identifikatoru zahteva za koji je pristiglo obaveštenje. Prototip funkcije je prikazan u Tabeli 4.1.

Prototip funkcije	**typedef void(*CMORE_Callback_t)(CMORE_CallbackType eventType, uint32_t requestID, uint32_t numberOfReceivedData)**
Parametri funkcije	eventType – tip obaveštenja koje se šalje.
	requestID – identifikator zahteva za koji se šalje obaveštenje.
	numberOfReceivedData – broj pristiglih podataka.

Tabela 4.1 Prototip funkcije za slanje obaveštenja o zahtevima klijentu

Takođe, prilikom prijave klijent mora proslediti pokazivač na funkciju putem koje će klijent biti obavešten o uspešnosti zahteva nad korisničkim profilima. Putem ove funkcije klijent će biti obavešten da li je zahtev bio uspešan, identifikatoru zahteva, i ukoliko se radi o zahtevu za

dobavljenje korisničkih naloga na drugim servisima, o broju korisničkih naloga. Izgled funkcije prikazan je u Tabeli 4.2.

Prototip funkcije	**typedef void (*CMORE_UserCallback_t)(CMORE_UserRequestType callbackType, uint32_t requestID, bool isRequestSuccessful, uint32_t numberOfCredentials)**
Parametri funkcije	callbackType – tip obaveštenja koje se šalje.
	requestID – identifikator zahteva za koji se šalje obaveštenje.
	isRequestSuccessful – da li je zahtev uspešno prihvaćen.
	numberOfCredentials – broj pristiglih podataka.

Tabela 4.2 Prototip funkcije za slanje obaveštenja o zahtevima nad korisničkim profilima

4.4 Funkcije

Bilioteka poseduje ograničen skup funkcija. Cilj je bio da se formira što jednostavnija sprega za razvoj programske podrške. Važno je naglasiti da su sve funkcije sinhrone i da im je obezbeđen siguran pristup iz više programskh niti.

4.4.1 Klijentska biblioteka

U nastavku će biti tabelarno prikazane funkcije aplikativne programske sprege klijentske biblioteke.

4.4.1.1 Komunikacija sa poslužiocem

Prototip funkcije	CMORE_ErrorCode CMORE_Init()
Opis funkcije	Inicijalizuje biblioteku za dalju upotrebu. Ovu funkciju je nepohodno pozvati pre početka rada sa bibliotekom, kako bi se zauzeli i inicijalizovali potrebni resursi.
Parametri funkcije	Nema parametara
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_Deinit()
Opis funkcije	Deinicijalizuje biblioteku. Ovu funkciju je nepohodno pozvati na kraju rada sa bibliotekom, kako bi se oslobodili svi predhodno zauzeti resursi.
Parametri funkcije	Nema parametara
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_RegisterClient(CMORE_Callback_t callbackFunc, CMORE_ClientHandle *clientHandle)
Opis funkcije	Prijavljuje klijenta za upotrebu biblioteke.
Parametri funkcije	callbackFunc – pokazivač na funkciju koja će biti pozvana kada bude potrebno klijentu poslati obaveštenje da su podaci za predhodno formirani zahtev spremni.
	clientHandle – klijentki rukovalac koji predstavlja povratnu vrednost nakon prijave. Ovaj rukovalac se koristi kao ulazni parametar pri pozivu ostalih funkcija biblioteke.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_UnregisterClient(const CMORE_ClientHandle clientHandle)
Opis funkcije	Odjavljuje klijenta. Poziva se kada klijent želi da završi sa upotrebom biblioteke.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_CreateContentItemRequest(CMORE_ClientHandle handle, const CMORE_ContentFilter* contentFilter, int32_t* requestId)
Opis funkcije	Formiranje zahteva za podacima.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	contentFilter – filter kojim se specificiraju podaci koji se zahtevaju od poslužioca. Na osnovu popunjenih polja ovog filtera, formira se URI ka kojem se upućuje zahtev za podacima.
	requestId – ID formiranog zahteva. Ovaj povratni parametar, koji se koristi prilikom poziva funkcija za dobavljanje podataka ili brisanje zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_CreateSourcesRequest(CMORE_ClientHandle handle, const CMORE_SourceFilter* sourceFilter, int32_t *requestId)
Opis funkcije	Formiranje zahteva za podacima o servisima.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	sourceFilter – filter kojim se specificiraju podaci o servisima koji se zahtevaju od poslužioca. Na osnovu popunjenih polja ovog filtera, formira se URI ka kojem se upućuje zahtev za podacima.
	requestId – ID formiranog zahteva. Ovaj povratni parametar, koji se koristi prilikom poziva funkcija za dobavljanje podataka o servisima ili brisanje zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_DeleteRequest(CMORE_ClientHandle handle, int32_t requestId)
Opis funkcije	Brisanje predhodno formiranog zahteva.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	requestId – ID formiranog zahteva. Ovaj povratni parametar, koji se koristi prilikom poziva funkcija za dobavljanje podataka o servisima ili brisanje zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_ContentFilterInit(CMORE_ContentFilter *contentFilter)
Opis funkcije	Inicijalizacija filtera za formiranje zahteva. Popunjava polja filtera inicijalnim vrednostima i vraća ga klijentu za dalju upotrebu.
Parametri funkcije	contentFilter – filter koji će se kasnije koristiti prilikom formiranja zahteva za podacima.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_SourceFilterInit(CMORE_SourceFilter *sourceFilter)
Opis funkcije	Inicijalizacija filtera za formiranje zahteva za podacima o servisima. Popunjava polja filtera inicijalnim vrednostima i vraća ga klijentu za dalju upotrebu.
Parametri funkcije	sourceFilter – filter koji će se kasnije koristiti prilikom formiranja zahteva za podacima servisima.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetContentItems(const CMORE_ClientHandle handle, int32_t requestId, uint32_t startIndex, uint32_t stopIndex, CMORE_ContentItem *items)
Opis funkcije	Dobavaljanje određene količine podataka.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	requestId – ID predhodno formiranog zahteva.
	startIndex – Indeks prvog podatka koji se dobavlja.
	stopIndex – Index poslednjeg podatka koji se dobavlja.
	items – podaci, povratni parametar funkcije.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetSources(const CMORE_ClientHandle handle, int32_t requestId, uint32_t startIndex, uint32_t stopIndex, CMORE_Source *sources)
Opis funkcije	Dobavaljanje određene količine podataka o servisima.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	requestId – ID predhodno formiranog zahteva za servisima.
	startIndex – Indeks prvog podatka o servisima koji se dobavlja.
	stopIndex – Index poslednjeg podatka o servisima koji se dobavlja.
	items – podaci o servisima, povratni parametar funkcije.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_SetServerIpAddress(CMORE_ClientHandle handle, const char *ipAddress)
Opis funkcije	Postavljanje nove IP adrese poslužioca.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	ipAddress – nova IP adresa poslužioca koja će zameniti postojeću.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_SetServerPort(CMORE_ClientHandle handle, const char *port)
Opis funkcije	Postavljanje novog prolaza poslužioca.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	port – novi prolaz poslužioca koja će zameniti postojeći.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetServerIpAddress(CMORE_ClientHandle handle, char *ipAddress)
Opis funkcije	Dobavljanje IP adrese poslužioca.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	ipAddress – povratni parametar koji sadrži trenutnu adresu poslužioca.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetServerPort(CMORE_ClientHandle handle, char *port)
Opis funkcije	Dobavljanje prolaza poslužioca.
Parametri funkcije	clientHandle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	ipAddress – povratni parametar koji sadrži trenutni prolaz poslužioca.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

4.4.1.2 Rukovanje korisničkim profilima i akcijama

U nastavku je prikazan spisak funkcija aplikativne programske putem kojih se rukuje korisničkim profilima.

Prototip funkcije	CMORE_ErrorCode CMORE_CreateUserProfile(CMORE_ClientHandle handle, const CMORE_UserProfile* user, int32_t* requestId)
Opis funkcije	Dobavljanje IP adrese poslužioca.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	user – novi korisnički profil koji se formira.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_DeleteUserProfile(CMORE_ClientHandle handle, const char* username, const char* password, int32_t* requestId)
Opis funkcije	Brisanje prednodno formiranog korisnika.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – korisničko ime profila koji se briše.
	password – šifra profila koji se briše.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_UpdateUserProfile(CMORE_ClientHandle handle, const CMORE_UserProfile* userProfile, int32_t* requestId)
Opis funkcije	Izmena postojećeg korisničkog profila.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	userProfile – izmenjeni korisnički profil koji je potrebno ažurirati.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_UserLogin(CMORE_ClientHandle handle, const char* username, const char* password, int32_t* requestId)
Opis funkcije	Prijava korisnika u sistem.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – korisničko ime.
	password – šifra korisnika.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_UserLogout(CMORE_ClientHandle handle, const char* username, const char* password, int32_t* requestId)
Opis funkcije	Odjava korisnika iz sistema.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – korisničko ime.
	password – šifra korisnika.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetActiveUser(CMORE_UserProfile* activeUser)
Opis funkcije	Dobavljanje trenutno prijavljenog korisničkog profila.
Parametri funkcije	activeUser – profil trenutno prijavljanog korisnika.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_AddUserProfileExternalCredential(CMORE_ClientHandle handle, const char* username, const char* password, const CMORE_ExternalCredential* externalCredential, int32_t* requestId)
Opis funkcije	Dodavanje naloga sa drugih Internet servisa.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – ime korisničkog profila za koji se dodaje novi nalog.
	password – šifra korisničkog profila za koji se dodaje novi nalog.
	externalCredential – korisnički nalog na nekom Internet servisu.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_DeleteUserProfileExternalCredential(CMORE_ClientHandle handle, const char* username, const char* password, const CMORE_ExternalCredential* externalCredential, int32_t* requestId)
Opis funkcije	Brisanje postojećeg korisničkog naloga na drugom Internet servisu.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – ime korisničkog profila čiji se nalog briše.
	password – šifra korisničkog profila čiji se nalog briše.
	externalCredential – korisnički nalog koji se briše.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_DeleteUserProfileExternalCredential(CMORE_ClientHandle handle, const char* username, const char* password, const CMORE_ExternalCredential* externalCredential, int32_t* requestId)
Opis funkcije	Brisanje postojećeg korisničkog naloga na drugom Internet servisu.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – ime korisničkog profila čiji se nalog briše.
	password – šifra korisničkog profila čiji se nalog briše.
	externalCredential – korisnički nalog koji se briše.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_CreateUserExternalCredentialsRequest(CMORE_ClientHandle handle, const char* username, const char* password, int32_t* requestId)
Opis funkcije	Formiranje zahteva za korisničkim nalozima nekog korisnika.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	username – ime korisničkog profila čiji se korisnički nalozi zahtevaju.
	password – šifra korisničkog profila čiji se korisnički nalozi zahtevaju.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_GetListOfUserCredentials(CMORE_ClientHandle handle, int32_t requestId, uint32_t startIndex, uint32_t stopIndex, CMORE_ExternalCredential* externalCredentials)
Opis funkcije	Dobavljanje liste sa korisničkim nalozima nekog korisnika.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	requestId – identifikator formiranog zahteva.
	startIndex – index prvog korisničkog naloga koji se dobavlja.
	stopIndex – index poslednjeg korisničkog naloga koji se dobavlja.
	externalCredentials - lista korisničkih naloga.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_DeleteUserRequest(CMORE_ClientHandle handle, int32_t requestId)
Opis funkcije	Brisanje predhodno formiranog korisničkog zahteva.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

Prototip funkcije	CMORE_ErrorCode CMORE_SubmitUserAction(CMORE_ClientHandle handle, CMORE_UserAction* userAction, int32_t* requestId)
Opis funkcije	Slanje akcije korisnika.
Parametri funkcije	handle – klijentki rukovalac koji je dobavljen prijavom, pozivom funkcije CMORE_RegisterClient.
	userAction – akcija korisnika koja se šalje.
	requestId – identifikator formiranog zahteva.
Povratna vrednost funkcije	Ukoliko pri izvršavanju funkcije nije došlo do greške funkcija će vratiti CMORE_NO_ERROR , u suprotnom biće vraćen kod greške.

4.4.2 JNI prilagodni sloj

JNI prilagodni sloj predstavlja spregu između *CMore klijent biblioteke* i korisničke aplikacije. Putem ovog sloja aplikacija direktno koristi API *CMore klijent biblioteke*. Ovaj prilogodni sloj sastoji se iz skupa funkcija koje grupišu sve funkcionalnosti biblioteke i omogućavaju potpunu kontrolu nad njom. Kako ove funkcije preslikavaju API *CMore klijent biblioteke* u nastavku će samo u okviru jedne tabele biti prikazan spisak svih funckija JNI prilagodnog sloja uz kratak opis za svaku od njih.

Naziv funkcije	*Opis funkcije*
initClient	Inicijalizacija klijentske biblioteke i prijava kllijenta. Ova funkcija se potiva na početku korištenja, pre poziva bilo koje druge funkcije. Njenim pozivom se zauzimaju neophodni resursi neophodni za dalji rad sa bibliotekom.
deinitClient	Deinicijalizacija klijentske biblioteke. Ova funkcija se poziva na kraju korištenja, pre poziva bilo koje druge funkcije. Njenim pozivom se oslobađaju resursi zauzeti tokom rada sa bibliotekom.
createCMoreContentRequest	Formiranje zahteva za podacima
deleteRequest	Brisanje predhodno formiranih zahteva. Ova funkcija se poziva nakon upotrebe podataka, kada oni više nisu potrebni. Pozivom funkcije oslobađaju se resursi, za formiranje novih zahteva.
getCMoreContentItems	Dobavljanje određene količine podataka. Poziva se nakon pristiglog obaveštenja da su podaci spremni.
createSourceRequest	Formiranje zahteva za servisima.
getContentSources	Dobavljanje određene količine podataka o servisima. Poziva se nakon pristiglog obaveštenja da su podaci spremni.
setServerIpAddress	Podešavanje IP adrese poslužioca.
setServerPort	Podešavanje prolaza poslužioca.
getServerIpAddress	Dobavljanje trenutno podešene IP adrese poslužioca.
getServerPort	Dobavljanje trenutno podešenog prolaza poslužioca.
createUserProfile	Formiranje novog korisničkog profila.
deleteUserProfile	Brisanje postojećeg korisničkog profila.
updateUserProfile	Izmena postojećeg korisničkog profila.
userLogin	Prijava korisnika u sistem.
userLogout	Odjava korisnika iz sistema.
addUserExternalCredential	Dodavanje korisničkih naloga sa drugih Interner servisa.

deleteUserExternalCredential	Brisanje korisničkog naloga nekog korisnika.
createUserCredentialsRequest	Formiranje zahteva za korisničkim nalozima nekog korisničkog profila.
getListOfUserExternalCredentials	Dobavljanje liste korisničkih naloga, na drugim Internet servisima, za nekog korisnika.
deleteUserRequest	Brisanje predhodno formiranog korisničkog zahteva.
submitUserAction	Slanje akcije korisnika na obradu poslužiocu.

Tabela 4.3 Pregled funkcija JNI prilagodnog sloja

4.5 Java klase i interfejsi

U ovom poglavlju biće nabrojane klase i interfejsi posredstvom kojih korisnička aplikacija komunicira sa *CMore klijent bibliotekom* odnosno sa serverskim srednjim slojem. Takođe, dat je prikaz klasa i interfejsa posredstvom kojih se obavlja komunikacija sa srednjim slojem uređaja. Treba napomenuti da za komunikaciju sa srednjim slojem uređaja i pristup DTV funkcionalnostima koristi već postojeće rešenje [19]. Ovo rešenje predstavlja *Android* servis koji posredstvom interfejsa i odgovarajućeg *Java* API-ja obezbeđuje DTV funkcionalnosti korisničkim aplikacijama.

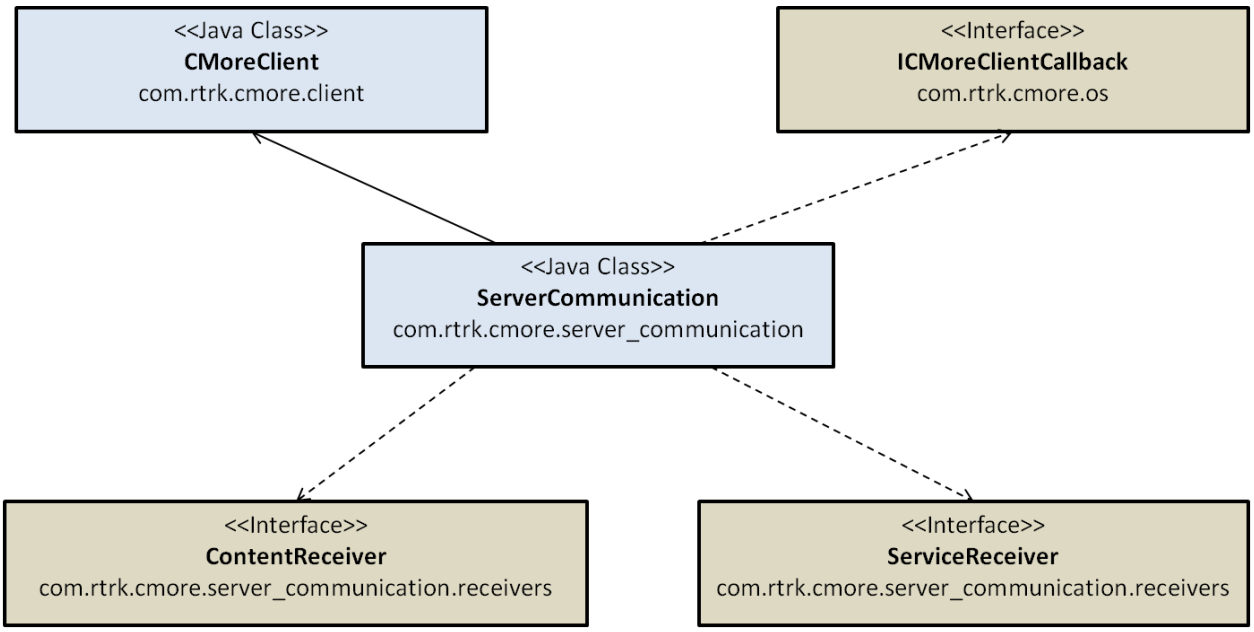
4.5.1 Klase i interfejsi za komunikaciju sa serverskim srednjim slojem

Klase i interfejsi posredstvom kojih korisnička aplikacija komunicira sa serverskim srednjim slojem su:

- *CMoreClient* klasa – posredstvom instance ove klase pozivaju se funkcije JNI prilagodnog sloja *CMore klijent biblioteke*,
- *ServerCommunication* – posredstvom instance ove klase korisnička aplikacija poziva funkcije *CMore klijent biblioteke* i tako obavlja komunikaciju sa serverskim srednjim slojem. Poziv funkcija biblioteke se obavlja posredstvom instance *CMoreClient* klase,
- *ICMoreClientCallback* je interfejs posredstvom kojeg se korisničkoj aplikaciji šalju obaveštenja o uspešnosti formiranih zahteva. Ovaj interfejs implementira *ServerCommunication* klasa,

- *ContentReceiver* – interfejs posredstvom kojeg *ServerCommunication* klasa obaveštava korisničku aplikaciju o uspešnosti zahteva i prosleđuje joj pristigle podatke. Instanca ovog interfejsa se prosleđuje prilikom formiranja zahteva,
- *ServiceReceiver* – interfejs posredstvom kojeg *ServerCommunication* klasa obaveštava korisničku aplikaciju o uspešnosti zahteva za servisima i prosleđuje joj pristigle podatke. Instanca ovog interfejsa se prosleđuje prilikom formiranja zahteva.

Na Slici 4.13. je prikazan dijagram klasa i interfejsa posredstvom kojih se obavlja komunikacija sa serverskim srednjim slojem.



Slika 4.13 Dijagram klasa za posredstvom kojih se obavlja komunikaciju sa serverskim srednjim slojem

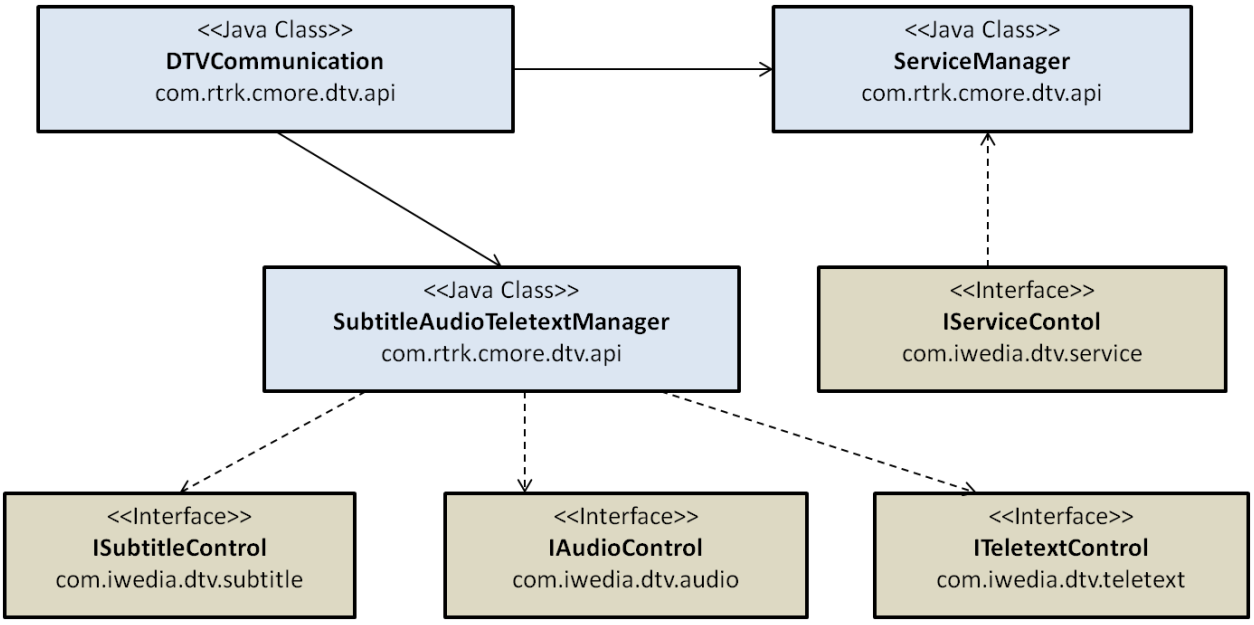
4.5.2 Klase i interfejsi za komunikaciju sa srednjim slojem uređaja

Klase i interfejsi posredstvom kojih se obavlja komunikacija sa srednjim slojem uređaja su:

- *DTVCommunication* – posredstvom instance ove klase korisnička aplikacija komunicira sa srednjim slojem uređaja i pristupa DTV funkcionalnostima,
- *ServiceManager* – posredstvom instance ove klase se vrši upravljanje listom servisa,
- *SubtitleAudioTeletextManager* – posredstvom instance ove klase se vrši upravljanje subtitle, audio i teletext trakama,
- *IServiceControl* – interfejs posredstvom kojeg klasa *ServiceManager* upravlja listom servisa,

- *ISubtitleControl* – interfejs posredstvom kojeg klasa *SubtitleAudioTeletextManager* upravalja subtitle trakama,
- *IAudioControl* – interfajs posredstvom kojeg klasa *SubtitleAudioTeletextManager* upravlja audio trakama,
- *ITeletextControl* – interfejs posredstvom kojeg klasa *SubtitleAudioTeletextManager* upravlja teletext trakama.

Na slici ispod je prikazan dijagram klasa i interfejsa za komunikaciju sa srednjim slojem uređaja.



Slika 4.14 Dijagram klasa i interfejsa posredstvom kojih se obavlja komunikacija sa srednjim slojem uređaja

5. Ispitivanje i rezultati

U ovom poglavlju dat je uvid u mehanizme za verifikaciju datog rešenja. Rezultati koji su dobijeni u procesu verifikacije su takođe predstavljeni u ovom poglavlju. U cilju potvrđivanja funkcionalnosti realizovanog rešenja, sprovedeno je nekoliko ispitivanja:

- Ispitivanje brzine izvršavanja
- Ispitivanje zauzeća radne memorije
- Ispitivanje u okviru realizovane grafičke korisničke sprege

5.1 Ispitivanje brzine izvršavanja

U okviru ovog ispitivanja vršeno je merenje vremena proteklog od trenutka formiranja zahteva za podacima do trenutka kada podaci budu spremni za isporuku klijentu. Kako ukupno vreme predstavlja sumu vremena potrebnog serverskom srednjem sloju da isporuči tražene podatke i vremena potrebnog biblioteci da pristigle podatke obradi i izgeneriše odgovorajući format, prihvatljiv klijentu, merenje je sprovedeno tako da se poredi ukupno vreme sa vremenom obrade pristiglih podataka. Takođe, uzeta je u obzir i različita količina podataka koja se zahteva, tako da su merenja sprovedena nad zahtevima za jedan, deset, stotinu i pet stotina podataka.. Rezultati merenja su prikazani u Tabelama 5.1, 5.2, 5.3, 5.4.

R. br.	Vreme izršavanja funkcije	Vreme obrade podataka	Procenat vremena odrade podataka u izvršavanju
1.	17.43 ms	0.44 ms	~2,6%
2.	18.463 ms	0,45 ms	~2,4%
3.	18.37 ms	0.47 ms	~2,5%
prosek	18.06 ms	0.45 ms	~2,5%

Tabela 5.1 Merenja vremena izvršavanja zahtreva za jednim podatkom

R. br.	Vreme izršavanja funkcije	Vreme obrade podataka	Procenat vremena odrade podataka u izvršavanju
1.	49.73 ms	2.65 ms	~5,3%
2.	46.79 ms	2.88 ms	~6,1%
3.	46.18 ms	2.69 ms	~5,8%
prosek	47.56 ms	2,74 ms	~5,7%

Tabela 5.2 Merenja vremena izvršavanja zahteva za 10 podataka

R. br.	Vreme izršavanja funkcije	Vreme obrade podataka	Procenat vremena odrade podataka u izvršavanju
1.	329.84 ms	24.97 ms	~7,6%
2.	308.31 ms	25.65 ms	~8,3%
3.	313.22 ms	22.05 ms	~7,1%
prosek	317.12 ms	24.22 ms	~7,7%

Tabela 5.3 Merenja vremena izvršavanja zahteva za 100 podataka

R. br.	Vreme izršavanja funkcije	Vreme obrade podataka	Procenat vremena odrade podataka u izvršavanju
1.	1289.40 ms	73.32 ms	~5,7%
2.	1279.55 ms	64,84 ms	~5,1%
3.	1281.55 ms	65,19 ms	~5,1%
prosek	1281.5 ms	67.78 ms	~5,3%

Tabela 5.4 Merenja vremena izvršavanja zahteva za 500 podataka

Na osnovu ispitivanja se može zaključiti da obrada pristiglih podataka, ne utroši više od 10% vremena izvršavanja same funkcije za formiranje zahteva za podacima. Takođe može se zaključiti da je obrada veće količine podataka efikasnija, prosečno vreme potrebno za obradu pristiglih podataka se smanjuje sa povećanjem broja podataka po jednom zahtevu.

Treba naglasiti da za ispitivanje korišćena mrežna infrastruktura koja je omogućila minimalno vreme potrebno za protok podataka. Tokom ispitivanja klijentski uređaj i serverski srednji sloj su bili smešteni u okviru iste lokalne mreže, što je u velikoj meri uticalo na vreme potrebno za prenos podataka. Upotreba drugačije mrežne infrastrukture, u kojoj se klijent i serverski srednji ne bi nalazili u istoj lokalnoj mreži, dovela bi do povećanja vremena potrebnog za protok podataka a samim tim i vremena izvršavanje funkcije. U tom slučaju, obrada pristiglih podataka bi bila još efikasnija, s obzirom da bi se procenat vremena potrebnog za obradu podataka bio manji.

Što se tiče same korisničke aplikacije, najvažnije da su izmerena vremena prihvatljiva i sa aspekta korisničkog doživljaja i da su u skladu sa očekivanim [41][42][43].

Prethodna merenja odnose se samo na prijem podataka u okviru biblioteke. Međutim nakon toga podatke je potrebno proslediti klijentu, u našem slučaju korisničkoj aplikaciji. Kada podaci budu spremni za isporuku, biblioteka putem povratne sprege, šalje obaveštenje aplikaciji o tome, nakon čega aplikacija može da traži određenu količinu podataka. Treba uzeti u obzir i to da se komunikacija sa aplikacijom vrši putem JNI sprege. Sledeća grupa merenja odnosi se na vreme potrebno da nakon slanja zahteva za dobavljanje podatka od strane aplikacije, traženi podaci budu isporučeni. U okviru ovog ispitivanja, vršeno je merenje vremena potrebnog za isporuku različite količine podataka, kao i poređenje ukupnog vremena sa vremenom potrošenim u JNI-u. Rezultati meranja su prikazani u Tabelama 5.5, 5.6, 5.7 i 5.8.

R. br.	Vreme potrebno da podaci budu dostavljeni aplikaciji	Vreme obrade u JNI-u	Procenat vremena odrade podataka u JNI-u
1.	19.79 ms	0.71 ms	~3,6%
2.	22.65 ms	0.91 ms	~4,03%
3.	19.80 ms	0.80 ms	~4,08%
prosek	20.75 ms	0,81 ms	~3,9%

Tabela 5.5 Merenje vremena dobavaljanja jednog podatka

R. br.	Vreme potrebno da podaci budu dostavljeni aplikaciji	Vreme obrade u JNI-u	Procenat vremena odrade podataka u JNI-u
1.	59.24 ms	9.01 ms	~15,2%
2.	54.78 ms	8.75 ms	~15,9%
3.	51.60 ms	7.05 ms	~13,6%
prosek	55.21 ms	8.27 ms	~14.9%

Tabela 5.6 Merenje vremena dobavljanja 10 podataka

R. br.	Vreme potrebno da podaci budu dostavljeni aplikaciji	Vreme obrade u JNI-u	Procenat vremena odrade podataka u JNI-u
1.	382.76 ms	68.80 ms	~17,9%
2.	396.79 ms	69.60 ms	~17,5%
3.	368.46 ms	53.68 ms	~14,5%
prosek	382.67 ms	64.02 ms	~16,6%

Tabela 5.7 Merenje vremena dobavljanja 100 podataka

R. br.	Vreme potrebno da podaci budu dostavljeni aplikaciji	Vreme obrade u JNI-u	Procenat vremena odrade podataka u JNI-u
1.	1654.33 ms	290,11 ms	~17,5%
2.	1724.53 ms	284,66 ms	~16,5%
3.	1691.76 ms	306.50 ms	~18,1%
prosek	382.67 ms	64.02 ms	~17,4%

Tabela 5.8 Merenje vremena dobavljanja 500 podataka

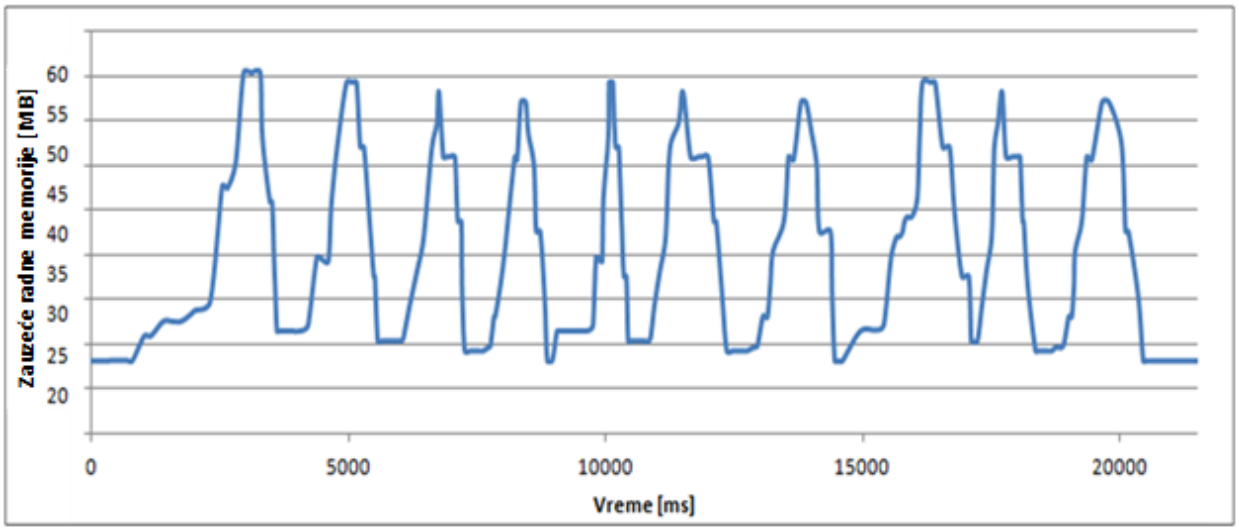
Ovim ispitivanjem je pokazano da vreme obrade podataka u JNI prilagodnom sloju, ne oduzima više od 20% ukupnog vremena potrebnog za dobavljanje podataka. Veoma je značajno da ovo vreme bude što manje, kako bi ukupno vreme ostalo u prihvatljivim okvirima. Na osnovu merenja može se zaključiti da se sa porastom količine podataka koja se dobavlja, smanjuje prosečno vreme potrebno za dobavljanje jednog podatka. I ova vremena su u skladu sa očekivanim vrednostima[41][42][43].

5.2 Ispitivanje zauzeća radne memorije

S obzirom da je biblioteka implementirana u nativnom kodu, od izuzetne vaznosti za funkcionalnost čitavog sistema je da nema curenja memorije. Ovim ispitivanjem se vrši provera zauzeća radne memorije od strane *CMore klijent biblioteke*.

Za nadgledanje zauzeča radne memorije iskorištena je postojeća *Android* funkcija *getNativeHeapAllocatedSize*. Ova fukcija vraća ukupnu količinu radne memorije zauzete od strane native dela izraženo u bajtima.

Ispitivanje je sprovedeno tako što je formirano 10 uzastopnih zahteva za 500 podataka i praćeno zauzeće radne memorije pozivima pomenute funkcije. Na sledećem grafiku su prikazani rezultati merenja.



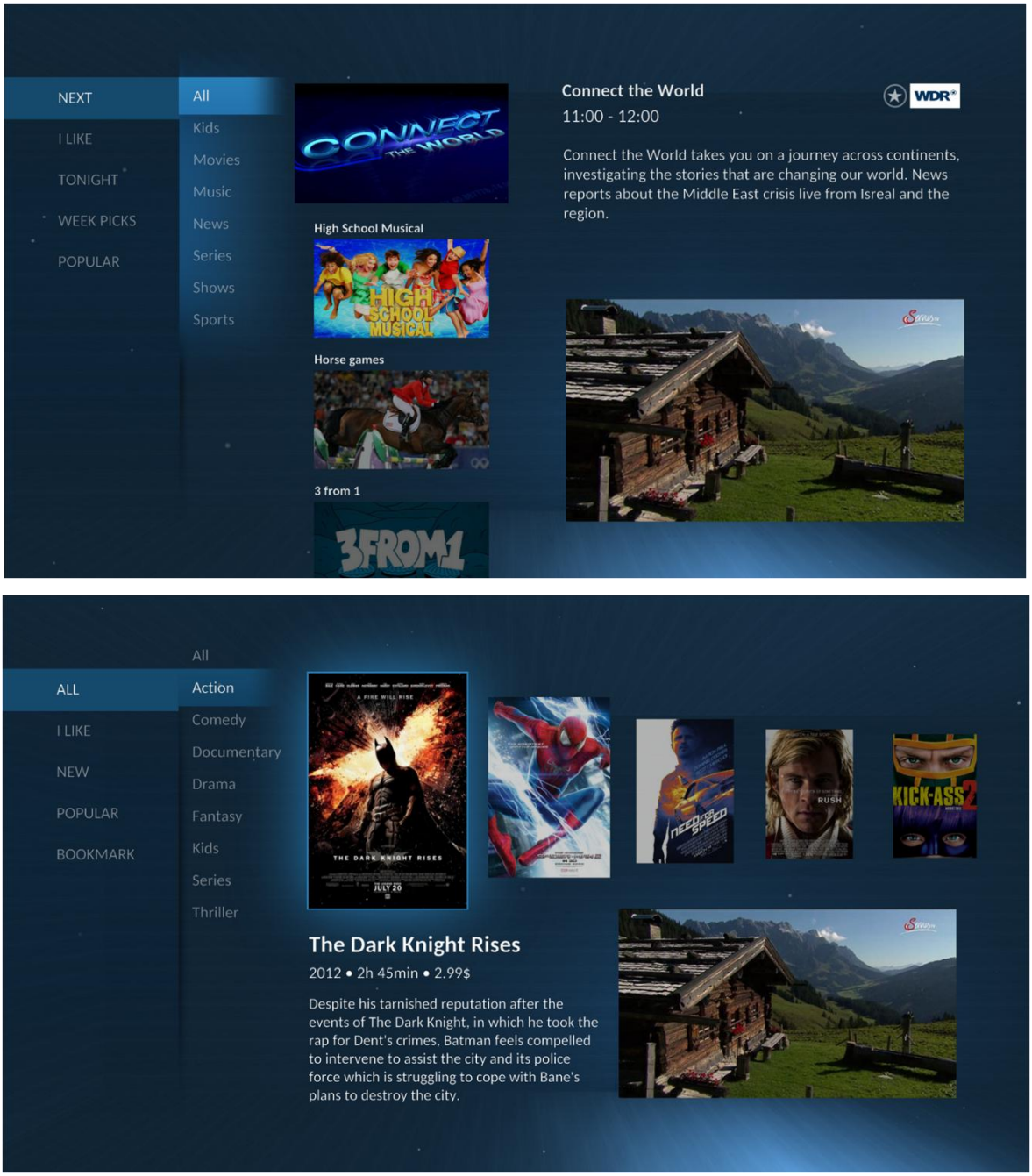
Slika 5.1 Grafik zauzeća radne memorije prilikom rada biblioteke

Sa grafika se mogu uočiti deset naglih promena zauzeća memorije. Ove promene odgovaraju formiranim zahtevima za podacima. Svako formiranje zahteva dovodi do naglog povećanja zauzeća memorije, što je rezultat obrade pristiglih podataka. Nakon obrade podataka, oslobađaju se predhodno zauzeti resursi i zauzeće radne memorije se vraća na ustaljenu vrednost, što se može videti na grafiku.

Na osnovu ovog ispitivanja može se zaključiti da realizovana programska podrška pri izvršavanju troši između 30 MB i 60 MB radne memorije. Ovo je zanemarljivo malo u odnosu na raspoloživu memoriju na *Android* platformi. Takođe, može se zaključiti da nema nepoželjnog efekta curenja memorije, u suprotnom bi funkcionalnost same korisničke aplikacije bila ugrožena.

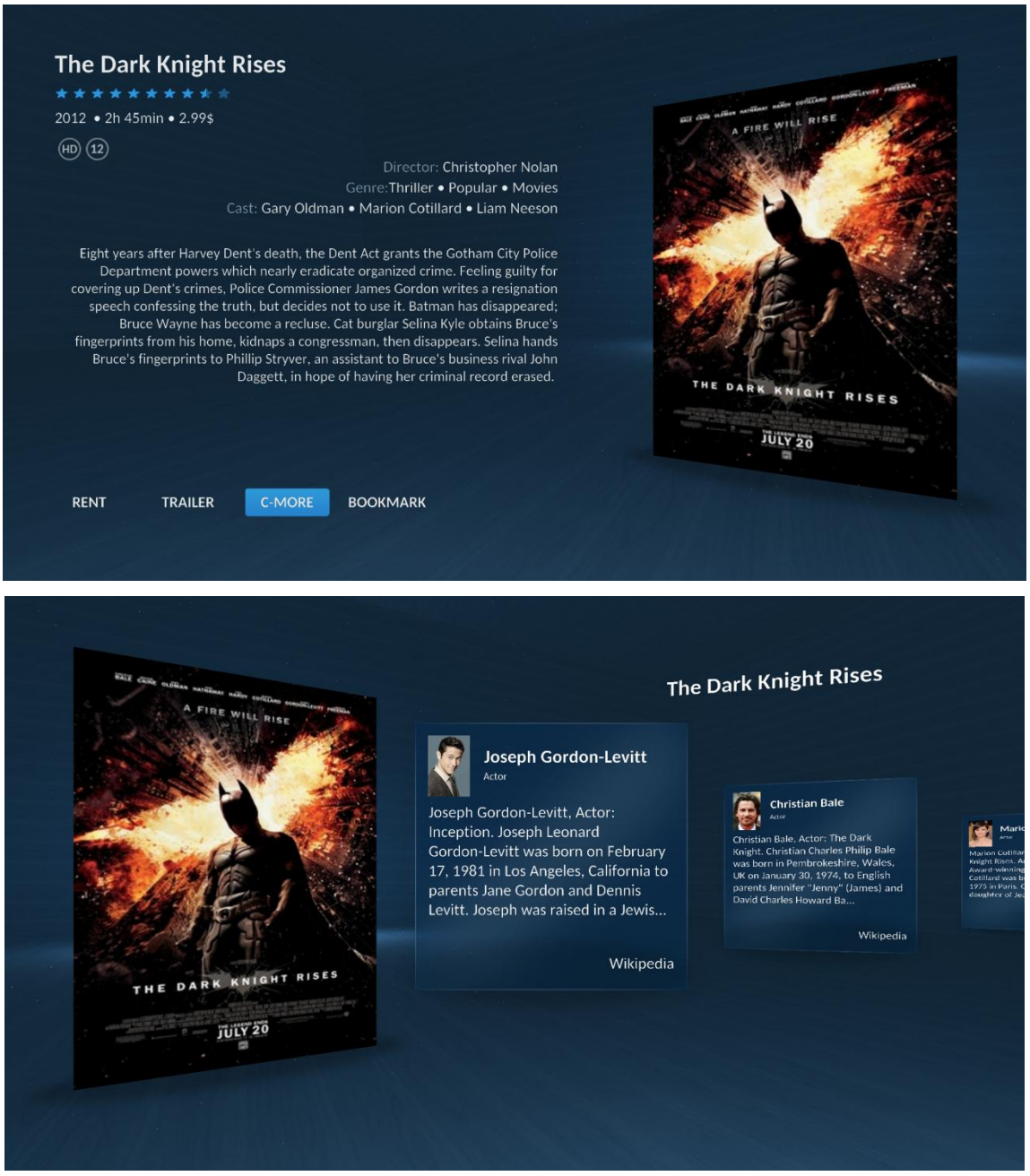
5.3 Ispitivanje u okviru TV aplikacije

Realizovana programska podrška je korišćena u okviru TV aplikacije. Aplikacija putem *CMore klijent biblioteke*, dobavlja podatke sa udaljenog serverskog srednjeg sloja i prikazuje ih korisniku, što se može videti na Slikama 5.2 i 5.3. Na ovaj način je u realnom okruženje testirana funkcionalnost realizovanog programskog rešenja. Takođe, može se zaključiti da li je vreme odziva prihvatljivo sa aspekta korisničkog iskustva ili nije.



Slika 5.2 CMore aplikacija – prikaz pristiglih EPG i VoD podataka

Ovim ispitivanjem je zaključeno da je odziv zadovoljavajući u odnosu na [41][42][43]. Ovo ispitivanje je bilo u velikoj meri zavisno od poslužioca na kome su se nalazili podaci, kao i od zagušenja mreže. Oba pomenuta faktora mnogo više utiču na odziv aplikacije, nego realizovana biblioteka, što je bio jedan od zahteva.



Slika 5.3 CMore aplikacija – prikaz dodatnog sadržaja

6. Zaključak

Jedan od ključnih problema prilikom razvoja modernih grafičkih korisničkih interfejsa za STB i TV uređaje, predstavlja heterogenost gradivnih komponenti softvera uređaja. Neophodna je komponenta koja će sve povezati u jednu skladnu i pre svega funkcionalnu celinu. Budući da je grafički korisnički interfejs sprega korinika sa uređajem, i jedino što korisnik vidi i posredstvom čega rukuje uređajem, brz odziv i funkcionalnost sistema su od izuzetne važnosti. Zbog toga obezbeđivanje odgovarajućih metoda za preuzimanje sadržaja uz minimalno kašnjenje predstavlja osnovni zadatak prilikom realizacije softverskih rešenja integracije.

U ovom radu prikazana je realizacija softverske komponente za integraciju grafičkog korisničkog interfejsa sa softverom hibridnog sloja STB uređaja. U tu svrhu bilo je neophodno korisničkoj aplikaciji obezbediti spregu za prihvat podataka sa udaljenog serverskog srednjeg sloja. Rešenje je implementirano u programskom jeziku C, kako bi se obezbedila lakša prenosivost koda i maksimalno iskoristile mogućnosti hardvera. Posredstvom jedinstvenog i univerzalnog API-ja, koji obezbeđuje realizovano rešenje, korisnička aplikacija ima pristup svim podacima i sadržajem neophodnim za funkcionalnost sistema i interakciju sa korisnikom. Sprovođenjem ispitnih slučajeva utvrđeno je da je realizovano rešenje odgovorilo na sve postavljene zahteve. Zadovoljavajući odziv korisničke aplikacije, prihvatljiv sa aspekta korisničkog iskustva, potvrdio je funkcionalnost i ispravnost realizovanog rešenja.

Dalji rad na unapređivanju ovog rešenja mogao bi da bude usmeren u pravcu prenosivosti na druge operativne sisteme (npr. Windows, Linux).

7. Literatura

[1] Marshall Brain: *How Digital Television Works*, http://electronics.howstuffworks.com/dtv.htm

[2] *Digital TV:A Cringley Crash Course – Digital Vs. Analog*.Pbs.org. Retrived 2014-01-13

[3] *OTT, Over the Top content*, https://en.wikipedia.org/wiki/Over-the-top_content

[4] *REST API*, https://developer.salesforce.com/page/REST_API

[5] *libcurl API*, http://curl.haxx.se/libcurl/c/

[6] *HTTP*, https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[7] *JNI*, http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html

[8] N. Jovanov, V. Ilkic, M. Knezevic, N. Smiljkovic: *Design solution of 3D graphical user interface for television receivers*, IEEE CE Workshop, Novi Sad, 2015

[9] Milan Z. Bjelica, Ištvan Pap, Nikola Teslić: *Moderni pravci razvoja grafičkih korisničkih sprega za STB i TV uređaje*, ETRAN, Srebrno jezero, 2015

[10] *ART and Dalvik*, https://source.android.com/devices/tech/dalvik/

[11] *OpenGL ES, The Standard for Embedded Accelerated 3D Graphics*, https://www.khronos.org/opengles/

[12] *DirectFB*, http://www.directfb.org/

[13] B. P. Toledo Freitas, A. A. Susin, A. Bonatto: *Ginga MiddleWare on a SoC for Digital Television Set-Top Box*, IEEE 5th Latin American Symposium on Circuits and Systems, 2014

[14] M. Bjelica, N. Teslić, M. Savić: *Upotreba Open Source okruženja za razvoj grafičke korisničke sprege za TV prijemnike*, Zbornik radova 53. konferencije za ETRAN, 2009

[15] S. Y. Lee, S. T. Park: *Design and Implementation of Basic Smart TV Functions on the Web Browser*, International Conference on ICT, 2013

[16] N. Jovičić: *Jedno rešenje grafičke korisničke sprege digitalnog TV prijemnika*, Master rad, Fakultet Tehničkih Nauka u Novom Sadu, 2012

[17] J. Svensson: *Next Generation Graphical User Interface for IPTV*, Master of Science Thesis, KTH, Sweden, 2013

[18] *WebGL*, https://get.webgl.org/

[19] M. Vidakovic, T. Maruna, N. Teslic, V. Mihic: *A Java API interface for integration of DTV services in embedded multimedia devices*, IEEE Transactions on Consumer Electronics, vol. 58, no. 3, pp. 1063-1069, 2012

[20] N. Kuzmanovic, T. Maruna, M. Savic, G. Miljkovic, D. Isailovic: Google's Android as an application environment for DTV decoder system, IEEE 14th International Symposium on Consumer Electronics (ISCE), 2010

[21] M. Milanović, B. Pavlović, I. Petrović, T. Maruna: *Jedno rešenje grafičke korisničke sprege TV aplikacije na prijemniku digitalnog televizijskog signala zasnovanog na Android platformi*, TELFOR, pp. 724-726, 2013

[22] *Android*, Patrick Brady dissects Android, http://www.zdnet.com/article/patrick-brady-dissects-android/

[23] *Android TV*, http://www.android.com/tv/

[24] *Android Anatomy and Psychiology*, http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf

[25] *10-foot user interface*, Graphical User Interface, https://en.wikipedia.org/wiki/10-foot_user_interface

[26] Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee (June 1999): *Hypertext Transfer Protocol -- HTTP/1.1. IETF. RFC 2616*, http://www.w3.org/Protocols/rfc2616/rfc2616.html

[27] *TCP/IP Protocol*, http://docs.oracle.com/cd/E23823_01/html/816-4554/ipov-6.html

[28] T. Dierks, E.Rescorla (August 2008): *The Transport Layer Security (TLS) Protocol Version 1.2*, https://tools.ietf.org/html/rfc5246

[29] *XML 1.0 Specification*, World Wide Web Consortium, http://www.w3.org/TR/1998/REC-xml-19980210.pdf

[30] *RSS 2.0 Specification*, http://cyber.law.harvard.edu/rss/rss.html

[31] *SOAP Specification*, http://www.w3.org/TR/soap/

[32] *XHTML 2.0 Specification*, http://www.w3.org/TR/xhtml2/

[33] *Atom Specification*, http://www.ietf.org/rfc/rfc4287.txt

[34] *XMPP Standards Foundation*, http://xmpp.org/

[35] *XMLTV*, http://wiki.xmltv.org/index.php/Main_Page

[36] H.Peić Tukuljac, D. Nađ, G. Stupar: *From EPG to Good Recommendation Results for TV Program*, IEEE CE Workshop, Novi Sad, 2015

[37] H. Peić Tukuljac, D. Nađ, G. Stupar, M. Z. Bjelica: *A solution of a DTV recommendation engine based on broadband and broadcast data*, TELFOR, 2014

[38] *libGDX*, http://libgdx.badlogicgames.com/

[39] *libXML2*, http://www.xmlsoft.org/

[40] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart (June 1999): *HTTP Authentication: Basic and Digest Access Authentication*, http://tools.ietf.org/html/rfc2617

[41] R. B. Miller: *2 second rule*, Response time in man-computer conversational transactions, 1968

[42] Jacob Nielsen: *Response Times: The 3 Important Limits*, January 1993, http://www.nngroup.com/articles/response-times-3-important-limits/

[43] Jacob Nielsen: *Powers of 10: Time Scales in User Experience*, October 2009, http://www.nngroup.com/articles/powers-of-10-time-scales-in-ux/