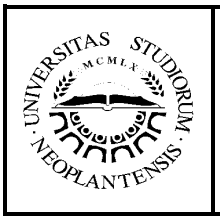


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: **Никола Зорић**
Број индекса: **РА97/2018**

Тема рада: **ИМПЛЕМЕНТАЦИЈА СЕРВИСА ЗА ГРУПНО ГЛЕДАЊЕ ВИДЕО САДРЖАЈА КОРИСТЕЋИ ‘LINE REST API’**

Ментор рада: **проф. др Илија Башичевић**

Нови Сад, јул, 2024

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**:		Завршни (Bachelor) рад
Аутор, **АУ**:		Никола Зорић
Ментор, **МН**:		проф. Др Илија Башичевић
Наслов рада, **НР**:		Имплементација сервиса за групно гледање видео садржаја користећи 'LINE REST API'
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2024
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)		7/27/0/0/11/0/0
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кqучне речи, **ПО**:		Дигитална телевизија, „Line" програмски интерфејс, HTTП протокол, База података
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овом раду описана је имплементација сервиса за подршку групном гледању видео садржаја користећи програмски интерфејс „Line" платформе. Такође, урађено је истраживање програмског интерфејса платформе те описан проблем затворености у тренутку издавања рада.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	
	Члан:	Потпис ментора
	Члан, ментор:	

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**:		Bachelor Thesis	
Author, **AU**:		Nikola Zorić	
Mentor, **MN**:		Ilija Bašičević, PhD	
Title, **TI**:		Implementation of a Service for Group Video Watching Using the 'LINE REST API'	
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**:		2024	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/27/0/0/11/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Digital television, Line REST API, HTTP protocol, Database	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper describes the implementation of a service supporting group video content viewing using the Line platform's application programming interface. Additionally, a study of the platform's API was conducted, and the issue of its closed nature at the time of publication is discussed.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:		
	Member:		Menthor's sign
	Member, Mentor:		

Захвалност

Захваљујем се ментору, проф. Др Илији Башичевићу и техничком ментору, Милици Лукић, на стручној помоћи и савјетима током израде овог рада.

Такође, захвалност дугујем мојој породици, дјевојци и пријатељима за сву подршку коју су ми пружали током школовања.

САДРЖАЈ

Списак слика

Скраћенице

API (*Aplication programming interface*) – Програмски интерфејс апликације

REST (*Representational state transfer*) – Пренос приказа стања

QR (*Quick response*) – Дводимензиони бар-код са брзим одзивом

HTTP (*Hypertext Transfer Protocol*) – Протокол за пренос хипертекста

HTML (*Hypertext Markup Language*) - Језик за означавање хипертекста

OSI (*Open System Interconnection*) – Отворено повезивање система

TCP (*Transmission Control Protocol*) – Протокол контроле преноса

IP (Internet Protocol) – Мрежни протокол

URL (*Uniform Resource Locator*) – Јединствени локатор ресурса

JSON (*JavaScript Object Notation*) – Текстуални формат за размјену порука

1. Увод

У доба доминације друштвених мрежа и потребом за кратким и директним садржајем може се примијетити да телевизијска индустрија губи корак с конкуренцијом. Корисници желе да брже добијају информације, желе краћи и прилагођенији садржај те интерактивно искуство у којем они неће бити само посматрач садржаја. Као одговор на такво стање на тржишту, јавља се концепт који корисницима нуди другачије искуство – концепт *Watchparty*. Идеја је да се корисницима омогући заједничко гледање телевизијског садржаја док комуницирају у реалном времену путем дигиталних платформи.

Сам пројекат пружа могућност креирања група у којима корисници могу истовремено гледати садржај, док коментаришу, реагују и дијеле утиске са осталима. Овај концепт трансформише начин на који људи доживљавају телевизију, чинећи корисничко искуство интерактивним. Такође, омогућава стварање такозваних дигиталних соба гдје корисници, иако физички удаљени, имају могућност да дијеле реакције у реалном времену.

Задатак овог рада је имплементација сервиса који ће омогућити размјену података између клијентске апликације за групно гледање садржаја и *Line* апликације на корисниковом мобилном телефону. Овај сервис омогућава да се корисници пријаве путем *Line* апликације, потом да се добаве основне информације о сваком кориснику те да се створе групе за гледање, коментаре и интеракцију током емитовања садржаја.

Приликом израде овог рада није било могуће имплементирати све жељене функционалности сервиса због ограничења *Line REST API*-ја о чему ће више бити речено у наставку.

Рад се састоји из 7 поглавља. У другом поглављу су дате теоријске основе, те објашњени основни протоколи и формати који су коришћени приликом израде. У трећем поглављу је представљен концепт рјешења. Четврто поглавље описује главне функције које су имплементиране. У петом поглављу су представљени резултати рада и описани су начини верификације исправности реализованог модула. Шесто поглавље даје закључак рада, док седмо поглавље даје приказ литературе коришћене приликом израде рада.

2. Теоријске основе

2.1 *HTTP* протокол

HTTP (Hypertext Transfer Protocol) је најчешће коришћен интернет протокол те је као такав коришћен и у овом раду. Основна намјена овог протокола је достављање *HTML* (Hypertext Markup Language) докумената, односно интернет страница. Припада апликативном слоју *OSI* (Open System Interconnection) референтног модела.

Комуникација се врши по систему клијент-сервер, гдје клијент започиње комуникацију, док сервер ослушкује те одговара на поруку. Најприје клијент успоставља везу са сервером путем *TCP/IP* протокола на одређеном порту. Сервер константно ослушкује на долазеће захтјеве. Након успостављене поуздане везе на транспортном нивоу, клијент шаље *HTTP* захтјев који се састоји од линије захтјева, заглавља и тијела. Линија захтјева састоји се од:

- *HTTP* методе
- назива траженог ресурса (*URL*)
- верзије *HTTP*-а

У заглављу клијент доставља више информација о вези и типу садржаја. Тијело захтјева је опционо и користи се код одређених метода ради слања података серверу, типично код *POST* методе [1].

Неке од најчешће коришћених метода су: *GET* - користи се за обављање интернет страница, *POST* - користи се за слање података серверу, *PUT* -

користи се за ажурирање постојећег ресурса на серверу, *DELETE* – брисање одређеног ресурса итд. Примјер једног *HTTP* захтјева дат је на слици 2.1.

Након примљеног захтјева, сервер шаље одговор. Структура одговора је сљедећа:

- статусна линија - садржи верзију проткола, статусни код и опис статуса
- заглавље одговора - садржи додатне информације о одговору
- тијело одговора - садржи сам тражени ресурс

Статусни код се састоји од три цифре и даје додатне информације о обављеној акцији. Може се подијелити у 5 група:

1xx - информативни кодови

2xx - потрвда успјешно обављене акције

3xx – преусмјеравање

4xx - грешка на клијентској страни

5xx - грешка на серверској страни.

Примјер *HTTP* одговора дат је на слици 2.2

```
GET /index.html HTTP/1.1
Host: www.examplehost.com
User-Agent: user
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
```

Слика 2.1 – Примјер *HTTP* захтјева

```
HTTP/1.1 200 OK
Server: Apache
Content-Encoding: gzip
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Слика 2.2 - Примјер *HTTP* одговора

2.2 REST API

API (Application Programming Interface) односно програмски интерфејс апликације је скуп дефинисаних правила и протокола који омогућавају различитим апликацијама да међусобно комуницирају. *API* је посредник који омогућава једној апликацији да користи функционалности и податке друге апликације без потребе да познаје детаље њене интерне имплементације. Омогућава модуларност и флексибилност у развоју софтвера, јер олакшава поновну употребу кодова и функционалности.

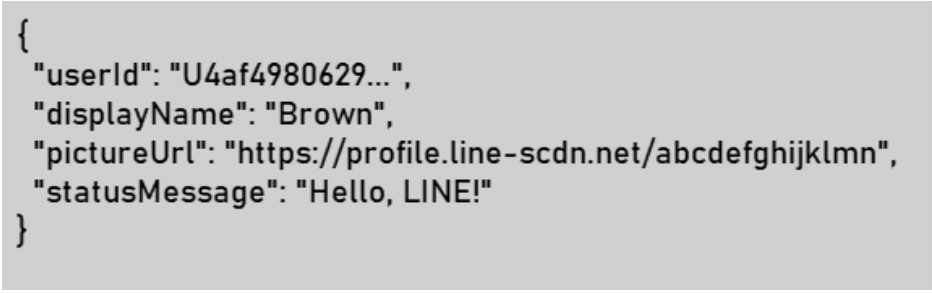
REST (Representational State Transfer) API односно пренос приказа стања је стил архитектуре за дизајнирање интернет апликација. Омогућава комуникацију између клијента и сервера коришћењем стандарних *HTTP* метода. Кључна идеја је да се интернет сервиси третирају као ресурси којима се може приступити преко јединствених *URI-ja (Uniform Resource Identifier),* тј. јединственог имена ресурса. Користи протокол без памћења стања што значи да сваки *HTTP* захтјев садржи све потребне информације за његово извршење. Ово омогућава једноставност интеракције јер сервер нема потребу да чува информације о стању клијента између захтјева. Ресурси у *REST API*-ју се представљају у различитим форматима, најчешће у *JSON* формату, што омогућава једноставну размјену података између различитих система. Једна од кључних предности је његова флексибилност и једноставност. Користи стандардне *HTTP* методе те се може лако интегрисати у различите врсте апликација [2].

Све горенаведене предности чине *REST API* једним од најпопуларнијих и најшире прихваћених стандарда за изградњу интернет сервиса.

2.3 JSON

JSON (JavaScript Object Notation) је текстуални формат за размјену података. Обично се користи за пренос података у интернет апликацијама. Иако је првенствено настао као дио *JavaScript* језика он је данас језички независан и већина језика посједује уграђене функције за манипулацију овим форматом. Формат је заснован по систему име – вриједност при чему је име увијек низ карактера, док вриједност може да буде број, низ итд. Кључна предност овог формата је његова једноставност. Људима је лако читљив, док је машинама

једноставан за рашчлањивање и генерисање [3]. Примјер једног *JSON* објекта дат је на слици 2.3.



Слика 2.3 – Примјер *JSON* формата поруке

2.4 POSTMAN

Postman је алат за тестирање *API*-ја који омогућава програмерима да шаљу *HTTP* захтјеве и анализирају одговоре. Једноставан је и интуитиван за коришћење. Корисницима омогућава слање разних *HTTP* захтјева те дозвољава лако подешавање свих параметара захтјева што омогућава детаљно и једноставно тестирање. Такође, нуди подршку за рад са разним форматима података, укључујући и *JSON*, што га чини погодним за рад са мрежним сервисима [4].

2.5 JAVASCRIPT

JavaScript је динамички типизиран програмски језик високог нивоа, развио га је 1995. године Брендан Ајк радећи за компанију Нетскејп. Иако је првенствено био намјењен динамичком креирању интернет страница, развојем *Node.js* платформе своју примјену налази и у креирању интернет сервера.

JavaScript је интерпретирани језик, што значи да се преводи линију по линију током извршавања, за разлику од компајлираних језика који се преводе у машински језик у цјелини прије самог извршавања. Користи асинхрони модел извршавања заснован на догађајима, што га чини погодним за мрежне сервере који морају да истовремено обраде велики број конекција без блокирања ресурса. Једнонитни је језик, али захваљујући механизму догађаја те неблокирајућим улазно-излазним операцијама може ефикасно да обради велики број захтјева истовремено [5].

Node.js је платформа заснована на *Chrom V8 JavaScript* машини која омогућава покретање кода изван интернет претраживача. Подршка механизму догађаја те неблокирајућим улазно-излазним операцијама учиниле су ову платформу веома погодном за креирање интернет сервера.

2.6 SQLITE

Sqlite је библиотека отвореног кода писана у *C* програмском језику чија је намјена подршка у креирању и управљању базом података. За разлику од већине других система за управљање базама података, не користи клијент-сервер архитектуру, већ се складиштење врши у обичну датотеку на диску. Погодна је за апликације мање величине које захтјевају лагану, локалну базу података. Сама библиотека не захтјева велике ресурсе јер заузима свега неколико мегабајта меморије што је чини погодном и за мобилне и уграђене апликације. Компатибилна је на више платформи гдје нуди подршку за разне оперативне системе, укључујући *MAC OS Linux*, *Windows* и др. Једна од кључних предности ове библиотеке је једноставност. Програмери могу једноставно преузети библиотеку, укључити је у свој пројекат и одмах почети користити. За њено коришћење нема потребе да се инсталира посебан сервер и да се врши додатна конфигруација. Упркос својој једноставности *Sqlite* пружа висок степен перформанси те се може оптимизовати за специфичне случајеве употребе [6].

Кључни недостатак *Sqlite* библиотеке је то што је намјењен једнокорисничким апликацијама, односно није погодан за апликације које захтјевају истовремени приступ бази података. Међутим, *Sqlite* пружа подршку за истовремени приступ кроз дијељени кеш и евидентирање унапред.

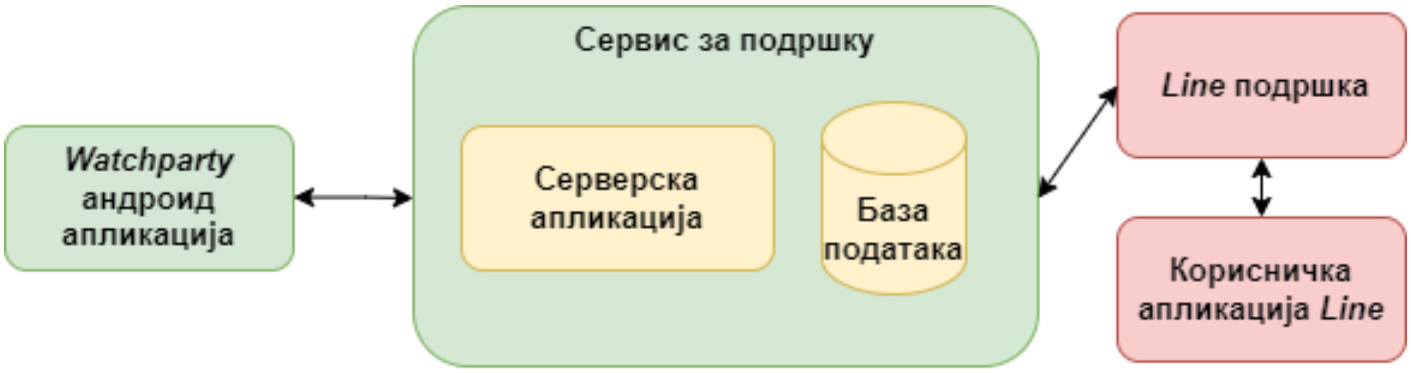
3. Концепт рјешења

У овом поглављу описан је концепт рјешења. Приказан је преглед структуре и основне идеје пројекта *Watchparty.*

У оквиру пројекта можемо да одвојимо четири ентитета који међусобно комуницирају, а то су:

- *Watchparty* андроид апликација
- Сервис за подршку
- *Line* серверска подршка
- Клијентска апликација *Line*

Приказ на слици 3.1



Слика 3.1 - Архитектура пројекта

3.1 *Watchparty* андроид апликација

Watchparty андроид апликација је корисничка апликација која омогућава заједничко гледање телевизијског садржаја. Кроз апликацију је омогућено да се корисник повеже са својим *Line* налогом. Добавља основне информације о кориснику, његову профилну слику, листу контаката итд. Апликација даље омогућава креирање нове *Line* групе за дописивање у коју корисник може да додаје пријатеље из листе контаката са којима ће да гледа заједнички садржај по избору. Сав садржај, односно све поруке, реакције и коментаре које корисници буду дијелили на заједничкој групи биће приказани на екрану у реалном времену. Након што је креирана група, корисник преко своје *Line* апликације на телефону коментарише садржај заједно са својим пријатељима.

3.2 Сервис за подршку

Као што је раније наведено, задатак овог рада је реализација сервиса који пружа подршку заједничком гледању телевизијског садржаја. Сервис представља спону између корисничке апликације и *Line* платформе. С једне стране обезбјеђује једноставан *REST API* корисничкој апликацији, поједностављујући процес корисничкој апликацији тако да не мора да улази у детаље интеракције са *Line* платформом, док с друге стране сам сервис користи *REST API Line* платформе да би омогућио све функционалности.

REST API овог сервиса имплементира функције које обезбјеђују следеће захтјеве:

- Пријављивање и одјављивање корисника
- Преузимање корисничке профилне фотографије са *Line* налога
- Преузимање информација о профилу
- Креирање нове *Line* групе за дописивање
- Додавање и брисање корисника у групи

Овај сервис у својој бази података чува информације о пријављеним корисницима као и остале релевантне податке за једног корисника.

Након успјешне пријаве корисника и креирања групе за заједничко гледање телевизијског садржаја овај сервис ослушкује на поруке унутар креиране групе. Претходно се на адекватан начин ауторизовао на *Line* платформи те добија информације о порукама које пристижу. Путем *FCM*

механизма сервис шаље обавјештења корисничкој апликацији о пристиглим порукама да би оне могле бити приказане.

3.3 *Line* подршка

Line подршка представља *API* који омогућава имплементацију жељених функционалности у оквиру сервиса. Кроз *Line* подршку је омогућено корисницима да се пријаве на платформу користећи своје *Line* налоге, чиме је поједностављен процес аутентификације, а такође је повећана безбједност услуге. *API* пружа могућност добављања детаљних информација о корисницима, што омогућава персонализацију услуга и унапређење корисничког искуства. Осим основних функционалности аутентификације, *Line API* нуди опције за интеграцију са другим апликацијама и сервисима, чиме се проширују могућности платформе.

Наведене функционалности значајно побољшавају интеракцију са корисницима те доприносе ефикасности при развоју пројекта. Коришћењем ових функционалности избјегнута је потреба да се развија потпуно нова платформа за размјену порука. Детаљи интеракције сервиса и *Line* подршке детаљније су описани у поглављу 4. које се бави програмским рјешењем.

4. Програмско рјешење

У овом поглављу дат је детаљнији опис имплементације функција и њихова намјена. За развој је коришћено *Visual Studio Code* окружење, а само рјешење је реализовано на бази *Node.js* платформе.

4.1 Пријављивање корисника

Као што је претходно наведено, да би обезбједио функције које су неопходне корисничкој апликацији, сервис комуницира са *Line* платформом која му омогућава да имплементира тражене функције. Да би се приступило *REST API-jy* платформе потебно је направити налог на самој платформи путем којег ће бити обезбјеђен идентификациони број и тајни кључ за приступ. Овај механизам обезбјеђује транспарентност и повећава безбједност при приступању функционалностима које нуди платформа.

Клијентска апликација започиње процес пријаве слањем захтјева сервису. Пошто се процес потврђивања идентитета клијента обавља у оквиру *Line* платформе, сервис као одговор на захтјев шаље јединствено име ресурса (енг. URI) са подешеним списком параметара који ће клијента да преусмјере на страницу за потврду идентитета.

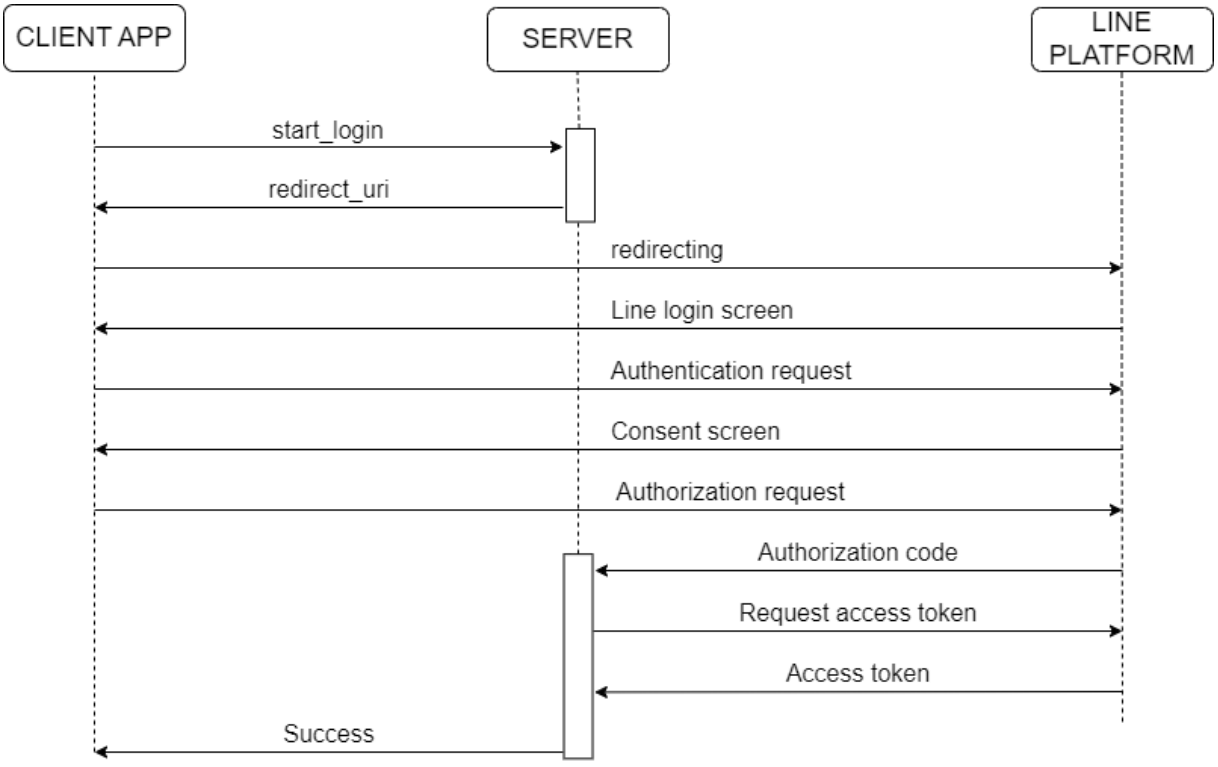
Параметри одговора су следећи:

- *response_type* – параметар који говори који је очекивани одговор, односно шта се тражи од платформе. У конкретном случају, очекивани одговор је типа *code,* што значи да као повратну информацију очекује код за потврду идентитета.

- *client_id* – идентификациони број налога на платформи преко којег се приступа *REST API-jy* платформе.

- *redirect_uri* – адреса на коју ће бити преусмјерен клијент након што заврши процес потврде идентитета.
- *state* – сигурносни код, односно јединствени алфанумерички низ који обезбјеђује сигурност комуникације. Генерише се при сваком новом захтјеву за пријаву. На овај начин трећа страна, која није извршила идентификацију, не може да се укључи у комуникацију и злоупотријеби информације.
- *scope* – параметар који говори које се све дозволе траже од корисника. У конкретном случају захтјевају се дозволе за приступ информацијама профила, идентификационом броју клијента те његовој електронској пошти.

Дијаграм тока комункације приказан је на слици 3.1



Слика 4.1 - Дијаграм тока пријаве

Кликом на линк који је добио као одговор сервиса, клијент бива преусмјерен на страницу за пријаву. Процес потврде идентитета се обавља директно између корисника и *Line* платформе, односно сервис није укључен.

Клијент се може пријавити на неколико начина укључујући уношењем електронске поште и лозинке, или скенирањем кода са брзим одзивом (енг. QR code). Након потврде идентитета, клијент се преусмјерава на страницу на којој су приказане дозволе за коришћење које захтјева сервис.

Пошто су добављене све потребне дозволе од клијента, даље се комуникација врши између сервиса и *Line* платформе као што је приказано на слици 3.1.

Сервис у оквиру одговора *Line* платформе добија захтјевани код за потврду идентитета те сигурносни код. Најприје је потребно потврдити да је послати сигурносни код адекватан, чиме се обезбјеђује додатна безбједност комуникације. Користећи добијени код за потврду идентитета сервис шаље захтјев за добијање приступног токена. Уколико није дошло до грешке у комуникацији *Line* платформа шаље приступни токен сервису, чиме му даје приступ потребним информацијама корисника. Приступни токен, као и други важни подаци везани за клијента чувају се у бази података о чему ће више бити ријечи у поглављу 4.6.

Одговор клијенту се шаље у *JSON* формату. Примјер успјешног одговора приказан је на слици 4.2

```
{
        "accessToken" : "1a3B97Hk90oO4F9gv11h5GD62Nv7..."
        "msg"         : "User authenticated successfully"
}
```

Слика 4.2 - Примјер поруке одговора на пријаву

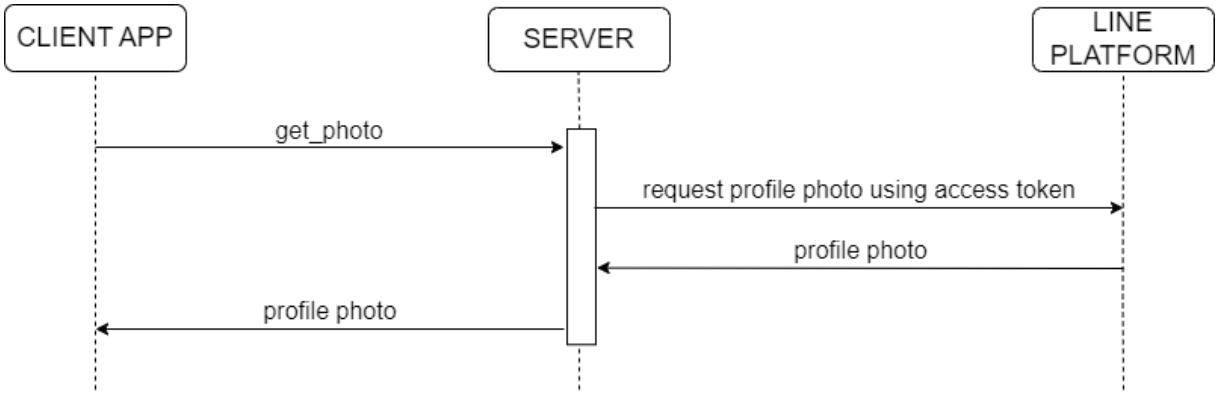
4.2 Добављање профилне фотографије

Приликом добављања профилне фотографије у заглављу *HTTP* захтјева, клијент шаље свој приступни токен који је добио као одговор на успјешну пријаву, након тога сервис провјерава валидност приступног токена у бази података. У случају да приступни токен није валидан, комуникација се завршава

и клијент добија поруку да није доставио валидан приступни токен, или да није пријављен на исправан начин. Примјер одговора у случају да приступни токен није валидан приказан је на слици 4.4

У случају да је приступни токен адекватан сервер приступа слању захтјева *Line* платформи за достављање слике. У заглављу захтјева доставља приступни токен за одговарајућег клијента. У случају да има потребне дозволе сервер добија профилну фотографију, односно добија јединствену адресу на којој се налази фотографија. Даље, сервер обезбјеђује преузимање фотографије те конверзију у одговарајући формат прије слања клијенту.

Дијаграм комуникације приказан је на слици 4.3



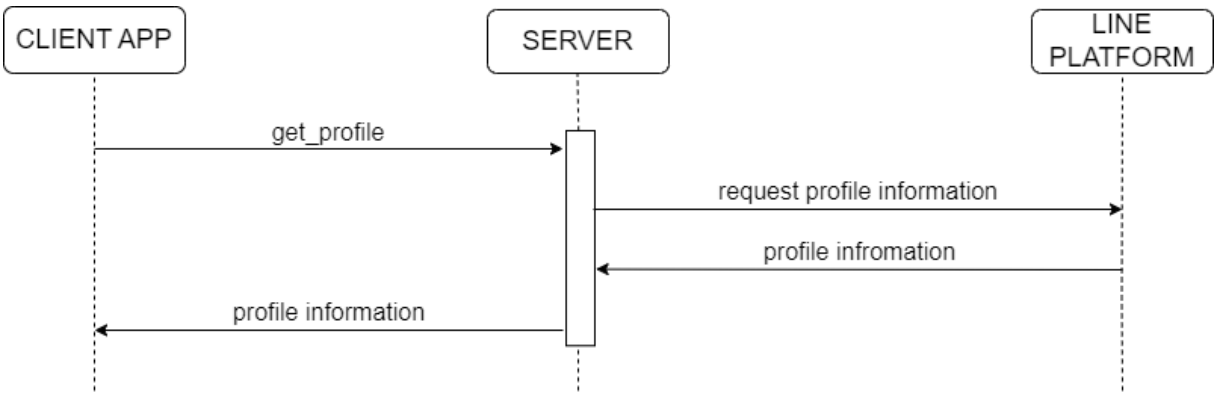
Слика 4.3 - Дијаграм тока комуникације при добављању фотографије

```
{
        "error" : "User is not authorised!"
}
```

Слика 4.4 – Примјер поруке негативног одговора на захтјев

4.3 Добављање информација о профилу

Информације о профилу се односе на корисничко име, електронску пошту те идентификациони број корисника. Комуникација се одвија слично као у примјеру добављања профилне фотографије и приказана је на слици 4.5



Слика 4.5 – Дијаграм тока при добављања информација о профилу

Клијент шаље захтјев у оквиру којег шаље приступни токен као потврду свог идентитета и потврду о обављеној пријави. Након што је провјерена валидност приступног токена сервер шаље захтјев *Line* платформи за добављање информација о електронској пошти, идентификационом броју те корисничком имену клијента. Одговор је дат у формату *JSON* и као такав се шаље назад до клијента.

Примјер валидног одговора приказан је на слици 4.6

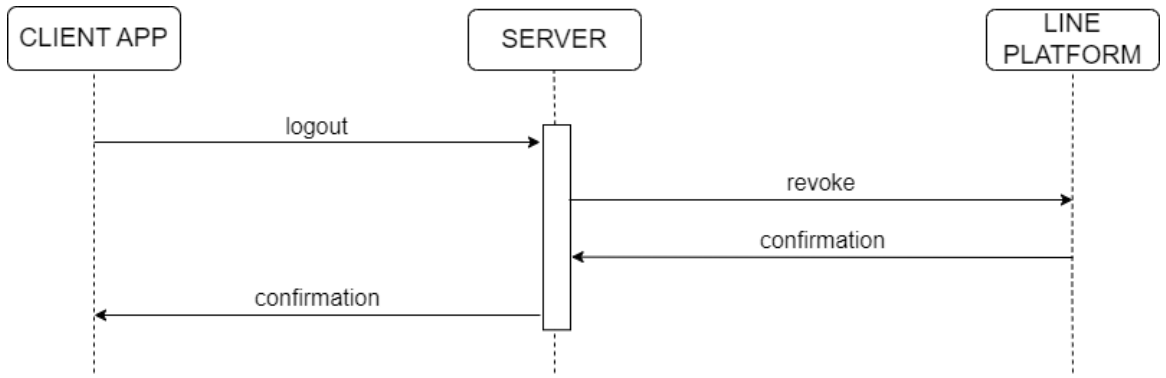
```
{
        "id"     : "U4af4980629..."
        "name"   : "Stefan"
        "e-mail" : "stefan90@gmail.com"
}
```

Слика 4.6 – Примјер повратне поруке информације о профилу

4.4 Одјављивање корисника

На слици 4.7 приказан је дијаграм тока комуникације при одјављивању корисника. Клијент започиње комуникацију захтјевом за одјављивање, као потврду свог идентитета шаље приступни токен. Након потврде идентитета клијента, сервер шаље захтјев *Line* платформи за одјављивање корисника. Након што је потврђено успјешно одјављивање, приступни токен више није важећи и не може се користити за приступ информацијама о кориснику.

Сервер води рачуна о томе да се у бази података све информације о кориснику обришу.



Слика 4.7 – Дијаграм тока при одјављивању

4.5 Креирање групе и читање порука у групи

Још један од задатака овог сервиса био је да омогући креирање група у име корисника те читање самих порука у групи да би се омогућило њихово емитовање на екрану клијентске апликације. Међутим, *Line* платформа у тренутку рада на овом задатку није омогућавала приступ таквим функцијама. Директно читање приватних порука порука корисника није дозвољено због заштите приватности, законских регулатива те корисничких уговора. Алтернативни приступ описан је у поглављу 4.7.

4.6 База података

База података реализована је коришћењем библиотеке *SQLite3.* База се састоји од једне табеле која има следеће колоне: примарни кључ, кориснички идентификациони број, приступни токен и сигурносни код. У оквиру базе података постоје следеће функције које омогућавају манипулацију над базом: функција за креирање тебеле, функција за додавање новог корисника, функција за ажурирање приступног токена, функција за ажурирање сигурносног кода, функција за ажурирање корисничког идентификационог броја те функција за брисање корисника или цијеле табеле.

4.7 Алтернативни приступ читању порука у групи

Опција коју *Line* платформа нуди као алтернативу директном читању порука клијента је читање порука преко бота задуженог за комуникацију. Идеја је да заједничку групу клијент креира мануелно, те да у њу поред пријатеља који

би гледали заједнички садржај дода и бота који је повезан са сервисом. Бота је могуће креирати и повезати са сервисом користећи платформу. Сваки бот добија свој идентификациони број и *QR* код помоћу којег га је могуће додати у групу. На тај начин би сва обавјештења, поруке и реакције унутар групе долазиле и на адресу бота као учесника комуникације и те поруке би сервис могао да чита и обрађује.

Проблем оваквог приступа је у томе што не постоји могућност креирања посебног бота за сваку нову групу, већ би се морао користити један исти бот за различите групе и различите кориснике. На тај начин све поруке из различитих група би долазиле на исту адресу и захтјевале би обраду и разврставање. У случају великог броја корисника овај приступ би био превише комплексан и тежак за обраду. Међутим, у случају да је укупан број група и клијената ограничен, овакав приступ би могао да буде користан у неким имплементацијама.

5. Резултати

Тестирање функционалности апликације извршено је ручно, коришћењем алата *Postman*, због недостатка могућности за аутоматизовано тестирање у овој фази развоја. Тестирање је обухватало провјеру свих кључних функционалности које се односе на процес пријаве корисника, добијање профилних информација и одјаву. Први корак је био тестирање иницијалног захтјева за пријаву, гдје је провјерено да ли сервис генерише исправан URI за преусмјеравање корисника на *Line* платформу са свим неопходним параметрима као што су *client_id*, *response_type*, *redirect_uri* и *state*.

Додатно, тестирано је добијање и провјера приступног токена након успјешне аутентификације корисника на *Line* платформи. Посебна пажња посвећена је валидацији токена, гдје су тестирани и позитивни (исправан токен) и негативни (неважећи или истекли токен) сценарији. Након валидације токена, тестирано је добијање профилних информација корисника, укључујући његово име, електронску пошту и профилну фотографију.

Сваки захтјев и одговор је анализиран у Postman-у, при чему је провјерено да ли су одговори у исправном *JSON* формату и да ли садрже све очекиване податке. Поред тога, тестирани су и сценарији са некомплетним захтјевима или погрешним параметрима како би се осигурало да систем правилно обрађује грешке и враћа адекватне поруке. Завршни корак био је тестирање процеса одјаве корисника и верификација да се приступни токен исправно поништава након одјаве.

6. Закључак

У овом раду је реализован пројекат Watchparty који омогућава заједничко гледање телевизијског садржаја уз интеграцију са *Line* платформом. Пројекат је успешно имплементирао основне функције пријављивања корисника, добијање профилних информација и одјава, користећи *REST API* и *Node.js* платформу.

Током развоја и тестирања апликације захтјеви су били испуњени у оквиру техничких могућности које је платформа пружала. Сви наведене функционалности су успјешно тестиране и провјерене у оквиру овог рада.

Важно је напоменути да неки захтјеви нису могли бити испуњени због отворености *LINE API*-ја. Наиме, у време израде овог рада политика *Line* платформе није дозволила одређене функције које су биле неопходне за потпуно реализовање свих планираних функционалности. Конкретно, било је ограничења у приступу одређеним подацима и функцијама које су биле потребне за креирање и читање порука унутар група, што је утицало на завршну имплементацију пројекта. Имплементација која би ријешила нека од ограничења описано је у поглављу 4.7. За будуће радове било би корисно пратити потенцијалне измјене у политици платформе који би могли омогућити испуњење свих функционалних захтјева.

7. Литература

[1] HTTP протокол: https://httpwg.org/specs/ , приступано мај 2024.

[2] REST API: https://restfulapi.net/ , приступано мај 2024.

[3] JSON , https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON , приступано мај 2024.

[4] Postman: https://learning.postman.com/docs/introduction/overview , приступано јун 2024.

[5] JavaScript: https://developer.mozilla.org/en-US/docs/Web/JavaScript , приступано јун 2024.

[6] SQLite: https://www.sqlite.org/docs.html , приступано јун 2024.