TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Oskar Voorel 185778IACB

# Meedia sünkroonse vaatamise platvormi arendamine

Bakalaureusetöö

Juhendaja: Meelis Antoi

Magistrikraad

Tallinn 2021

# Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Oskar Voorel

08.02.2021

# Annotatsioon

Käesoleva bakalarusetöö eesmärgiks on luua töötav veebirakendus meedia sünkroonseks vaatamiseks, võimaldamaks võimalikult elulähedase kinokogemuse saamist füüsiliselt kinno minemata.

Arendusprotsessi käigus luuakse veebirakendus, mis võimaldab kasutajatel luua ruume, ruumidega ühineda, ruumide vahel liikuda, ruumides reaalajas vestelda ja sünkroonis voogedastusmeediat vaadata. Arendus on jagatud mitmesse ossa, mis käsitlevad veebiteenuse, meedia failide töötlemise mootori ja meedia voogedastusmootori ehitamist ning kõigi vastavatesse serveritesse paigaldust.

Töö rõhk on ruumide moodulil, kuid ka moodulite eraldiseisvusel, et potentsiaalselt võimaldada erinevatel osapooltel kasutada juba endal olemas olevaid mooduleid. Muuhulgas katab lõputöö kasutatuid tehnoloogiaid, arhitektuuri, testimist ning sellega seonduvat pidevat integratsiooni.

Arendusprotsessi tulemuseks on töötav modulaarne[1] veebirakendus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 46 leheküljel, 6 peatükki, 9 joonist, 1 tabelit.

[1] Koosneb üksteisest sõltumatutest, täpselt määratletud komponentidest

# Abstract

Synchronous media viewing web application

The aim of this thesis is to create a working web application for synchronous media viewing to be able to get the cinema experience without physically going to the cinema.

During development a web application is created which enables people to create rooms, join rooms, move between rooms, chat in rooms in real time and synchronously watch a media stream. Development is split into multiple parts which address developing the webservice, media file conversion engine and media streaming engine, all of which also contain moving each to the appropriate server.

The emphasis of the work is on the webservice module as well the modularity of the designed system, to allow potential users to use their own premade media streaming or media file conversion modules. The paper also covers used technologies, architecture, testing and continued integration that is associated with it. The result of the development process is a functional modular web application.

The thesis is in Estonian and contains 46 pages of text, 6 chapters, 9 figures, 1 table.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| API | *Application Programming Interface* – rakendusliides |
| AJAX | *Asynchronous Javascript and XML* – tehnoloogiate kogum, mis võimaldab asünkroonseid päringuid |
| *backend* | Teenusepoolne keskkond |
| *Bash* | GNU projekti raames kirjutatud Unix-i kest |
| Body | API päringu keha ehk sisu |
| CI | *Continuous Intergration* – pidev integratsioon |
| CLI | *Command-line Interface* – käsurea liides |
| CORS | *Cross-origin Resource Sharing* – mehhanism, mis lubab domeenivälist infovahetust |
| CSS | *Cascading Style Sheets* – veebilehtede küljendamisel kasutatav märgistuskeel |
| Deno | JavaScripti ja TypeScripti käituskeskkond |
| DOM | *Document Object Model* – dokumendi objektimudel |
| *front-end* | Kasutajaliides – kliendipoolne keskkond |
| *Front-end* | Autori poolt kasutatav mooduli nimi |
| GPU | *Graphics Processing Unit* - graafikaprotsessor |
| HTML | *Hyper Text Markup Language* – veebilehe märgendamise keel |
| IDE | *Integrated Development Environment* – rakendus, mis pakub arendajatele keskkonda koodi kirjutamiseks |
| JSON | *Javascript Object Notation* – Javascript'il põhinev andmevahetusvorming |
| JSX | Eelprotsessor, mis lubab kasutada XML süntaksit Javascripti koodis |
| JWT | *JSON Web Token* – JSON-põhine standard ligipääsu sõnede koostamiseks |
| *metadata* | Metaandmed – andmed, mis kirjeldavad teisi andmeid |

| | |
|---|---|
| NodeJS | JavaScripti käitussüsteem |
| *production environment* | Keskkond, mida kasutavad reaalsed kasutajad |
| PWA | *Progressive Web App* – veebileht kohaliku mobiilirakenduse kujul |
| REST | *Representational State Transfer* – veebiteenusega suhtlemise liidese arhitektuur |
| *runtime* | Käitusaeg – programmi jooksutamise aeg |
| *skaleerimine* | *To scale up* vaste eesti keeles |
| *TypeScript* | JavaScript-i täiendus, lisades muutujate tüübid |
| UI | *User Interface* – kasutajale näidatava vaate disain |
| UUID | Universaalne unikaalne identifikaator |
| UX | *User Experience* – kasutaja kogemus rakendust kasutades |
| *WebSocket* | Uudne tehnoloogia, mis võimaldab kahepoolset suhtlust *browser*-i ja serveri vahel [1] |
| XML | *Extensible Markup Language* – standardne üldotstarbeline märgistuskeel |

# Sisukord

1 Sissejuhatus ..... 11
1.1 Metoodika ..... 12
2 Ülevaade probleemist ..... 13
2.1 Eksisteerivad lahendused ..... 13
2.1.1 Watch2Gether ..... 13
2.1.2 twoseven ..... 13
2.1.3 Plex – Watch Together ..... 13
2.1.4 Sync-Tube ..... 14
2.2 Eksisteerivate platvormide tugevuste ja nõrkuste võrdlus ..... 15
2.3 Uue lahenduse skoop ..... 15
2.4 Eeldused ..... 16
2.4.1 Nõuded lõppkasutaja vaatepunktist ..... 16
2.4.2 Nõuded süsteemi haldaja vaatepunktist ..... 16
2.4.3 Arenduskeskkonna valik ..... 17
3 Süsteemi arhitektuur ..... 19
3.1 Andmeliikluse arhitektuur ..... 22
3.2 Andmebaaside arhitektuur ..... 23
3.2.1 Andmebaasi tehnoloogia valik ..... 23
3.2.2 *Room manager* mooduli andmebaasi arhitektuur ..... 23
3.2.3 *Media DB* mooduli andmebaasi arhitektuur ..... 26
4 Moodulid ..... 28
*4.1 Front-end* ..... 28
4.1.1 Nõuded moodulile ..... 28
4.1.2 Tehnoloogia valik ..... 28
4.1.3 Mooduli töö kirjeldus ..... 29
*4.2 Room manager* ..... 31
4.2.1 Nõuded moodulile ..... 31
4.2.2 Tehnoloogia valik ..... 31
4.2.3 Mooduli arhitektuur ..... 33
*4.3 Streaming server* ..... 34

4.3.1 Nõuded moodulile ..... 34
4.3.2 Kaalutud tehnoloogiad ..... 35
4.3.3 Mooduli töö kirjeldus ..... 35
*4.4 Movies DB* ..... 35
4.4.1 Nõuded moodulile ..... 35
4.4.2 Tehnoloogia valik ..... 36
4.4.3 Mooduli töö kirjeldus ..... 36
*4.5 Video digestion engine* ..... 36
4.5.1 Nõuded moodulile ..... 36
4.5.2 Tehnoloogia valik ..... 37
4.5.3 Mooduli töö kirjeldus ..... 38
5 Tekkinud probleemid ..... 39
5.1 CORS-i vead erinevate võrkude vahel ..... 39
5.2 Hilised arhitektuurilised muudatused ..... 39
5.3 *Video digestion engine* mooduli jaoks valitud riistavara ..... 39
6 Tulemused ja edasiarenduse plaanid ..... 40
7 Kokkuvõte ..... 41
8 Summary ..... 42
9 Kasutatud kirjandus ..... 43
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks ..... 45

# Jooniste loetelu

Joonis 1 - Arenduse progressi järgimine Githubi keskkonnas ..... 18
Joonis 2 - Andmeliikluse skeem ..... 22
Joonis 3 – *Room manager* mooduli andmebaasi arhitektuuri skeem ..... 24
Joonis 4 - *Media DB* mooduli andmebaasi arhitektuuri skeem ..... 26
Joonis 5 - Ruumi sisenemisvaate näide, kui *Ticketing Module´*it ei ole kasutuses ..... 30
Joonis 6 – Suhtlusakna näide ..... 31
Joonis 7 - *Room manager* mooduli sisene andmeliiklusskeem ..... 34

# Tabelite loetelu

Tabel 1 - Olemasolevate lahenduste tugevused ja nõrkused ..... 15

# 1 Sissejuhatus

Üks kõige populaarseimaid meelelahutusmeetodeid tänapäeval on filmide, seriaalide ja muude videote vaatamine ning neile kaasa elamine. Inimesed on sotsiaalsed olendid, mis tähendab, et enamasti nad eelistavad asju teha koos teiste inimestega, see põhimõte laieneb ka filmide vaatamisele. Selle inimese loomuomaduse tõttu on välja arenenud suured firmad ja lahendused nagu Twitch[1], Facebook Live[2] ja YouTube Live[3].

Antud platvormid keskenduvad kõik kasutaja loodud sisule, kus ruumi looja filmib ennast või midagi muud huvitavat, kõik ülejäänud ruumis olijad vaatavad tema tegevusi reaalajas ning saavad nii üksteisega kui ka sisu loojaga teksti teel samuti reaalajas suhelda. Puudus on pühendatud keskkondadest, kus oleks võimalik vaadata koos teiste inimestega kolmanda osapoole meediakogu, sarnaselt nagu seda tehakse kinodes.

Käesolev lõputöö analüüsib probleemi Eesti kontekstis ja erinevate lahenduste all vaadeldakse keskkondi, mis on saadaval Eestist internetti ühenduvatele inimestele. Probleemi pakutud lahenduseks on pühendatud ühiselt vaadatavate filmide / seriaalide / videote vaatamise keskkonna välja arendamine, koos erinevate sotsiaal-liidestega.

Lõputöö autoril on ka endal tekkinud olukordi, kus on olnud sooviks vaadata koos teise(te) inimesega samaaegselt sama filmi, kuid füüsiliste piirangute tõttu pole sellise asja korraldamine olnud võimalik, mistõttu koos filmi või sarja vaatamine on osutunud võimatuks või erakordselt tülikaks. Sellest tulenevalt järeldab autor, et sarnane probleem eksisteeris ühiskonnas juba enne Covid-19 esinemist, mis aga omakorda on oluliselt suurendanud sellise olukorra/probleemi tekkimise võimalust.

---

[1] Hetkel kõige populaarsem voogedastuse platvorm: https://www.twitch.tv/

[2] Facebooki loodud voogedastuse platvorm: https://www.facebook.com/formedia/solutions/facebook-live

[3] Googlei loodud voogedastuse edasiarendus YouTube platvormile: https://www.youtube.com/howyoutubeworks/product-features/live/#youtube-live

## 1.1 Metoodika

Käesoleva lõputöö käigus selgitatakse esmalt lahti eksisteeriv probleem, eksisteerivad lahendused ja nende puudused ning pakutakse välja probleemile sobiv lahendus, mis võimaldaks edasi arendamist tulevikus ja kommertseesmärgil kasutuselevõttu.

Lahenduse analüüsi käigus tuuakse välja erinevad kasutajagrupid ning neid puudutavad nõuded kasutajaloode kujul, mis on grupeeritud funktsionaalseteks ja mitte-funktsionaalseteks nõueteks. Samuti analüüsitakse erinevaid tehnilisi lahendusi ning põhjendatakse nii teenusepoolsete ja kliendipoolsete tehnoloogiate valikuid kui ka arendus- ning haldusvahendite valikuid.

Lahenduse valmimist on kirjeldatud erinevate osadena, mis käsitlevad rakenduse kasutajaliidese ning andmebaasi disaini ja serveripoolse ning kliendipoolse lahenduse valmimist. Lahenduskäigu lõpuosas on kirjeldatud ka lahenduse edasiseid samme väljaspool lõputöö skoopi.

# 2 Ülevaade probleemist

Üha globaliseeruvas maailmas on inimesed, kes omavahel suhtlevad ja aega soovivad veeta, tihti üksteisest eraldatud suurte vahemaadega või ei ole neil võimalik kokku saada mõnel teisel põhjusel. Selle tõttu on ka üha enam meelelahutusplatvorme üritanud luua endast interneti vahendusel tarbitavaid versioone. Üks meelelahutusvorm, millel hetkel puudub hea interneti teel tarbitav lahendus, on kinos käimine. On palju platvorme, mis pakuvad võimalust meediat vaadata koos teiste inimestega interneti vahendusel, kuid ükski ei suuda pakkuda kinos käimisega sarnast kogemust ning on tihti väga raskesti kasutatavad.

## 2.1 Eksisteerivad lahendused

### 2.1.1 Watch2Gether

Watch2Gether on veebiplatvorm, mis võimaldab koos sünkroonselt vaadata video-meediat paljudest erinevatest keskkondadest, sealhulgas enda ekraani jagamist. Platvormil on palju funktsionaalsust, kuid kasutajal on keeruline selles orienteeruda ning kasutajaliidese disain on aegunud. Keskkonda on võimalik kasutada ilma pühendatud kasutajat tegemata.

### 2.1.2 twoseven

twoseven on veebiplatvorm, mis võimaldab koos sünkroonselt vaadata videot kõige populaarsematest meedia voogedastusplatvormidest. Platvormil on aktsepteeritavas koguses funktsionaalsust, kasutajaliides ja -kogemus jääb natukene alla Eestis disainitud lehtede tänaseks väljakujunenud standarditele.

### 2.1.3 Plex – Watch Together

Plex – Watch Together on Plex meedia serveri edasiarendus, mis võimaldab kasutajatel sünkroonselt kasutada Plex-i võimekust. Hetkel pakutavatest lahendustest kõige professionaalsem, kuid nõuab väga palju seadistamist lõppkasutajate poolt. See välistab aga antud lahenduse puhul suure hulga potentsiaalseid kliente.

### 2.1.4 Sync-Tube

Sync-Tube on veebiplatvorm, mis võimaldab YouTube’i videote koos sünkroonset vaatamist. Visuaalset kõige ilusam ja kasutajakogemuse poolest kõige lihtsam lahendus soovitud probleemile. Väga suureks miinuseks antud kontekstis on võimalike voogedastusplatvormide variatsioonide puudumine, kuna võimaldab ainult YouTube´i platvormi videote sünkroonset vaatamist.

## 2.2 Eksisteerivate platvormide tugevuste ja nõrkuste võrdlus

Tabel 1 - Olemasolevate lahenduste tugevused ja nõrkused

| Platvorm | Tugevused | Nõrkused |
|---|---|---|
| **Watch2Gether** | • Toetab paljusid voogedastus-platvorme, sealhulgas enda ekraani edastust<br>• Ei nõua kliendilt pühendatud kasutaja loomist<br>• Toetab suhtlemiseks mikrofoni ja veebikaamera kasutamist | • Aegunud disain ja keeruline kasutaja kogemus<br>• Täis reklaame |
| **twoseven** | • Toetab suhtlemiseks mikrofoni ja veebikaamera kasutamist<br>• Integreeritud *friends-list* ehk kasutaja(te) poolt loodud sõprade nimekiri | • Nõuab pühendatud kasutaja loomist<br>• Nõuab lisa veebibrauseri laiendi installimist, et toetada populaarseimaid meedia voogedastuskeskkondi<br>• Nõuab, et kõikidel ruumis viibijatel oleks autoriseeritud kolmanda osapoole konto |
| **Plex – Watch Together** | • Ilus ja lihtne kasutaja kogemus | • Nõuab lisatarkvara installimist<br>• Nõuab pühendatud kasutaja loomist |
| **Sync-Tube** | • Ilus ja lihtne kasutaja-kogemus<br>• Lihtne ja võimalusterohke ruumi kontroll<br>• Ei nõua kliendilt pühendatud kasutaja loomist | • Võimaldab vaadata meediat ainult YouTube´i keskkonnast |

## 2.3 Uue lahenduse skoop

Eksisteerivate lahenduste suurim probleem on nende eraldatus meedia originaali pakkuvast platvormist. See tekitab tihti problemaatilisi olukordi autoriõigusega ja hea kasutajakogemuse tagamisega. Antud probleemi lahendamiseks pakub autor välja uue,

modulaarse keskkonna loomise, mida voogedastusplatvormidel oleks endal võimalik lihtsasti integreerida otse, juba olemasoleva, platvormiga ning mida oleks soovi korral võimelised püsti panema ka iseseisvad lõppkasutajad, kui on soov enda video- või audiomeediat jagada sõprade või lähedastega.

Lahendus võimaldab kasutada kõiki mooduleid eraldiseisvalt, mis omakorda võimaldab lihtsat edasiarendust ning vajadusel ühe või mitme mooduli lihtsat asendamist kahjustamata kogu süsteemi töövõimekust.

Projekt piirdub vaid veebibrauseri põhise lahendusega, kuna see on kõige suurema tõenäosusega kättesaadav meedium potentsiaalsetele lõppkasutajatele, tänu veebi-brauserite väga laialdasele levikule.

Rakenduse eesmärgiks on autori enda meediaserveri täiustamine, kuid kuna on esinenud soovi projektimooduleid osta, on projekti käigus kirjutatud kood hetkel kinnine.

## 2.4 Eeldused

Loodava süsteemi eelduste määramisel arvestatakse, et projektil on lootus minna globaalsesse kasutusse, veebirakendus peab olema piisavalt ahvatlev, et olla võimeline iseseisvalt klientuuri suurendama puhtalt kliendilt kliendile soovitustega ning projekti moodulid peavad olema võimelised funktsioneerima täielikult iseseisvalt.

Nõudeid, mida soovitakse edasi arendada pärast praeguse skoobi valmimist, käsitletakse antud lõputöö lisades.

### 2.4.1 Nõuded lõppkasutaja vaatepunktist

- Tavakasutajana soovin siseneda platvormi lihtsalt ja kiiresti
- Tavakasutajana soovin saada reaalajas suhelda teistega ruumis olles
- Tavakasutajana soovin saada filmi vaadata sujuvalt, ilma hakkimisteta

### 2.4.2 Nõuded süsteemi haldaja vaatepunktist

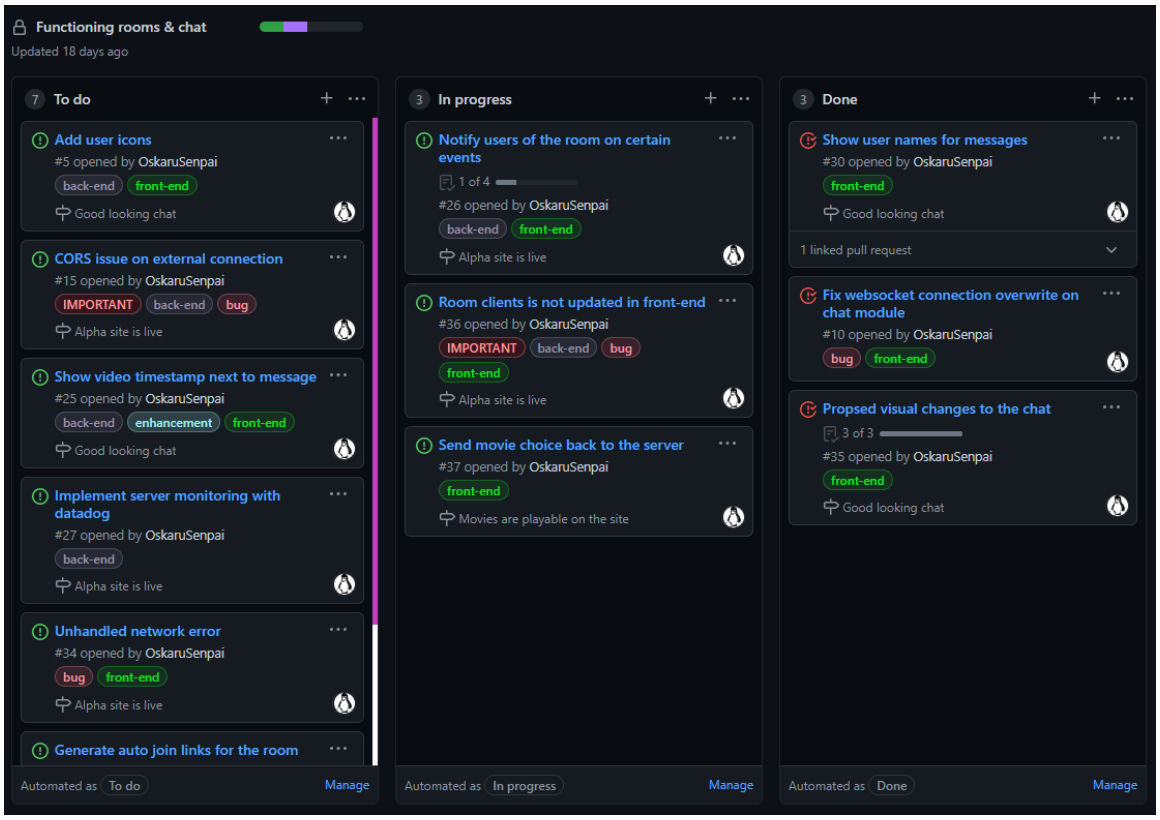
- Süsteem peab töötama võimalikult väikese administraatori sekkumisega
- Süsteemile peab olema lihtne teha muudatusi ja uuendusi

- Süsteem peab olema turvaline
- Süsteem peab olema modulaarne
- Süsteem peab genereerima kasulikku statistikat kasutajate ja filmide kohta
- Süsteem peab väljastama paljusõnalisi ja detailseid veateateid

### 2.4.3 Arenduskeskkonna valik

Arenduskeskkonna valikud jagunevad vastavalt: versiooni haldus, koodi arendus- & testimiskeskkond ning arendust toetav tarkvara.

Versiooni halduseks valiti Git-i tarkvara koos Github-i keskkonnaga. Git võimaldab lihtsat liikumist versioonide vahel ning tükeldatud arendust, mis oluliselt kiirendab arendusprotsessi. Github valiti tänu temaga integreeritud võimalustele nagu *Github Pages*, mis võimaldab tasuta serveerida staatilisi veebilehti ning mida kasutatakse ka esialgseks *front-end* mooduli serveerimise platvormiks klientidele. Lisaks on Githubil olemas ideaalne progressi jälgimise ja haldamise võimekus ülesannete, vahe-eesmärkide ja projektide näol, mis võimaldab Kanban arendusstiilis jälgida tegemata, parasjagu tegemisel olevaid ja tehtud ülesandeid (Joonis 1). Githubi ülesanded teeb eriti kasulikuks võimalus märgistada antud ülesande tüüp ja siduda see vastava koodi muudatusega.




Joonis 1 - Arenduse progressi järgimine Githubi keskkonnas

Githubi suurim nõrkus on tema nõue, et tasuta kasutajakontod saavad luua vaid avalikke projekte, kuid kuna Github pakub tudengitele tasuta *premium*-õigustega kontosid, koos väga suure hulga teiste tarkvarade integratsioonidega ja *premium*-õigustega kontodega, ei ole antud olukorras Githubi suurim nõrkus aktuaalne. [2]

Koodi arenduskeskkonnaks (IDE-ks) valiti Microsofti poolt loodud ja arendatav tasuta tarkvara Visual Studio Code tulenevalt selle erakordselt heale veebiarenduse toele, kõrgele integratsioonitasemele Git-i tarkvaraga ja Githubi keskkonnaga ning autori suurele kogemusele valitud IDE-ga.

Abistavate tarkvaradena tuleb välja tuua Insomnia, mis on modernne API-de disainimise ja testimise tarkvara, mida kasutati väga aktiivselt *backend* moodulite testimise ja arendamise käigus.

# 3 Süsteemi arhitektuur

Veebirakenduse lõppkasutajale ligipääsetavaks tegemise jaoks on vaja serverilahendust. Traditsiooniline veebirakenduse serveerimise meetod on omada või rentida füüsilist serverit. Lahenduse eeliseks on täielik süsteemi kontroll, mis võimaldab määrata täpseid riistvara tingimusi ja tarkvara optimeerida vastava riistvara jaoks. See nõuab tihti suurt esialgset kulu, kuid püsiv kulu on enamasti odavam kui sarnase võimsusega pilveteenus.

Teine võimalus tänapäeval veebirakenduse serveerimiseks on kasutada pilveteenuseid, kus arendaja loodud tarkvara jookseb kellegi teise loodud virtuaalmasina peal, kus arendajal on tihti piiratud valikud tema tarkvarale kasutatava riistvara suhtes. See-eest on pilveteenuste puhul teatud osa funktsionaalsusest garanteeritud teenusepakkuja poolt. Pilveteenuste eelised on tihti oluliselt lihtsam skaleeritavus, paigaldatavus, süsteemi taastamise võimalus kriisi olukorras, väiksem hooldusvajadus ning maksumuse vastavus veebirakenduses toimuva liiklusega.

Pilveteenused jagunevad omakorda järgnevatesse peamistesse kategooriasse [3]:

- IaaS ehk *Infrastructure as a Service*[1] – kõige traditsioonilisema lahendusega sarnasem pilveteenus. Pakub täielikku kontrolli vastava virtuaalmasina üle, võttes arendajalt ära vajaduse hooldata füüsilist serverit. Tuntuimad näited on AWS[2], Microsoft Azure[3] ja DigitalOcean[4].

- PaaS ehk *Platform as a Service*[5] – vastutab lisaks füüsilisele serverile ka operatsioonisüsteemi, vahevara ning rakenduse keskkonna haldamise eest. Sobib hästi, kui on soov keskenduda äriloogika arendamisele, kuna nõuab üsna vähest

[1] Pilvepõhine teenus, mis haldab infrastruktuurilisi elemente nagu server

[2] Amazon Web Services on Amazoni poolt pakutav erinevate virtuaalmasinate kogum

[3] Microsoft Azure on Microsofti poolt pakutav erinevate virtuaalmasinate kogum

[4] DigitalOcean on pilveteenuste pakkumisele pühendunud firma

[5] Pilvepõhine teenus, kus teenusepakkuja haldab kõike, mis ei ole seotud rakenduse enda või äriloogikaga

seadistamist. Peamisteks nõrkusteks on kolmanda osapoole raamistike ja teekide installimise võimetus ja toetatud keelte vähesus. Näiteks ei toeta paljud PaaS platvormid *TypeScript*-is kirjutatud programmide kompileerimist ja jooksutamist. Lisaks on tihti süsteemidele lisatud teenusepakkuja enda vahevara andmete ja statistika kogumiseks tekitades potentsiaalseid turvaprobleeme, kui tegemist on sensitiivsete andmete kogumise, hoidmise või käsitlemisega. Populaarseimad näited on Heroku[1] ja Google App Engine[2]

- SaaS ehk *Software as a Service*[3] – pakub juba valmis toote sarnast lähenemist, kus arendaja ei pea tegelema seadistamisega ega riistvaraliste või tarkvaraliste muudatustega. Populaarseimad näited on Google Apps (näiteks Google Drive), DropBox, Slack[4] ja Github Pages[5].

- FaaS ehk *Function as a Service*[6] – üks kõige uuem ja kiiremini populaarsust koguv versioon pilveteenustest, mis töötab sündmuste põhisel arhitektuuril, kus arendaja peab ainult kirjutama ühe funktsiooni ning määrama, mis olukorras seda funktsiooni jooksutatakse. Selle suured eelised on täielik keskendumine äri-loogikale, täielikult automaatne skaleeritavus ning maksumuse täpne vastavus kasutusele. FaaS lahenduste nõrkuseks on väike keelte toetatavus, kus enamus pakkujaid toetavad maksimaalselt nelja kõige populaarsemat veebiarenduse keelt ning nii nimetatud *cold start* [4], kus funktsiooni esmane käitamine üle mingi aja-perioodi võtab oluliselt rohkem aega kui tavaliselt. Tuntuimad näited on AWS Lambda[7], Microsoft Azure Functions[8] ja Google Functions[9].

[1] Heroku on Heroku Inc. poolt loodud ja arendatud PaaS lahenduste pakkumisele keskendunud platvorm

[2] Google App Engine on Google´i poolt pakutav PaaS platvormide kogum

[3] Pilvepõhine teenus, kus teenusepakkuja haldab nii platvormi kui ka funktsionaalsuseid

[4] Slack on Slack Technologies poolt loodud ja arendatud ärisuhtlusplatvorm

[5] Github Pages on Githubi poolt pakutav staatiliste veebilehtede serveerimise platvorm

[6] Pilvepõhine teenus, kus teenusepakkuja haldab kõike, peale ühe arendaja poolt loodud funktsiooni

[7] Amazoni poolt pakutud FaaS platvorm

[8] Microsofti poolt pakutud FaaS platvorm

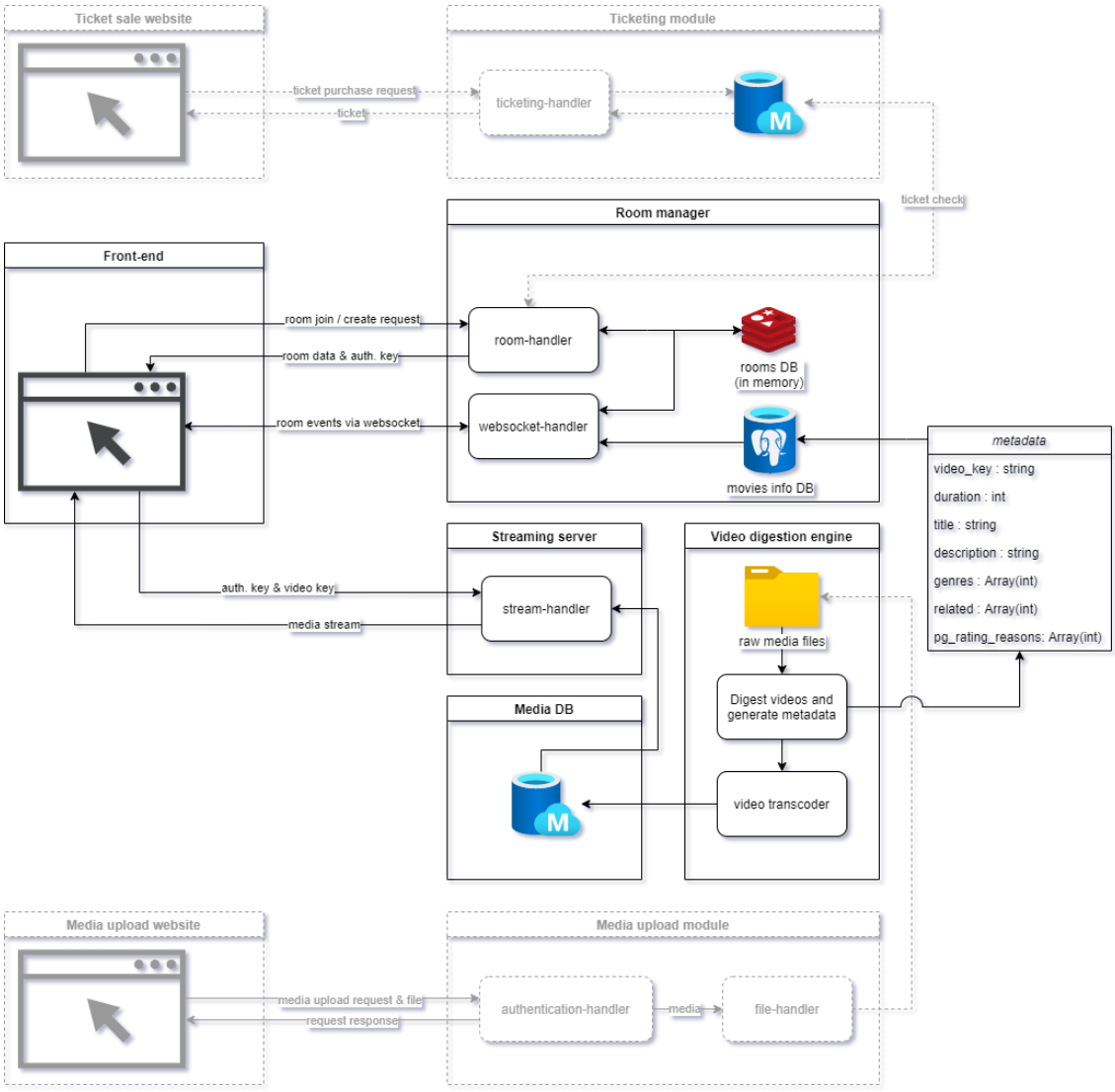
[9] Google´i poolt pakutud FaaS platvorm

Tänu süsteemi modulaarsusele on võimalik kasutada erinevaid tehnoloogiaid erinevate moodulite puhul vastavalt mooduli eesmärgile, nõutele ning arendaja soovile. Autor soovitab moodulid paigaldada vastavalt:

- *Front-end* – PaaS platvorm serveerimaks staatilist veebi lehte (Github Pages).
- *Room manager* – IaaS platvorm tagamaks usaldusväärset ja reaalajas töötavaid ruume.
- *Streaming server & Media DB* – samal IaaS platvormil võimaldades kiiret meedia serveerimist olenemata lõppkasutaja globaalsest asukohast.
- *Video digestion engine* – Pühendatud server võimaldamaks riistvara ja tarkvara optimeerimist meediafailide töötlemiseks, kus süsteemi vastuste kiirus internetist tulevatele päringutele ei ole oluline. Teine võimalus on videotöötlusele pühendatud IaaS platvorm.

## 3.1 Andmeliikluse arhitektuur

Andmeliikluse skeemil (**Error! Reference source not found.**) on välja toodud kõik tehtud moodulid ning ka neli moodulit, (*Ticket sale website*, *Ticketing module*, *Media upload website* ja *Media upload module*) mida antud töö skoop ei sisalda, kuid millega tuleb arvestada süsteemi loomisel, kui tulevikus potentsiaalne süsteemi haldaja soovib projekti kasutada raha teenimiseks või projekti edasi arendada.




Joonis 2 - Andmeliikluse skeem

## 3.2 Andmebaaside arhitektuur

### 3.2.1 Andmebaasi tehnoloogia valik

Kaalutud tehnoloogiad:

- MySQL – üsna kiire ja lihtsasti õpitav andmebaasi tarkvara. MySQL on lihtsasti skaleeritav ja lubab erineva õigustega kasutajate loomist andmebaasi kasutamiseks. On avatud lähtekoodiga, kuid kasutab GPL litsentsi, mis keelab kommertseesmärkidel andmebaasi kasutamist. Kommertseesmärkidel kasutatav litsents on ostetav. [5]

- PostgreSQL – avatud lähtekoodiga ning suure kasutajate baasiga, mille tõttu omab ka väga suurt tuge. Teoreetiliselt võimaldab andmebaasi lõpmatut skaleerimist ning erinevate õigustega kasutajate loomist. Lihtsate operatsioonide käitamisel töötab aeglasemalt kui MySQL [6].

- MongoDB – kergelt skaleeritav, võimaldab paremat protsessimistõhusust ning paindlikke päringuid, kui neid ei tehta üle mitme dokumendi, mis tähendab, et andmebaas ei sobi relatsiooniliste andmete jaoks ning ei oma sisemist transaktsioonide haldust. [7]

Valitud tehnoloogiaks osutus PostgreSQL, tänu selle avatusele, laiale kasutajaskonnale, väga headele monitoorimise rakendustele (pgAdmin[1]) ning autori soovile antud tehnoloogiat paremini õppida.

Järgides modulaarsuse põhimõtet, on andmebaasid võimalikult väikseteks osadeks segmenteeritud võimaldades tulevikus lihtsat täiendamist ja osade välja vahetamist.

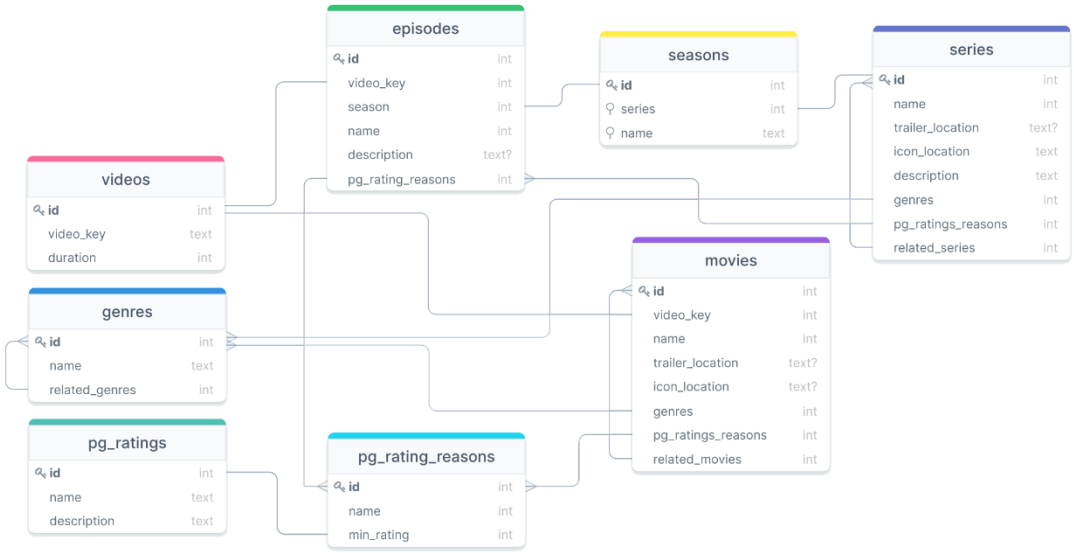
### 3.2.2 *Room manager* mooduli andmebaasi arhitektuur

Arenduse käigus käidi läbi neli iteratsiooni andmebaasi disaine ja loogikaid. Andmebaas on loodud DrawSQL[2] keskkonnas, mis võimaldab graafilist andmebaasi disainimist ning

[1] Avatud lähtekoodiga PostgreSQL andmebaaside administreerimise rakendus: https://www.pgadmin.org/

[2] Andmebaasi disaini ja jagamise keskkond: https://drawsql.app/

hiljem vastava andmebaasi paigaldusfaili alla laadimist, muutes andmebaasi paigaldamise väga lihtsaks.




Joonis 3 – *Room manager* mooduli andmebaasi arhitektuuri skeem

Kõikides tabelites on igal real UUID tabelite indekseerimiseks ja üksteisega sidumiseks.

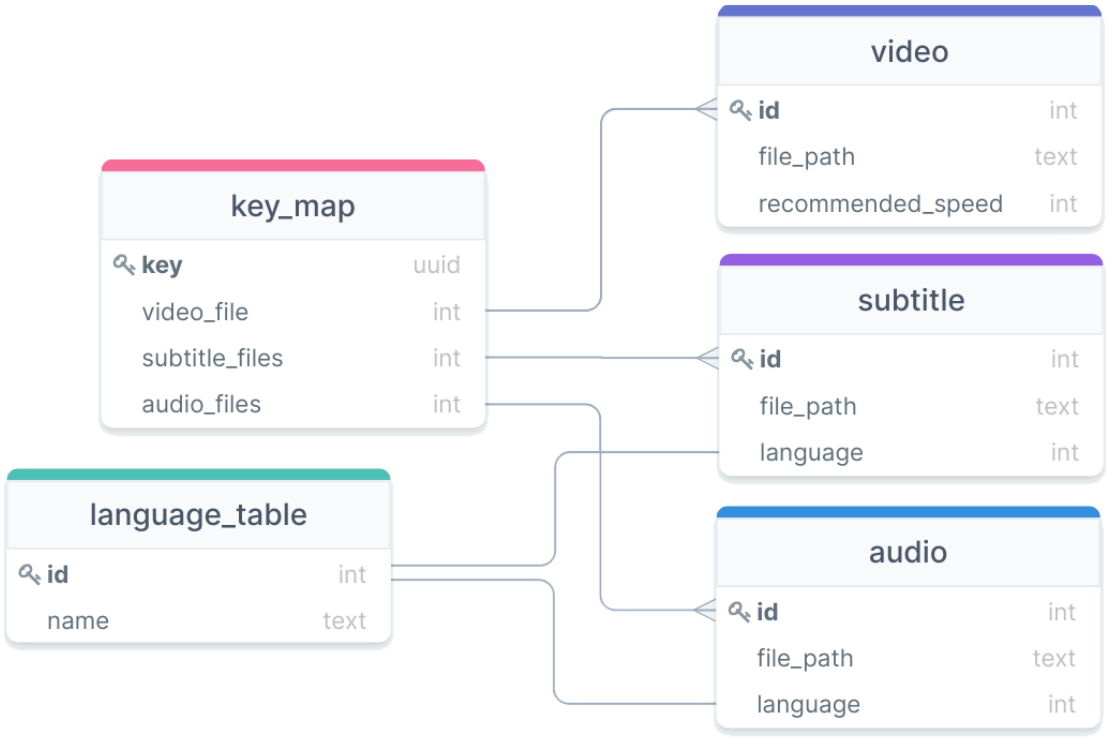
Andmebaasi moodulite kirjeldused:

- *videos*
  - *video_key* – video faili võti *Streaming server* mooduli jaoks
  - *duration* – video kestvus
- *genres*
  - *name* – žanri nimi
  - *related_genres* – massiiv sarnaste žanrite tabeli UUID-dega
- *pg_ratings*
  - *name* – vanusepiirangu nimi
  - *description* – vanusepiirangu pikem kirjeldus
- *pg_rating_reasons*
  - *name* – vanusepiirangu põhjuse nimi
  - *min_rating* – minimaalne vanusepiirang videol vastava põhjendusega

- *movies*
    - *video_key* – vastava *videos* tabeli elemendi UUID
    - *name* – filmi tiitel
    - *trailer_location* – vastava filmi treileri URL
    - *icon_location* – vastava filmi postri URL
    - *genres* – massiiv filmi kohta sobivate žanrite tabeli UUID-dega
    - *pg_rating_reasons* – massiiv filmi kohta käivate vanusepiirangu põhjuste UUID-dega
    - *related_movies* – massiiv sarnaste filmide tabeli UUID-dega
- *episodes*
    - *video_key* – vastava *videos* tabeli elemendi UUID
    - *season* – vastava seriaali hooaja UUID
    - *name* – episoodi nimi
    - *description* – episoodi lühikirjeldus
    - *pg_rating_reasons* – massiiv episoodi kohta käivate vanusepiirangu põhjuste UUID-dega
- *seasons*
    - *series* – vastava seriaali UUID
    - *name* – hooaja nimi / number
- *series*
    - *name* – seriaali nimi
    - *trailer_location* - seriaali treileri URL
    - *icon_location* – seriaali postri URL
    - *description* – seriaali lühikirjeldus
    - *genres* – massiiv seriaali kohta sobivate žanrite tabeli UUID-dega
    - *pg_rating_reasons* – massiiv seriaali kohta käivate vanusepiirangu põhjuste UUID-dega
    - *related_series* – massiiv sarnaste seriaalide tabeli UUID-dega

### 3.2.3 *Media DB* mooduli andmebaasi arhitektuur

*Media DB* mooduli andmebaasi (Joonis 4) disainimsel oli peamisteks eesmärkideks süsteemi kiire lugemiskäskude täitmine ja lihtne arhitektuur.




Joonis 4 - *Media DB* mooduli andmebaasi arhitektuuri skeem

Andmebaasi moodulite kirjeldused:

- *key_map*
  - *key* – vastava video UUID
  - *video_file* – võtmele vastav *video* tabeli ID
  - *subtitle_files* – võtmele vastav *subtitle* tabeli ID-de massiiv
  - *audio_files* – võtmele vastav *audio* tabeli ID-de massiiv
- *video*
  - *id* – tabeli sisendi ID
  - *file_path* – reaalse faili asukoht kõvakettal
  - *recommended_speed* – soovitatud minimaalne interneti kiirus vastava video sujuvaks voogedastuse saavutamiseks

- *subtitle*
    - *id* – tabeli sisendi ID
    - *file_path* – reaalse faili asukoht failisüsteemis
    - *language* – subtiitri failile vastav *language_table* tabeli ID
- *audio*
    - *id* – tabeli sisendi ID
    - *file_path* – reaalse faili asukoht failisüsteemis
    - *language* – subtiitri failile vastav keelte tabeli ID
- *language_table*
    - *id* – tabeli sisendi ID
    - *name* – keele nimetus

Andmebaasi tulevate päringute esimene samm on leida üles vastavad failid otsitud võtme kohta. Päringu lisaargumentidena võivad olla määratud:

- Ühenduse kiirus kliendiga, et tagastada optimaalse resolutsiooniga videofaili asukoht.
- Subtiitrite valimise soov ja vastavalt mis keelelisi subtiitreid otsitakse.
- Võimalikud helifailid vastava videofaili kohta

Andmebaasile on planeeritud ka teha pidevaid päringuid videole kõikide vastavate võimalike subtiitrite ja helifailide kohta, et anda kliendile nimekiri, mille hulgast on võimalik valikut teha. Antud arhitektuuri tugevaks küljeks on võimalus sama filmi samal ajal vaatavatel kasutajatel vaadata filmi neile kõige paremini sobivas keeles ja mitte olla lukus valitud keelega.

# 4 Moodulid

Planeeritud on 9 moodulit, millest neli (*Ticket sale website*, *Ticketing module*, *Media upload website* ja *Media upload module*) ei ole antud lõputöö skoobis.

Kõikide moodulite tehnoloogiate valik on tehtud vastavalt mooduli eesmärgile ja nõuetele ning autori arvamusega tehnoloogia sobivusest antud ülesande täitmiseks.

Moodulite nõuded on järjestatud ülevalt alla vastavalt kui tähtsad nad antud moodulile on.

## *4.1 Front-end*

Mooduli eesmärk on hallata suhtlust kasutaja ja teiste moodulite vahel ning siduda kliendi jaoks teiste moodulite funktsionaalsus, pakkudes võimalikult sujuvat kasutajakogemust.

### 4.1.1 Nõuded moodulile

- Moodul peab töötama ühte moodi kõikide populaarsemate brauserite peal
- Kasutajale näidatav veebileht peab saama *Google Lighthouse*-i testis kõik tulemused üle 95%
- Kasutajakogemus peab olema võimalikult sujuv ka arvutit harva kasutavatele inimestele

### 4.1.2 Tehnoloogia valik

Kaalutud tehnoloogiad [8]:

- React – Facebooki poolt loodud ja arendatud JSX-i kasutav JavaScripti pakett, mis on mõeldud kasutajaliideste loomiseks. ReactJS on deklaratiivne raamistik võimaldades kiiret reageerimisaega ning komponendipõhist arhitektuuri, muutes rakenduse kergesti skaleeritavaks.

- Angular – Google´i poolt loodud ja arendatud JavaScriptil põhinev raamistik. Angular on mõeldud suurte mitmeleheliste rakenduste loomiseks.

- Vue – endise Google´i töötaja poolt loodud ja arendatud JavaScriptil põhinev raamistik, sobib hästi suurematele ühelehelistele rakendustele.

Kõik kaalutud valikud on vabavaralised projektid, mis sobivad kiireks projekti arenduseks.

Valitud tehnoloogiaks osutus React tänu selle struktuuri potentisaalsele modulaarsusele, skaleeritavusele, efektiivsusele ning suurele populaarsusele arendajate hulgas, mis võimaldab tulevikus projekti kiiret edasiarendamist kolmandate osapoolte poolt.

### 4.1.3 Mooduli töö kirjeldus

Moodul näitab kasutajatele kahte või kolme peamist vaadet, olenevalt kas ruumi sisenemisel oli juba film ette määratud vastavalt *Ticketing Module*-ist saadud koodile:

- Ruumi sisenemisvaade (Joonis 5) – vaatest on kaks versiooni vastavalt kas *Ticketing Module* on tehtud või mitte

- Filmi valiku vaade – ei näidata, kui ruumil on juba ette määratud film, mida vaadata

- Filmi vaatamise vaade

Kuna antud lõputöö skoobis ei ole kaasatud *Ticketing Module´*it, siis on koodina realiseeritud ainult vaadete versioonid, kus *Ticketing Module´*it ei kasutata. Teised vaadete versioonid on realiseeritud Figma[1] vaadete arendamise keskkonnas. Lisavaadete realiseerimine koodis on tehtud erakordselt lihtsaks tänu Reacti modulaarsele vaadete ehitamise põhimõttele ja Figma võimalusele stiili reeglite kuvamisele CSSi keeles.

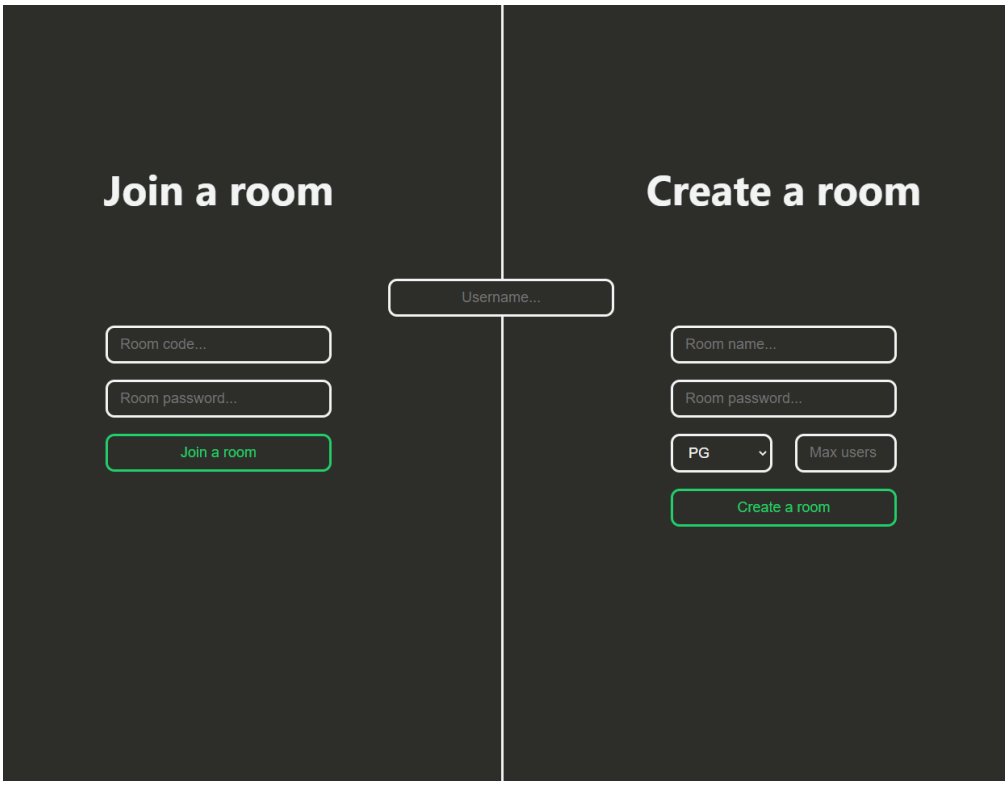
Vaadete disainimisel oli eesmärgiks minimalistlik ja intuitiivne kasutajaliides. Kõik vaated on disainitud ujuvalt, ehk suudavad kohanduda erinevate ekraanide suurustega. Hetkel puudub vaadetel spetsiaalne disain mobiilseadmete jaoks, tulenevalt projekti

[1] Vektor jooniste ja veebilehe UI / UX disaini loomise tarkvara

teostamiseks võimalikust piiratud ajast. Planeeritud on eraldi vaated mobiili jaoks, kus mobiili saab kasutada tavalise vaatava kliendina, kuid ka spetsiaalne mobiilile kohandatud, eksklusiivne vaade, mis spetsialiseerub vastava ruumi kontrollimisele, teisisõnu ruumi pult.

Kõik veebiliiklus teiste moodulitega toimub üle krüpteeritud HTTPS ühenduse, tagamaks turvalise JWTide ja päringute transpordi *front-endi* ja teiste moodulite vahel. Ühendus kasutab vastavalt vajadusele kas REST tüüpi või *WebSocketi* päringut.

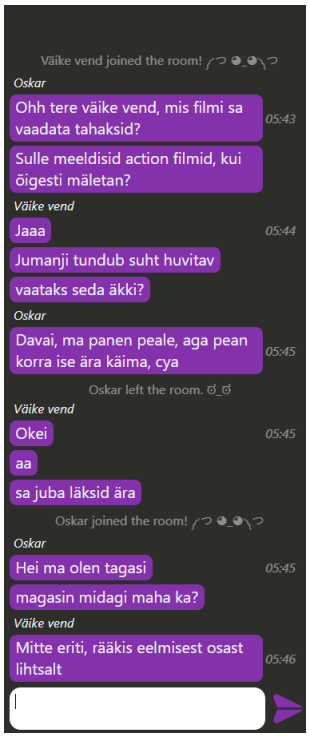
Planeeritud on kasutada PaaS pilveteenust mooduli lõppkasutajale kättesaadavaks tegemiseks, seega pole vaja vastavaid lisateenuseid nagu Nginx[1] või Apache[2]. Kui on soov serveerida moodulit IaaS pilveteenuse või traditsioonilise serveri kaudu on vaja installida ja seadistada vastavalt soovile serveerimisveebilehtede serveerimisplatvorm.




Joonis 5 - Ruumi sisenemisvaate näide, kui *Ticketing Module´*it ei ole kasutuses

[1] Tasuta, avatud lähtekoodiga, suure jõudlusega HTTP-server, pöördproksi (*reverse-proxy*) ning IMAP / POP3 puhverserver

[2] Kõige populaarsem HTTP-server veebiteenuste ja veebilehtede serveerimiseks




Joonis 6 – Suhtlusakna näide

## *4.2 Room manager*

Mooduli eesmärk on hallata ruumide loomist, ruumides olevaid kasutajaid, ruumidesse sisenemist ja ruumidest lahkumist ning ruumide sünkroonis hoidmist erinevate kasutajate jaoks.

### 4.2.1 Nõuded moodulile

- Moodul peab olema stabiilne
- Moodul peab töötama peaaegu reaalajas
- Moodul peab olema monitooritav

### 4.2.2 Tehnoloogia valik

Kaalutud tehnoloogiad

- PHP – Rasmus Lerdorfi loodud ja PHP grupi poolt arendatav populaarne üldotstarbeline skriptimiskeel, mis on spetsialiseerunud veebiarendusele.

Suurimaks puuduseks antud projekti puhul on PHP aegunud loomus ja väga nõrk tugi *WebSocket* ühenduste loomiseks.

- NodeJS – Ryan Dahli loodud avatud lähtekoodiga populaarne JavaScripti koodi jooksutamise keskkond, mis seob Google'i arendatud V8 JavaScripti jooksutamismootori, *event loopi*[1] põhise käitlustsükli ja madalatasemelise sisend/väljund API.

- Deno – 2020 aasta mais välja tulnud, Ryan Dahli loodud JavaScripti ja TypeScripti koodi jooksutamise keskkond, mis seob Google'i arendatud V8 JavaScripti jooksutamismootori ning Rust programmeerimiskeele. Peetakse paljude poolt NodeJS järeltulijaks. Paneb rõhku arenduse produktiivsusele ja lõpptulemuse turvalisusele [9].

- Python – Guido van Rossumi poolt loodud ja Python Software Foundationi poolt arendatav, interpreeritav, kõrgtasemeline, üldotstarbeline programmeerimiskeel. Üks kõige populaarseimaid programmeerimiskeeli projekti teostamise ajal.

- C++ – C programmeerimiskeele edasiarendus Bjarne Stroustrupi poolt, kaalutud tehnoloogiate nimekirjas ainuke kompileeritav keel, mis annab kiiruse eelise teiste kaalutud tehnoloogiate ees.

Valitud tehnoloogiaks osutus NodeJS. Deno, millel on potentsiaal olla turvalisem ja lihtsamini arendatav versioon NodeJSist, on projekti esialgse arenduse käigus liiga uudne tehnoloogia, mis ei toeta veel kõiki samu funktsionaalsusi ja teeke, mis on välja arenenud NodeJSi eluaja jooksul. Projekti enda eluaja jooksul on suur tõenäosus, et moodul kirjutatakse ümber Deno baasile, kui Deno jaoks on välja arendatud piisavalt vaja-minevat funktsionaalsust.

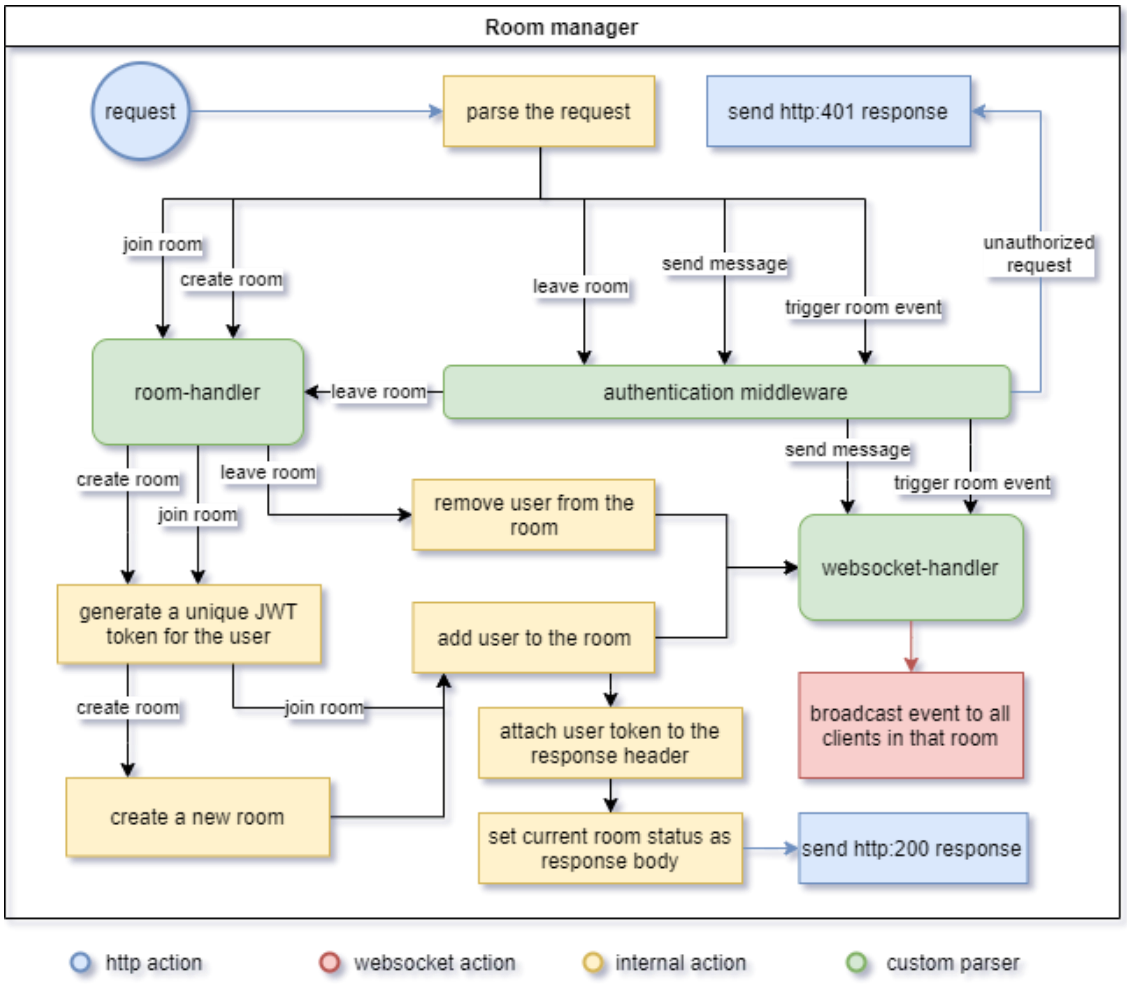
[1] Käskude käitulustsükli meetod, kus peaprotsess saadab ja ootab sündmusi või sõnumeid programmi sees

### 4.2.3 Mooduli arhitektuur

*Room manager* moodul (Joonis 7) on võimeline töötlema 5 peamist tüüpi päringuid:

- Ruumi loomine – skeemil *create room* – välja lülitatud, kui *Ticketing Module* on ühendatud
- Ruumiga ühinemine – skeemil *join room*
- Ruumist lahkumine – skeemil *leave room*
- Sõnumite saatmine ruumis olevate klientide vahel – skeemil *send message*
- Ruumis sündmuse välja kutsumine – skeemil *trigger room event*

*Room manager* moodul peab tegelema ka kasutajate autentimisega teiste moodulite jaoks. Autentimise jaoks vajalike andmete edastamiseks kasutatakse *JSON Web Tokeneid* (JWT), mis võimaldab kompaktselt saata allkirjastatud infot kasutades JSON formaati. Pärast serveri poolt JWT genereerimist ja allkirjastamist, saadetakse JWT kliendile, kes võib seda kasutada teistele moodulite tõestamaks, et klient on autoriseeritud päringu tegemiseks.




Joonis 7 - *Room manager* mooduli sisene andmeliiklusskeem

JWT koosneb kolmest komponendist: *header* ehk päis, *payload* ehk tokeni omaniku kohta käiv informatsioon ning *signature* ehk allkiri. Päis määrab ära tokeni ehk sõne tüübi (JWT) ning algoritmi (näiteks HMAC SHA256). *Payloadis* olev info võib sisaldada kasutaja nime ning rolli ning allkirja, mida kasutatakse teenuse poolt sõnumi verifitseerimiseks. [10]

## *4.3 Streaming server*

Mooduli eesmärk on tagada kiire, stabiilne ja turvaline meedia voogedastus kliendile.

### 4.3.1 Nõuded moodulile

- Moodul peab töötama reaalajas
- Moodul peab olema suuteline kannatama suuri andmevoogusid

- Moodul peab olema stabiilne
- Moodul peab olema võimeline kiiresti ja potentsiaalselt lõpmatult suurenema

### 4.3.2 Kaalutud tehnoloogiad

- C++
- Deno
- NodeJS
- Python
- PHP

Antud mooduli suur rõhk reaalajalise funktsioneerimise nõudlusel suurendas oluliselt C++ kasutamiseks kaalumise osakaalu. Vaatamata sellele sai valitud tehnoloogiaks NodeJS, kuna C++ kasutamine veebiarenduseks on oluliselt keerulisem ja aeganõudvam ning arvestades soovi tulevikus projekti edasi arendada ja kaasata teisi arendajaid, on tõsiasi, et C++ veebiarendajad on suhteliselt vähe suureks takistuseks C++ kasutamisele antud mooduli arendamisel. Sarnaselt *Room manager* mooduliga on suur tõenäosus, et projekti eluaja jooksul kirjutatakse moodul ümber Deno baasile, kui Deno on saavutanud piisavalt suure teekide toe ja populaarsuse.

### 4.3.3 Mooduli töö kirjeldus

Moodul kõigepealt kontrollib, kas päringu päises olevate küpsiste hulgas on *Room manager* mooduli poolt allkirjastatud JWT. Kui autoriseeritud JWT on päringuga kaasa pandud, siis hakatakse töötlema päringu sisu. *Movies DB* mooduli abiga leitakse soovitud meediafailid ja serveeritakse need kliendile.

## *4.4 Movies DB*

Mooduli eesmärk on hoiustada optimaalses kodeeringus meediafaile ning neid edastada voogedastusmoodulile.

### 4.4.1 Nõuded moodulile

- Kiire andmebaasist lugemise operatsioon

- Võime sooritada sügavalt seotud päringuid
- Võime toetada väga suurt arvu kirjeid
- Võimalikult väike andmebaasile kuluv ketta maht

### 4.4.2 Tehnoloogia valik

Kaalutud tehnoloogiad:

- MySQL
- PostgreSQL
- MongoDB

Valitud tehnoloogiaks osutus taaskord PostgreSQL tänu tasuta litsentsile, mis võimaldab moodulit kasutada kommertseesmärkidel.

### 4.4.3 Mooduli töö kirjeldus

Moodul töötab lihtsa andmebaasi põhimõttega. Mooduli sisendiks on *Streaming server* moodulilt saadud parameetrid, millele moodul tagastab soovitud andmed.

## *4.5 Video digestion engine*

Mooduli eesmärk on töödelda võimalikult suurt hulka erinevas kodeeringus meediafaile, muutes nende kodeeringu optimaalseks voogedastuse jaoks, vähendades faili kogusuurust ning genereerides metaandmeid failide kohta nende paremaks kategoriseerimiseks ning töötlemiseks.

### 4.5.1 Nõuded moodulile

- Moodul peab olema võimeline töötlema väga erineva kodeeringuga sisendfaile
- Moodul peab raporteerima oma staatust pidevalt
- Moodul peab olema võimeline looma ja hoidma järjekorda
- Moodul peab olema võimeline ise üle saama sisendfaili kodeeringuvigadest või nendest raporteerima

### 4.5.2 Tehnoloogia valik

Kaalutud tehnoloogiad:

- VLC – ei võimalda videofaili formaadi muutmist ilma ümberkodeerimiseta
- FFmpeg – kõige populaarsem videofailide töötlemise programm [11]
- HandBreak – ei võimalda videofaili formaadi muutmist ilma ümberkodeerimiseta

Valitud tehnoloogiaks osutus FFmpeg koos *bash* skriptidega tänu selle suurepärasele kiirusele ja toetatud kodeeringute rohkusele. Lisaks sellele on FFmpeg-l hea käsurea tugi võimaldades lihtsamat automatiseerimist. Veel võimaldab FFmpeg-i kasutamine tulevikus lihtsalt *bash* skriptide väljavahetamist pühendatud C, C++ või C#-s kirjutatud rakenduse jaoks, tänu ametlikult toetatud C ja C# raamistikele. Võimalik on ka *bash* välja vahetatud Pythonis kirjutatud skriptide vastu, kuigi Pythoni jaoks kirjutatud FFmpeg raamistikud ei ole ametlikult toetatud. [12]

Kaalutud videokoodeksid [13]:

- H.264 / AVC (*Advanced Video Coding*)
- H.265 / HEVC (*High-Efficiency Video Coding*)
- H.266 / VVC (*Versatile Video Coding*)
- AV1
- VP9

Autori enda majutatud platvormi jaoks valitud video kodeeringuks sai Google´i poolt arendatud vabavaraline VP9 koodeks tänu selle erakordselt heale video kvaliteedi ja faili suuruse suhtele võrreldes H.265-ga, voogedastusstandardite toele ja vabavaralisele litsentsile. VP9 koodeksi üheks suureks miinuseks on Safari toe puudumine.

Väike lisaprobleem tekkib seoses ümberkodeerimise suure ajakuluga. Enamus algallikaid kasutab aegunud H.264 kodeeringut, mille ümberkodeerimisel topelt ülekäimise meetodil kulub 2 kuni 17 (olenevalt algmaterialist) korda rohkem aega, kui on originaalvideo faili kestus. Tänu mooduli võimalusele töötada järjepidevalt ja autonoomselt ei ole tegu probleemiga, mis segab rakenduse toimimist ning mille lahendamine pole akuutne.

### 4.5.3 Mooduli töö kirjeldus

Moodul on jagatud nelja kausta:

- *scripts* – sisaldab erinevaid *bash* skripte
    - töötlemise skript – Kui *input*-kaustas on fail nimega *.lock,* siis lõpetab operatsiooni, vastasel juhul, loob *input*-kausta uue tühja faili nimega *.lock*, pärast mida võtab kõik meediafailid hetkel *input*-kaustas ja hakkab neid ükshaaval töötlema, muutes videokodeeringut, jaotades failid eraldi audio- ja videofailideks, ning genereerides JSON formaadis metaandmete faili ning kustutades *input*-kaustast originaalfailid. Kui skript on failide töötlemise lõpetanud, kutsutakse välja üleslaadimise skript.
    - üleslaadimise skript – võtab *output*-kaustast meedia- ja metaandmete failid ja laeb need üles vastavatesse andmebaasidesse. Kui üleslaadimine on edukas, kustutab *input*-kaustast *.lock* faili.
    - logimise skript – skript, mida kutsutakse välja, kui midagi tahetakse kirjutada logidesse. Tegeleb õigesse faili kirjutamisega ja sõnumi õiges formaadis kirjutamisega. Käsurea argumentidena võtab sisse sõnumi tüübi (*error, warn, info*) ja sõnumi teksti ning kirjutab sõnumi õigel kujul õigesse faili.
- *input* – sisaldab töötlemata meediafaile.
- *output* – sisaldab optimeeritud ja töödeldud video-, audio- ja metaandmete faile.
- *logs* – sisaldab logifaile videotöötlusprotsessidest ja ajaloost.

Töötlemise skript kutsutakse välja iga teatud aja tagant. Antud lahendus tagab selle, et skriptid jooksevad kõige optimaalsemalt ja võimaldab uusi faile lisada *input*-kausta samal ajal, kui juba olemasolevaid faile töödeldakse.

Tänu V9 kodeeringu valikule on siiani töödeldud meediafailide suurust vähendatud keskmiselt 8% ainult ümberkodeerimise tulemusena.

# 5 Tekkinud probleemid

## 5.1 CORS-i vead erinevate võrkude vahel

Kui on soov rakendada modulaarsuse täispotentsiaali, paigutades kõik moodulid erinevatesse nendele optimeeritud keskkondadesse, tuleb tegeleda palju moodulite vaheliste õiguste ja ühenduste seadistamisega. Algselt tekitas probleeme just CORS-i reeglite seadistamine, mis võttis oluliselt rohkem aega, kui algselt oli ülesandele planeeritud.

## 5.2 Hilised arhitektuurilised muudatused

Üsna hilisel ajal projekti arenduse käigus selgus, et parimaks kasutajakogemuse pakkumiseks ei ole võimalik olemasoleva arhitektuuriga optimaalset lahendust luua. Parim lahendus, mis tagaks ka tulevikus jätkusuutliku arendusprotsessi, oli muuta andme-suhtlus- ja -käitlusarhitektuuri *Room manager*, *Front-end* ja *Streaming server* moodulite puhul, mis nõudis antud moodulite suurt ümber kirjutamist. Tänu algfaasis otsustatud modulaarsele lähenemisele oli võimalik teatud osad koodist uut arhitektuuri sisse viies taaskasutada.

## 5.3 *Video digestion engine* mooduli jaoks valitud riistavara

Enne mooduli arendamise alustamist oli teada suur ajaline kulu, mida mooduli operatsioonid nõuavad. Selle jaoks planeeritud optimeering oli kasutada GPU kiirendatud videokodeerimist. Optimeerimise ajal selgus, et moodulile valitud GPU (Nvidia GeForce 730M) ei ole toetatud FFmpeg-i poolt. Probleemi lahenduseks on mooduli liigutamine masinasse, millel on FFmpeg-i poolt toetatud GPU.

# 6 Tulemused ja edasiarenduse plaanid

Kõikide planeeritud moodulite põhifunktsionaalsus sai valmis ja optimeeritud vastavalt moodulile kehtivate nõuetega. Kõik skoobis olevate moodulitega seotud Githubis olevad ülesanded ja vahe eesmärgid said täidetud ning uued, arenduse jätkuks mõeldud ülesanded loodud. Veebirakendust on võimalik serveerima hakata, testimaks serveritele kuluvat jõudlust suurte koguste kasutajate puhul.

Edasiarenduse plaanid ja võimalused:

- *Video digestion engine* mooduli liigutamine masinale, millel on FFmpeg poolt toetatud GPU, vähendamaks oluliselt ümberkodeeringuks kuluvat aega
- Luua ja implementeerida mobiilile kohandatud vaated *Front-end* moodulile
- Luua antud töö skoobist välja jäänud neli moodulit
- Luua parem UI/UX disain *Front-end* moodulile
- Luua automaattestid kõikidele moodulitele
- Lisada erinevate keelte tugi *Front-end* moodulile
- Integreerida moodulitesse automaatsed dokumentatsiooni generatsiooni süsteemid
- Luua PWA mobiilseadmete jaoks
- Muuta suhtlusakna visuaale, et erinevate inimeste sõnumid oleksid üksteisest paremini eristatavad

# 7 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli valmis saada veebiplatvorm, kus oleks võimalik kasutajatel koos samaaegselt samu filme ja seriaale vaadata olles üksteisest füüsiliselt eraldatud ning reaalajas vaadatava meedia kohta oma muljeid ja arvamusi vahetada.

Lõputöö esitamise ajaks on projektist valmis tehtud algselt planeeritud viis moodulit, kus kõik moodulid täidavad antud moodulile esitatud nõudeid. Projekti on võimalik hakata kasutama ilma muudatusi tegemata, järgides moodulite seadistamiseks Giti koodivaramus välja toodud samme.

Suurimaks probleemiks osutus hiline arhitektuuriline muudatus, kuid tänu modulaarsele süsteemi disainile vähendades ümberkirjutatava koodi osakaalu ja ette arvestatud puhverajaga eesmärkide täitmiseks ei olnud probleem ületamatu.

Kõige aeganõudvam ja autori hinnangul ebaefektiivsem ajakulu esines *front-end* mooduli arenduse käigus, kuna autori peamine kogemus on olnud etteantud disainide põhjal rakenduste loomisel ja mitte UI/UX disainide loomisel.

Autori hinnangul on projekt olnud siiani edukas ja projektil on lootust leida kasutust nii autori enda kui ka meediafirmade poolt. Projekti on plaanis oluliselt edasi arendada, kaasates teisi arendajaid arenduse kiirendamiseks ja potentsiaalsete probleemide parimaks võimalikuks lahendamiseks ning samuti kasutajakogemuse parandamiseks.

# 8 Summary

The main objective of this bachelor´s thesis was to create a web platform, where users would be able to view movies and series together at the same time while being physically apart and share their thoughts and opinions about the given media.

At the moment of submission of this thesis, the five modules of the project have been completed enough, where they all meat the requirements set out for them. A functional version of the project can be deployed by following the steps laid out in the Git repository to setup the modules.

The biggest problem turned out to be the late architectural changes needed to be made to the system, although thanks to the modular approach decreasing the amount of code needed to be rewritten as well as a dedicated buffer time meant to handle unexpected problems.

The most inefficient and time-consuming usage of time in the opinion of the author occurred during the development of the front-end module, because the author's main experience has been in creating applications based on predefined designs and not in creating UI / UX designs.

In the opinion of the thesis´ author the project has thus far been successful and it has a potential to be used by the author themselves as well as larger media producing companies. There are plans to continue development of the project by involving other developers to accelerate the development, to best solve potential problems and to improve the user experience.

# 9 Kasutatud kirjandus


[1] MDN Web Docs, „The WebSocket API (WebSockets) - Web APIs | MDN,“ Mozilla, [Võrgumaterjal]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API. [Kasutatud 04 05 2021].

[2] Microsoft, „GitHub Student Developer Pack - GitHub Education,“ [Võrgumaterjal]. Available: https://education.github.com/pack. [Kasutatud 04 05 2021].

[3] P. Sroczkowski, „Cloud: IaaS Vs PaaS Vs SaaS Vs DaaS Vs FaaS Vs DBaaS | Brainhub,“ [Võrgumaterjal]. Available: https://brainhub.eu/blog/cloud-architecture-saas-faas-xaas/. [Kasutatud 04 05 2021].

[4] J. Manner, M. Endreß, T. Heckel ja G. Wirtz, „(PDF) Cold Start Influencing Factors in Function as a Service,“ Otto-Friedrich-Universität Bamberg, [Võrgumaterjal]. Available: https://www.researchgate.net/publication/328450988_Cold_Start_Influencing_Factors_in_Function_as_a_Service. [Kasutatud 04 05 2021].

[5] Oracle, „Oracle MySQL Licensing,“ [Võrgumaterjal]. Available: https://greymatter.com/corporate/licensing-draft/#:~:text=Licensing-,MySQL%20is%20an%20open%20source%20database%20governed%20by%20the%20terms,same%20open%20source%20GPL%20terms.. [Kasutatud 04 05 2021].

[6] M. Drake ja ostezer, „SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean,“ DigitalOcean, 19 03 2019. [Võrgumaterjal]. Available: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems. [Kasutatud 04 05 2021].

[7] B. Al-Saeedi, „Strengths and Weaknesses - Factors Influencing NoSQL Adoption,“ [Võrgumaterjal]. Available: http://alronz.github.io/Factors-Influencing-NoSQL-Adoption/site/MongoDB/Results/Strengths%20and%20Weaknesses/. [Kasutatud 04 05 2021].

[8] S. Daityari, „codeinwp.com,“ 15 03 2021. [Võrgumaterjal]. Available: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#:~:text=Vue%20provides%20higher%20customizability%20and,two%20is%20an%20easy%20option.. [Kasutatud 03 05 2021].

[9] D. Schiemann, „Deno: Secure V8 TypeScript Runtime from Original Node.js Creator,“ 26 12 2018. [Võrgumaterjal]. Available: https://www.infoq.com/news/2018/12/deno-v8-typescript/. [Kasutatud 05 05 2021].

[10] auth0, „JSON Web Token Introduction - jwt.io,“ auth0, [Võrgumaterjal]. Available: https://jwt.io/introduction. [Kasutatud 05 05 2021].

[11] FFmpeg, „FFmpeg,“ FFmpeg, [Võrgumaterjal]. Available: https://www.ffmpeg.org/about.html. [Kasutatud 04 05 2021].

[12] K. Kroening, „kkroening/ffmpeg-python: Python bindings for FFmpeg - with complex filtering support,“ [Võrgumaterjal]. Available: https://github.com/kkroening/ffmpeg-python. [Kasutatud 12 05 2021].

[13] U. Stanimirovic, „What Are the Best Video Codecs and How to Pick the Right One,“ 09 2020. [Võrgumaterjal]. Available: https://www.brid.tv/best-video-codecs-to-use-and-how-to-choose-the-right-one/#7-vp9. [Kasutatud 08 05 2020].

[14] P. LePage ja S. Richard, „What are Progressive Web Apps?,“ [Võrgumaterjal]. Available: https://web.dev/what-are-pwas/. [Kasutatud 12 05 2021].

# Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Oskar Voorel

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Meedia sünkroonse vaatamise platvormi arendamine“, mille juhendaja on Meelis Antoi
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

15.02.2021

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.