TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Simo Savila 185579IADB

Nutiteleri eesrakenduse loomine Jupiteri keskkonnale

Bakalaureusetöö

Juhendaja: Joel Kivi
Bakalaureusekraad

Tallinn 2022

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: Simo Savila

02.01.2022

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua rakendus erinevate operatsiooni süsteemidega nutiteleritele. Töö teoreetilises osas koosneb programmeerimiskeelte analüüsist, kasutatavate tööriistade kirjeldusest ja süsteemi nõuetest. Töö praktilises osas tehakse ülevaade loodud rakendusest.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 4 peatükki, 14 joonist.

Abstract

SmartTV application development for Jupiter platform

The goal of this Bachelor's Thesis is to develop SmartTV applications for most knows TV operating systems. The theoretical part of the work consist of analysis of programming languages, frameworks and tools that are used . The practical part gives overview how application is working.

The thesis is in Estonian and contains 25 pages of text, 4 chapters, 14 figures.

Lühendite ja mõistete sõnastik

API	Application Programming Interface – rakenduse programmerimise liides
DRM	Digital Rights Management – digiõiguste haldus
HTML	Hypertext Markup Language, hüperteksti märkimise keel
HTTP	HyperText Transfer Protocol – hüperteksti edastusprotokoll
IP	Internet Protocol, internetiprotokoll
JavaScript	Objektorienteeritud programmeerimiskeel
TCP	Transmission Control Protocol, edastusohje protokoll
Tizen	Samsungi nututelerites kasutatav operatsioonisüsteem
tvOS	Apple nutitelerites kasutatav operatsioonisüsteem
WebOS	LG nututelerites kasutatav operatsioonisüsteem

Sisukord

1 Sissejuhatus 9
1.1 Aktuaalsus 9
1.2 Metoodika 10
2 Analüüs 11
2.1 Rakenduse nõuded 11
2.2 Digitaalsete õiguste haldamine 11
2.3 Raamistike valik 14
2.3.1 Samsung ja LG 14
2.3.2 Android 16
2.3.3 AppleTV tvOS 17
2.4 Kasutatud tööriistad ja tehnoloogiad 18
2.4.1 Visual Studio Code 18
2.4.2 Tizen Studio 19
2.5 Kasutajaliidese disain 19
3 Valminud rakenduse kirjeldus 21
3.1 Arhitektuur 21
3.1.1 Eesrakenduse arhitektuur 21
3.2 Andmetüüpide struktuur 22
3.3 Rakendus vaated ja komponendid 23
3.3.1 Menüüriba 23
3.3.2 Pealeht 23
3.3.3 Sisu vaade 25
3.3.4 Otsevaatamine 27
3.4 Minu Jupiter vaade 28
3.5 Otsingu vaade 29
3.6 Videomängija 30
3.6.1 Adaptiivne bitikiirus ja HLS ning DASH 30
3.6.2 WebOS ja ShakaPlayer 30
3.6.3 Tizen 30

3.6.4 Flutter 31
3.6.5 Swift 31
4 Edasiarenduse võimalused 32
Kokkuvõte 33
Kasutatud kirjandus 34
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 37

Jooniste loetelu

Joonis 1. DRM tööpõhimõte 13
Joonis 2. DRM süsteemide ja operatsiooni süsteemide kooslus 14
Joonis 3. Java ja Kotlini süntaksi erinevus 17
Joonis 4. Jupiteri andmeobjektide tüüpide struktuur 22
Joonis 5. Jupiteri otse-eetri andmeobjekt 23
Joonis 6. Jupiteri menüü ja pealeht 24
Joonis 7. kategooria tüübid 24
Joonis 8. Jätka vaatamist ja lemmikud rubriik 25
Joonis 9. Sisu vaade 26
Joonis 10. Hooaja ja osa valimine 26
Joonis 11. Otsevaatamise võimalused 27
Joonis 12. DRM-ga kaitstud voog 28
Joonis 13. Minu lemmikud ja jätka vaatamist 29
Joonis 14. Otsingu vaade 29

1 Sissejuhatus

Alates aastast 2016 on hüppeliselt kasvanud nutitelerite arv inimeste kodudes, mis annavad võimaluse lisaks teleteenusepakkuja pakutavatelt kanalitelt tulevale sisule vaadata filme, seriaale, dokumentaale ja muud sisu erinevatelt voogedastusplatvormidelt.

Nutitelerite veebilehitsejad ei ole, aga disainitud esitama voog sisu, kuigi mõned brauserid seda suudavad, ei ole kasutajakogemus ja sisu esitamine sujuv ja mõistlik. Seetõttu on rakendused nagu Netflix, HBO, YouTube ja Disney loonud omale rakendused, mis on disainitud selleks, et kasutajal oleks võimalikult lihtne oma nutiteleri juhtimisseadet kasutades rakenduses ringi liikuda ja nautida platvormide sisu suurelt ekraanilt.

Seetõttu ongi ERR otsustanud luua oma nutiteleri rakenduse Jupiteri keskkonnale, et kõik Eesti kodanikud saaksid nautida parimat sisu oma kodus asuvast nutitelerist. Antud lõputöös kirjeldatakse kuidas loodi ERR poolt tellitud nutiteleri rakendused neljale enam levinud nutiteleri operatsiooni süsteemile, milleks on Andorid, tvOS, Tizen ja WebOS.

1.1 Aktuaalsus

Viimastel aastatel, kui kinod ja muud meelelahutus asutused on olnud suletud, soovivad inimesed siiski nautida uusi filme ja muud sisu oma kodus olevatest nutiteleritest. 2019. aastaks on kodudesse paigaldatud üle 1,26 miljardi nutiteleri üle maailma [15].

2020. aasta neljanda kvartali seisuga toodeti ja saadeti üle maailma laiali umbes 318 miljonit nutiteleri ja oletatav seadete arv kodudes kasvas peaaegu 140 miljoni võrra. Viies kodudesse paigaldatud nutitelerite koguarvu natuke alla 1,4 miljardi. [16]

Samas kui 2021. aasta esimeses kvartalis kasvas nutiseadmete globaalne saadetiste hulk 70,6 miljonini viies kodudesse paigaldatud nutitelerite koguarvu üle 1.4 miljardi, mis tähendab, et iga viies inimene maailmas omab nutitelerit. [17]

Jupiteri keskkond ongi loodud just selleks, et inimene saaks nautida kvaliteetset sisu, oma nutitelerist. Lisaks sellele ei maksa Jupiteri rakendus ega sisu kasutajale mitte midagi.

Samuti on väga palju Eesti kodanikke, kes soovivad vaadata kodumaised saateid või Eesti kanaleid mõnest välisriigist ning leidub ka inimesi, kes pole omale tellinud kanalite paketti kuid soovivad siiski vaadata erinevaid filme ja sarju. Loodav rakendus võimaldab vaadata ETV, ETV2 ja ETV+ otse kõikidest Euroopa riikidest.

1.2 Metoodika

Antud lõputöö analüütilises osas selgitatakse välja nõuded, mida rakendus peab suutma täita. Seejärel võrreldakse erinevaid raamistikke ja programmeerimiskeeli ja valitakse välja raamistikud ja keeled, millega hakatakse eesrakendust arendama. Põhjendatakse ka tööriistade ja arenduskeskkondade valikut.

Töö praktilises osas kirjeldatakse töö arhitektuuri. Kirjeldatakse andmetüüpe selgitatakse, kuidas liiguvad andmed ning miks ja kuidas antud lahendus välja töötati.

2 Analüüs

Kuigi tegemist ei ole arenduse esimese etapiga, soovitakse siiski välja selgitada, kas loodud ja arendatav lahendus on parim või tuleks kaaluda programmeerimiskeelte ja raamistike väljavahetamise peale.

2.1 Rakenduse nõuded

Antud alapeatükis selgitab autor süsteemi nõudeid, mis on loodud koostöös kliendiga. Selged nõuded aitavad kaasa tehnoloogiate valiku tegemisel ja hoiavad aega kokku arendusprotsessi käigus.

Süsteemi nõuded:

- Süsteem peab töötama nii Android, tvOS, Tizen, kui ka WebOS nutitelerite peal.
- Loodav süsteem pead olema suuteline edastama nii digitaalse autorikaitse litsentsiga (DRM) kaitstud voogu kui ka ilma selleta.
- Süsteem peab edastama adaptiivset voogu.
- Kliendil peab olema võimalik oma nutiteleri juhtimisseadmega rakenduses kiirelt ja mugavalt ringi liikuda.
- Rakendus peab nii välimuselt, kui ka kasutajakogemuse poolest olema sarnane ERR jupiter.err.ee portaalile.
- Loodavad rakendused peavad olema lihtsasti edasiarendatavad.

2.2 Digitaalsete õiguste haldamine

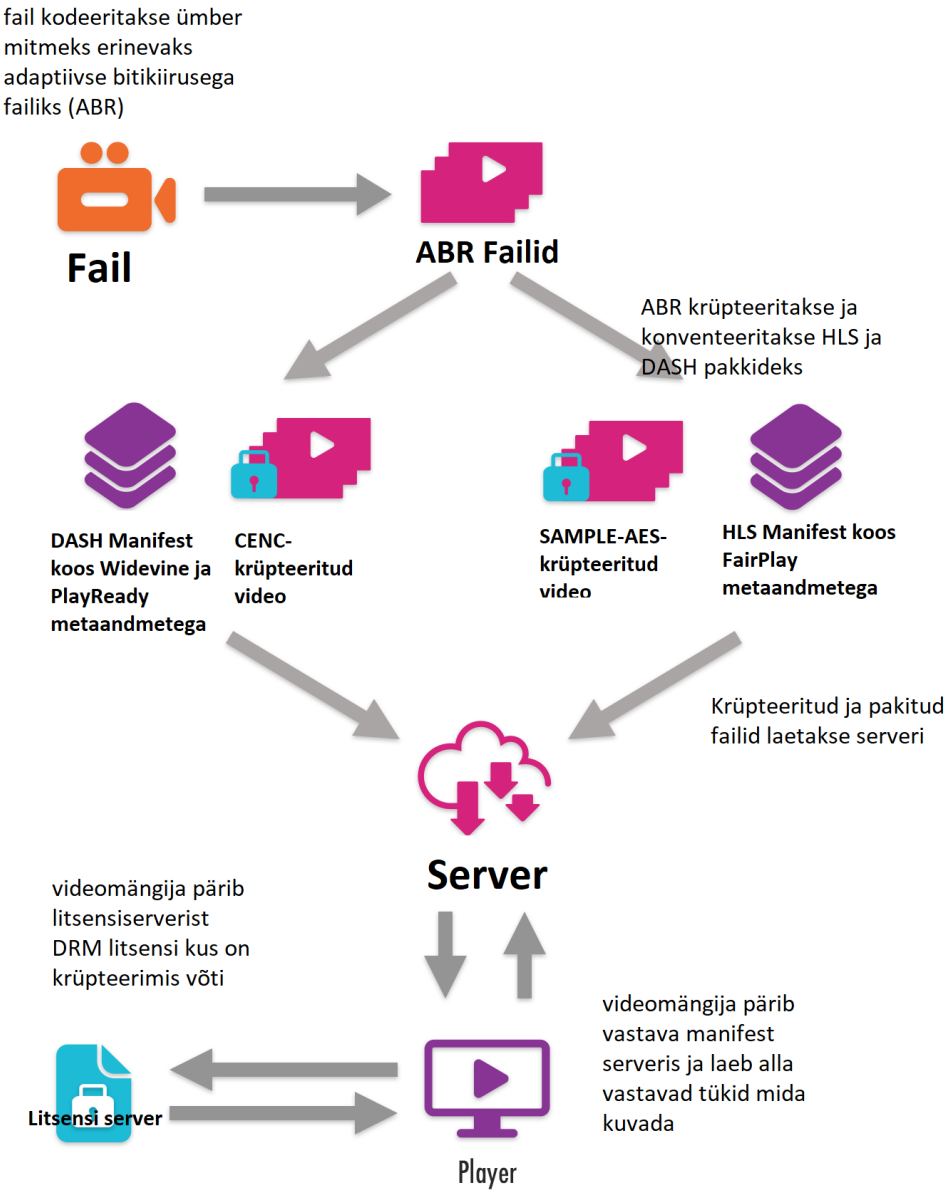
DRM (Digital rights management) ehk digitaalõiguste haldamine on viis kuidas kaitsta autoriõigustega kaitstud digitaalset meediat. DRM ei suuda tuvastada neid, kes tegelevad piraatluse ja illegaalsete failide jagamisega, vaid muudab sisu varastamise ja jagamise raskendatuks. [18]

DRM sisaldab koodi mis keelab kopeerimist või limiteerib aega ja seadmete arvu, kust antud teenust saab kätte. Autorid, ettevõtted ja teised sisuloojad kasutavad rakendust, mis krüpteerib sisu ning on kätte saadav ainult omades kindlat võtit. [18]

DRM võimaldab sisu omanikul kaitsta oma vara erinevatel viisidel. Järgnevalt on välja toodud erinevad DRM võimalused:

- Keelata kasutajal muuta või salvestada sisu.
- Keelata kasutajal jagada või edasi anda sisu.
- Keelata kasutajatel teha kuvatõmmiseid või jagada ekraani salvestusi.
- Anda sisu kättesaamise õigus ainult kindlatele IP aadressidele, seadmetele või riikidele. Näiteks kui sisu on kätte saadava ainult Eesti elanikele siis, ei ole antud sisu kättesaadav teistes riikides.
- Vesimärgiga kaitsta oma vara.

DRM litsentsiga kaitstud sisu mängimiseks on, aga nii süsteemile, kui ka sisule endale mõningad nõuded. Nimelt peab sisu, mida edastatakse olema ümberkodeeritud, krüpteeritud ja kokku pakitud formaati, mis sobib DRM süsteemiga. Järgneval joonisel ongi kuvatud DRM töö põhimõte (Joonis 1). [30]



Joonis 1. DRM tööpõhimõte [30]

Samuti peab videomängija, mida kasutatakse suutma teha päringuid, et saada võti litsentsi serverilt, et video krüpteerimisvõtmega esitatav oleks. Lisaks sellele võivad erinevad platvormid vajada erinevaid videomängijaid. [30]

Kolmandaks on vajalik litsentsi serveri olemasolu, kus hoitakse ja tagastatakse krüpteerimisvõtmed iga kord, kui sisu hakatakse kuvama. Litsentsi serveri tööks ongi autentida päringuid ning vastata päringutele. [30]

Kuigi erinevaid DRM süsteeme on palju saab suuremas osas hakkama kolme suurima DRM süsteemiga. Google loodud Widevine, Apple FairPlay ja Microsofti loodud PlayReady. Allpool näidatud joonisel on välja toodud milliseid DRM süsteeme toetavad antud lõputöös käsitletavad nutitelerite operatsiooni süsteemid. [30]

Joonis 2. DRM süsteemide ja operatsiooni süsteemide kooslus [30]

2.3 Raamistike valik

Kuna antud lõputöös käsitletakse ainult eesrakenduse arendust, siis pööratakse antud peatükis tähelepanu erinevate platvormide arendus võimalustele ning analüüsitakse ja tuuakse välja, mis on erinevate keelte ja raamistike tugevused ning miks just antud raamistikud ja keeled osutusid valituks.

2.3.1 Samsung ja LG

Samsungi ja LG nutiteleri tarkvara arenduskeskkonnad ja tööriistad suudavad JavaScriptil loodud rakenduse kokku pakkida endale vajalikuks rakenduseks. Kuna erinevaid eesrakenduse kirjutamiseks mõeldud raamistikke ja teeke on väga palju otsustas töö autor võrrelda ja analüüsid kolme kõige levinumat – ReactJS, Vue, Angular. Järgnevalt kirjeldataksegi antud raamistike eeliseid ja kasutusvõimalusi.

React on Facebooki poolt arendatud JavaScripti teek, mis avaldati 2013 aastal [6]. React on loodud, et luua interaktiivseid ning kiiresti laaditavaid kasutajaliideseid nii veebi- kui ka mobiilirakendustele. React on komponendi põhine teek, mis vastutab ainult rakenduse kuvamise eest. Üheks Reacti populaarsuse põhjusteks on suur jõudlus, mis tuleb tema Virtual DOM kasutamises. Virtual DOM võrdleb komponendi eelnevate seisu hetkese

seisuga ning uuendab ainult neid osasid ja andmeid, mis on muutunud, ilma kogu lehte uuendamata. [8]

Vue on nendest kolmest kõige uuem raamistik, esimene versioon tuli välja 2014. aastal [9]. Selle raamistiku suureks eeliseks peetakse kiiret õppimiskõverat. Kasutajal kel on eelnev kogemus ja arusaam HTML, CSS ja JavaScriptist on Vue'd väga lihtne õppid ja kasutada. Vue raamistiku looja Evan You ideeks oli antud raamistikku luues, kasutada Angulari ja Reacti parimaid omadusi ja teha neid omadusi kasutades uus raamistik, mida on lihtne kasutada. Vue on nendest kolmest raamistikus ainuke, mis kasutab ühe faililist komponenid süsteemi, mis tähendab et, CSS, HTML ja JavaScript kirjutatakse kõik ühte faili. [9]

Angular on arendatud Google poolt ning esimene versioon tuli välja 2010. aastal, ehk antud raamistik on eelpool mainitutest kõige vanem [6]. Viimane stabiilne versioon on 13 [7]. Angular on JavaScripti raamistik, mida kirjutatakse TypeScriptis. Üheks Angulari populaarsuse põhjuseks ongi TypeScripti kasutus. [6]

TypeScript on staatiliselt tüübitud keel, mis tähendab, et muutujad saavad olla alati kindlat tüüpi ning muutujate tüübid peavad olema teada juba kompileerimise ajal. 2017 aastal tehtud uuring näitas, et 15% JavaScripti programeerimis vigadest on võimalik ära hoida kasutades TypeScripti [10].

Angular nagu ka mõlemad eelpool mainitud raamistikud on komponendi põhine, mis tähendab, et sama koodi saab kasutada mitmes kohas, ilma ennast kordamata, mis omakorda vähendab koodikordust. Iga komponent koosneb tavaliselt kolmest osast HTML laiendiga failst, mis sisadab HTML elemente ning vastutab vaate kuvamise eest. CSS laiendiga fail, mis sisaldab kujundus element ja TypeScript fail laiendiga TS mis kontrollib komponendi käitumist. [11]

Kuigi kõikidel raamistikel on üpris sarnased omadus, ning kõigil on omad suuremad või väiksemad positiivsed küljed. Antud töös otsustati jätkata rakenduse kirjutamist Angulari kasutades järgnevatel põhjustel:

- Koodi puhtuse ja struktuuri poolest on Angular kõige parem valik sest, CSS, HTML ja JavaScript'i kood hoitakse eraldi failides.

- Angular nendest kõige vanem ning selle kohta on kõige lihtsam leida abimaterjale.

- Kolmandaks Tizen ja WebOS kasutamise kohta koos Angulariga on kordades rohkem materjale, kui teise kahe eelpool mainitu kohta.

2.3.2 Android

Androidi nutiteleritele rakenduse loomiseks on kolm peamist valikut Kotlin/Java, Flutter ja React Native. Järgneval tuuakse välja nende kolme tugevused, nõrkused ning võrreldakse neid omavahel.

Flutter on kasutaja liidese loomiseks mõeldud rakendus tööriist, mis on loodud Google poolt. See võimaldab luua rakendusi, mis töötavad nii IOS kui ka Android seadmetel [5]. Flutterit kirjutatakse Dart nimelises programmeerimis keeles, mis on samuti Google poolt arendatud. Flutteri rakendus ei kompileerita, mitte native komponentideks ja rakenduseks, vaid kuvatakse oma visualiseerimis mootoriga [5].

Flutteri populaarsuse ja kiire arendamise põhjuseks võib lugeda hot reload funktsionaalsust, mis võimaldab kiiresti katsetada rakenduse funktsionaalsust, kasutaja liidese disaini ja parandada programmeerimisvigu. Hot reload lisab uuendatud lähtekoodi failid hetkel jooksvasse Dart virtuaal masinasse ning seejärel laadib Flutteri raamistik kõik komponendid uuesti. Ilma koodi uuesti kompileerimata. [24]

React Native võimaldab samuti arendada rakendusi nii IOS kui ka Androidi seadmetele ning on loodud Facebooki poolt. [5]

Reacti Native ja Flutteri vahe seisnebki selles, et React teisendab Javascripti komponendid native komponentideks. See võib olla nii hea, kui ka halb, kui kasutaja soovib, et rakendus näeks koguaeg samasugune välja on Flutter hea valik. Kui kasutaja soovib, aga et Native komponendid näeksid välja samasugused nagu uuendustega kaasa tulevad, siis tuleks kasutatada React Native.[5] Nagu ka Flutter omab React Native hot reload sarnast funktsionaalsust, mis muudab arendamisprotsessi kiiremaks.

Kotlin on vabavaraline, avatud lähtekoodiga staatiliselt tüübitud programmeerimiskeel, mis on loodud Android rakenduste arendamiseks. Kotlin on objekt orienteeritud

programmeerimiskeel, mis on loodud 2010. aastal ettevõtte JetBrains poolt ning on avatud lähtekoodiga alates 2012. aastast. [22]

Kotlin on väga sarnane oma süntaksi poolest Javale nagu on näha ka jooniselt 3. Üheks peamiseks erinevuseks on see, et Kotlini puhul ei ole muutuja andme tüüpi vaja välja tuua, kui muutujale antakse algväärtus.[23]

```java
public class JavaCode {
  public String toJSON(Collection<Integer> collection) {
    StringBuilder sb = new StringBuilder();
    sb.append("[");
    Iterator<Integer> iterator = collection.iterator();
    while (iterator.hasNext()) {
      Integer element = iterator.next();
      sb.append(element);
      if (iterator.hasNext()) {
        sb.append(", ");
      }
    }
    sb.append("]");
    return sb.toString();
  }
}
```

```kotlin
class JavaCode {
  fun toJSON(collection:Collection<Int>):String {
    val sb = StringBuilder()
    sb.append("[")
    val iterator = collection.iterator()
    while (iterator.hasNext())
    {
      val element = iterator.next()
      sb.append(element)
      if (iterator.hasNext())
      {
        sb.append(", ")
      }
    }
    sb.append("]")
    return sb.toString()
  }
}
```

Joonis 3. Java ja Kotlini süntaksi erinevus [23]

Flutteri eeliseks Kotlini ees on kiirem arendus tänu eelpool mainitud hot reloadile, sest sellel on suurem hulk dokumentatsiooni ja näidisrakendusi ja kordades suurem kasutajate hulk [25].

Kuna loodavas rakenduses soovitakse, et rakendus näeks välja koguaeg stabiilne ja native komponentide kujundusest ei hoolita sobib Flutter antud eesrakenduse loomiseks. Lisaks eelpool mainitule on Flutter oma konkurentidest kõige kiirema arendusprotsessi, õppimis kõveraga ning on ka kõige suurema jõudlusega.

2.3.3 AppleTV tvOS

Kuigi Flutter ja React Native pakuvad võimalust arendada nii IOS kui ka Android rakendusi puudub neil tvOS arenduse tugi. Seega jäi ainsateks valikuteks Objective-C ja Swift.

Objective-C on 1984. aastal loodud objekt orienteeritud programmeerimiskeel C keele super komplekt ning on siiani üks põhilisi iOS rakenduste arenduskeeli. Üheks eeliseks peetaksegi selle pikaaegset kasutamis ajalugu ning stabiilsust. Objective-C süntaks on, aga üsna keeruline ja koodivigade leidmine ja probleemide lahendamine üsna keeruline.

Samas ei ole ka teada kaua Apple kavatseb jätkata erinevate teekide toetamist. Seetõttu soovitaksegi järjest vähem töötada Objective-C keelega, kuna selle tulevik on ebaselge.[27]

Swift 2014. aastal Apple poolt loodud programmeerimiskeel, mille eesmärkideks on, et loodud koodi oleks mugav ja lihtne hallata ja arendada. Swifti enda dokumentatsioonis on kirjas, et Swift on loodud asendamaks C keelele baseeruvaid keeli. Nagu C, C++ ja Objective-C. [26] Swifti üheks eeliseks on ka väiksem koodi hulk, mis on vajalik rakenduse loomiseks, mis omakorda kiirendab arendamise protsessi. Näiteks USA-s tegutsev sõidu tellimis teenust pakkuv ettevõte Lyft vähendas oma koodibaasi 75000 rea võrra kui nad kirjutasid oma rakendus ümber objective-c'lt Swifti. Muutes koodibaasi 2/3 võrra väiksemaks. [27]

Lisaks Swiftile kasutatakse veel ka raamistikku SwiftUI mis on samuti Apple poolt loodud. SwiftUI peamiseks eesmärgiks on luua kasutajaliideseid kõikidele Apple operatsiooni süsteemidele – iOS tvOS, macOS ja watchOS [32]. Lihtsalt öeldes SwiftUI ja Swifti vaheks on keel milles kirjutatakse ja SwiftUI on hulk tööriistu, mis aitavad luua ja kontrollida kasutajaliidest [33].

Nendel põhjustel otsustatigi kirjutada rakendus kasutades SwiftUI raamistikku.

2.4 Kasutatud tööriistad ja tehnoloogiad

Selles peatükis kirjeldatakse täpsemalt milliseid tööriistu ja tehnoloogiaid on kasutatud antud lõputöö praktilise osa teostamisel.

2.4.1 Visual Studio Code

Rakenduse arendusekeskkonnaks valis töö autor Microsoft Visual Studio Code, mis on kõige populaarsem arendus keskkond 2021. aasta uuringute kohaselt [1]. Lisaks sellele on antud arenduskeskkond tasuta kättesaadav ja sellel on ka palju erinevaid pistikprogramme ning suur valik laiendusi, mis aitavad jälgida stiilireegleid ja hoida koodi puhtana [31]. Samuti on töö autoril eelnev töötamise kogemus antud arenduskeskkonnaga.

2.4.2 Tizen Studio

Tizen Studio pakub täispakett lahendust erinevatest tööriistadest ja programmidest mis on vajalik Samsungi nutiteleritele nii veebi, kui native lahenduse arendamiseks. Tizen Studio arendusplatvorm on ehitatud Eclipse arenduskeskkonna peale. [2]

Tizen käsurea liides võimaldab kasutada kõike funktsioone ilma kasutamata Tizen stuudio arenduskeskkonda. See sisaldab kõiki arendus protsesse alates projekti loomisest kuni selle käivitamiseni.[4] See tähendab, et kogu arendusprotsess oleks võimalik läbi teha ilma Tizen Studio arenduskeskkonnata, siiski soovib töö autor kasutada nii Tizeni pakutavat arendus platvormi kui ka käsurea liidest, kuna Tizen studio pakub mõnevõrra paremat koodi silumise võimalust.

Tizen Studio sisaldab endas ka paketihaldurit, mis võimaldab alla laadida platvorme ja profiile millele on võimalik rakendusi luua, alustades nutikellast ja lõpetades nutiteleritega. [2]

Üheks tööriistaks, mida pakutakse on ka Tizen Studio Emulator, mis võimaldab jooksutada vastava seadme emulaatorit ning selle peal testida rakenduse funktsionaalsust. Kuigi emulaator ja päris seade ei anna kunagi täpselt sama tulemust, on disaini ja lihtsama funktsionaalsuse testimiseks antud tööriist siiski kasulik.

Veel üheks oluliseks tööriistaks on Tizen seadmehaldur, mis võimaldab laadida rakenduse päris seadmesse.

Viimaseks oluliseks komponendiks on Certificate Manager ehk sertifikaadi haldur. Enne rakenduse paigaldamist seadmesse või Tizen Store keskkonda laadimist tuleb rakendus allkirjastada. Allkiri kinnitab et rakendust ei ole vahepeal muudetud pahavaraliseks ning et on üles laetud sama tootja poolt. [3]

2.5 Kasutajaliidese disain

Eesrakendust arendama hakates ei olnud loodud kindlat disaini, ainukeseks kriteeriumiks oli, et nutiteleri rakendus peab olema võimalikult lihtne ja sarnane nii kasutajakogemuse kui ka disainis poolest Jupiter veebikeskkonnale. Kuigi Jupiteri portaali arendusmeeskonna käest oleks saanud küsida disaini failid, otsustati disain siiski luua töö

käigus kuid võimalikult sarnane värvivaliku ja kasutajakogemuse poolest. Arvestati sellega, et kasutajal oleks lõpptulemust võimalikult lihtne ja mugav kasutada.

3 Valminud rakenduse kirjeldus

Antud peatükis kirjeldatakse kuidas valminud rakendus töötab. Tutvustatakse rakenduse ülesehitust, valminud eessüsteemi ja andmete struktuuri ning erinevate lahenduste integreerimist.

Nagu eelmises peatüks mainitud on Angular, Flutter ja Swift komponendi põhised raamistikud. Mis võimaldab koodi taaskasutada ja tänu sellele vähendab koodi kordust ning hoiab projekti puhtana.

Seetõttu ongi rakendus jagatud mitmeks väikeseks komponendiks, mis teevad ära rakenduse põhilised toimingud ning näevad igal pool samasugused välja.

3.1 Arhitektuur

Kuna autor ei ole osalenud antud süsteemi taga rakenduse arendamisel ning on kasutanud ainult API lõpp-punkte ei kirjeldata antud peatükis taga rakenduse tööd ega andmebaasi suhtlust.

3.1.1 Eesrakenduse arhitektuur

Eesrakenduse loomiseks on kasutatud kolme erinevat raamistikku Flutter, Angular, SwiftUI.

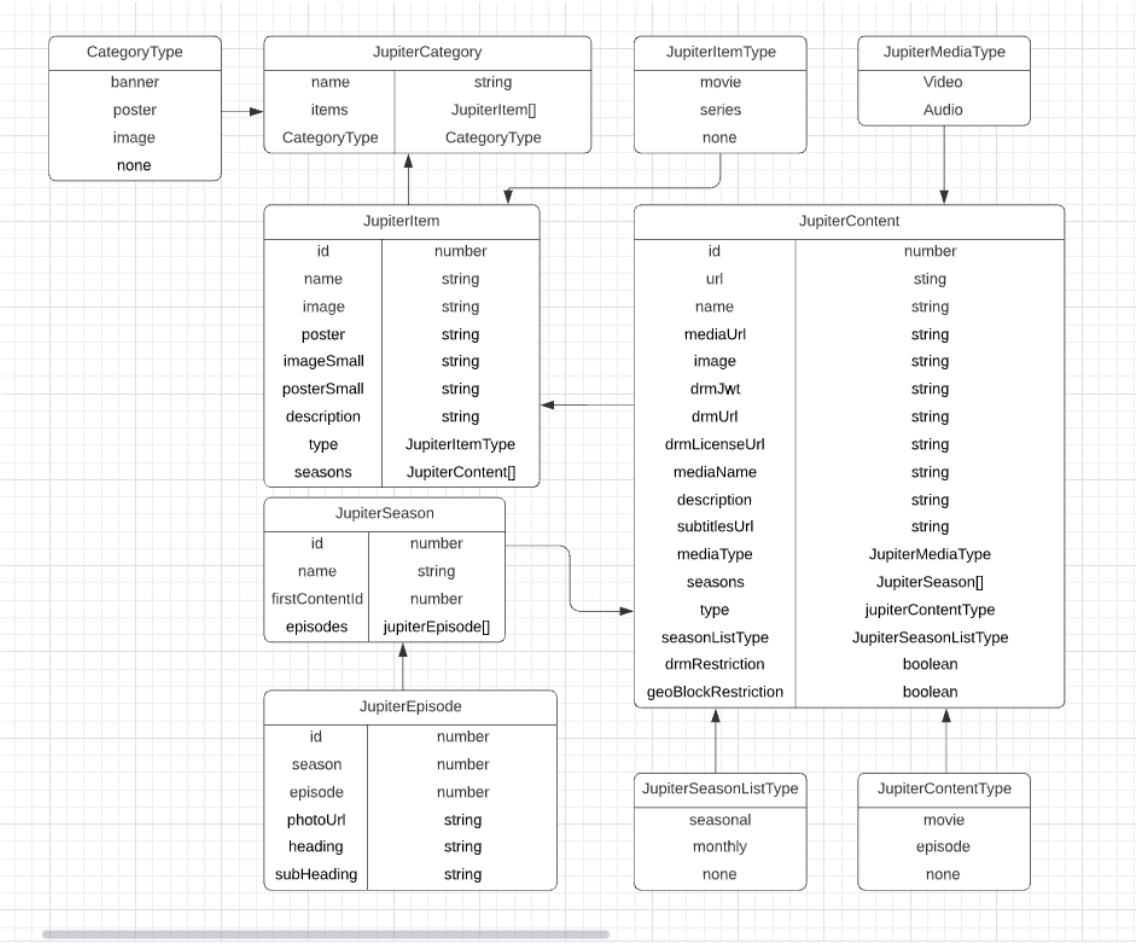
Kuna tegemist on kolme erineva keele ja raamastikuga on ka arhitektuur mõneti erinev, kuid siiski on suurem osa sama, järgnevas loetelus tuuakse välja rakenduse struktuuri ühised osad:

- Komponentide kaust - sisaldab kasutajaliidese komponente, mida on võimalik kasutada rakenduse erinevates osades.
- Teenuste kaust - sisaldab endas erinevaid teenuseid, mis suhtlevad API-ga ning omavad olulist rolli andmete saatmisel, tagastamisel ja vajadusel ka sobivaks andmetüübiks muutmisega.
- Mudelite kaust - sisaldab andmete mudeleid, mis tagastatakse taga rakendusest ning on vajalikud andmete liikumiseks.

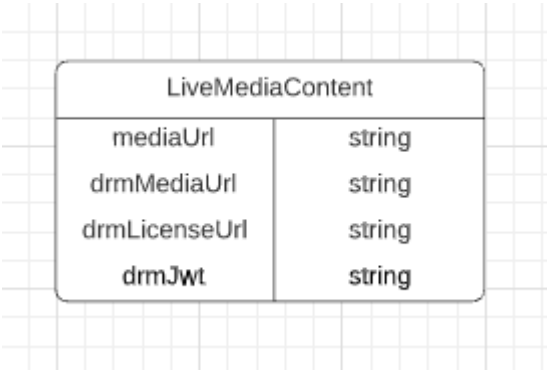
- Hoidla kaust - sisaldab erinevas formaadis pilte ja vektorgraafikat, mida kasutatakse rakenduses.

3.2 Andmetüüpide struktuur

Järgnevalt näitab autor andmetüüpide seoseid ja TypeScriptiga määratud staatilisi andmetüüpe (joonis 4). Teisel joonisel (joonis 5) on näha andmeobjekti mis on vajalik otse-eetris oleva sisu edastamiseks ning selle andmemudelit.



Joonis 4. Jupiteri andmeobjektide tüüpide struktuur



Joonis 5. Jupiteri otse-eetri andmeobjekt

3.3 Rakendus vaated ja komponendid

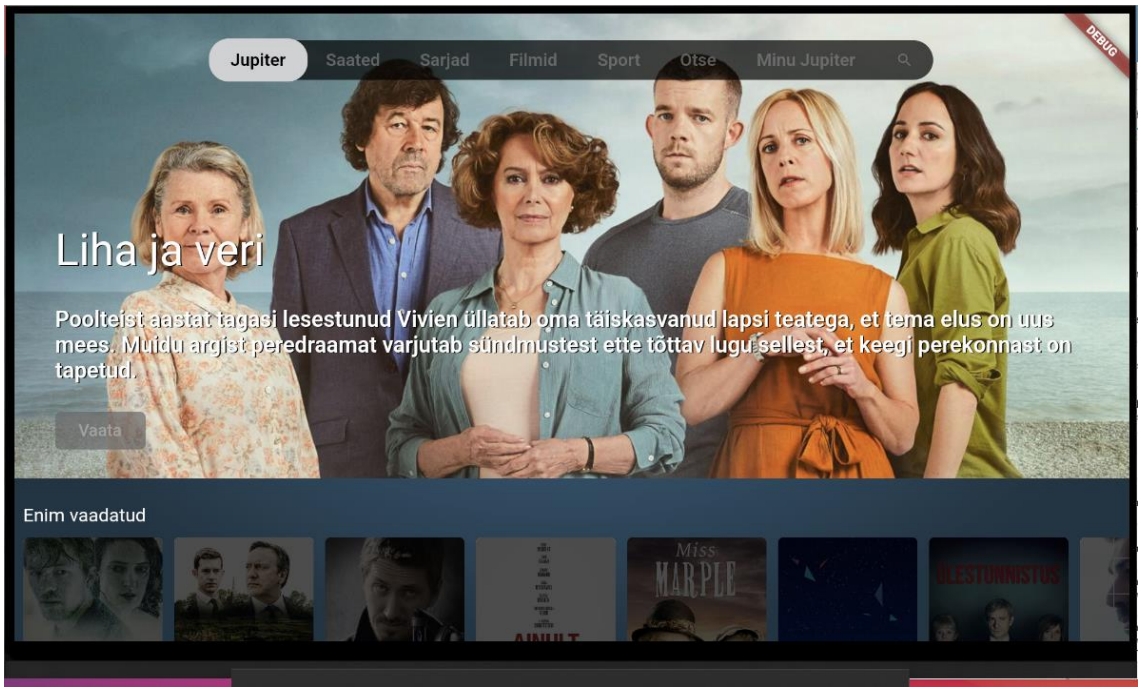
Antud alapeatükis keskendutakse natuke täpsemalt rakenduse vaadete tööpõhimõtetele.

3.3.1 Menüüriba

Kogu rakenduse tööd ja sisu, mida hetkel tuleb kuvada teeb ära menüüriba. Menüüriba asub lehe üleval servas peaaegu kõikide vaadete korral (joonis 6). Nimelt kui liikuda menüüs ühe vaate pealt teisel tehakse päring vastava taga rakenduse API lõpp-punkti vastu. Seejärel ehitatakse vaade üles kas pealehe komponendi või mõne teise komponendi poolt.

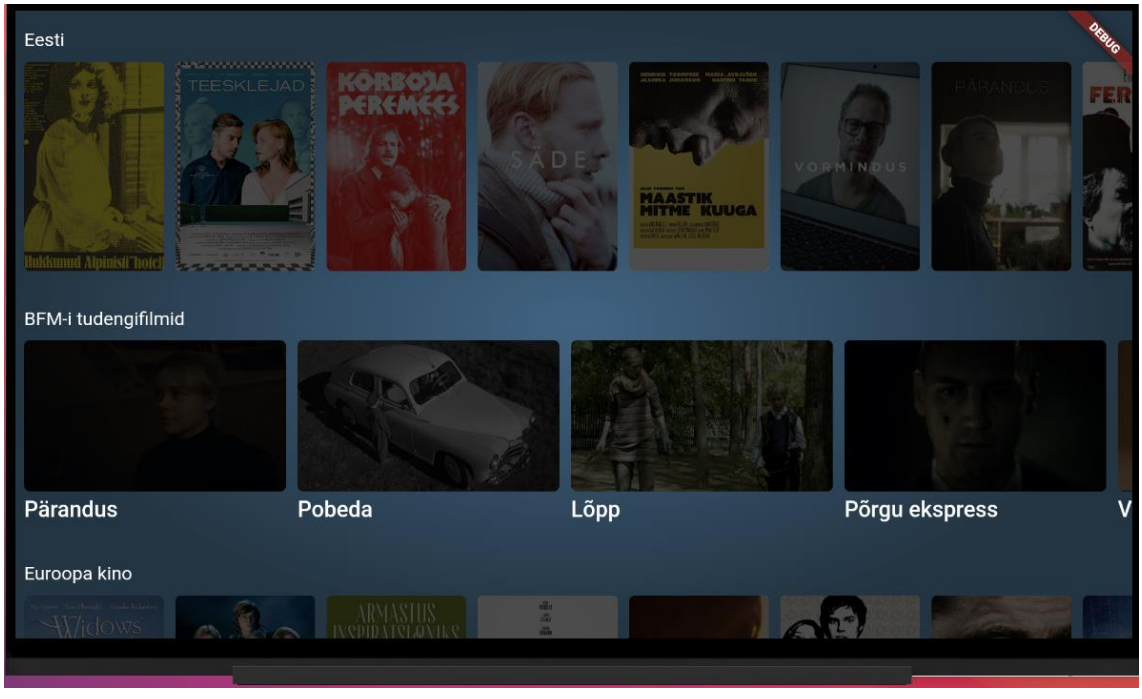
3.3.2 Pealeht

Pealehe komponent jälgib milline on hetkel aktiivne rubriik menüü ribal ning rubriigi muutudes laseb vastaval teenusel teha päringu, mis tagastab vastava rubriigi sisu. Pealehel kujutatakse asju vastavalt sellel, mis on konkreetse kategooria tüüp (joonis 4). Kui tegemist on *banner* ehk loosungi tüübiga, kuvatakse antud kategooria suurelt vaate ülemises osa, koos lühikirjeldusega (joonis 6).



Joonis 6. Jupiteri menüü ja pealeht

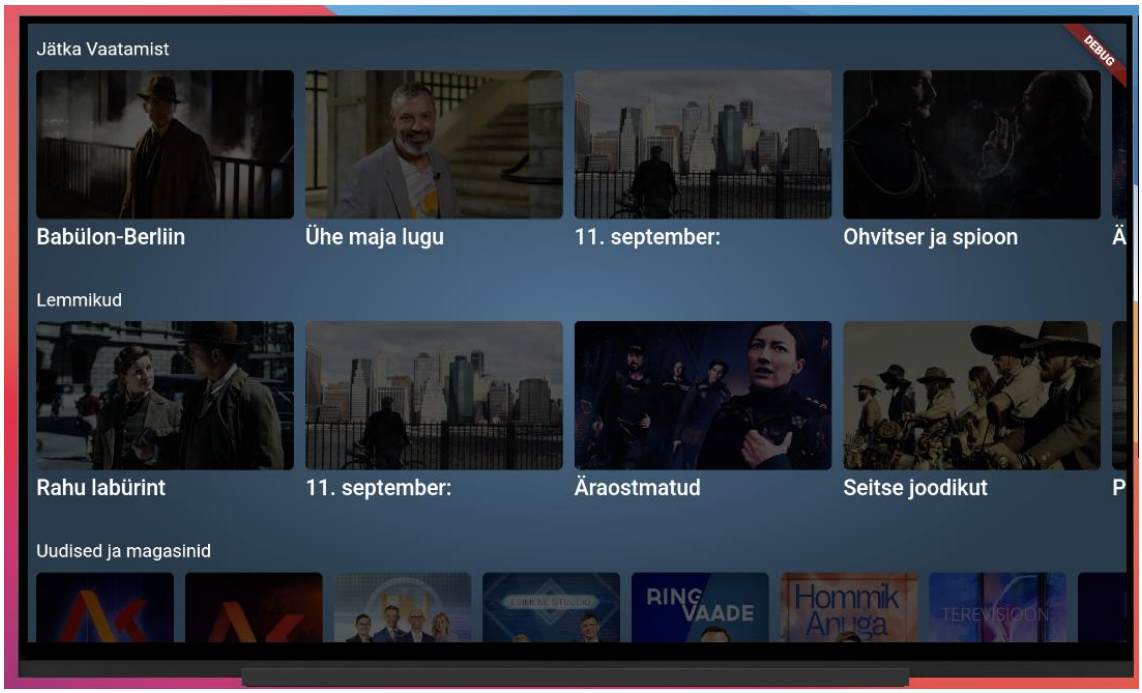
Kui kategooria on *poster* tüüpi kuvatakse kategooria elemente vertikaalses asendis ning ning kui tüübiks on *image*, siis kuvatakse horisontaalselt koos pealkirjaga (Joonis 7). Seda, mis tüüpi kategooriaga on tegu otsustab tagarakendus.



Joonis 7. kategooria tüübid

Autoriseeritud kasutaja puhul kuvatakse kasutajale pealehel ka tema viimati vaadatud või pooleli jäänud saated ning kasutaja poolt lemmikuks märgitud saated. Kui kasutaja

otsustab valida midagi, jätka vaatamist kategooriast, viiakse ta videomängijasse ning küsitakse, kas kasutaja soovib jätkata vaatamist kohast, kus ta pooleli jäi või alustada algusest (Joonis 8).

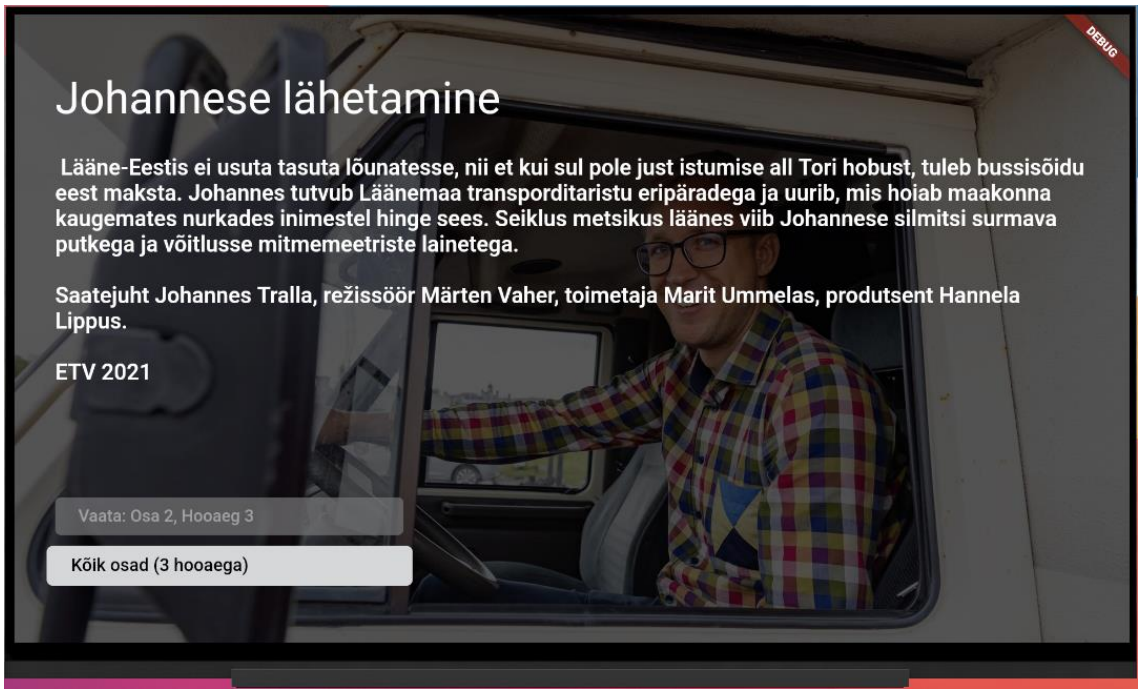


Joonis 8. Jätka vaatamist ja lemmikud rubriik

3.3.3 Sisu vaade

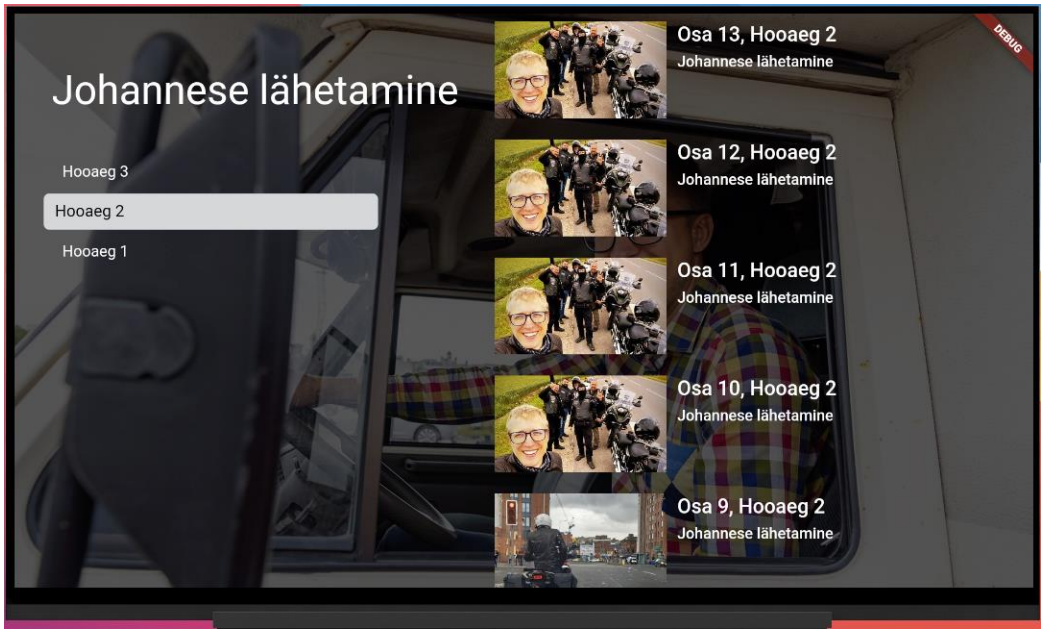
Menüü lehelt edasi liikudes suunatakse kasutaja edasi sisu vaatesse kus kuvatakse filmi, saate, seriaali või audio sisu kirjeldus. Komponendi sisendparameetriks on sisu ID mille alusel tehakse päring ning saadakse tagasi vajalik info, mida lehel kuvatakse – Pealkiri, sisu lühitutvustus posteri pilt. Seriaalide ja saadete puhul ka hooajad ja episoodid.

Lehe vasakul alumises servas asub nupp, millele vajutades hakatakse valitud sisu mängima (Joonis 9).



Joonis 9. Sisu vaade

Saadete ja seriaalide korral kuvatakse võimalus vaadata kõige uuemat episoodi või valida hooaega ja osa, mida soovitakse vaadata nagu on näidatud joonisel 10.



Joonis 10. Hooaja ja osa valimine

Enne sisu esitamise käivitamist kontrollitakse, kas sisu on kaitstud DRM'iga või mitte, kui antud sisu pole kasutajale kättesaadav kuvatakse vastav teade. Seda kas kasutaja saab antud sisu vaadata sõltub kolmest asjast:

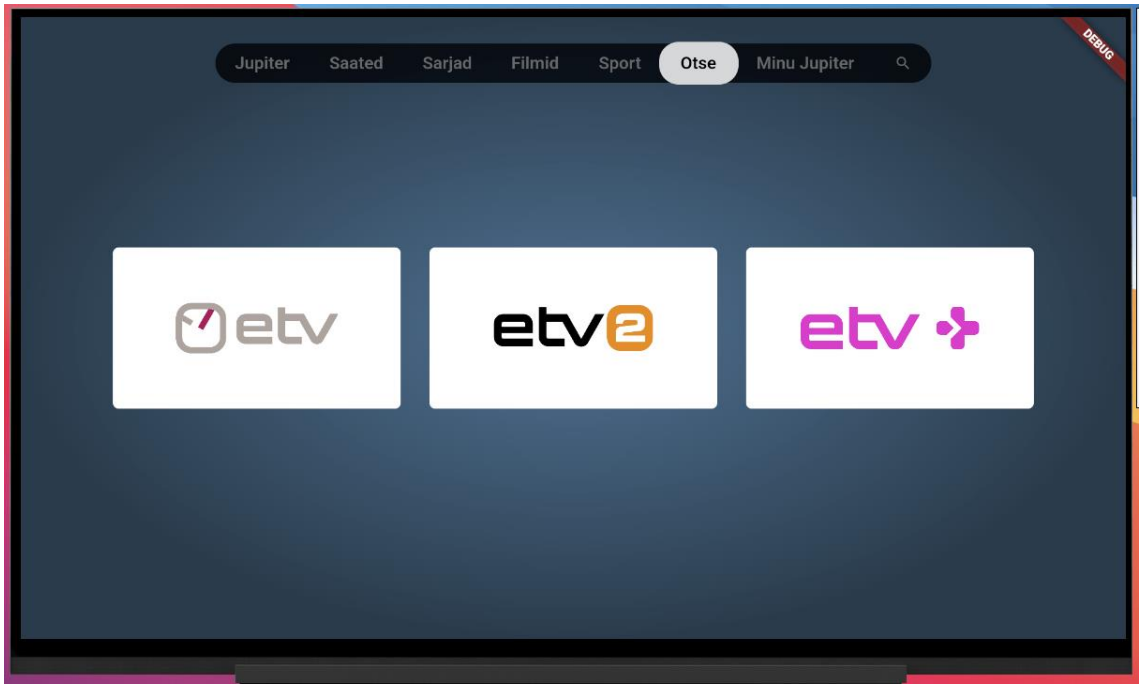
- Kas kasutaja on sisse loginud oma kasutajaga.
- Asukohast
- kas kasutaja on Eesti kodanik.

Kui kasutaja ei asu Eesti piirides kuvatakse talle ilma DRM nõuetega kaitstud voogu. Kui kasutaja asub, aga Eestis, siis näidatakse talle DRM litsentsiga kaitstud voogu. Seda, kus riigis kasutaja asub kontrollitakse API päringuga seadme IP alusel.

Audio sisu esitamise puhul näidatakse kasutajale sama taustapilti, mida ta nägi ka eelnevalt.

3.3.4 Otsevaatamine

Otsevaatamise lehele kuvatakse kolm valikut ETV, ETV2 ja ETV+ (joonis 11).



Joonis 11. Otsevaatamise võimalused

Otse eetri puhul nagu ka muu sisu puhul on kaks valikut, kas näidatakse DRM-ga kaitstud voogu või ilma. Erinevuseks on aga see, et kui eetris on saade, mida ei saa kasutajale näidata kuvatakse antud teade (joonis 12).

Joonis 12. DRM-ga kaitstud voog

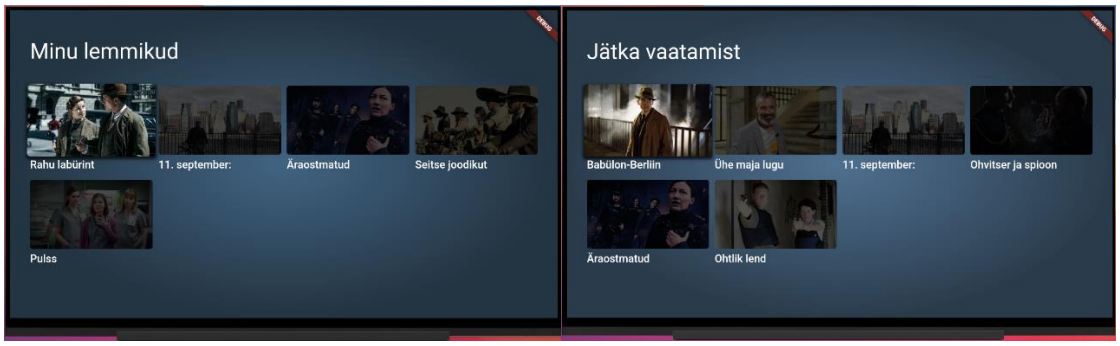
Kui DRM-ga kaitstud saade lõpeb läheb voog automaatselt edasi ilma, et kasutaja peaks midagi tegema.

3.4 Minu Jupiter vaade

Minu jupiteri vaates kuvatakse kasutajale ühte kahest valikust:

1. Kui kasutaja pole sisse loginud näidatakse talle sisselogimis vormi ja juhiseid kuidas registreerida uut kontot.
2. Kasutajal kelle on juba konto ning on sellega rakendusse sisse logitud, kuvatakse kolme nuppu, mis võimaldavad tal teha erinevaid valikuid:
    2.1. Kas ta soovib näha loetelu sisust, mida ta on eelnevalt vaadanud kuid pooleli jätnud
    2.2. Kas ta tahab vaadata hoopis enda poolt lemmikuks määratud sisu.
    2.3. Kas ta soovib välja logida.

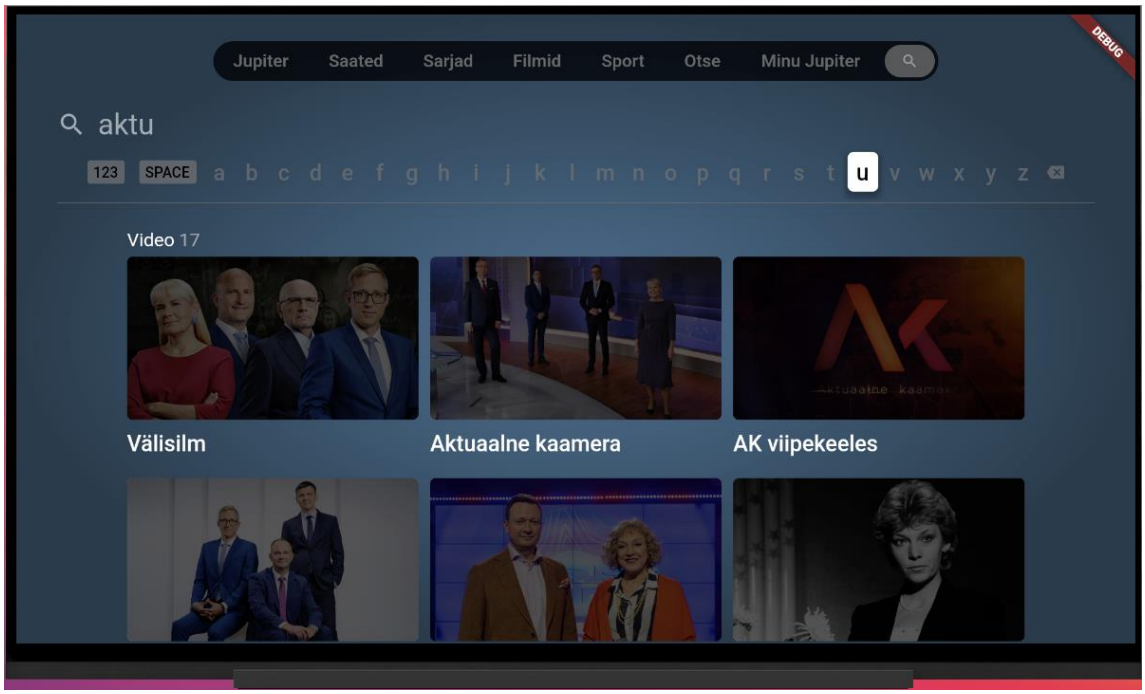
Kui kasutaja valib, aga ühe kahest esimesest valikust (2.1 või 2.2) viiakse ta vaatesse, mis kuvab vastavalt valikule, kas kasutaja lemmikuid või siis pooleli jäänud sisu (joonis 13).



Joonis 13. Minu lemmikud ja jätka vaatamist

3.5 Otsingu vaade

Kasutajal on võimalus ka otsida sisu mida ta soovib vaadata. Kuna nutiteleritega kirjutamine on üpriski aeglane tegevus, ei saadeta päringuid iga klahvivajutuse peale, vaid ainult siis, kui kasutaja on sisestanud uue tähemärgi ja sellest on möödunud 1 sekund. Otsing tagastab nii video, kui audio vasted, ning arvu kui palju tulemusi saadi.



Joonis 14. Otsingu vaade

3.6 Videomängija

Kõikidel videomängijatele on samad nõudmised – nad peavad suutma mängida kas HLS või DASH formaadis adaptiivse bitikiirusega voogu ning toetama nii DRM litsentsiga kaitstud voogu kui ka ilma selleta.

3.6.1 Adaptiivne bitikiirus ja HLS ning DASH

Adaptiivne bitikiirusega voo edastamine on tehnika mida kasutatakse et dünaamiliselt muuta esitatava voo video kvaliteeti vastavalt vaataja interneti ühenduse kiirusele [28].

Voogedastus on andmete edastamise viis üle interneti ja see võimaldab alustada sisu kuvamist enne kui kogu fail on alla laetud. See tähendab, et sisu edastatakse kliendi seadmesse ilma, et kasutaja peaks faili tervenisti alla laadima. [29]

DASH on üks kahest peamisest voogedastus protokollist. DASH on lühend eesti keelde tõlgitult tähendab Dünaamiline adaptiivne voogedastus üle HTTP. Teine peamine voogedastusprotokoll on HLS ehk eesti keelde tõlgitult HTTP voogedastus. Neil kahel protokollil on palju ühist nagu nimigi ütleb kasutavad mõlemad protokollid HTTP-d. Lisaks sellele kasutavad nad TCP andmeedastus protokolli ning jagavad video väikesteks tükkideks mis edastatakse koos indeksifailga. [29]

Nende suurimaks erinevuseks on see et HLS on ainuke formaat, mida Apple toodetud seadmed suudavad esitada. Teiseks mõneti väiksemaks erinevuseks on video tükkide suurus, HLS puhul on standard 6 sekundit, DASH puhul, aga 2-4 sekundit. [29]

3.6.2 WebOS ja ShakaPlayer

LG WebOS telekatele osutus sobivaks videomängijaks Shaka Player. Shaka Player on vabavaraline JavaScripti teek, mis on loodud adaptiivse meedia esitamiseks. Antud videomängija suudab esitada voogu nii HLS kui ka DASH formaadis nii brauseris kui ka mõne teleka mudeli peal. [12]

3.6.3 Tizen

Kuigi ShakaPlayer sobiks ka väidetavalt Tizeni peal video ja audio edastamiseks [12], kasutatakse Samsungi nutitelerite peal Tizeni enda native videomängijat. Põhjuseks miks otsustati Tizen native playeri kasuks oli märgatavalt kiirem reageerimis aeg juhtkäskudele. Tizeni native videomängijaks on AVPlay. AVPlay toetab DASH ja HLS

adaptiivse voo esitamist ning võimaldab lihtsalt kasutada funktsioone nagu kerimine, paus, jätka ning esitamis kiirus valikut. Samuti toetab AVPlay subtiitrite mängimist erinevates formaatides [13]. Sisu kaitsmiseks DRM-ga kasutatakse DASH voogu ja PlayReady DRM süsteemi.

3.6.4 Flutter

Android rakenduses kasutatakse Google poolt loodud, ExoPlayer nimelist video ja audio mängijat. ExoPlayer'i suurimateks eelisteks Androidi enda multimeediamängija ees on selle modifitseerimis võimalus ning DASH voo esitamise tugi [19]. ExoPlayer toetab ka WideVine DRM krüpteeringut ja esineb vähem seadme spetsiifilisi vigasid ja käitumis variatsioone [20]. DRM sisu näitamiseks kasutatakse DASH voogu ja WideVine DRM süsteemi.

ExoPlayerit kasutatakse ka YouTube ja Netflixi androidi rakendustes. [21]

3.6.5 Swift

AVPlayer on Apple arendus tiimi poolt loodud kontroller objekt mida kasutatakse erinevate failide edastamiseks. AVPlayer suudab edastada nii faili põhiseid video ja heilfaile kui ka veebipõhist HTTP Live Steam (HLS) voogu. AV player suudab esitada ühte meedia objekti korraga. AVPlayer ise on dünaamiline objekt mille seis muutub järjepidevalt. AVPlayer ise on aga mittevisuaalne objekt, mis tähendab seda et ise nad ei suuda videot kuvada. Selleks et kuvada ekraanil videot kasutatakse AVKit raamistiku AVPlayerViewController klassi. Lisaks videole annab see klass ka täis lahenduse erinevatest meedia kontrolleritest nagu paus, jätka, kerimine ja nii edasi. [14]

4 Edasiarenduse võimalused

Edasiarenduse võimaluseks on lisada eesrakendusele tõlge ning valida ka subtiitreid erinevates keeltes. Lisaks võiks eesrakendusel olla kogumahus sama funktsionaalsus mis Jupiter veebiportaalil, ehk raadio kuulamise võimalus ning saatekava kuvamine.

Kuigi antud lõputöös käsitletud arendus hõlmas rakenduse kirjutamist ainult vastava operatsioonisüsteemiga nutitelereitele, tuleks kaaluda ka sarnase rakenduse arendamist erinevate teleteenuse pakkuja digiboksidele. Mis võimaldaks Jupiteri vaadata ka vanematest teleritest läbi teenusepakkuja vaheseadme.

Kui kasutaja vaatab seriaali või mitme episoodilist sisu võiks ühe osa lõppedes hakata automaatselt peale järgmine osa.

Lisaks sellele, tuleks kaaluda ka Samsungi Tizen rakenduse videomängija väljavahetamist Shaka Playeri vastu, mis omakorda vähendaks veelgi koodi hulka ja muudaks koodi lihtsamaks, kuid nagu ka eelnevalt mainitud tuleks uurida kas Shaka Playerit on võimalik muuta kiiremaks juht käsklustele.

Kokkuvõte

Bakalaureuse töö eesmärgiks oli luua eesrakendus, Samsungi, LG, AppleTV ja Sony telekatele mille operatsioonisüsteemiks on Tizen, WebOS, tvOS või Android. Töö käigus valminud lahendus võimaldab kasutajal vaadata või kuulata oma nutitelerist erinevaid saateid, sarju, filme ja palju muud sisu, mis on saadaval Jupiteri portaalis. Lisaks sellele on rakenduses võimalik vaadata reaalajas kas ERR kanaleid. Ainukeseks erinevuseks on aga see, et kui saatel, mis hetkel eetris on puudub internetis näitamise õigus kuvatakse kasutajale ka selle kohane teade. Rakendus on kättesaadava ja kasutatav kõikides Euroopa riikides.

Lisaks sellele on võimalus rakendusse siseneda ka oma Jupiteri kontoga, mis võimaldab kasutajal jätkata sisu vaatamist sealt kust eelmine kord pooleli jäädi. Lisaks on võimalik ka valitud sisu märkida lemmikuks et seda järgmisel korral kiiresti üles leida.

Kasutatud kirjandus

[1] "Stack Overflow Developer Survey 2021", Stack Overflow, 2021. [Online]. Available: https://insights.stackoverflow.com/survey/2021#most-loved-dreaded-and-wanted-language-love-dread. [Accessed: 06- Nov- 2021].

[2] "Overview | Tizen Developers", Developer.tizen.org, 2021. [Online]. Available: https://developer.tizen.org/development/tizen-studio. [Accessed: 07- Nov 2021].

[3] "Certificate Manager | Tizen Docs", Docs.tizen.org, 2021. [Online]. Available: https://docs.tizen.org/application/vstools/tools/certificate-manager/. [Accessed: 07- Nov- 2021].

[4] "Command Line Interface Commands | Tizen Developers", Developer.tizen.org, 2021. [Online]. Available: https://developer.tizen.org/development/tizen-studio/web-tools/cli. [Accessed: 07- Nov- 2021].

[5] M. Zaleski, "Flutter vs. React Native in 2021", Nomtek, 2021. [Online]. Available: https://www.nomtek.com/blog/flutter-vs-react-native. [Accessed: 12- Nov- 2021].

[6] E. Elliott, "Top JavaScript Frameworks and Tech Trends for 2021", Medium, 2021. [Online]. Available: https://medium.com/javascript-scene/top-javascript-frameworks-and-tech-trends-for-2021-d8cb0f7bda69. [Accessed: 12- Nov- 2021].

[7] "Angular", Angular.io, 2021. [Online]. Available: https://angular.io/guide/update-to-latest-version. [Accessed: 12- Nov- 2021].

[8] T. Sufiyan, “What is React: Definition, Why ReactJS, its Features and Installation,” Simplilearn.com, 04-Jun-2020. [Online]. Available: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs. [Accessed: 15-Nov-2021].

[9] “What is Vue.js, and Why is it Cool?,” Linuxhint.com. [Online]. Available: https://linuxhint.com/about_vue_js. [Accessed: 15-Nov-2021].

[10] Z. Gao, C. Bird, and E. T. Barr, “To type or not to type: Quantifying detectable bugs in JavaScript,” in 2017 [Online]. Available: https://earlbarr.com/publications/typestudy.pdf. [Accessed: 15-Nov-2021].

[11] F. Jerga, “Why should one use Angular? - Eincode - Medium,” Eincode, 25-Mar-2021. [Online]. Available: https://medium.com/eincode/why-should-one-use-angular-490e00cb64c5. [Accessed: 15-Nov-2021].

[12] shaka-player: JavaScript player library Github.com [Online]. Available: https://github.com/google/shaka-player. [Accessed: 20- Nov- 2021]

[13] P.-J. Speelmans, “Going big screen: How to use Samsung tizen’s AVPlay?,” Theoplayer.com. [Online]. Available: https://www.theoplayer.com/blog/how-to-use-samsung-tizens-avplay. [Accessed: 20-Nov-2021].

[14] “Apple Developer Documentation,” Apple.com. [Online]. Available: https://developer.apple.com/documentation/avfoundation/avplayer. [Accessed: 25-Nov-2021].

[15] R. M. Player, “Radiant Media Player,” Radiantmediaplayer.com. [Online]. Available: https://www.radiantmediaplayer.com/docs/latest/dash-drm-documentation.html. [Accessed: 25-Nov -2021].

[16] D. Watkins, “Consumer Electronics Reports,” Strategyanalytics.com. [Online]. Available: https://www.strategyanalytics.com/access-services/devices/connected-home/consumer-electronics/reports/report-detail/global-connected-tv-device-vendor-market-share-q4-2019. [Accessed: 25-Nov-2021].

[17] D. Watkins, “Consumer Electronics Market Data,” Strategyanalytics.com. [Online]. Available: https://www.strategyanalytics.com/access-services/devices/connected-home/consumer-electronics/market-data/report-detail/global-connected-tv-device-vendor-market-share-q4-2020. [Accessed: 25-Nov-2021].

[18] F. Dingledy and A. B. Matamoros, “What is Digital Rights Management?,” Digitalguardian.com. [Online]. Available: https://digitalguardian.com/blog/what-digital-rights-management. [Accessed: 25-Nov-2021]

[19] ExoPlayer: An extensible media player for Android. Github.com. [Online]. Available: https://github.com/google/ExoPlayer [Accessed: 25-Nov-2021]

[20] ExoPlayer, “Pros and cons,” Exoplayer.dev. [Online]. Available: https://exoplayer.dev/pros-and-cons.html. [Accessed: 25-Nov-2021].

[21] “ExoPlayer in Android with Example,” Geeksforgeeks.org, 23-Dec-2020. [Online]. Available: https://www.geeksforgeeks.org/exoplayer-in-android-with-example/. [Accessed: 25-Dec-2021].

[22] “Kotlin vs Flutter - comparison of popularity, performance in 2021,” Aglowiditsolutions.com, 09-Apr-2021. [Online]. Available: https://aglowiditsolutions.com/blog/kotlin-vs-flutter/. [Accessed: 26-Nov-2021].

[23] J. Hartman, “Kotlin vs java: What’s the difference?,” Guru99.com, 22-Feb-2020. [Online]. Available: https://www.guru99.com/kotlin-vs-java-difference.html. [Accessed: 26-Nov-2021].

[24] “Hot reload,” Flutter.dev. [Online]. Available: https://docs.flutter.dev/development/tools/hot-reload. [Accessed: 26-Nov-2021].

[25] S. Osadchuk, “Flutter vs. Kotlin: Which one to choose in 2021,” Doit.software, 02-Apr-2020. [Online]. Available: https://doit.software/blog/flutter-vs-kotlin-which-is-best-for-cross-platform-app-development. [Accessed: 27-Nov-2021].

[26] Swift.org. [Online]. Available: https://www.swift.org/about/. [Accessed: 27-Nov-2021].

[27] J. Szczęsny, “Swift vs Objective-C: Which language should you choose for your project?,” Concisesoftware.com, 22-Nov-2019. [Online]. Available: https://concisesoftware.com/swift-vs-objective-c/. [Accessed: 27-Nov-2021].

[28] W. Jones, “Adaptive bitrate streaming (ABR),” Haivision.com, 17-Aug-2018. [Online]. Available: https://www.haivision.com/resources/streaming-video-definitions/adaptive-bitrate-abr-streaming/. [Accessed: 27-Nov-2021].

[29] Cloudflare.com. [Online]. Available: https://www.cloudflare.com/en-gb/learning/video/what-is-mpeg-dash/. [Accessed: 27-Nov-2021].

[30] J.D. Russell “How to produce protected content: Understanding digital rights management,” Brightcove.com. [Online]. Available: https://www.brightcove.com/en/resources/blog/dealing-drm-understanding-drm-and-how-produce-protected-content/. [Accessed: 27-Nov-2021].

[31] "Why Visual Studio Code?," Visualstudio.com. [Online]. Available: https://code.visualstudio.com/docs/editor/whyvscode. [Accessed: 27-Nov-2021].

[32] "What is swiftui," *Cocoacasts*. [Online]. Available: https://cocoacasts.com/swiftui-fundamentals-what-is-swiftui. [Accessed: 15-Dec-2021].

[33] P. Hudson, "What's the difference between Swift and Swiftui?," *Hacking with Swift*, 28-May-2020. [Online]. Available: https://www.hackingwithswift.com/quick-start/understanding-swift/whats-the-difference-between-swift-and-swiftui. [Accessed: 15-Dec-2021].

[34]

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

Mina, Simo Savila

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Nutiteleri eesrakenduse loomine Jupiteri keskkonnale, mille juhendaja on Joel Kivi
   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.

02.01.2022

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.