TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Robert Barisovets 179256IADB

TALTECH SALVESTUSTARKVARA ARENDUS

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2020

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: Robert Barisovets

09.02.2020

Annotatsioon

Antud bakalaureusetöö eesmärgiks on pakkuda alternatiivne lahendus Tallinna Tehnikaülikoolis kasutuses oleva salvestustarkvarale. Tarkvara analüüsi käigus toodi välja eksisteeriva lahenduse tugevused ja nõrkused. Toetudes saadud andmetele koostati plaan optimaalse lahenduse loomiseks.

Eksisteeriva salvestustarkvara analüüsimise käigus järeldati, et kasutuses olev lahendus põhineb vananenud tehnoloogial ning vajab asendust. Bakalaureusetöö tulemusena loodi rakenduse prototüüp, mis vastab salvestustarkvara baas funktsionaalsuse nõuetele ning kasutab modernset ja levinud tehnoloogiat, mis soodustab projekti edasiarendust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 4 peatükki, 14 joonist.

Abstract

TalTech Recording Software Development

The aim of this bachelor’s thesis is to offer an alternative solution to the recording management software used at Tallinn University of Technology. During the software analysis, the strengths and weaknesses of the existing solution were highlighted. Based on the obtained data, a plan to achieve an optimal solution was created.

The analysis of the existing software concluded that the solution in use is based on outdated technology and needs to be replaced. As a result of the bachelor's thesis, a prototype of the application was created, which meets the requirements of the basic recording software functionality. Provided solution uses modern and widespread technology, which facilitates the further development of the project.

The thesis is in Estonian and contains 27 pages of text, 4 chapters, 14 figures.

Lühendite ja mõistete sõnastik

AIR	Adobe Integration Runtime
API	Application programming interface
BLL	Business Logic Layer
CSS	Cascading Style Sheet
DAL	Data Access Layer
DPI	Dots per inch
DTO	Data Transfer Object
FXML	User interface markup language Extended from XML
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
ID	Identification
LTS	Long-Term-Support
MVC	Model View Controller
MXML	Macromedia Extensible Markup Language
OS	Operating system
PoE	Power over Ethernet
RIA	Rich internet Application
UI	User interface
WPF	Windows Presentation Foundation
XML	Extensible Markup Language

Sisukord

1 Sissejuhatus 9
2 Analüüs 10
2.1 Funktsionaalsuse määramine 10
2.1.1 Esmajärguline funktsionaalsus 10
2.1.2 Salvestusprotsessi toetav funktsionaalsus 11
2.1.3 Väheoluline funktsionaalsus 11
2.2 Riistvaralised lahendused 12
2.3 Tarkvaralised lahendused 14
2.3.1 Olemasoleva EchoMonitori platvorm 14
2.3.2 Olemasoleva EchoMonitor projekti struktuur 15
2.3.3 Alternatiivsed platvormid ja lahendused 17
2.4 Valitud lahendus 19
3 Arenduskäik 20
3.1 Tööriistad 20
3.2 Salvestusriistvara sõnumite kaardistamine 21
3.3 Tehnoloogia 23
3.3.1 Valitud platvormi komponentide versioonid 23
3.3.2 Muu tehnoloogia 23
3.4 Rakenduse struktuur 24
3.5 SpringBoot tagarakendus 25
3.5.1 Business Logic Layer 25
3.5.2 Data Access Layer 26
3.5.3 Data Transfer Object 26
3.5.4 Domain 26
3.6 JavaFX esirakendus 27
3.6.1 FXML 27
3.6.2 Controller 28
3.6.3 Spring 30
3.6.4 Kasutajaliidese tegevuste skeem 30

3.7 Kasutajaliidese välimus 33
3.8 Lahenduse testimine 34
4 Kokkuvõte 36
Kasutatud kirjandus 37
Lisa 1 – Rakenduse lähtekood 40
Lisa 2 – IT Kolledži multimeediaspetsialisti tagasiside 41

Jooniste loetelu

Joonis 1. TODO nimekiri programmkoodis. 16
Joonis 2. Kommentaarid, mis ei paku lisaväärtust. 17
Joonis 3. Konstandid ja kohad, kus neid võiks kasutada. 17
Joonis 4. Echo360 seadme juhtpaneeli kinnipüütud päringud. 22
Joonis 5. new_capture ja confidence_monitor POST päringute keha. 22
Joonis 6. extend POST päringute keha. 22
Joonis 7. Echo360 salvestusriistvara lõppsõlmed. 23
Joonis 8. scaleY ja sacleX väärtuste arvutamine FXML failis. 27
Joonis 9. @FXML annotatsiooniga märgitud kontrolleri meetodid ja muutuja. 28
Joonis 10. Taaskäivitav toiming, mis loob automaatselt endale lõime. 29
Joonis 11. Joonis 10 kujutatud toimingu kasutamine kontrolleris 30
Joonis 12. Kasutajaliidese tegevuste skeem. 32
Joonis 13. Photoshopis loodud kasutajaliidese avavaate välimus. 34
Joonis 14. Avavaade SceneBuilderis. Vaate komponendid vasakul, valitud komponendi omadused paremal. 34

1 Sissejuhatus

TalTech loenguid salvestatakse Echo360 lahenduse abil. Ruumid, mis toetavad loengute salvestusi on varustatud Echo360 videosalvestiga, mis on ühendatud Echo360 veebiteenusega. Veebiteenuses määratakse salvestuste graafik, mille järgi sorteeritakse tehtud salvestusi automaatselt. Graafikuväliseid salvestusi on vaja hallata käsitsi.

Loengu salvestuse kontrollimiseks on olemas salvestusriistvara juhtpaneel, mis nõuab autentimist. Alternatiivselt käivitatakse salvestus läbi Echo360 veebiteenuse. Mõlemad viisid on sammude rohked ja seetõttu ebamugavad. Antud probleemi adresseerimiseks loodi esialgne EchoMonitori lahendus. Salvestusega ruumidesse paigutati sülearvutid, kuhu oli paigaldatud EchoMonitor nimeline töölauarakendus. Sülearvutid on ühendatud koos salvestusriistvaraga ühisesse lokaalsesse võrku. Rakendus korraldab autentimist ning andmevahetust vastava ruumi salvestusriistvaraga. Loengut pidava isiku ülesandeks jääb vastavate nuppude vajutamine.

EchoMonitori rakenduse kasutamise käigus avastati tarkvaravigu ning tekkis vajadus uue funktsionaalsuse järele. Antud bakalaureusetöö eesmärgiks on asendada olemasolev EchoMonitor uue versiooniga, mis kõrvaldaks eelkäia puudujääke, oleks jätkusuutlik ning vastaks püstitatud nõuetele.

Analüüsi käigus määratakse nii olemasoleva kui ka soovitava tarkvara funktsionaalsus. Vaadatakse üle eksisteerivaid puudujääke. Tuuakse välja erinevaid probleemi lahendusviise. Valitakse optimaalseim teekond tulemuse saavutamiseks ning põhjendatakse tehtud otsus.

Bakalaureusetöö praktilises osas käsitletakse arendusprotsessi kulgemist, vaadatakse üle töökäigu jooksul tehtuid otsused tuuakse esile tekkinud probleemid koos vastava lahenduskäiguga ning teostatakse tarkvara testimist ja kontrollitakse tulemusi.

2 Analüüs

2.1 Funktsionaalsuse määramine

Uus EchoMonitor täidab olemasoleva versiooni olulisemaid funktsionaalsuseid ning rakendab läbiräägitud uuendusi tagamaks parima kasutajakogemuse loengu salvestuste juhtimisel. Järgnevalt on välja toodud eelnevalt mainitud eesmärgi täitmiseks vajaliku funktsionaalsuse loetelu.

2.1.1 Esmajärguline funktsionaalsus

Tarkvara sätted on paindlikud, tagamaks ühilduvuse erinevate Echo360 salvestusseadmetega. Riistvaraga ühenduse loomiseks määratakse vastava seadme lokaalvõrgu aadress, kasutajanimi ja salasõna. Autoriseerimiseks vajalikud andmed salvestatakse esmasel seadistamisel. Salvestatud andmeid saab jooksvalt muuta sätetes. Tarkvara haldab iseseisvalt salvestusriistvara autentimisprotsessi, kõrvaldades vajaduse kasutajanime ja parooli küsimise järele enne loengu salvestuse algust. Salvestuse päritolu jälgimiseks määratakse tarkvara asukoht ruumi numbri järgi.

EchoMonitor loob ühenduse lokaalvõrgus asuva salvestusriistvaraga ning veendub seadme valmisolekus. Tarkvara teavitab kasutajat kui ka Echo360 salvestusriistvarahaldajat programmi töökäigus tekkinud võrgutõrgetest. Kasutajale kuvatakse vastav veateade. Konfiguratsioonis määratud vastutava isiku e-mailile saadetakse kokkuvõtte juhtumist. Kasutaja saab teavitada administraatorit tekkinud tõrgetest kastutades etteantud tüüpvigade valikut.

Programm käivitab Echo360 veebiteenuses etteplaneeritud salvestusi. Samuti on võimalik alustada graafikuväliseid *ad-hoc* salvestusi. Käsitsi käivitatud salvestustele määratakse vaikimisi salvestusperioodi pikkus. Ülikooli süsteemi administraatori e-mailile saadetakse välja informatiivne märguanne, tagades parema käsitsi käivitatud salvestuse päritolu jälgimise. *Ad-hoc* salvestuse kuuluvust määratakse läbi Echo360 veebiteenuse. Üldise *„Ad Hoc Capture“* salvestuse nimetuse tõttu on raske määrata salvestus kindla aine alla.

Alustatud salvestust on võimalik seisata, pikendada või lõpetada. Antud funktsionaalsus on salvestuse jooksul kergesti kättesaadav. Seisatud salvestuse olukorras kuvatakse vastav märguanne ning antakse valik sessiooni jätkamiseks või lõpetamiseks. Salvestuse pikendamisel lükatakse vaikimisi määratud lõpetamise aeg valitud ühiku võrra edasi, suurendades sellega salvestuse pikkust. Ülekande lõpetamisel suunatakse kasutaja avavaatele.

2.1.2 Salvestusprotsessi toetav funktsionaalsus

EchoMonitor näitab Echo360 veebiteenuses etteplaneeritud ruumi salvestuse graafikut, kas salvestuse või päeva kaupa. Tuuakse välja salvestuse alguskuupäev, loengu nimi, salvestuse kestus ning kui palju aega on jäänud järgmise salvestuse alguseni. Salvestuse käigus kuvatakse oletatav loengu lõppaeg.

Salvestuse käigus kuvatakse Echo360 riistvaraga ühendatud videosisendeid nagu õppejõu ekraan või ruumi videokaamera. Vastavalt lektori valikule kuvatakse korraga üks või mitu videoallikat. Meediat päritakse salvestusriistvara käest videoallikate olemasolu kontrollimise eesmärgiks. Pildi puudumise korral kuvatakse vastav teadaanne.

Loengu jooksul jälgitakse salvestuse helikvaliteeti. Kasutatava mikrofoni jaoks seadistatakse valjususe ülem- ja alampiir helikvaliteedi tuvastamiseks. Pikema vaikuse korral juhitakse sellele tähelepanu kuvatud sõnumiga kui ka hoitava heliteatega.

Väljaspool salvestamise sessiooni on võimalik käivitada monitooring video sisendite silumise eesmärgiks. Monitooringu käigus simuleeritakse tavaline loenguseanss, salvestuse pilve laadimist ei teostata. Kasutaja näeb Echo360 riistvaraga ühendatud seadmetest väljuvaid signaale. Monitooring on kasulik uute seadmete ühendamisel Echo360 riistvaraga veendumaks seadmete tõrgete vabas koostöös. Monitooringut tasub teostada ka peale Echo360 tarkvara uuendamist.

2.1.3 Väheoluline funktsionaalsus

Olemasolevas EchoMonitoris esineb funktsionaalsust, mis ei paku tunduvat lisaväärtust püstitatud eesmärgist lähtudes. Arenduse perspektiivist koheldakse taolist funktsionaalsust kõige madalama prioriteediga või kaalutakse eemaldamise kasuks.

Esialgne EchoMonitor toetab keelte valikut. Püstitatud eesmärgi saavutamiseks luuakse võimalikult lihtsa kasutajaliidesega tarkvara. Loengute juhtimiseks mõeldud käsud peavad olema intuitiivsed, seega puudub vajadus keeruliste lauseehituste jaoks. Vaikimisi kasutajaliidese keeleks valitakse inglise keel.

EchoMonitoris rakendatud klaviatuuri otseteed võivad olla kasulikud klaviatuuri olemasolul. Käskude efektiivne kasutamine eeldab nende meeldejätmist, mis on vastuolus eesmärgiga luua võimalikult mugav salvestusi juhtiv tarkvara. Arvestades puutetundliku ekraani kasutamisega, ei paku mainitud omadus lisaväärtust.

Sätetes toetatakse kasutajaliidese visuaalsete komponentide väljalülitamist. Pürgides minimalistliku ja informatiivse kasutajaliidese suunas on antud funktsionaalsus üleliigne, kuna programmi eesmärgiks on näidata lektorile ainult vajaliku teavet.

Silumisrežiim, mida võib sisse lülitada tarkvara sätetes. Antud lahendus on vananenud arvestades uue funktsionaalsusega, mille eesmärgiks on saata administraatori e-mailile raport juhtumi kokkuvõttega. Uue lahenduse raames teostatakse andmevahetust vastutava isikuga, tagades kiirema tehnilise toe reageerimisaja.

Ekraanirežiimide vahetuse aktuaalsus on küsitav, kuna salvestuste juhtimiseks on paigaldatud eraldi riistvara, millel käivitatakse EchoMonitori tarkvara. Antud funktsionaalsus võib osutuda vajalikuks lahenduses, kus EchoMonitor on mõeldud jooksma paralleelselt teiste rakendustega. Lähtudes olemasolevast riistvaralisest lahendusest on sarnase funktsionaalsuse kasulikkus ebatõenäoline.

2.2 Riistvaralised lahendused

Uue EchoMonitori eesmärgiks on kokkusobivus erinevate riistvaraliste lahendustega. Programm peab toetama lisaks hiirele ja klaviatuurile ka puutetundlikku lahendust ja efektiivselt realiseerima etteantud ekraanipinda, mille suurus võib varieeruda. Potentsiaalne riistvara jõudlus vastab EchoMonitori nõuetele ning võib kasutada ARM või x86 tüüpi protsessoreid. Riistvara toetab ekraani väljundit, vähemalt ühte kasutaja sisendit ja omab võrgukaabli pesa.

Hetkel kasutatakse rakendust sülearvutitel, millel on võrgukaabli sisend. Sülearvuti on ühendatud võrgukaabliga lokaalsesse võrku, milles asub kogu majasisene Echo360

salvestusriistvara. Tarkvara juhtimiseks kasutatakse sülearvuti puutepaneeli ja klaviatuuri. Antud lähenemisviis on säästlik, kuna realiseeritakse vana riistvara, mille efektiivsus jääb tänapäeva mudelitele tunduvalt alla Lisaks on sülearvuti kõik ühes platvorm, mis hõlmab endas kõiki vajalikke sisend- ja väljundliideseid: klaviatuur, hiir, ekraan, kõlarid. Seevastu on antud lähenemisviisil hulk puudujääke. Kasutatav riistvara kohandati etteantud ülesande täitmiseks. Tegemist on tootega, mille esialgne eesmärk on olla kaasaskantav töömasin. Salvestuse juhtimise vaatepildist on sülearvuti kasutamine kohmakas ja ebamugav. Vaatamata kasutajaliidese lihtsusele ja komponentide suurusele on kasutajaliideses orienteerumiseks vaja keskenduda väikesele hiirekursorile.

Salvestuste juhtimiseks sobivad puutetundlikud terminalid nagu HP Engage One või Lenovo THINKSMART HUB. Antud tootegrupi liikmed on varustatud puutetundliku ekraaniga ja enda kompaktsuse juures keskpärase sülearvuti võimsusega riistvaraga, mis kujutab endast sülearvuti gabariitidega puutetundliku ekraaniga lauaarvutit. Toote disain eeldab kasutamist terminalina konverentsi ruumides või jaemüügi valdkonnas, kus on vaja saavutada eesmärk võimalikult mugavalt ja kiiresti tehes võimalikult vähe liigutusi. Tegemist on nišitootega, mille eesmärgiks on olla kompaktne, statsionaarne ja puutetundlik. Loetletud omadused sobivad kokku loengute salvestuste juhtimiseks ning oleks moderne asendus sülearvuti lahendusele. Riistavaral on stiilne korpus ning see on juba komplekteeritud. Kasutamiseks on vaja seade ainult võrku ühendada ja panna peale soovitud tarkvara. Riistvara toetab Windows ja Linux operatsioonisüsteeme. Puutetundliku terminali puudujäägiks on kõrge hind, mis on tuletatud nišitoote omadustest. Universaalsuse mõttes on terminali paindlikus, võrreldes sülearvutiga, samuti madalam. Lauarvuti vaatenurgast on tegemist kompaktse masinaga, mille hind on kõrgklassi lauarvutite piirides, kuid jõudlus on võrreldav madalama klassi süsteemiga. [1] [2]

Alternatiivne riistvara probleemi lahendamiseks on miniarvutid. Antud lahenduse raames vaadatakse täpsemalt Raspberry Pi tooteid. Raspberry Pi 4 model B on suurust arvesse võttes piisavalt võimas riistvara koos neljatuumalise protsessoriga, taktsagedusega 1,5 GHz ja kuni 4GB operatiivmäluga. Arvutil on olemas kõik vajalikud sisend ja väljund ühendused. Tegemist on energiasäästliku riistvaraga. Raspberry Pi pakub seitsme tollist puutetundlikku ekraani, mis ühildub Raspberry Pi 4 plaadiga. Ekraani nõrkuseks võib osutuda madal, 800 x 480 pikslit, resolutsioon kui ka väike

ekraani pindala, mis võib piirata efektiivset lugemiskaugust. Antud puudujääki saab kompenseerida kasutajaliidese komponentide suurusega. Tuleb arvestada, et kasutajaliides peab olema võimalikult lihtne. Raspberry Pi 4 toetab Power over Ethernet moodulit. PoE komponenti kasutus võimaldab eemaldada toiteploki, elimineerides sõltuvuse pistikute asukohast. Miniarvutite lahendusele lisab keerukust vajadus korpuse järgi. Eelistatud on puutetundlik, pööratav ja kallutatav kuvari korpus. Ekraani paigutuse paindlikkuse tagamiseks, tuleb projekteerida ja trükkida nõuetele vastav lahendus või leida valmis korpuse lahendus. Võrreldes HP ja Lenovo puutetundlike terminalidega on Raspberry Pi lahenduse tüki hind vähemalt poole odavam ning võimsus poole nõrgem. Miniarvuti riistvara kasutusele võtmine vajab leiutamist, kokkupanemist ja sätestamist, kuna tegemist on erinevateks projektideks mõeldud universaalse platvormiga. Seevastu riistvara modulaarsus võimaldab mugavat üksikute komponentide asendamist. Taaskasutamise mõttes võib Raspberry Pi riistvara sobitada teisteks eesmärkideks, näiteks robootikaklubi projektidesse. [3] [4] [5] [6]

2.3 Tarkvaralised lahendused

2.3.1 Olemasoleva EchoMonitori platvorm

EchoMonitor on Windows, Mac OS X ja Linux toega töölauarakendus. Programm on kirjutatud ActionScript keeles, kasutades Adobe Flex tarkvaraarenduskomplekti ja Adobe AIR käituskeskkonda. [7]

Adobe Flex on mõeldud Adobe Flash baasil multiplatvormsete rakenduste loomiseks. 2011. aasta omanikuvahetuse käigus sai Apache uueks Flex projekti omanikuks. Apache Flex toetab järgmiseid platvorme: Microsoft Windows, Mac OS X, Apple iOS, Google Android ja RIM BlackBerry. Linuxi tugi on eksperimentaalses seisundis ning selle arendus rohkesti sõltub projekti huvilistest. Viimane arendus tehti 2017. aasta detsembris, mis võib viidata Apache Flex populaarsuse langusele. [8] [9] [10]

Adobe Air käituskeskkond kasutab HTML, JavaScript, Adobe Flash, Apache Flex ja ActionScript tehnoloogiaid. Adobe AIR eesmärgiks on käivitada Flash Player rakendusi kui ka veebirakendusi töölauarakendustena Windows, Mac OS, Android ja BlackBerry Table OS platvormidel. Adobe AIR rakendus on pakitud arhiivi, mille käivitamine tagab tarkvara automaatse paigalduse. Adobe lõpetab AIR tuge 2020. aasta lõpuks.

Kohustuse võtab üle Harman, Samsungile kuuluv ettevõtte, mille tegevusharude sekka kuuluvad autoelektroonika ja heli. [11] [12]

ActionScript on JavaScriptiga sarnanev objektorienteeritud programmeerimiskeel, mis on mõeldud Adobe Flash Player ja Adobe AIR käituskeskkondadele. ActionScripti esialgseks eesmärgiks oli kontrollida Adobe Flash animatsioone. ActionScripti viimast, 3.0, versiooni integreeriti Adobe AIR keskkonnaga. Lähtudes ActionScript 3.0 arendaja juhendist toimus viimane programmeerimiskeele uuendus 2017. aastal. [13] [14] [15] [16]

2.3.2 Olemasoleva EchoMonitor projekti struktuur

EchoMonitori sissejuhatav *readme* fail kirjeldab projekti eesmärki, funktsionaalsusese ja paigalduse käiku. Projekti sees on kaks kausta: *assets* ja *locale*. Assets kaustas on graafilise kasutajaliidese poolt kasutatavad pildid ja ikoonid. *Locale* kataloogis asuvad keeltega seotud failid. *EchoMonitor-app.xml* sätestab Adobe AIR käituskeskkonnale vajalikud parameetrid EchoMonitor tarkvara kohta nagu tarkvara nimi, versioon, autoriõigused ja kuvatava akna omadused. Failid *configuration.xml* ja *default_configuration.xml* on identse sisuga – mõlemad sisaldavad vaikimisi seadeid tarkvara tööaja jaoks. Projekti loogika on määratletud *EchoMonitor.mxml* failis. Graafilise kasutajaliidese kuju on kirjeldatud MXML keeles. Ülejäänud loogika on ActionScript keeles. [17]

EchoMonitor.mxml faili suurus, mis on üle tuhande rea, raskendab arendus protsessi. Koodi lugemine, mõistmine ja navigeerimine on häiritud liigse kerimise tõttu. Koodi komponendid ei asu enam lähestikku. Õige järje leidmine vajab rohkem keskendumist ning aega. Mitme keele kasutamine ühes failis aeglustab koodi lugemist, kuna lugeja peab kohanema erinevate keelte süntaksidega. Graafilise kasutajaliidese ja loogika paiknemine samas kohas läheb SOLID tava ühe vastutuse (*Single responisibility*) idee vastu. Loetavuse parandamiseks on tarvis eraldada graafiline kasutajaliides ja loogika. Loogika koodi on omakorda vaja jaotada alamrühmadesse vastavalt projekti omadustele, näiteks vaadete kohaselt või funktsionaalsuse järgi. Clean Code autor leiab, et optimaalse ridade arvuga fail mahub tervenisti ekraanile. Antud mõõteviis on suhteline, võttes arvesse monitoride suuruse kasvu ja arendaja fondisuuruse eelistusi. Antud rusikareegli äärmuslik järgimine võib tekitada olukorra, kus projektis on raske

orienteeruda suure failide koguse tõttu. Tuleks lähtuda projekti suurusest ning leida kuldne vahetee failide koguse ja suuruse vahel. [18]

EchoMonitor.mxml alguses on väljatoodud TODO nimekiri, kus on korraga nii tehtud kui ka ootel tööd. Vastavalt Clean Code tavadele on tegemist aegunud töökäigu dokumenteerimis meetodiga. Tänapäeva versioonihaldustarkvara on piisavalt võimas, et antud ülesandega toime tulla. [18]

```
TODO:
 - move the user information to EncryptedLocalStore
 - handle monitoring from different timezones than the
       appliance with correct dates and times
 - handle locale change with more grace :)
 - [DONE] implement audio level monitoring
 - [DONE] clear audio error message when a capture ends
 - [DONE] notify the user of any network/connection
       problems that cause EchoMonitor not to be able
       to connect to the selected appliance
 - [DONE] optimize connections to the capture appliance
 - [DONE] improve the display of monitoring area
       (hide scrollbars)
```

Joonis 1. TODO nimekiri programmkoodis.

Koodis esineb tühja kommenteerimist nagu funktsiooni nime mainimine enne ja pärast funktsiooni. Antud käitumise põhjal võib oletada, et autor püüab koodi loetavust parandada. Tõenäoliselt on kommentaaridega ümbritsetud funktsioon liiga pikk või funktsiooni nimi ei suuda piisavalt selgelt selgitada funktsiooni kavatsust. Tühja sisuga korduvad kommentaarid ei pakku lisa väärtust, kuna koodi lugeja õpib kiiresti neid ignoreerima. Parimaks lahenduseks oleks pikk ja keeruline funktsioon jaotada väiksemateks osadeks. [18]

```
// infoHandler
private function infoHandler(event:ResultEvent):void {
     var result:XML = event.result as XML;
     var now:Date = new Date();
     var startTime:Date;
     var durationInMinutes:int;
     var difference:Number;
     var minutesToNextCapture:int;
     var hoursToNextCapture:int;

     deactivateInfoBox(); // Hide infoBox
     networkErrorCount = 0; // Set the error count of network
     errors to 0
```

Joonis 2. Kommentaarid, mis ei paku lisaväärtust.

Projektis on rakendatud konstandid, mis annavad määratud numbritele konteksti ja seega parandavad koodi loetavust. Kohakuti jäi valitud lähenemisviis rakendamata.

```
// Define some constants
private const day:int = 1000 * 60 * 60 * 24;
private const hour:int = 1000 * 60 * 60;
private const minute:int = 1000 * 60;
private const monitorImageWidth:int = 320;

audioInputLevel.y = audioInputLevelVUMeter.y - (audioPeakLevel
* (240 / 32768));

configuration.infoRequestTimeout = 3; // in seconds
configuration.infoTimerActive = 3000; // in milliseconds
configuration.infoTimerIdle = 10000; // in milliseconds
```

Joonis 3. Konstandid ja kohad, kus neid võiks kasutada.

Joonises 3 välja toodud koodi lõigus, kus määratakse audioInputLevel, ei saa kindalt väita, mida numbrid 240 ja 32768 tähendavad. Mõnedes kohtades kompenseerib autor konstantide puudumist kommentaaridega. Selle asemel saaks luua kirjeldava nimega konstandi. Koodis olevate konstantide nimed ei vasta kokkulepitud ActionScript3 stiili nõuetele. *EchoMonitor.mxml* failis esineb mitmeid korduvaid sõnesid, mida tasuks muuta üheks konstandiks võimalike muudatuste tegemise lihtsuseks ja näpuvigade minimiseerimiseks. [18] [19]

2.3.3 Alternatiivsed platvormid ja lahendused

Uus EchoMonitor jääb klient töölauarakenduseks, mis teostab andmevahetust Echo360 salvestusriistvaraga. Küsimuse all on platvorm, mille peal tarkvara realiseerida. Potentsiaalne platvorm kasutab populaarset objektorienteeritud keelt ning on suure

kasutajabaasiga, mis viitab platvormi heale jätkusuutlikkusele. Toetatud on multiplatvormsus, graafiline kasutajaliides, teostakse HTTP päringuid, võetakse vastu ja tõlgendatakse vastust, mis võib olla STREAM tüüpi. Antud lõputöö käigus võrreldakse C#, Java ja Python programmeerimiskeeli kasutavaid raamistikke, kuna eelnevalt nimetatud programmeerimiskeeled on kohustuslikult läbivõetud Infotehnoloogia süsteemide arenduse õppekavas.

C# sai alguse 2000ndate alguses olles C++ ja Java segu. Tegemist oli Microsoft .NET raamistike jaoks loodud range tüüpsusega programmeerimiskeelega. NET Core ilmumisega, leidis C# kasutust multiplatvormses maailmas. Net Core 2.1 sai realiseerida tagarakendusi nii Windows, Mac OS ja Linuxi peal. Otsene töölauarakenduse tugi Linux ja Mac operatsioonsüsteemide jaoks puudub. NET Core 3.1 versiooniga tekkis Windows Forms teegi tugi, kuid see ei käivitu Windows platvormi väliselt. WPF on ainult Windowsil toetatud uuem graafilisekasutajaliidese raamistik. Kasutades kolmanda osapoole poolt arendatud Mono käituskeskkonda saab käivitada .NET Foundation rakendusi nii Linuxil kui ka Mac OSil. Mono toetab hulka graafilise kasutajaliidese platvorme, nende seas on Windows Forms. Avalonia UI on alternatiivne valik graafilise kasutajaliidese lahenduseks, mille tugevuseks on .Net Core tugi. Vastukaaluks, Avalonia UI võib olla ebastabiilne, kuna projekt on alles arenguetapis. Kombineerides .Net Core ja Avalonia UI kaotatakse vajadus Mono käituskeskkonna järgi. [20] [21] [22] [23] [24]

Java sai alguse 1991. aastal Green Team projektina televisiooni riistvara juhtimise eesmärkideks. Algusest peale oli programmeerimiskeele omaduste seas objektorienteeritus ja platvormist sõltumatus. Java ilmus avalikult turule 1995. aastal ja on sellest ajast alates edasi arenenud. Java raamistikest paistab silma eelkõige Spring. Tegemist on populaarse platvormiga, mis sisaldab endas hulka raamistikke, mida annab vastavalt vajadusele projekti juurde lisada. Spring eesmärgiks on kiirendada arendusprotsessi, eemaldades väärtuse loomist aeglustavaid korduvaid komponente abstraktsiooni abil. Ilmunud 2003. aastal, tegemist on ajale vastupidanud raamistikuga, mida ümbritseb suur kommuun. Graafilise kasutajaliidese loomiseks on mitmeid valikuid nagu Swing ja JavaFX. Swing on Java käituskeskkonnaga kaasa antud hästi dokumenteeritud raamistik, mis on eksisteerinud pikemat aega. Eesialgu Oracle poolt arendatud JavaFX on Swingi järeltulija. JavaFX muutus avatud lähtekoodiga OpenJFX

projektiks. Alates Java 11. versioonist on JavaFX teek Javast eraldatud. [25] [26] [27] [28] [29]

Python, objektorienteeritud dünaamiliselt tüübitud multiplatvormne programmeerimiskeel, mille eesmärgiks on olla lihtne ja mugav. 1991. aastal ilmunud ja eksisteerimise jooksul mitmeid olulisi uuendusi adapteerinud Pythoni populaarsus on aktiivselt kasvanud. Keelel on palju teeke ja raamistikke, mis tõstavad arendaja tõhusust. Töölauarakenduste loomise mugavdamiseks on Pythonil pakkuda mitmeid rammistike nagu Kivy, Tkinter ja WxPython. Kivy on riistvaralist kiirendust toetav multiplatvormne raamistik. Tkinter on kaua eksisteerinud populaarne hästi dokumenteeritud graafilise kasutajaliidese raamistik, mis on lihtne ning tuleb Pythoniga vaikimisi kaasa. WxPython põhineb WxWindows C++ teegil, mida adapteeriti Pythoni jaoks. Nagu Tkinter tuleb WxPython Pythoniga vaikimisi kaasa. HTTP päringute loomise lihtsustamiseks saab kasutada *Requests* teegi. [30] [31] [32] [33] [34]

2.4 Valitud lahendus

Sobiva lahenduse valimiseks võeti arvesse rakendusele määratuid nõudeid kui ka autori pädevusi. Valitav lahendus peab olema levinud, ajakohane ning sobima seatud eesmärkide saavutamiseks. Lahendus toetab vähemalt Windows ja Linux operatsioonsüsteeme. Lahendus on õppekavas läbivõetud ja seega on autorile tuttav.

Autor otsustas uue EchoMonitor projekti kasuks, kuna otsingu tulemuste põhjal on Adobe platvormi jätkusuutlikus kõvasti madalam, võrreldes eelnevalt loetletud alternatiivsete lahendustega. Projekti struktuur kui ka kasutajaliidese välimus vajavad ümberehitust, mis on võrdeline uue rakenduse kirjutamisega.

Uutest lahendustest vastab püstitatud nõuetele paremini Java keelel põhinev lahendus. Autoril on kolmest valitud keelest kõige rohkem kogemust Java keelega. Mõlemad JavaFX ja Spring raamistikud on õppekavas läbivõetud. Java vastab multiplatvormsuse nõudele ning on suuteline konkureerima C# ja Pythoniga.

C# raamistikel on eelis Windows platvormi peal, kuid arengu etapis oleva Linuxi toe tõttu ei ole C# püstitatud eesmärkide täitmiseks sobilik. Pythoni keelel põhinev lahendus jäi valimata, kuna autoril puudus vajalik Pyhtoni keele kogemus.

3 Arenduskäik

Käesolevas peatükis tuuakse ülevaade töökäigust, alustades tööriistade valikust kuni rakenduse protsessi kirjeldamiseni.

3.1 Tööriistad

Projekt arendati JetBrains IntelliJ programmeerimiskeskkonnas. Tegemist on subjektiivse valikuga, kuna otsustavaks faktoriks oli autori positiivne kogemus valitud tööriistaga õppetöö ja praktika käigus.

SceneBuilder on Oracle poolt levitatud avalik graafiline töörist FXML vaadete ehitamise lihtsustamiseks. SceneBuilder kasulikkus seisneb kasutajaliidese vaadete ehitamise protsessi lihtsustamises. Arendaja ei pea käsitsi kodeerima vaadet ja veenduma, et kasutajaliides näeb välja korrektne. Komponentide lisamine vaatele ehk FXML faili arendus toimub lohistamise viisil. SceneBuilder näitab lõplikku tulemust ning genereerib vastava FXML koodi iseseisvalt. JavaFX teegis on hulk komponente, mis täidavad erinevaid eesmärke. Komponente võib paigutada teiste komponentide sisse moodustades vastastikuse suhte pea- ja alamkomponendi vahel. Vajutades valitud komponendile saab muuta komponendi omadusi, mis kirjeldavad elemendi välimust, sisu ja ühendust programmi käitusaja sündmustega. Tuvastades vaate elementide seost CSS stiilifailiga on SceneBuilder võimeline kuvama CSS stiilifailis määratud komponentide eripära.

On olemas olukordi, kus SceneBuilder on kasutu. Dünaamiliste objektide puhul, mille olekud ja arv on ettearvamatu, tuleb loogika määrata vastava vaate kontrolleris. SceneBuilder ei suuda tõlgendada FXML keele poolt toetatud primitiivseid avaldisi. Sellisel juhul tuleb avaldise korrektsuses veenduda rakenduse käivitamisel. [35]

Postman tuli kasuks Echo360 salvestusriistvara päringute kaardistamisel. Rakendus võimaldab sorteerida päringuid vastavalt soovile. Salvestatud päringuga seotakse tagastatud tulemus. Igat päringut on võimalik korrata. Arenduse käigus loodi uus päringute kollektsioon ühise päise seadistusega. Saadud päringute tulemusi kasutati hiljem lõppsõlmede simuleerimiseks.

Adobe Photoshopi abil loodi soovitav kasutajaliidese välimus, kasutades kihte ja soovituid efekte. Autor otsustas Photoshopi heaks eelnevalt omandatud pädevuse tõttu. Alternatiivselt oleks võimalik kasutada avatud lähtekoodiga pilditöötlusprogrammi GIMP, mis samuti toetab kihtide kasutamist.

3.2 Salvestusriistvara sõnumite kaardistamine

Echo360 veebiteenusel on olemas avalik API dokumentatsioon. Seda ei saa väita Echo360 salvestusriistvara API kohta, kuna autor pole infot leidnud kättesaadavatest allikatest. Annab järeldada, et Echo360 suunab kasutama seadmeid läbi enda veebiteenuse või seadme enda kohaliku juhtpaneeli. [36]

Salvestusseadme juhtimeseks vajalikud lõppsõlmed avastati käsitsi kohalikus võrgus. Echo360 seadme juhtpaneel sarnaneb põhimõtte poolest ruuteri juhtpaneeliga. Ligipääsu saamiseks on vaja teada masina lokaalset aadressi ning läbida autoriseerimine.

Sisseloginud tavakasutaja näeb kolme vaadet: *Captures*, *Monitoring*, *System*. *Captures* ehk salvestuste vaates kuvatakse järgmise salvestuse kohta teavet. Antakse võimalus käivitada planeerimata salvestus sobiva kirjeldusega ja ajavahemikuga. *Monitoring* (jälgimise) alt kuvatakse viivitusega aktiivse salvestuse helitaset ja videosisendeid. Seadme sisendite häälestamiseks saab käivitada jälgimisfunktsiooni. Süsteemivaade pakub üldist infot nagu tarkvara versioon. *Admin* õigustega kasutaja saab lisaks üleval loetletud funktsionaalsusele vaadata seadmesse laetud seadistusandmeid, salvestuste tunniplaan kaasaarvatud.

```
GET päringud
http://{ip_aadress}/status/captures
http://{ip_aadress}/status/system
http://{ip_aadress}/status/monitoring
http://{ip_aadress}/monitoring/{pildi_nimi.laiend}
http://{ip_aadress}/diagnostics/system-info/tasks

POST päringud
http://{ip_aadress}/capture/confidence_monitor
http://{ip_address}/capture/stop
http://{ip_address}/capture/new_capture
http://{ip_address}/capture/pause
http://{ip_address}/capture/extend
http://{ip_address}/capture/record
```

Joonis 4. Echo360 seadme juhtpaneeli kinnipüütud päringud.

EchoMonitori toimimise jaoks vajalikud lõppsõlmed saadi kätte jälgides brauseri konsoolis võrguliiklust, liikudes mööda eelnevalt nimetatud vaateid (vt. Joonis 4). Echo juhtpaneeli teenus küsib iga sekundi tagant süsteemi infot *(…/status/system),* millele lisandub vaatele vastav päring *(…status/captures)* või *(…status/monitoring)*. Videosisendi näitamiseks päritakse .jpg formaadis videosisendi tüübi nimest tuletatud nimega pildid *(…/monitoring/{pildi_nimi.jpg}*). Pärides admin õigustega tunniplaani tõmmatakse alla *(…/diagnostics/system-info/tasks)* lingilt semestri pikkuse ruumi salvestuste graafik.

POST päringud tekkisid nuppude vajutamise tulemusena. Stopp *(…/capture/stop),* paus *(…/capture/pause)* ja jätka *(…/capture/record)* päringud saadeti tühja kehaga. Jälgimise aktiveerimise *(…/capture/confidence_monitor)* ja planeerimata salvestuse alustamise *(…/capture/new_capture)* käsud nõuavad järgmist keha sisu, kus „*description*“ määrab salvestuse pealkirja, *„duration“* väärtus seadistab salvestuse pikkuse sekundites ning *„capture_profile_name“* valib sobiva salvestusprofiili (vt. Joonis 5).

```
description="{salvestuse_pealkiri}"
&duration="{periood_sekundites}"
&capture_profile_name={salvestuse_profiili_nimi}
```

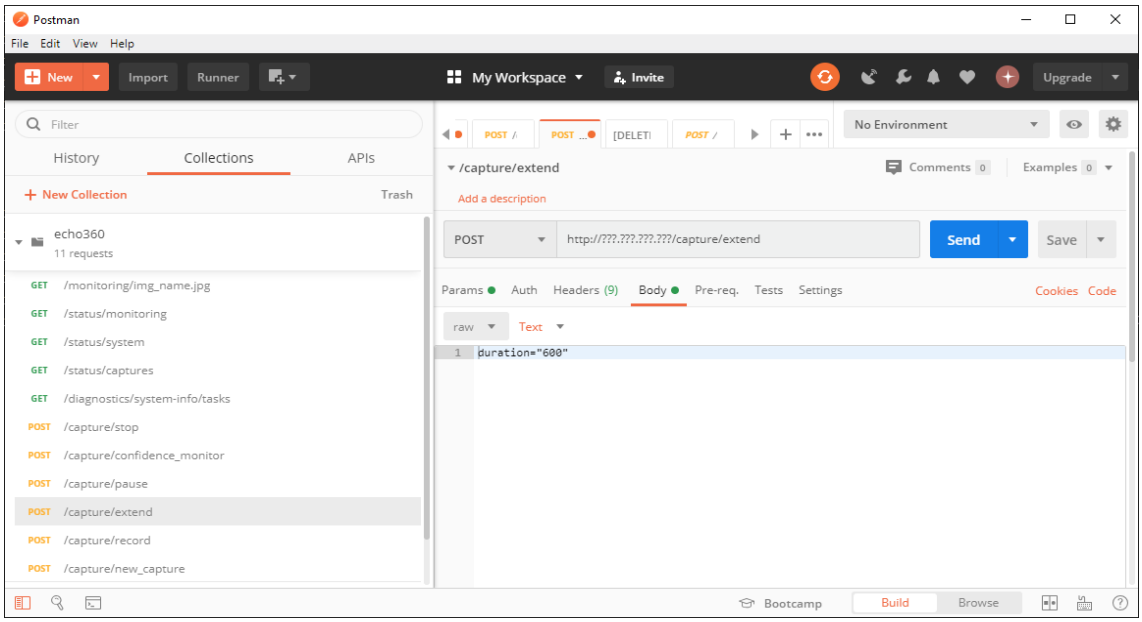
Joonis 5. new_capture ja confidence_monitor POST päringute keha.

Salvestuse pikendamise käsk *(…/capture/extend)* vajab pikkuse *„duration“* väärtust sekunditest (vt. Joonis 6)

```
duration="periood_sekundites"
```

Joonis 6. extend POST päringute keha.

Kõik eelnevalt nimetatud päringud sisestati Postman rakenduses eraldi loodud kausta ning seejärel käivitati järjest. Veenduti linkide korrektsuses ning kätte saadud päringute vastused salvestati, et jätkata rakenduse arendust salvestusriistvara asukohast sõltumata.



Joonis 7. Echo360 salvestusriistvara lõppsõlmed.

3.3 Tehnoloogia

3.3.1 Valitud platvormi komponentide versioonid

Stabiilsuse tagamiseks valiti platvormi aluseks Java 11 LTS versioon, mille tugi kestab 2023. aasta septembrini. Valitud Java versioon on kooskõlastatud tänapäeva operatsiooni süsteemidega. Seoses olemasoleva riistvaralise lahenduse toetamise nõudega, tuli arvestada, et Java 8 on viimane versioon, mis käivitub Windows XP peal. Järelikult peab rakendus toetama mitut Java versiooni. [37]

Projekti arendusprotsessi käigus konfigureerimise vähendamise eesmärgiks langetati otsus Spring Boot raamistiku heaks. Valiti hiljutisem 2020. aasta märtsis ilmunud 2.2.6 versioon [38]

JavaFX raamistiku versiooni valik sõltub projekti Java versioonist. Viimane avalik JavaFX versioon Java11 jaoks on 11.0.2. [39]

3.3.2 Muu tehnoloogia

Gradle, projektide ehitamiseks kui ka sõltuvuste haldamiseks mõeldud tööriist, laeb automaatselt sätestatud sõltuvusi alla. Gradle tagab ühtlast keskkonna projekti arenduseks, sõltumata masina eripäradest. Maven, Gradle alternatiiv, jäi valimata autori

väheses kogemuse tõttu. Projekti loomise käigus laadis IntelliJ alla automaatselt viimase Gradle 6.3 versiooni. [40]

Lombok vähendab korduva koodi kirjutamist, asendades objekti muutujate *get* ja *set* meetodite kirjutamist @ annotatsioonidega. Korduvate elementide vältimine tagab parema koodi loetavuse, kuna sisu on kompaktsem ning arendaja ei pea kulutama keskendumisvõimet korduvate elementide lugemisele. Projektis rakendati Lomboki 2020. aasta veebruari 1.18.12 versioon [41]

JAXB on XMLi ja Java klassi siduv raamistik. Projektis kasutatakse seda Echo360 salvestusriistavaralt saabuvate sõnumite tõlgendamiseks. Sidumisprotsess toimub Java klassis, kus valitud muutujale määratakse XMLi vastand annotatsiooniga. Kasutusel on viimane stabiilne 2.3.3 versioon. [42]

3.4 Rakenduse struktuur

Analüüsi käigus jõuti järeldusele, et programmikoodi hoidmine ühes failis vähendab loetavust. Antud lähenemisviis ei ole tulevikukindel programmi komponentide tiheda omavahelise seose tõttu. Uues projektis rakendatakse kihilist struktuuri. Projekt jaotati kolmeks loogiliseks kihiks: esitluskiht (*presentation layer*), äriloogikakiht (*business logic layer*) ning andmekiht (*data access layer*).

Esitluskiht juhib andmevahetust kasutaja ja teenuse vahel. Antud projekti raames kuulub esitluskihi alla JavaFX esirakendus. Äriloogikakiht vahendab infot kasutajaliidese ja Echo360 salvestusriistvara vahel. Andmekihis teostatakse päringud Echo360 riistvara suhtes ning võetakse vastu riistvara poolt saadetud XML keeles kujutatud infot. Äriloogika– ja andmekiht asuvad SpringBoot tagarakenduse moodulis.

Kihilise struktuuri kasutamine tagab rakenduse paindlikust võimalikke muudatuste suhtes. Kihiline struktuur piirab rakenduse komponentide omavahelist sõltuvust, mis võimaldab teostada kihisiseseid muudatusi, mis otseselt ei mõjuta teisi kihte. Echo360 salvestusriistvara API struktuuri muudatuse korral tuleb kohandada andmekihti. Esitluskiht ning äriloogikakiht jäävad muutumatuks. Samuti ei mõjuta esitluskihi muudatus ülejäänud kihte.

3.5 SpringBoot tagarakendus

Web-client tagarakenduse mooduli ülesandeks on vahendada informatsiooni Echo360 salvestusseadme ja JavaFX kasutajaliidese vahel. Andmete kaardistamise protsess on jaotatud neljaks pakiks: *bll*, *dal*, *dto* ning *domain*.

3.5.1 Business Logic Layer

Business Logic Layer pakkis teostatakse XML sõnumist kätte saadut *dto* objekti teisendamist domeen objekti, mida kasutab kasutajaliides. Echo360 seade saadab sõnumites palju informatsiooni, millest kasutajaliidesele on kasulik ainult osa. Domeen objekti loomise ajal kaotatakse ebavajalikud andmed. Teisendamine tehakse käsitsi. Sarnane lähenemisviis nõuab aega esmasel seadistusel, kuid pikemas perspektiivis tasub ära muudatuste tegemiste olukorras.

MonitoringStatusService tagastab seade järgmiseid seisundeid: *idle, waiting, active ja paused. Idle* viitab sellele, et masin on ooterežiimis ning hetkel salvestust ei toimu. *Waiting* staatuses olev masin valmistub ette monitooringuks või salvestuseks. Aktiivse salvestuse korral tagastatakse *active* väärtusega sõnum, millele lisanduvad salvestuse sisendite ja väljundite andmed nagu heli tugevus või ekraanipildi asukoht. *Paused* olek käivitub, kui kasutaja on salvestuse peatanud.

CaptureStatusService teeb kättesaadavaks informatsiooni järgmise ja hetkel toimuva salvestuse kohta. Teenuse peamiseks eesmärgiks on tagastada salvestuse nimi, algus kuupäev, periood ja valitud salvestusprofiil.

PostCommandService haldab kasutajaliidese poolt küsitud POST päringuid. POST päringud kasutavad erinevaid lõppsõlmi, kuid sõnumi keha ja käitumine on sarnane. (vt. Joonis 4). Päringud, mis vajavad sisendit saavad kasutajaliidese poolt *CaptureParameters* objekti.

ScheduleService pärib Echo360 riistvaralt tunniplaani faili. Kasutajaliides saab kätte korrastatud sisuga Schedule objekti, mis hoiab endas salvestuse nime, algusaega ja lektori nime.

SystemService tagastab üldist infot salvestusseadme kohta nagu ruumi number, kus masin asub.

ImageService saab kätte etteantud nimega pildifaili ning tagastab baitide massiivi, mida kasutajaliides kuvab pildina.

Kasutajaliides suhtleb ainult *bll* pakis asuvate teenustega. Igal domeen objekti tagastaval teenusel on vastav *mapper* klass, mis võtab sisse päringust saadud *dto* objekti ning loob selle põhjal vastava domeen objekti.

3.5.2 Data Access Layer

Data Access Layer vastutab veebi päringute saatmisest ja vastuvõtmisest. *ClientConfiguration* klassis ehitatakse veebipäringute põhi, mida hiljem rakendatakse päringute ehitamisel Echo360Client klassis. Konfiguratsiooni käigus rakendatakse sätetes määratud parameetrite põhjal lingi esimene osa masina ip-aadressi kujul. Päringu päisesse lisatakse Echo360 seadme poolt autoriseerimiseks nõutud kasutaja andmed.

Echo360Client võtab sisse *ClientConfigration* klassis ehitatud *WebClient* objekti. Igale Peatükis 3.2 kaardistatud sõnumitele (vt. Joonis 4) vastab üks päring. Päringu koostamisel lisatakse masina ip-aadressile lõppsõlme aadress, määratakse vastuse kaardistamiseks mõeldud klass ning päringu elutsükli periood.

3.5.3 Data Transfer Object

DTO pakis asuvad XML sõnumite kaardistamiseks mõeldud objektid, mis omakorda koosnevad alamobjektidest. Igas objektis on hulk muutujaid, millele määratakse @ annotatsiooniga valitud XML element. Selguse mõttes kannavad *dto* juurobjektid vastava teenuse nime, kus seda kasutatakse.

LocalDateTimeAdapter seob XML sõnumis olevaid aega näitavaid elemente. JAXB ei suuda automaatselt tuvastada aja formaati ning luua sellest *LocalDateTime* Java objekti.

3.5.4 Domain

Domain pakis on kõik domeen objektid, mida tagastab teenus kasutajaliidesele kasutamiseks. Antud objektid on mõeldud XML päringu sisu kaardistamise tulemuse edastamiseks ega sisalda äriloogikat.

3.6 JavaFX esirakendus

Kasutajaliidese *gui* mooduli struktuur kujunes Oracle poolt välja toodud parimate JavaFX tavade põhjal. Kasutati MVC *Model-View-Controller* struktuuri, kus *„model“* on domeeni objektid, *„view“* on SceneBuilderi abil ehitatud *fxml* vaate alus ning *„Controller“* juhib vaate tööaja protsesse. Kasutajaliidese vaadete kujundamiseks kasutatakse *resource* kataloogis asuvaid pilte, fondi faile ning CSS stiili faile. [43]

3.6.1 FXML

Fxml pakkis asuvad kõik *.fxml* laiendiga vaadete kujud. Igale vaatele vastab üks *fxml* fail. Parema ettekujutuse loomiseks võib *fxml* fail avada SceneBuilderis.

FXML on XML-põhine märgistuskeel Java objektide konstrueerimiseks. Tegemist on mugava alternatiiviga vaate konstrueerimiseks võrreldes vaate kujundamisega koodis. Antud lähenemisviis sobib JavaFXi kasutajaliidese arendamiseks, kuna XML dokumendi hierarhiline struktuur on paralleelne JavaFXi vaate struktuuriga. [44]

EchoMonitori rakenduse raames vastutab FXML fail vaate staatiliste objektide paigutuse eest ja nende suurusest vastavalt etteantud resolutsiooni suurusele. Vaate komponentide paigutus on suhteline, mis võimaldab kohanemist erinevatele kuvari resolutsioonile. Minimaalne resolutsiooni piiriks määrati *800 x 480* pikslit, mis on Raspberry Pi 7 tollisse puutetundliku ekraani resolutsiooniks. Arenduse käigus kõrgeim kasutatud resolutsioon oli *1920 x 1080* pikslit. Osade vaate elementide suuruse kohandamine toimub läbi *scaleX* ja *scaleY* kordaja, mis on ühendatud primitiivse avaldisega. Saadud väärtus jääb ühe ja kahe vahele.

```
scaleX="${1 + (1 - mainPane.minHeight / mainPane.height)}"
scaleY="${1 + (1 - mainPane.minHeight / mainPane.height)}"
```

Joonis 8. scaleY ja sacleX väärtuste arvutamine FXML failis.

FXML vaatele määratakse kontroller *fx:controller* parameetriga. Kontroller saab pöörduda vaate komponentide poole, millel on määratud *fx:id*. Igale vaate elemendile saab määrata valitud sündmuse käivitamise korral vastava kontrolleri meetodi *onAction="#startCapture"*, kus *onAction* on sündmuse nimi ja *#startCapture* on kontrolleris loodud meetodi nimi.

Ühendamaks CSS stiili fail vaatega kastutakse *stylesheets* seadet, mis viitab faili asukohale. Stiili fail tasub määrata ülemkomponendis, kuna alamkomponendid saavad pärida stiili. Komponendile saab määrata nii CSS stiili ID kui ka stiili klass.

3.6.2 Controller

Controller pakis asuvad kontrollerid on ühendatud vasta fxml failiga. Iga kontroller juhib ühte kindlat vaadet. Mitmes kontrolleris korduvad tegevused toodi välja *helper* alampaki korduste vältimiseks, tagades parema koodi loetavuse. Mitmes kohas kasutatavad elemendid on näiteks kella ja vaate vahetamise käigus käivitavad animatsioonid.

Initsialiseerimismeetodiga *initialize();* saab määrata kontrolleri käitumist vaate laadimise käigus. Meetodit kutsutakse välja ainult üks kord, kui seotud dokumendi sisu on täielikult laaditud. See võimaldab rakendusklassil teha vajalikke sisu järeltöötlusi. *@FXML* annotatsiooniga märgitud muutujad ja meetodid on tehtud FXML failile kättesaadavaks (vt. Joonis 9). [35]

```
@FXML
private Label statusLabel;

@FXML
private void statusLabelClick() {
statusLabel.setText(„Tere“)
}

@FXML
public void initialize() {
statusLabel.setText(„Initialize“)
}
```

Joonis 9. @FXML annotatsiooniga märgitud kontrolleri meetodid ja muutuja.

Kontroller suhtleb SpringBoot tagarakenduses *bll* pakis olevate teenustega. Kuna JavaFX rakenduse lõim vastutab kasutajaliidese uuenduse eest, on oluline vältida pikaajaliste ülesannete rakendamist samas lõimes. Vastasel juhul kasutajaliidese reageerimisaeg muutub aeglaseks. Antud probleemile parimaks lähenemisviisiks on luua mitmeid toiminguid ühel või mitmel taustalõimel ning lasta peamisel JavaFX rakenduse lõimel kontrollida toimingute poolt loodud sündmusi. [45]

```
@Component
public class MonitoringStatusClient extends
ScheduledService<MonitoringStatus> {

     private final MonitoringStatusService
     monitoringStatusService;

     public MonitoringStatusClient(MonitoringStatusService
     monitoringStatusService) {
          this.monitoringStatusService =
          monitoringStatusService;
    }

    @Override
    protected Task<MonitoringStatus> createTask() {
        return new Task<>() {
            @Override
            protected MonitoringStatus call() {
                updateMessage("Requesting monitoring status");
                return monitoringStatusService
                    .getMonitoringStatus();
            }
        };
    }
}
```

Joonis 10. Taaskäivitav toiming, mis loob automaatselt endale lõime.

```
@Component
public class Controller {

	@FXML
	private Label statusLabel;

	private final MonitrongStatusClient
	monitoringStatusClient;

	public Controller(MonitoringStatusClient
	monitoringStatusClient) {
		this.monitoringStatusClient =
		monitoringStatusClient;
	}

	private void initMonitoringStatusClient() {
		monitoringStatusClient.setOnSucceeded(event -> {
			statusLabel.setText(„Success“);
		}
		monitoringStatusClient.setOnFailed(event -> {
			statusLabel.setText(„Failed“);
		}
		monitoringStatusClient.start();
	}

	@FXML
	public void initialize() {
		initMonitoringStatusClient()
	}
}
```

Joonis 11. Joonis 10 kujutatud toimingu kasutamine kontrolleris.

3.6.3 Spring

Spring kataloogis ühendatakse JavaFX kasutajaliides Spring raamistikuga. Tulemusena käivitub JavaFX Spring raamistiku osana. Antud lähenemisviis võimaldab kasutada kasutajaliidese komponentides Spring funktsionaalsust nagu sõltuvuste süstimine ja konfiguratsiooni failist väärtuste lugemine.

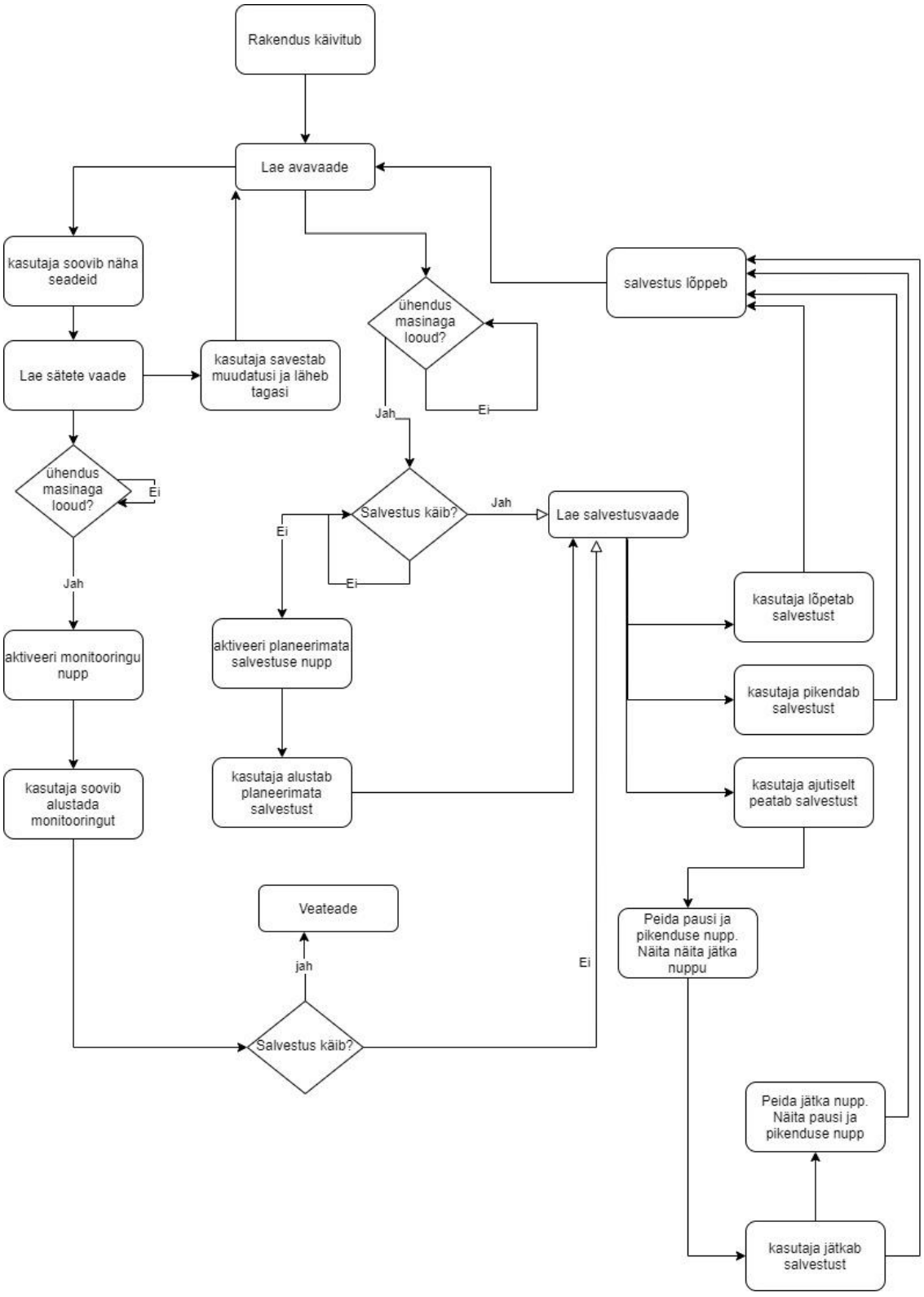
3.6.4 Kasutajaliidese tegevuste skeem

Rakenduse käivitamisel *Main* klassist laetakse Spring rakendus, mille sees on omakorda JavaFX kasutajaliides. Peale edukat sõltuvuste seadistamist laetakse kasutajaliidese avavaate FXML fail ja vastava vaate kontroller *StageInitializer* klassis. Määratakse programmi akna nimi, minimaalne suurus ning täisaknarežiim.

Vaate kontrolleris kutsutakse välja *initialize()* meetod, kus käivitatakse tagarakenduselt päringuid küsivaid taustalõimesid. Küsitakse salvestuse plaani, järgmist salvestust ning Echo360 riistvara olekut. Eduka ühenduse loomisel aktiveerub planeerimata salvestuse käivitamise võimalus, mille käivitamisel suunatakse kasutaja salvestuse vaatele samal ajal kui tagataustal päritakse ekraani pildid ja helitugevuse andmed.

Määratud ajavahemiku tagant küsib EchoMonitor riistvara oleku kohta uuendust. Juhul, kui salvestusriisvara olek muutus aktiivseks toimub salvestuse vaate ehitamine, mille käigus antakse masina olekut küsiva lõime kontroll üle. Kasutaja võib salvestussessiooni peatada, lõpetada, pikendada ja jätkata. Enneaegsel salvestuse lõpetamisel toimub ümberlülitus tagasi avavaatele samal ajal, kui tagataustal toimub salvestuse lõpetamise protsess. Vaikimisi tehakse automaatne ümberlülitus avavaatele, kui salvestusriistvara olek muutub vabaks.

Avavaatelt on võimalik liikuda sätete vaatele. Sätetes on võimalik käivitada jälgimisfunktsioon sisendseadmete seadistamiseks. Antud tegevus annab veateate juhul, kui samal ajal toimub juba aktiivne salvestus.



Joonis 12. Kasutajaliidese tegevuste skeem.

3.7 Kasutajaliidese välimus

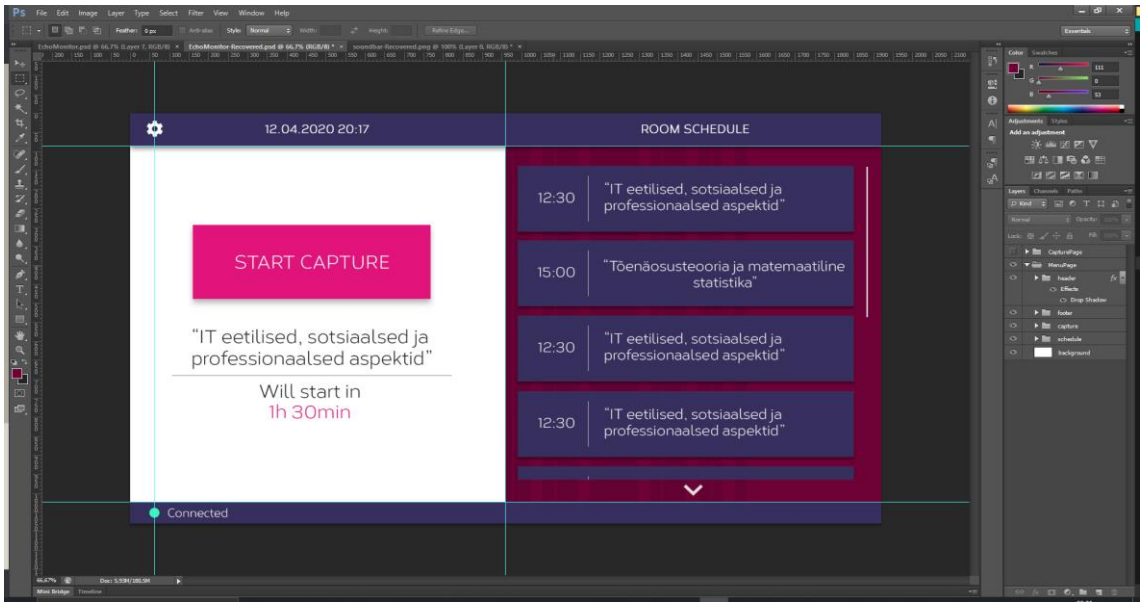
Kasutajaliidese välimuse aluseks võeti olemasoleva EchoMonitori rakenduse kasutajaliides. Arvestades sellega, et EchoMonitor on loodud salvestuse juhtimise lihtsustamiseks on oluline roll rakenduse kasutatavusel. Praeguse lahenduse raames käivitatakse rakendus õppejõu laual paigutatud sülearvutil. Kuna õppejõud võib eelistada esinemisel loenguruumis ringi liikuda, peab EchoMonitori kasutajaliides olema võimalikult kaugelt nähtav. Selleks on vaja teha vaata elemente suuremaks ning samal ajal minimiseerida vaate komponentide arv kuvades ekraanile olulisemat infot. Nii puutetundlikul ekraanil kui ka tavalise hiirega orienteerudes on suuremaid nuppe kergem vajutada tänu suuremale pindalale.

Kasutajaliidese välimuse loomine toimus pilditöötlusprogrammis. Kasutades keskkonna poolt pakutud võimalusi on kasutajaliidese välimuse kujundamine kordades kiirem kui algusest peale luua välimus koodis.

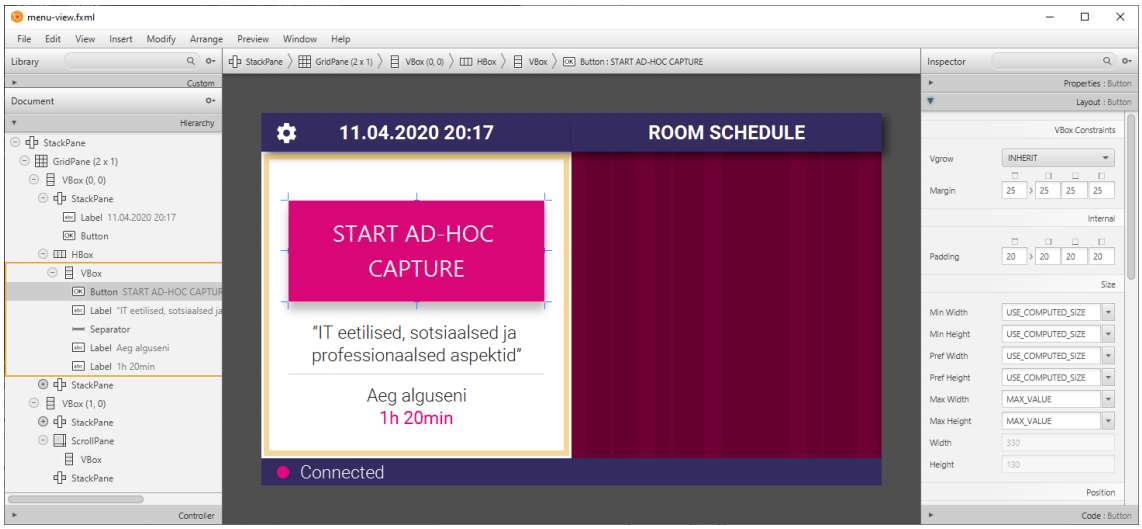
Värvi valikul võeti eeskujuks Tallinna Tehnikaülikooli süsteemide disain. Värvikoodi ja välimust kujundavad komponendid saadi kätte TalTech kodulehe CSS stiili failist.

Olemasoleva EchoMonitori kasutajaliidese analüüsi käigus tuldi järeldusele, et mikrofoni riba tuleb paigutada ekraani piltidest eemale, et säilitada sümmeetria paarituarvu monitoride korral. Kuigi video sisendite arvu muutus on vähetõenäoline, õppides olemasoleva rakenduse vigadest tasub ekraanipiltide elemendid hoida eemale teistest elementidest, kuna pildi resolutsioon võib muutuda.

Sobiva šrifti ja ikoonide leidmisel oli abiks Google Material Design ikoonide kogumik kui ka Google Fonts teenus.



Joonis 13. Photoshopis loodud kasutajaliidese avavaate välimus.



Joonis 14. Avavaade SceneBuilderis. Vaate komponendid vasakul, valitud komponendi omadused paremal.

3.8 Lahenduse testimine

Arenduse käigus lisatud funktsionaalsus kontrolliti kõigepealt lokaalselt, kasutades eelnevalt kätte saadud Echo360 sõnumite sisu. Echo360 riistvara simuleerimiseks loodi Spring raamistikul põhinev rakendus, mis tagastas Echo360 riistvara sõnumeid. Antud lähenemis viis andis kiireid tulemusi ning oli arenduseprotsessi suunava otstarbega.

Veendudes tarkvara stabiilsuses korraldati rakenduse testimine ülikooli loenguruumis. Rakendus ühendati loenguruumis asuva Echo360 salvestusriistvaraga ning jälgiti selle

käitumist erinevates olukordades. Testimise käigus avastatud puudujäägid salvestati ning analüüsiti multimeediaspetsialistiga. Korrastatud kood läbis uuesti testimis protsess, veendumaks tarkvara loogika korrektsuses.

4 Kokkuvõte

Lõputöö raames loodi rakendus, mis täidab olulisemaid salvestuste juhtimise tarkvara funktsionaalsuse nõudeid. Rakenduse välimus on kaasaegne ning sobib kokku teiste Tallinna Tehnikaülikooli rakenduste stiiliga. Loodud projekt kasutab populaarseid tehnoloogiaid. Projekti kood ja aredusprotsessi käigus tehtud otsused on dokumenteeritud, soodustades projekti edasist arengut. Kasutusel olevad tehnoloogiad olid Infotehnoloogia süsteemide arenduse õppekava raames läbitud. Autoril sai õppeprotsessi käigus omandatud teadmisi proovile panna.

Kasutatud kirjandus

[1] „HP Engage One All-in-One System Specifications,“ [Võrgumaterjal]. Available: https://support.hp.com/us-en/document/c05638650#AbT0. [Kasutatud 17.03.2020].

[2] „ThinkSmart Hub 500 for Microsoft Teams Rooms,“ [Võrgumaterjal]. Available: https://www.lenovo.com/us/en/virtual-reality-and-smart-devices/smart-office/thinksmart/Hub-500/p/11SP1TSH500. [Kasutatud 17.03.2020].

[3] „Raspberry Pi 4 Tech Specs,“ [Võrgumaterjal]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Kasutatud 17.03.2020].

[4] „Raspberry Pi PoE HAT,“ [Võrgumaterjal]. Available: https://www.raspberrypi.org/products/poe-hat/. [Kasutatud 17.03.2020].

[5] „Raspberry Pi Touch Display,“ [Võrgumaterjal]. Available: https://www.raspberrypi.org/products/raspberry-pi-touch-display/. [Kasutatud 17.03.2020].

[6] „Raspberry Pi 4 Model B,“ [Võrgumaterjal]. Available: https://raspberrypi.dk/en/product/raspberry-pi-4-model-b/?attribute_pi-4-model-b-memory=4GB+Memory. [Kasutatud 17.03.2020].

[7] M. Puusaar. [Võrgumaterjal]. Available: https://github.com/mpuusaar/EchoMonitor. [Kasutatud 17.03.2020].

[8] „Apache FLEX SDK,“ [Võrgumaterjal]. Available: https://git-wip-us.apache.org/repos/asf/flex-sdk/repo?p=flex-sdk.git;a=summary.

[9] „Adobe Flex FAQ,“ [Võrgumaterjal]. Available: https://www.adobe.com/products/flex/faq.html. [Kasutatud 17.03.2020].

[10] „Apache Flex,“ [Võrgumaterjal]. Available: http://flex.apache.org/about-history.html. [Kasutatud 17.03.2020].

[11] „Adobe AIR FAQ,“ [Võrgumaterjal]. Available: https://www.adobe.com/products/air/faq.html. [Kasutatud 17.03.2020].

[12] „The Future of Adobe AIR,“ [Võrgumaterjal]. Available: https://theblog.adobe.com/the-future-of-adobe-air/. [Kasutatud 17.03.2020].

[13] „ACTIONSCRIPT 3.0 Developer's Guide,“ [Võrgumaterjal]. Available: https://help.adobe.com/en_US/as3/dev/as3_devguide.pdf. [Kasutatud 17.03.2020].

[14] „ActionScript 3.0 overview,“ [Võrgumaterjal]. Available: https://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html. [Kasutatud 17.03.2020].

[15] „Learning ActionScript 3,“ [Võrgumaterjal]. Available: https://www.adobe.com/devnet/actionscript/learning.html. [Kasutatud 17.03.2020].

[16] „ActionScript basic for JavaScript developers,“ [Võrgumaterjal]. Available: https://help.adobe.com/en_US/air/html/dev/WS5b3ccc516d4fbf351e63e3d118666ade46-7fa6.html. [Kasutatud 17.03.2020].

[17] „Github/EchoMonitor,“ [Võrgumaterjal]. Available: https://github.com/mpuusaar/EchoMonitor/tree/master/EchoMonitor/src. [Kasutatud 03 04 2020].

[18] R. C. Martin, „Clean Code A Handbook of Agile Software Craftsmanship,“ 2008.

[19] „ActionScript 3 Fundamentals: Variables,“ [Võrgumaterjal]. Available: https://www.adobe.com/devnet/actionscript/learning/as3-fundamentals/variables.html#articlecontentAdobe_numberedheader_4. [Kasutatud 03 04 2020].

[20] „WinForms,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/core/compatibility/winforms. [Kasutatud 18.03.2020].

[21] „Mono,“ [Võrgumaterjal]. Available: https://www.mono-project.com/. [Kasutatud 18.03.2020].

[22] „Mono Gui,“ [Võrgumaterjal]. Available: https://www.mono-project.com/docs/gui/. [Kasutatud 18.03.2020].

[23] „Avalonium,“ [Võrgumaterjal]. Available: https://avaloniaui.net/. [Kasutatud 18.03.2020].

[24] „The history of C#,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history. [Kasutatud 03 04 2020].

[25] „OpenJFX,“ [Võrgumaterjal]. Available: https://openjfx.io/. [Kasutatud 18.03.2020].

[26] „Future of JavaFX,“ [Võrgumaterjal]. Available: https://blogs.oracle.com/java-platform-group/the-future-of-javafx-and-other-java-client-roadmap-updates. [Kasutatud 18.03.2020].

[27] „Swing documentation,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/tutorial/uiswing/components/index.html. [Kasutatud 18.03.2020].

[28] „Spring Framework Overview,“ [Võrgumaterjal]. Available: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html. [Kasutatud 18.03.2020].

[29] „The History of Java Technology,“ [Võrgumaterjal]. Available: https://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html. [Kasutatud 03 04 2020].

[30] „Requests Project description,“ [Võrgumaterjal]. Available: https://pypi.org/project/requests/. [Kasutatud 03 04 2020].

[31] „WxPython,“ [Võrgumaterjal]. Available: https://wiki.python.org/moin/WxPython. [Kasutatud 03 04 2020].

[32] „TIOBE Index,“ [Võrgumaterjal]. Available: https://www.tiobe.com/tiobe-index/. [Kasutatud 03 04 2020].

[33] „Kivy,“ [Võrgumaterjal]. Available: https://kivy.org/#home. [Kasutatud 03 04 2020].

[34] „Tkinter,“ [Võrgumaterjal]. Available: https://wiki.python.org/moin/TkInter. [Kasutatud 03 04 2020].

[35] Oracle, „Introduction To FXML,“ [Võrgumaterjal]. Available:

https://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html#expression_binding. [Kasutatud 29 04 2020].

[36] Echo360, „Echo360 public api Swagger documentation,“ [Võrgumaterjal]. Available: https://echo360.org.uk/api-documentation. [Kasutatud 30 04 2020].

[37] Oracle, „Oracle Java SE Support Roadmap,“ [Võrgumaterjal]. Available: https://www.oracle.com/java/technologies/java-se-support-roadmap.html. [Kasutatud 29 04 2020].

[38] Spring, „Spring Boot 2.2.6.RELEASE,“ [Võrgumaterjal]. Available: https://spring.io/blog/2020/03/26/spring-boot-2-2-6-available-now. [Kasutatud 29 04 2020].

[39] Gulon, „JavaFX,“ [Võrgumaterjal]. Available: https://gluonhq.com/products/javafx/. [Kasutatud 29 04 2020].

[40] Gradle, [Võrgumaterjal]. Available: https://gradle.org/releases/. [Kasutatud 29 04 2020].

[41] Project Lombok, „Lombok Changelog,“ [Võrgumaterjal]. Available: https://projectlombok.org/changelog. [Kasutatud 29 04 2020].

[42] Jakarta XML, „Jakarta XML Binding API,“ [Võrgumaterjal]. Available: https://mvnrepository.com/artifact/jakarta.xml.bind/jakarta.xml.bind-api. [Kasutatud 29 04 2020].

[43] Oracle, „Implementing JavaFX Best Practices,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm. [Kasutatud 29 04 2020].

[44] Oracle, „Introduction to FXML,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html#overview. [Kasutatud 29 04 2020].

[45] Oracle, „Concurrency in JavaFX,“ [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/interoperability-tutorial/concurrency.htm. [Kasutatud 29 04 2020].

[46] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[47] „Flash & The Future of Interactive Content,“ [Võrgumaterjal]. Available: https://theblog.adobe.com/adobe-flash-update/. [Kasutatud 17.03.2020].

Lisa 1 – Rakenduse lähtekood

https://bitbucket.org/ttu_robari/echomonitor

Lisa 2 – IT Kolledži multimeediaspetsialisti tagasiside

Töö eesmärgiks oli luua tarkvara, mis juhiks loengusalvestuse seadmeid Echo360 ja töötaks mitme erineva operatsioonisüsteemi peal.

Eelmine tarkvara sõltus suuresti Adobe Air ja Flashi tööst.

Echo360 keskkonna Pilvplatvormile kolimisega hakkas esialgne tarkvara tõrkuma ja tänu sellele oli vaja uut tarkvara.

Robert teostas tarkvara ning lisas sellele mitmeid võimalusi, mida eelmisel tarkvaral ei olnud.

Tarkvara on testitud IT Kolledži seadmete peal ja suhtleb seadmetega korrektselt.

Robert oma töö teostamisel on tubli olnud ja käinud mitmeid kordi IT Kolledži seadmetega testimas, et tarkvara korrektselt tööle saada.