TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kirill Juferev 185304IADB

# Funktsionaalsuse kihi arendamine ja refaktoreerimine meditsiinirakenduse näitel

Bakalaureusetöö

Juhendaja: Nadežda Furs
MBA

Tallinn 2021

# Autorideklaratsioon

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

Autor: Kirill Juferev

29.04.2021

# Annotatsioon

Tänapäeval kasutatakse tehnoloogiaid üha rohkemates valdkondades, erandiks pole ka meditsiin. Raviasutuste töö kiirendamiseks, töö täpsuse parandamiseks ja ka inimfaktori mõju vähendamiseks kasutatakse mitmesuguseid tehnoloogiaid. Meditsiinivaldkonnas ilmub üha enam lahendusi, mis säästavad väärtuslikku aega.

Käesolev töö on projekt firmale, mis tegeleb lahenduste arendamisega haiglate jaoks. Ettevõtte lahendused võimaldavad luua mitmesuguseid patsientide ja protseduuridega seotud meediafaile. Vaja on lahendust, mis annaks juurdepääsu kogu sellele infole väljaspool operatsioonisaale ja patoloogialaboreid. Käesolevas töös kirjeldatakse funktsionaalsuse kihi arendamise protsessi kõnealuse rakenduse jaoks. Töö käigus lisab autor rakendusele uued funktsioonid, teeb andmebaasis muudatusi ja parandab olemasolevad vead. Samuti kaitstakse selles töös rakendust dekompileerimise eest, rakendatakse ajutiste litsentside süsteem ja luuakse võimalus rakenduse versiooni lihtsaks värskendamiseks. Projekti arendatakse UWP platvormil C# keeles.

Rakenduse väljatöötamise tõhusaks lõpuleviimiseks jagati töö kolmeks osaks: funktsionaalne kiht, kasutajaliidese kiht ja testkiht. Tänu niisugusele lähenemisviisile oli võimalik luua toimiv ja stabiilne rakendus koos kõigi vajalike funktsioonidega.

Arenduse tulemusena saadakse rakendus, mida kasutatakse tulevikus meditsiiniasutustes. Töö käigus välja arendatud rakendus töötab koos ettevõtte teiste rakendustega, täiendades neid ja pakkudes uusi võimalusi. Töö lõpus saadetakse rakendus esimestele kasutajatele testimiseks, seejärel analüüsitakse programmi tulevaste kasutajate esimest tagasisidet.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 6 peatükki, 11 joonist, 0 tabelit

# Abstract

## Functional Layer Development and Refactoring Based On Example Of Medical Application

Nowadays, technologies are used in an increasing number of different areas, and the medical field is no exception. Various technologies are used to speed up the work of medical institutions, improve the work accuracy, and reduce the influence of the human factor. In the medical field the number of solutions that allow you to save valuable time increases from year to year.

This work is a project for a company that develops solutions for hospitals. The products of this company allow users to create and annotate various media files that are linked to patients and procedures. The company needs a solution that allows it to interact with all this information outside of operating rooms and pathology laboratories. This bachelor thesis describes the process of developing a functional layer for this application. This work is part of a group project where the author develops the functional layer of the application. In the course of this work, the author adds new features to the application, makes changes to the database, and fixes existing bugs. Also, the application is protected from decompilation, a system of temporary licenses is developed, and the foundation for a simple update method for application is created. The project is developed on the UWP platform in C#.

To effectively complete the development of the application, the work was divided into 3 parts: the functional layer, the UI layer, and the test layer. Thanks to this approach, we managed to create a working and stable application with all the required functionalities.

As a result of the development, the application that will be used in medical institutions in the future was developed. The application developed in the course of this work will work together with other applications of the company, complementing them and providing new features. At the end of this work, the application will be sent for testing to the first users,

after which the first feedback from end users of the program will be received and analyzed.

The thesis is in Estonian and contains 33 pages of text, 6 chapters, 11 figures, 0 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| .Net Native | Eelkompileerimise tehnoloogia. |
| Automaattekst | Salvestatud sõna või fraas, mida saab pildile lohistada. |
| Backend | Tarkvara ja riistvara teenuste osa. *Backend* vastutab rakenduse sisemise osa toimimise eest. |
| C# | Microsofti välja töötatud objektorienteeritud programmeerimiskeel. |
| Cue Sheet | *Cue sheet*, or *cue file* on metaandmete fail, mis kirjeldab, kuidas CD või DVD lood on paigutatud. |
| DICOM | Meditsiiniline standard piltide ja dokumentide loomiseks, hoidmiseks, edastamiseks ja visualiseerimiseks. |
| Dekompileerimine | Allika taastamine masinakoodist või pöördkonstrueerimine. |
| Drag and Drop | Kasutajaliideses elementide kasutamise viis. Inglise keelest: “tõmba ja lase lahti”. |
| Id | Identifikaator. |
| Jetbrains dotPeek | Tasuta *.NET* dekompilaator ja assemblerite brauser. |
| Konteiner | Konteiner proovikassettide hoidmiseks. |
| Kassett | Inimese kehaosa proovikassett. |
| Logid | Sündmuste kirjed kronoloogilises järjekorras, lihtsaim viis protokollimiseks. |
| MS SQL | Relatsiooniline andmebaaside haldussüsteem. |
| UWP | (*Universal Windows Platform*) platvorm *Windows 10* rakenduste loomiseks. |
| Vahemälu | Rakenduse lokaalne andmete hoidla. |
| XAML | *XML*-põhine rakenduse märgistuskeel [1]. |
| XML | *Extensible Markup Language* – ’laiendatav märgistuskeel’. |

# Sisukord

# Jooniste loetelu

# 1 Sissejuhatus

Meditsiinivaldkond on alati olnud inimeste elus väga oluline, mis omakorda tähendab, et see peab ajaga kaasas käima. Maailm areneb pidevalt ja koos sellega arenevad ka tehnoloogiad. Iga aastaga saavad inimesed inimkeha üha paremini tundma, teevad üha keerulisemaid operatsioone.

Meditsiinivaldkonna tähtsaimad aspektid on kiirus ja täpsus. Selles valdkonnas on protseduurile kulutatud aeg otsustav. Ajast sõltub mitte ainult konkreetse patsiendi elu, vaid ka patsientide arv, keda määratud aja jooksul vastu võtta saab. Seetõttu on tervishoius hakatud üha enam infotehnoloogia lahendusi kasutama.

### 1.1. Digilahenduste väljakutsed tervishoius

Digitaalse tervishoiu reform toob esile uue tarbija paradigma, mis käsitleb andmete ja informatsiooni käitlemist ja teeb tervishoiusüsteemis võimalikuks tsentraliseeritud lahenduste kasutamise, mille abil saab salvestada materjale patsiendi raviprotsessist ja teha need raviarstidele kättesaadavaks turvalisel digiplatvormil.

Tervisehoiuga seotud organisatsioonid ja riiklikud terviseinstitutsioonid (National Health Service) otsivad pidevalt lahendusi, mis aitaksid vähendada bürokraatiat ja suurendada tõhusust, mis omakorda võimaldaks arstidel rohkem tegeleda patsientidega, mitte paberite täitmisega.

Whelan (2018: 40) toonitab, et kättesaadavad digilahendused võimaldavad patsiendile õigete ravimite määramist ja tema ravile suunamist [2].

Savage (2019: 593) toob esile, et digiandmete kasutamine võib vähendada tervishoiusüsteemi kulutusi ja parandada ravi kvaliteeti, sellest võidavad nii patsient kui ka tervishoiuorganisatsioon ja kogu süsteem tervikuna [3].

Swindells (2013: 19) rõhutab digilahenduste kasutuselevõtmise ja andmete salvestamise olulisust patsientide ravis, kuna see võib aidata patsiendi diagnoosi määrata ja kiiremas korras tema seisundit hinnata [4].

Shivank (2018) tõdeb, et digitaalne tervisesüsteem koosneb tarkvara- ja riistvaratehnoloogiatest, mis toimivad terviseteenuseid osutades ja korraldades koostöös. Digitaalse tervisesüsteemi lahendustes on esindatud nii tehisintellekti toimimisloogika, kui ka interaktiivsed meetmed ja andmete kogumise mudelid [5].

Skandinaaviamaad panustavad digilahenduste töötamisse tervishoiusüsteemis märkimisväärselt, arvestades demograafilisi väljakutseid, vajadust jagada informatsiooni ja saavutada tõhusat kommunikatsiooni antud valdkonnas [6].

Bait Partner OÜ ettevõtte lahendused on otseselt suunatud tavaliselt palju aega nõudvate probleemide kiirele lahendamisele. Ettevõttel on kaks põhisuunda: lahendused patoloogialaborite ja kirurgiaruumide jaoks. Kirurgiaruumidele mõeldud lahendus annab suure hulga lisavõimalusi operatsiooniruumis, võimaldades hallata mitmesugust tehnikat ühest kohast. Kirurgiaruumide jaoks mõeldud programm hõlmab erineva valgustuse, aknaeesriiete, monitoride ja nendele väljastatava teabe ning mitmete laborite kaamerate juhtimist. Kõik need võimalused on ühes programmis ja neid teostatakse ühest arvutist. Samuti pakub see programm võimaluse operatsiooniprotsessi salvestamiseks, seejärel saab avada konkreetse patsiendi ja konkreetse uuringuga seotud lehekülje ning saada kõik protseduuriga seotud meediafailid. See muudab protsessi väga palju lihtsamaks, sest arstid ei pea kulutama aega failide sorteerimisele ja hilisemale otsimisele. Kõik ilmub patsiendi ja protseduuri valimisel. Nii saab operatsiooni ajal salvestada video toimuvast, teha pildi iga kättesaadava kaameraga või lihtsalt salvestada audiofaili, kus arst selgitab, mida ta teeb. Seejärel salvestatakse kõik need andmed andmebaasi. Seal võib neid meediafaile iga hetk näha, samuti lisada uusi, redigeerida olemasolevaid või kustutada need, mis pole vajalikud.

## 1.1 Tööprotsess

Paljudes riikides kohustab seadus operatsioone salvestama. Seda tehakse ohutuse huvides, samuti selleks, et vajaduse korral oleks alati võimalik aru saada, mis operatsiooni

ajal toimus ja mida arstid tegid. Selleks kasutatakse kirurgiale mõeldud lahendust, mis hõlmab videofailide, audiofailide ja piltide loomist.

Videofaili loomisel saab selle sektoriteks jagada ja anda sektoritele nimetused, et video järgmiste vaatamiste käigus kiiresti orienteeruda ja aru saada, mis toimub konkreetsetes fragmentides. Samuti saab videot lõigata, kui on vaja ainult teatud hetk esile tuua.

Audiofailide loomisel on samuti võimalik faili lõigata, kui tervikfaili hoidmine ei ole vajalik või on tarvis konkreetset hetke.

Pildi loomisel on mitmeid võimalusi selle redigeerimiseks. On olemas tööriistade komplekt – joon, ruut, ring ja nool –, mille abil saab luua vajaliku mustri, samuti on võimalik joonistada käsitsi. Pildile saab lisada oma teksti või mõne endistest. Pilti saab kalibreerida, et saada teada pildil olevate objektide mõõtmed.

Lisaks eeltoodule võib saata faili üldisesse meediakausta või laadida programmist arvutisse. Kõik need tegevused säilivad patsiendiga seotud eraldi meediafailide kujul või muudavad esialgset faili. Kuid on üks miinus – praegu on kõik see kättesaadav üksnes operatsiooniruumides või patoloogialaborites olevates programmides. See muudab juurdepääsu patsientide andmetele tavapärastest ruumidest keerukamaks.

Seetõttu on vaja välja töötada rakendus, mis on orienteeritud arstide kabinettides olevatele arvutitele, et oleks võimalik saada juurdepääs protseduuride käigus tehtud failidele.

## 1.2 Rakenduse roll

Uus rakendus annab juurdepääsu konkreetse patsiendi sorteeritud ja struktureeritud meediafailidele. Andmed pärinevad ühest andmebaasist ja väljastatakse vajalikus vormis.

Nii on arstidel võimalus tutvuda meediafailidega varasematest protseduuridest ja teostada nendega mitmesuguseid manipulatsioone. Uus rakendus võimaldab teha märkmeid piltidele, eraldada vajalikke fragmente, lisada piltidele teksti. See lubab meditsiinitöötajatel väga lihtsalt ja kiiresti vahetada konkreetse patsiendiga seotud informatsiooni. Lisaks sellisele stsenaariumile on võimalik uurida faile patsiendi kohta väljaspool operatsiooniruumi, salvestada helifailina kommentaare või planeerida järgmist operatsiooni.

Antud lahendus aitab säästa palju aega andmete edastamisel, säilitamisel ja salvestamisel ning vähendab ka inimfaktori mõju, kuna arst võib ekslikult vajutada vale nuppu, salvestada faili valesse kohta või anda sellele vale nime. Programm automatiseerib protsessi ja arst ei pea selle pärast muretsema, ta saab rahulikult oma tööd teha.

Pärast kõiki tegevusi on uues programmis tehtud muudatused nähtavad ka teistes programmides. Kirurgiaruumi programmis on võimalik avada patsiendi kohta muudetud andmed ja kõik muudatused jõustuvad, sest programmid kasutavad üht andmebaasi ning meediafailide ühesuguseid lugemis- ja muutmismeetodeid.

## 1.3 Probleemi püstitamine

Hetkel on firmal *UWP* platvormil kirjutatud projekt, mida hakkas arenema teine meeskond. Nad lõid programmi aluse, kuid programmi ei saa viia meditsiiniasutustesse, sest selles on palju probleeme. Programmis on palju vigu, mille tõttu see töötab ebastabiilselt ja täidab mõningaid funktsioone valesti. Programmi disain ei ole intuitiivne ega kaasaegne. Arstid peaksid kulutama palju aega programmi mõistmiseks ja õppimiseks, kuidas sellega kiiresti töötada. Programmi kood on sageli kirjutatud nii, et on raske lisada uut loogikat või muuta olemasolevat. Programmis ei ühildu ettevõtte teiste programmidega, mis muudab võimatuks informatsiooni ühesuguse kuvamise programmides. Ka puuduvad programmis paljud tähtsad funktsioonid.

Kõige selle tõttu ei saa programmi haiglas sisse viia, see ei vasta meditsiinilistele standarditele ega rahulda klientide vajadusi. Projekt tuleb lõpule viia ja muuta meditsiiniasutuste jaoks sobivaks.

## 1.4 Töö eesmärk

Selle töö eesmärk on töötada välja valmis rakendus, kus kõik funktsioonid toimivad õigesti ja stabiilselt. Töö hõlmab programmis puuduvate moodulite väljatöötamist ja olemasolevate, mis oma ülesandeid korrektselt ei täida, parandamist. Samuti tuleb kaitsta koodi kolmanda isiku lugemise eest.

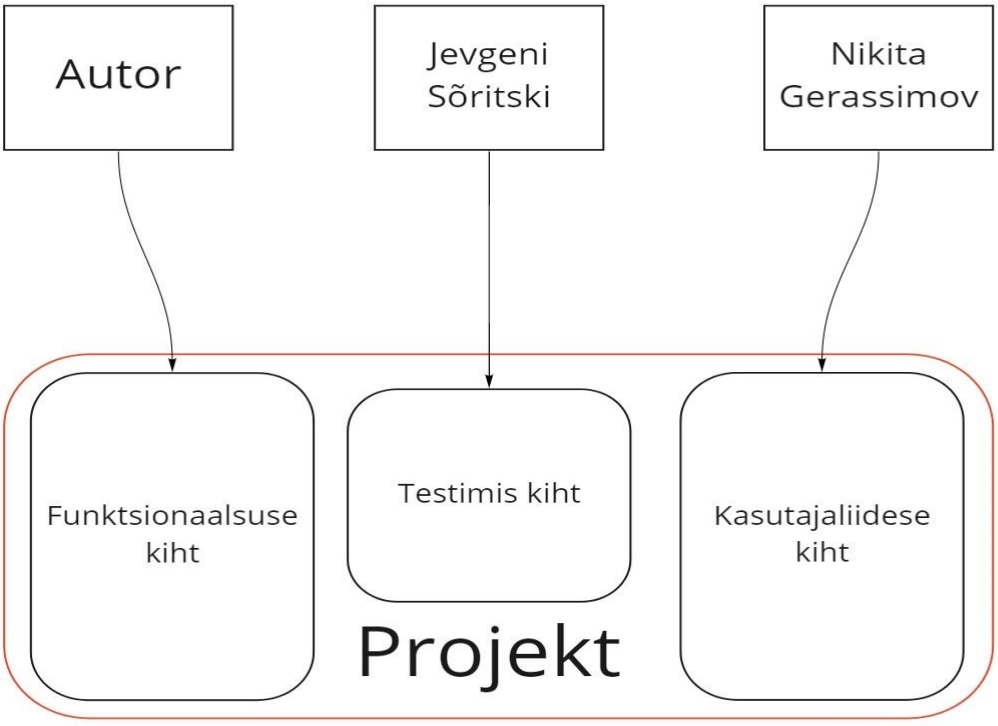
Lõpptulemuseks peab olema valmis rakendus, mis töötab stabiilselt ja turvaliselt, millel on kogu vajalik funktsionaalsus, mis on täielikus kooskõlas firma teiste programmidega, läbib automaatse testimise ja on valmis testimiseks esimeste kasutajate poolt.

Lõpus kavatseb autor koguda ja analüüsida tagasisidet esimestelt kasutajatelt, kellele programm vahetult enne haiglatesse ja teistesse meditsiiniasutustesse viimist testimiseks lähetatakse.

## 1.5 Projekti kirjeldus

See töö on osa grupiprojektist, milles osaleb kolm inimest. Töö autor töö vastutab *Backend*'i, teiste sõnadega rakenduse sisemise loogika eest. See hõlmab niisuguseid asju nagu andmete kogumine ja säilitamine andmebaasis, koodi turvalisus, andmete transportimine rakenduse sees, erinevate failide lugemine rakenduses, ühilduvus ettevõtte teiste lahendustega, rakenduse prooviversiooni arendamine, piltidele joonistamise loogika, piltide salvestamine ning muud väikesed muudatused ja uuendused.

Projektis osaleb veel kaks inimest, kes töötavad kasutajaliidesse kihi arendamise ja automaattestide kihi lisamisega. Töö projektis on jagatud järgmiselt:




Joonis 1 Tööjaotus

Töö projektis on osalejate vahel selgelt jagatud, igaüks tegeleb oma kihi arendamisega. Tänu sellele võib iga arendaja keskenduda oma tööosale ega pööra tähelepanu muudele osadele.

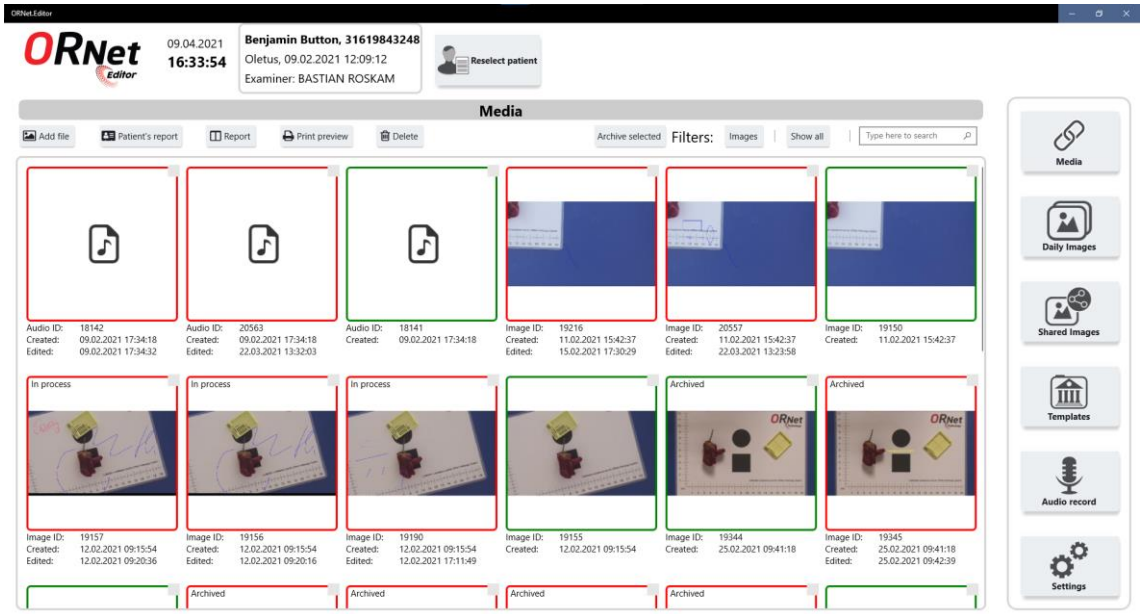
# 2 Probleemi analüüs

Selles osas autor selgitab probleemi olemust ja kirjeldab oma probleemi lahenduse visiooni. Samuti analüüsib autor olemasoleva lahenduse puudusi.

## 2.1 Olemasoleva lahenduse analüüs

Hetkel on firmal *UWP* platvormil kirjutatud rakenduse alus. Lahendusel on olemas põhifunktsionaalsus, mis teatud määral meenutab niisuguse rakenduse toimimist, mida oodatakse lõpptulemusena.

Antud hetkel näeb rakendus välja järgmine:

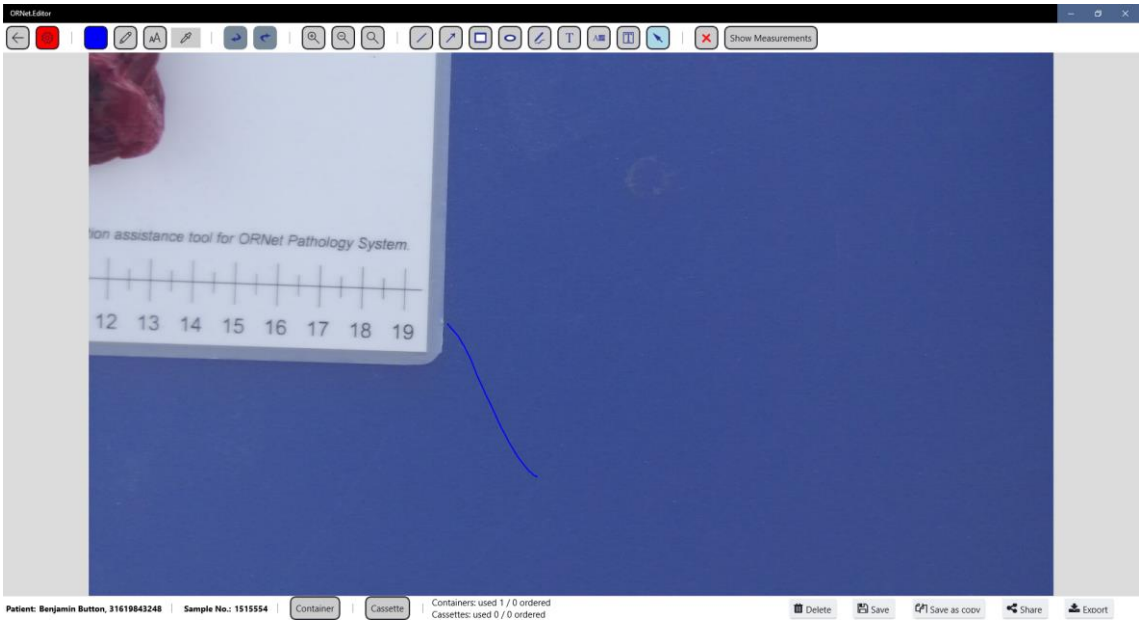



Joonis 2 Rakenduse vana kasutajaliides

Rakenduse üks olulisim ja keerulisim osa on pildiredaktor. Praegu on selle võimalused väga piiratud. Hetkel on võimalikud ainult ruut, ring, joon ja nool. Joonistada saab ka käsitsi, kuid see funktsioon ei toimi sõrmega joonistamisel. On võimalik muuta figuuride suurust ja neid liigutada. Käsitsi joonistatud joonistega ei saa midagi teha, sest seda funktsionaalsust ei ole veel olemas.

Pärast muudatuste salvestamist salvestatakse kõik joonistatud objektid faili, mille abil saab pildi uuesti avada ja näha kõiki joonistatud objekte. Teised ettevõtte programmid ei saa seda faili aga õigesti lugeda, sest puudub ühtne salvestamise ja objektide lugemise loogika.

Pildiredaktoris ei ole funktsiooni „automaattekstid“, mis võimaldaks salvestada teatud sõnu või lauseid ja seejärel neid hõlpsalt pildile kanda, et säästa aega ja mitte kirjutada sama asja mitu korda.




Joonis 3 Vana pildiredaktori kasutajaliides

Rakenduses ei ole võimalik lugeda faile laiendusega *.cue*. Neid faile kasutatakse videolõikude ja nende nimede salvestamiseks. Selline funktsionaalsus on ettevõtte teistes programmides, niisiis peab see olema ka meie rakenduses. Videoredaktoris puudub vajalik loogika, näiteks video lõikamine, teatud fragmentide määramine ja fragmentide ühendamine uue videosse.

Pildid on sageli seotud konteinerite ja kassettidega, mille nimed võivad olla erineval kujul. Praegusel hetkel saab rakendus kujutada ainult konteineri või kassetti alusnime ja ei ole võimalik nime konvertida.

Praegu programmis on rakendatud loger, kuid see ei kata kogu vajaminevat koodi. Paljudes olulistes etappides ei ole rakenduse töötsüklis mingit logimist, nii et kui esineb

vigu, siis on raskem kindlaks teha, kus need juhtusid. Samuti puudub logide jagamine ühislogile ja kasutajalogile. Kõik logid salvestatakse ainult arvutis.

Rakendus ei paku lihtsat võimalust versiooni tõstmiseks. Praegu tuleb rakendus uuendamiseks täielikult arvutist eemaldada ja pärast paigaldada uus versioon. Selline lähenemine on väga aeglane, ebapraktiline ja tekitab palju raskusi.

Antud hetkel ei ole rakendus kaitstud dekompilatsiooni eest, see võimaldab kergesti avada rakenduse lähtekoodi ja kasutada seda omakasupüüdlikel eesmärkidel.

Praegust rakenduse versiooni ei saa testimisele saata, sest see ei oma litsentse. Kasutaja võib piiranguteta kasutada rakenduse prooviversiooni, see on suur puudus ega luba saata rakendust esmasele testimisele lõppkasutajale.

Kasutajate sätteid hoitakse praegu eraldi *XML* failis.

Lisaks kõigele eeltoodule on rakenduses ka palju väiksemaid puudujääke, mis vajavad tähelepanu ja parandamist.

## 2.2 Lahenduse visioon

Lõpptulemuseks peab olema valmis rakendus, mis vastab kõikidele nõuetele ja standarditele. Rakendus peab töötama stabiilselt ja korrektselt.

Rakenduse käivitamisel peab arstil olema võimalus valida patsiendi ja temaga seotud uuringute andmed. Pärast patsiendi ja protseduuri valikut avatakse aken selle patsiendiga seotud meediafailidega. Meediafailid on pildid, audiod ja videod.

Piltide redaktoris töötavad kõik vajalikud funktsioonid. On võimalik joonistada nii hiirega kui ka sõrmedega ekraanil. On võimalik joonistada nii ettevalmistatud mustritega kui ka käsitsi. Samuti on võimalik pildile teksti lisada. Lisatud on võimalus salvestada teatud sõnad või laused ja pärast lohistada need nimekirjast pildile. Kõiki pildi objekte saab lohistada mujale, suurendada, vähendada, valida ja kustutada. Kõik muudatused salvestatakse nõuetekohaselt, seejärel on pilt täpselt samasugune ettevõtte teistes programmides.

Videoredaktorisse on lisatud võimalus fragmentide salvestamiseks. Kõik fragmendid salvestatakse videofailiga seotud *.cue* laiendusega failis. Pärast salvestamist näeb fragmente õigel kujul ka ettevõte teistes programmides. Ka videoredaktorisse lisatakse funktsioonid video lõikamiseks ja fragmentide ühendamiseks eraldi videosse, mis salvestatakse patsiendi meediafailide listi.

Logid katavad kogu olulise funktsionaalsuse. Kõik logid on jagatud kahte kategooriasse: kasutajate logid ja süsteemsed logid. Kasutajate logid salvestatakse ainult andmebaasis. Seejärel saab vajadusel saata lihtsa päringu andmebaasi ja vaadata, millega kasutaja programmis tegeles. Süsteemsed logid salvestatakse nii lokaalse arvuti faili kui ka andmebaasi. Logides ei kasutata patsientide personaalseid andmeid, patsientide nimesid, ja isikukoode ning neid ei salvestata failidesse ega andmebaasi logide tabelitesse.

Kassettide ja konteinerite nimede konverterid on rakendusse integreeritud.

Rakenduse töö kiiruse ja ligipääsu parandamiseks on kasutajate sätted viidud eraldi *XML* failist vahemälusse.

Uue versiooni paigaldamine on optimeeritud, see läheb kiiresti ega nõua töötaja füüsilist kohalolekut.

Rakendus on kaitstud dekompileerimise eest ja tarkvarakood on tugevamini kaitstud kolmandate isikute eest.

Rakendusel on versiooni kontrollimise süsteem, mis piirab programmi tegevust teatud aja möödudes. See võimaldab luua programmi testversiooni ja kaitsta seda volitamata kasutamise eest.

# 3 Lahenduse analüüs

Selles peatükis analüüsib autor nõudeid lahendusele vaatleb kasutatud tehnoloogiaid. Selles osas on kirjeldatud viis tehnoloogiad (C#, UWP, MS SQL, Cue Sheet, .NET Native).

## 3.1 Nõuded lahendusele

Lisaks toimivale funktsionaalsusele peab rakendus vastama ka mõnedele nõuetele. Need nõuded on järgmised:

Kuna projekti vundament on juba üles ehitatud UWP platvormil, peab rakenduse sellega lõpuni arendama. Tuleb arvestada platvormi kõigi funktsioonidega ja leida lahendused seal, kus platvorm piirab arendajat. Projekt peab olema rakendatud *C#* keeles.

Rakendus peab suhtlema olemasoleva andmebaasiga. Hetkel töötab andmebaas *Microsoft SQL Serveris*.

Rakendusel peab olema versioneerimise võimalus. Uue versiooni paigaldamine peab olema lihtne, kiire ja kaugjuhitav.

Rakendusel peab ühilduma ettevõtte teiste toodetega. Andmed peavad ühtviisi välja nägema kõikides ettevõtte programmides.

Uus funktsionaalsus peab olema ehitatud niimoodi, et seda oleks võimalik ja mugav testida. Samuti peab see olema kättesaadav automaattestimiseks.

Pärast arendamise lõpetamist saadetakse rakendus testimiseks esimestele kasutajatele. Seejärel on vaja koguda tagasiside ja seda analüüsida.

## 3.2 Tehnoloogiad

Selle töö käigus peab autor tutvuma tehnoloogiatega, millega ta ei ole varem kokku puutunud. Kuna projekti alus oli juba valmis, ei olnud mõnes valdkonnas võimalust valida, milliseid tehnoloogiaid kasutada.

Esiteks, arendamine toimub programmeerimiskeeles *C#*. *C#* on *C*-suguse süntaksiga keel ja objektipõhine keel programmeerimiskeel [7]. Autoril õnnestus tutvuda keelega ülikoolis õppimise ajal ning luua selle abil mitmeid projekte.

Rakendus on realiseeritud *UWP* platvormil. *Uwp (Universal Windows Platform)* on Microsoft loodud platvorm, selle abil saab luua universaalseid rakendusi, mida käivitada *Windows 10* seadmetes [8].

*Microsoft SQL Server* toimib rakenduse andmebaasina. Seda andmebaasi kasutavad kõik ettevõtte rakendused ja see kujutab endast relatsioonbaasihalduse süsteemi, mille lõi *Microsoft* [9].

Videofragmentidega seotud info hoidmiseks kasutatakse *Cue Sheet* faili. Need on metaandmetega tekstifailid, mis kirjeldavad trekkide järjestust ja kestust. Seda failivormingut toetavad paljud *CD*-de kirjutamise tarkvarad ja meediumipleierid [10]. Ettevõtte teistes rakendustes toimub videofragmentide töötlus selle tehnoloogia abil, niisiis on vaja välja töötada lahendus just seda tehnoloogiat kasutades.

Koodi dekompileerimise keerukamaks muutmiseks ja programmi töö kiirendamiseks kasutatakse projektis tehnoloogiat nimega *.NET Native*. *.NET Native* on eelkompileerimise tehnoloogia, mida kasutatakse universaalsete Windowsi rakenduste loomiseks [11]. See lahendus kompileerib tägid masinkoodiks, muutes rakenduse kiiremaks, vähendades mälu tarbimist ja raskendades ka juurdepääsu rakenduse koodile.

# 4 Lahenduse realiseerimine

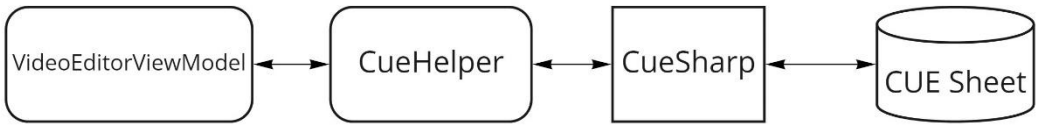
Selles peatükis on kirjeldatud rakendamise protsesse. Iga lahenduse kohta on toodud koodi näited või skeemid. Samuti on toodud andmebaasi tabelite skeemid.

## 4.1 Videoredaktor

Esialgu lubas videoredaktor videot vaadata ja teostada failiga erinevaid toiminguid, näiteks faili kustutada või videost ekraanipildi luua.

Esimene samm oli arendada tägide süsteemi. Selleks kasutas autor tasuta *CueSharp'*i teeki, mis on mugav *.cue* failide parser ja mida kasutatakse juba ettevõtte teistes rakendustes. Kuna fail on andmete hoidla, on vajalik eraldada andmetega suhtlus programmi funktsionaalsusest ja panna see eraldi klassi – *CueHelper.cs*.

Selle tulemusena sain järgmise skeemi:




Joonis 4 *Cue Sheet*'iga tööskeem

*Cue Sheet* on andmetega fail, *CueSharp* on teek, mis loeb ja kirjutab andmeid faili, *CueHelper* on autori kirjutatud klass, kuhu on kirjutatud kõik vajalikud meetodid andmetega suhtlemiseks. *VideoEditorViewModel* on antud juhul klass, mis kasutab failist andmete saamiseks *CueHelper*'it.

Järgmisena on antud näidiskood failist *CueHelper.cs*:

```
public Track[] GetAllTracks()
{
    return _cue?.Tracks;
}

public async Task AddNewTrack(Track track)
{
    if (_cue.Tracks.Length.Equals(0))
        track.TrackNumber = 0;
    else
    {
        var number = _cue.Tracks.Last().TrackNumber + 1;
        track.TrackNumber = number;
    }
    _cue.AddTrack(track);
    await UpdateFile(_cue.ToString());
}

public async Task RemoveTrack(int index)
{
    if (index > _cue.Tracks.ToArray().Length - 1 || index < 0) return;
    var list = _cue.Tracks.ToList();
    list.RemoveAt(index);
    _cue.Tracks = list.ToArray();
    await UpdateFile(_cue.ToString());
}
```

Järgmisena lisame funktsioonid video lõikamiseks ja fragmentide ühendamiseks üheks videoks. Videofailide töötlemise funktsionaalsus otsustati teisaldada eraldi klassi *MediaEditingHelper.cs*, kuhu hiljem lisatakse veel meediafailide töötlemise meetodid. See fail sisaldab meetodeid video lõikamiseks ja fragmentide ühendamiseks. Video lõikamise meetod võtab parameetritena faili, millest lõikamine toimub, uue faili, intervalli alguse ja lõpu ning ja *System.Progress* klassi elemendi, et kuvada kasutajale teavet videotöötluse protsessi protsentides. Pärast meetodi töötamise lõppu paigutatakse kogu loodud sisu faili, mis edastati läbi konstruktori, ja video muutub edasiseks manipuleerimiseks kättesaadavaks [12].

Videofragmentide ühte faili ühendamise eest vastutav meetod peab kõigepealt looma ajutised failid igast eraldi tägist ja seejärel kõik need ühte faili ühendama. Koodi dubleerimise vältimiseks saab fragmendid olemasolevate meetodite abil eraldada eraldi videofailideks. Selleks itereerin tägid läbi, luues igaühest videolõigu, seejärel ühendan saadud lõigud üheks tervikuks.

Järgnevalt on antud on video lõikamiskoodi näide:

```
public static async Task CutVideo(StorageFile inputFile, StorageFile
outputFile, double fromStart, double fromEnd, double duration,
Progress<double> progress)
{
    var clip = await MediaClip.CreateFromFileAsync(inputFile);
    var transcoder = new MediaTranscoder()
    {
        TrimStartTime = TimeSpan.FromSeconds(fromStart),
        TrimStopTime = TimeSpan.FromSeconds(duration - fromEnd)
    };
    clip.TrimTimeFromStart = TimeSpan.FromSeconds(fromStart);
    clip.TrimTimeFromEnd = TimeSpan.FromSeconds(fromEnd);
    var composition = new MediaComposition();
    composition.Clips.Add(clip);
    MediaEncodingProfile profile = MediaEncodingProfile.
             CreateMp4(VideoEncodingQuality.HD1080p);
    await composition.RenderToFileAsync(outputFile,
    MediaTrimmingPreference.Precise, profile).AsTask(progress);
    var transcode = await transcoder.PrepareFileTranscodeAsync(inputFile,
outputFile, profile);
    await transcode.TranscodeAsync().AsTask(progress);
}
```

## 4.2 Piltide redaktor

Piltide redaktoris on 3 suurt osa, mida on vaja arendada. Need osad koosnevad automaattekstidest, kujunditega toimingutest ja andmete salvestamisest. See osa vajas andmebaasi muutmist.

### 4.2.1 Automaattekstid

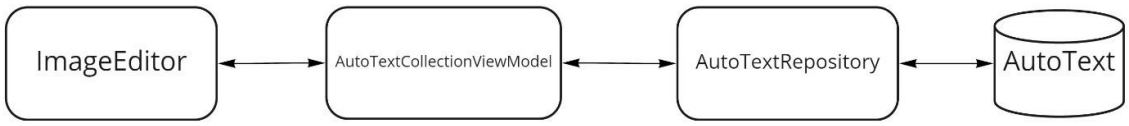
Esimene samm oli lisada tekstide salvestamise võimalus ja seejärel nende pildile lohistamine. See töö nõuab andmebaasis muudatusi, kuna puudub vajalik tabel, kuhu tekstid salvestatakse. Otsustati teha tabel nimega AutoText järgmiste väljadega:

- (int) Id
- (int) WorkstationId
- (string) Text
- (bool) Favorite

- (int) Count

Id – identifitseerimisnumber tabelis, *WorkstationId* – selle arvuti ID, millega element seotakse. Tekst – kuvatav tekst. *Favorite* – kas element on lemmikute loendisse lisatud. *Count* – kasutuste arv. Samuti on vaja lisada uus automaattekstide hoidla, mille abil edaspidi andmebaasis vajalikule tabelile viitame.

Tulemuseks saame järgmise skeemi:




Joonis 5 Automaattekstidega töötamise skeem

Automaattekstidega seotud loogika asub klassis *AutoTextViewModel*, see on automaattekstide kogu. See klass sisaldab niisuguseid meetodeid nagu näiteks uue automaatteksti lisamine, kogu sortimine ja kogust otsimine. Kokku on neli sorteerimisvõimalust: tähestikulises järjekorras, kasutussageduse järgi, lemmikute järgi ja teksti järgi otsimiseks. Lisaks ehitatakse see klass otse pildiredaktori sisse, kus automaattekstid tööle hakkavad.

Järgnevalt on toodud näited automaattekstide sorteerimise kohta:

```
private void SortByFresh()
{
    AuditLogger.LogInfo<AutoTextCollectionViewModel>($"Sort by fresh button
clicked");
    _autoTexts.Clear();
    var labels = new List<AutoTextViewModel>(_labels);
    labels.Reverse();
    foreach (var label in labels)
    {
        _autoTexts.Add(label);
    }
}

private void SortByAlphabet()
{
    AuditLogger.LogInfo<AutoTextCollectionViewModel>($"Sort by alphabet
button clicked");
    _autoTexts.Clear();
    var sortedList = new List<AutoTextViewModel>(_labels);
    sortedList.Sort((a, b) => string.Compare(a.Text, b.Text,
StringComparison.Ordinal));
    foreach (var label in sortedList)
    {
        _autoTexts.Add(label);
    }
}

private void SortByFavorites()
{
    AuditLogger.LogInfo<AutoTextCollectionViewModel>($"Sort by favorites
button clicked");
    _autoTexts.Clear();
    var sortedList = new List<AutoTextViewModel>(_labels);
    sortedList.Sort((a, b) => b.Count.CompareTo(a.Count));
    foreach (var label in sortedList.Where(label => label.Favorite))
    {
        _autoTexts.Add(label);
    }
    foreach (var label in sortedList.Where(label => !label.Favorite))
    {
        _autoTexts.Add(label);
    }
}
```

Kui kogu on seadistatud ja toimiv, saame selle lehele panna. Selleks peame lehe XAML-i koodi lisama loendi, mille allikaks on kollektsioon *AutoTextCollectionViewModel*

klassist. Automaattekstide loendi kõrval on ka nupud sorteerimiseks, uue teksti lisamiseks, samuti olemasoleva kustutamiseks või muutmiseks. Automaattekstide *Drag And Drop* rakendamiseks peame lisama meetodid, mis sündmustele reageerivad [13]. Esimene sündmus on *DragItemsStarting*. See sündmus toimub hetkel, kui kasutaja võtab hiirega loendist automaatteksti ja hakkab seda lohistama. Seejärel peame kirjutama *Drop* sündmuse meetodi. *Drop* sündmus lisab teksti pildil soovitud asukohta [14]. Lisati ka võimalus lisada pildile automaatteksti, kui sellel klõpsate. Vajutamisel ilmub tekst pildi keskele.

### 4.2.2 Kujunditega toimingud

Kuna rakendusel puudub käsitsi joonistatud jooniste liikumise ja suuruse muutmise võimalus, peame selle loogika olemasolevale koodile lisama. Joonistuse liigutamiseks on vaja kirjutada meetod, mis nihutab kõik punktid nihkevektorile.

Alljärgnevalt on antud rakendamise näide:

```
private Path MoveFreeHand()
{
    var data = (PathGeometry) ((Path) _element).Data;
    var segments = data.Figures.First().Segments.First() as PolyLineSegment;
    var vector = new Vector2((float)(_startPoint.X - data.Bounds.Left),
(float)(_startPoint.Y - data.Bounds.Y));
    var pointsCount = segments.Points.Count;
    for (var i = 0; i < pointsCount; i++)
        segments.Points[i] = new Point(segments.Points[i].X + vector.X,
segments.Points[i].Y + vector.Y);
    data.Figures.First().StartPoint = segments.Points.First();
    return (Path) _element;
}
```

Kuna vabakäejoonist ei saa joonistada kahest punktist, peame selle suuruse muutmiseks arvutama iga punkti uue positsiooni. Selleks on vaja võrrelda vana ja uue pildi piire ning arvutada neist hälbekoefitsiendid, tänu millele on võimalik arvutada iga punkti uus asukoht. Edasi antakse tsükli abil iga punkti jaoks hälbevektor. Iga vektor on kaugus vanade piiride keskpunktist korrutatult koefitsiendiga. Järgmises tsüklis kirjutatakse iga punkt üle ja seatakse uude asukohta, mis arvutatakse järgmise valemi järgi: uute piiride keskpunkt + vektor. Selle tulemusel muutub joonis õigel viisil. Veel lisatud piirangud, et vältida joonise liiga väikseks jäämist. See ei tohi olla väiksem kui 50x50 pikslit.

Alljärgnevalt on antud jooniste suuruse muutmise rakendamine:

```
private Path ResizeFreeHand()
{
    var data = (PathGeometry) ((Path) _element).Data;
    var segments = data.Figures.First().Segments.First() as PolyLineSegment;
    var newBounds = new Rect(_startPoint, _endPoint);

    var kx = newBounds.Width > 50 ? newBounds.Width / data.Bounds.Width : 1;
    var ky = newBounds.Height > 50 ? newBounds.Height / data.Bounds.Height :
1;

    var oldCenterPoint = new Point(data.Bounds.Left + data.Bounds.Width / 2,
data.Bounds.Top + data.Bounds.Height / 2);
    var newCenterPoint = new Point(newBounds.Left + newBounds.Width / 2,
newBounds.Top + newBounds.Height / 2);

    var vectors = new List<Vector2>();
    var pointsCount = segments.Points.Count;
    for (var i = 0; i < pointsCount; i++)
    {
        var x = (float) ((segments.Points[i].X - oldCenterPoint.X) * kx);
        var y = (float) ((segments.Points[i].Y - oldCenterPoint.Y) * ky);
        vectors.Add(new Vector2(x, y));
    }

    var vectorsCount = vectors.Count;
    for (var i = 0; i < vectorsCount; i++)
        segments.Points[i] =
            new Point(newCenterPoint.X + vectors[i].X, newCenterPoint.Y +
vectors[i].Y);
    data.Figures.First().StartPoint = segments.Points.First();

    return (Path) _element;
}
```

Pärast probleemi uurimist selgus, et sõrmega joonistamine ei toimi kahe vastuolulise sündmuste tõttu. Ekraanile sõrmega vajutamisel algas joonistamine ja samal ajal hakkas liikuma pilt. Probleem lahendati, lisades tingimuse, et kui valitud tööriist on käsi, siis on pildi liikumine lukustamata. Kõikide teiste tööriistade puhul on pildi liikumine blokeeritud ega häiri joonistamismeetodeid.

```
private void MyScrollViewer_PointerEntered(object sender,
PointerRoutedEventArgs e)
{
    if (_activeTool == Tools.Select)
    {
        ScrollViewer.HorizontalScrollMode = ScrollMode.Enabled;
        ScrollViewer.VerticalScrollMode = ScrollMode.Enabled;
        ScrollViewer.ZoomMode = ZoomMode.Enabled;
    }
    else
    {
        ScrollViewer.HorizontalScrollMode = ScrollMode.Disabled;
        ScrollViewer.VerticalScrollMode = ScrollMode.Disabled;
        ScrollViewer.ZoomMode = ZoomMode.Disabled;
    }
}
```

### 4.2.3 Andmete salvestamine

Pärast ühilduvuse kontrollimist selgus, et ettevõtte teistes programmides tõmmatud ring on mõnikord meie programmis joone kujul. Pärast küsimuse uurimist selgus, et see on tingitud ringi joonistamise erinevast loogikast. Teistes programmides joonistatakse ring keskpunkti ja raadiuse abil, meie programmis aga ringi alguspunkti ja lõpp-punkti abil. Seetõttu joonistati mõnikord failist kujundit lugedes vale kuju. Kuna meil on keskpunkt ja alguspunkt, leiame lõpp-punkti hõlpsasti üles. Elemendi lugemisele loogika lisamisega sain töötava parseri, mis hakkas salvestatud ringe õigesti kuvama.

## 4.3 Logimine

Kuna eri tüüpi logisid töödeldakse erineval viisil, otsustati olemasolevale logijale lisada veel üks, mis salvestab kasutaja logid. Uus loger lisatakse kohtadesse, kus kasutaja suhtleb programmi liidesega ja teostab mingisuguseid manipuleerimisi failidega või lihtsalt vaatab andmeid. Mõlemad logijad salvestavad teabe ühte tabelisse, seega loome neile ühise mudeli, mis vastab andmebaasi tabelile.

Tulemuseks saime järgmise mudeli:

- ID
- Salvestusaeg

- Kohalikus võrgus oleva arvuti nimi
- Klassi nimi, kust logi kutsuti
- Logi tekst
- Erand, kui see on olemas
- Logeri nimetus
- Windowsi kasutajanimi

Lisatud ka uus logide hoidla, mille kaudu logerid suhtlevad andmebaasiga.

Näide meetodist hoidlas:

```
public void WriteLogToDataBase(Log log)
{
    Connection.Open();
    using (SqlCommand cmd = new SqlCommand("INSERT INTO Logs (Logged, Source,
Level, Message, Callsite, Exception, Logger, Login) " +
                                           "VALUES (@Logged, @Source, @Level,
@Message, @Callsite, @Exception, @Logger, @Login)",
(SqlConnection)Connection))
    {
        cmd.Parameters.AddWithValue("Logged", log.Logged);
        cmd.Parameters.AddWithValue("Source", log.Source);
        cmd.Parameters.AddWithValue("Level", log.Level);
        cmd.Parameters.AddWithValue("Message", log.Message);
        cmd.Parameters.AddWithValue("Callsite", log.Callsite);
        cmd.Parameters.AddWithValue("Exception", log.Exception == null ?
DBNull.Value : (object)log.Exception);
        cmd.Parameters.AddWithValue("Logger", log.Logger);
        cmd.Parameters.AddWithValue("Login", log.Login);
        cmd.ExecuteNonQuery();
    }
    Connection.Close();
}
```

Pärast lisas autor logimist nendesse kohtadesse kus seda oli vaja, aga kust enne seepuudus. Logimine algab programmi käivitamisega ja toimub kogu töötsükli jooksul. Kõik logid salvestatakse edukalt andmebaasi.

Näide andmetest, mis programmi töötades andmebaasi satuvad:

Results Messages

| | Id | Logged | Source | Level | Message | Callsite | Exception | Logger | Login |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1755 | 2021-04-25 23:20:49.3933333 | TEST-MACHINE | TRACE | Filter list by All. | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 2 | 1756 | 2021-04-25 23:20:50.0033333 | TEST-MACHINE | DEBUG | Media list found for patient (ID: 12) and examination (I... | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 3 | 1757 | 2021-04-25 23:20:52.8800000 | TEST-MACHINE | INFO | Media item (ID: 156, name: 'ORNet-20210211213209.... | ORNet.Editor.Views.Media | NULL | Facade.Logging.AuditLogger | MED1 |
| 4 | 1758 | 2021-04-25 23:20:52.8900000 | TEST-MACHINE | DEBUG | Some media files were selected (count: 1). | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 5 | 1759 | 2021-04-25 23:20:52.9600000 | TEST-MACHINE | INFO | Media was double tapped: 156. | ORNet.Editor.Views.Media | NULL | Facade.Logging.AuditLogger | MED1 |
| 6 | 1760 | 2021-04-25 23:20:52.9700000 | TEST-MACHINE | INFO | Video file (ID: 156, name: 'ORNet-20210211213209.m... | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 7 | 1761 | 2021-04-25 23:20:53.4366667 | TEST-MACHINE | INFO | Patient (ID: 12), examination (name: 16) and examiner ... | ORNet.Editor.Views.MainPage | NULL | Facade.Logging.LogManager | MED1 |
| 8 | 1762 | 2021-04-25 23:20:53.4500000 | TEST-MACHINE | INFO | Window was navigated to VideoEditor. | ORNet.Editor.Views.VideoEditor | NULL | Facade.Logging.AuditLogger | MED1 |
| 9 | 1763 | 2021-04-25 23:20:53.6466667 | TEST-MACHINE | TRACE | Loading media file for VideoEditorViewModel. | ORNet.Editor.ViewModels.VideoEditorViewModel | NULL | Facade.Logging.LogManager | MED1 |
| 10 | 1764 | 2021-04-25 23:20:53.6800000 | TEST-MACHINE | INFO | Media file 'ORNet-20210211213209.mp4' was loaded ... | ORNet.Editor.ViewModels.VideoEditorViewModel | NULL | Facade.Logging.LogManager | MED1 |
| 11 | 1765 | 2021-04-25 23:20:54.4900000 | TEST-MACHINE | INFO | GetMediaList method was canceled | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 12 | 1766 | 2021-04-25 23:20:56.3500000 | TEST-MACHINE | INFO | BackWard button clicked | ORNet.Editor.Views.VideoEditor | NULL | Facade.Logging.AuditLogger | MED1 |
| 13 | 1767 | 2021-04-25 23:20:56.4300000 | TEST-MACHINE | INFO | Patient (ID: 12), examination (name: 16) and examiner ... | ORNet.Editor.Views.MainPage | NULL | Facade.Logging.LogManager | MED1 |
| 14 | 1768 | 2021-04-25 23:20:56.4433333 | TEST-MACHINE | INFO | Window was navigated to Media. | ORNet.Editor.Views.Media | NULL | Facade.Logging.AuditLogger | MED1 |
| 15 | 1769 | 2021-04-25 23:20:56.4533333 | TEST-MACHINE | INFO | Loading media for patient with id 12. | ORNet.Editor.ViewModels.MediaPageViewModel | NULL | Facade.Logging.AuditLogger | MED1 |
| 16 | 1770 | 2021-04-25 23:20:56.4766667 | TEST-MACHINE | INFO | Checking missing patient folders | ORNet.Editor.ViewModels.MediaPageViewModel | NULL | Facade.Logging.LogManager | MED1 |
| 17 | 1771 | 2021-04-25 23:20:56.5000000 | TEST-MACHINE | INFO | Filter button was clicked. | ORNet.Editor.ViewModels.MediaPageViewModel | NULL | Facade.Logging.LogManager | MED1 |
| 18 | 1772 | 2021-04-25 23:20:56.5133333 | TEST-MACHINE | TRACE | Filter list by All. | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |
| 19 | 1773 | 2021-04-25 23:20:57.0500000 | TEST-MACHINE | DEBUG | Media list found for patient (ID: 12) and examination (I... | ORNet.Editor.Views.Media | NULL | Facade.Logging.LogManager | MED1 |

Joonis 6 Logide näide andmebaasis

Isikuandmete turvalisuse tagamiseks kasutatakse logide loomisel identifikaatoreid andmebaasist, mis ei sisalda personaalset infot ja ei ütle midagi patsiendi kohta.

## 4.4 Vahemälu ja sisesätted

Rakendus kasutab sellist tööriista nagu vahemälu. See on rakenduse lokaalne konteiner, kuhu saab andmeid salvestada [15] [16]. Olles vahemälu õigesti konfigureeritud, pääseme sinna salvestatud andmetele juurde peaaegu kõikjal rakenduses. Vahemälu töötab nagu sõnastik: sinna pannakse võti ja määratakse võtmele väärtus. Pärast määramist saame väärtuse võtme järgi. Vahemälu on väga võimas tööriist, kuna võimaldab rakendust kiirendada ja lihtsustada juurdepääsu teabele. Selle põhjal otsustati, et tõhusam on kasutaja seadete salvestamine vahemällu.

Kõigepealt viidi vahemällu rakenduse keele ja teema seaded ning mõned rakenduses objektide kuvamise sätted. Selleks loodi vahemällu vajalikud võtmed ja seejärel määrati neile väärtused. Nüüd ei vaja seadete hankimine eraldi *XML*-faili lugemist ja näeb välja niimoodi:

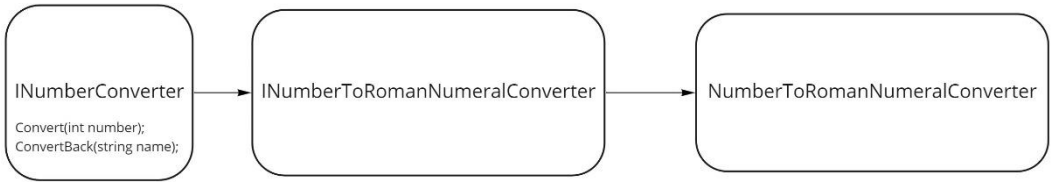
```
public static SettingsProperties ReadFromCacheSettings()
{
    var settingsProperties = new SettingsProperties();
    settingsProperties.Theme = (Enums.Theme) Cache.GetTheme();
    settingsProperties.DisplayContainers =
Cache.IsDisplayContainersEnabled();
    settingsProperties.DisplayTartu = Cache.IsTartuSettingEnabled();
    settingsProperties.Language =
string.IsNullOrEmpty(Cache.GetAppLanguage()) ? settingsProperties.Language :
Cache.GetAppLanguage();
    return settingsProperties;
}
```

Lisaks baasseadele otsustati vahemälus hoida sorteerimise ja kuvamise sätted. Seetõttu kanti vahemällu ka eelistatud meediafailide parameetrite kuvamine ja meediafailide sortimise parameetrid.

## 4.5 Kassettide ja konteinerite nimede konverterid

Väga oluline funktsioon on võimalus teisendada programmis olevate konteinerite ja kassettide nimesid. Ettevõtte põhiprogrammides saate määrata, millises vormis nimesid kuvatakse. Praeguses rakenduses puudub niisugune funktsionaalsus täielikult, seetõttu on vaja see rakendada nullist. Kuna nii konteinerid kui ka kassetid on andmebaasis salvestatud ühte tabelisse ja erinevad mõnede omaduste poolest, on võimalik valmistada universaalmuundureid, mis teisendavad nii konteinerite kui ka kassettide nimesid. Esiteks on vaja luua *INumberConverter* liides ja määratleda seal meetodid *Convert*, mis võtab parameetriks numbri, ja *ConvertBack*, mis võtab parameetrina stringi. Iga muundur realiseerib neid meetodeid omal moel. Praegu on ainult kahte tüüpi kuvatavaid nimesid: rooma numbrid ja araabia numbrid. On vaja luua üks konverter, kuna algselt salvestatakse andmed araabia numbritena. [17]

Esialgu oli koostatud hierarhia skeem, mille järgi arendas autor rooma numbritesse konverteri. Hierarhia skeem on järgmine:




Joonis 7 Konverterite ja nende liideste hierarhia

Araabia numbritest rooma numbriteks konverteri loomiseks loome liidese *INumberToRomanNumeralConverter*, millest päritakse klass *NumberToRomanNumeralConverter*. Järgmisena peame kirjutama meetodite *Convert* ja *ConvertBack* rakendamise klassis *NumberToRomanNumeralConverter*. *Convert*-meetodi juurutamine on *while* tsükkel, mis igal iteratsioonil kontrollib numbri suurust, lisab nõutava tähe lõpureale ja lahutab numbrist, mis tuli parameetrina, vajaliku numbri. Siis tsükkel kordub, kuni arvud on suuremad kui null. *ConvertBack* meetod on rakendatud rekursiooni abil. Esiteks kontrollitakse stringi õigsust, seejärel on tingimusi, mis kontrollivad, milliste tähtedega string algab. Pärast tingimuse edastamist tagastab meetod vajaliku numbri + kutsub ennast, parameeterina edastatakse string ilma esimeste märkideta, tänu millele tingimus möödus. See juhtub seni, kuni ükski tingimus ei möödu ja meetod tagastab 0. Tulemusena tagastatakse number, mis vastab algsele numbrile Rooma formaadis.

Osa *ConvertBack* meetodi rakendamisest:

```
if (romanUpperCase.StartsWith("X"))
{
    return 10 + ConvertBack(romanUpperCase.Substring(1));
}
if (romanUpperCase.StartsWith("IX"))
{
    return 9 + ConvertBack(romanUpperCase.Substring(2));
}
if (romanUpperCase.StartsWith("V"))
{
    return 5 + ConvertBack(romanUpperCase.Substring(1));
}
if (romanUpperCase.StartsWith("IV"))
{
    return 4 + ConvertBack(romanUpperCase.Substring(2));
}
if (romanUpperCase.StartsWith("I"))
{
    return 1 + ConvertBack(romanUpperCase.Substring(1));
}
return 0;
```

Edasi *ViewModel*'i loomisel määratakse sellele õiged konverterid, mis muudavad nime nõutavasse vormingusse. Samuti tänu sellele realisatsioonile on võimalik tulevikus lisada rohkem konvertereid, pärides need lihtsalt *INumberConverter* liideselt.

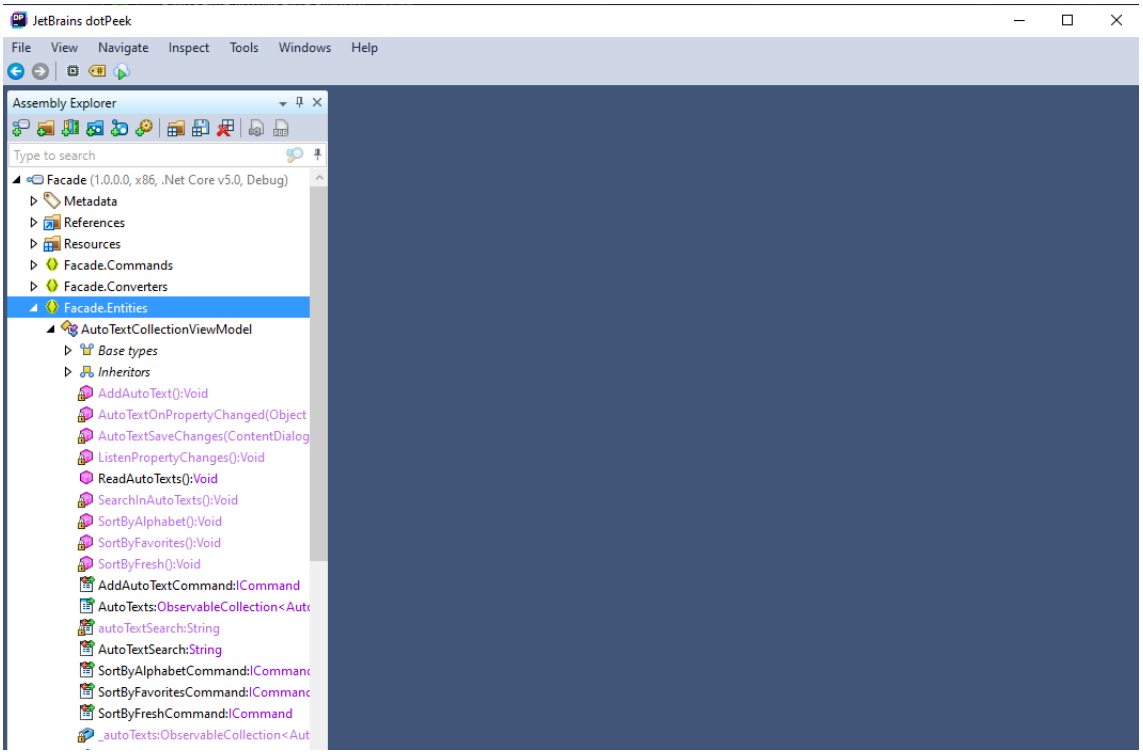
## 4.6 Turvalisus

Rakenduse turvalisus tuleb tagada kahes kohas: koodi turve ja ajutised litsentsid. Kood peab olema turvatud, et kolmandatel isikutel oleks selle hankimine ja enda huvides kasutamine raskendatud. Samal ajal on koodi täielik turvamine võimatu. Seetõttu on vaja leida piir koodikaitse ja rakenduse kiiruse vahel. Sama kehtib ajutiste litsentside kohta. Meetod ei tohiks programmi tööd aeglustada, kuid samal ajal peaks see muutma pärast määratud aja möödumist juurdepääsu tööprogrammile keerulisemaks. [18] [19]

### 4.6.1 Koodi turvalisus

Ilma ühtegi koodikaitse tehnoloogiat kasutamata saab koodi *Jetbrains dotPeek* programmi abil väga hõlpsalt dekompileerida. Programm võimaldab avada lähtekoodi ja pääseda juurde kõigile programmi klassidele. [20]

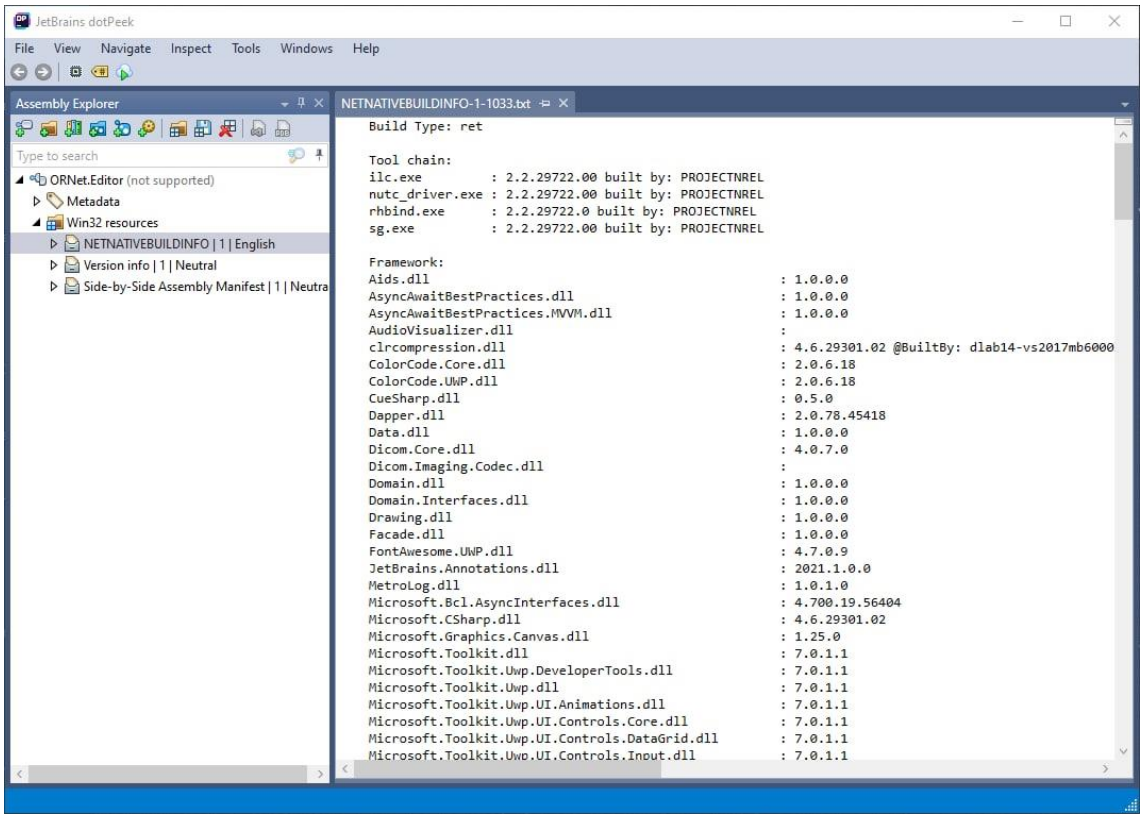
Näide programmi toimimisest:




Joonis 8 dotPeek'iga dekompileeritud kood

Koodi turvalisuse parandamiseks otsustati kasutada tehnoloogiat nimega *.NET Native*. See lahendus kompileerib programmi mitte vahekeelde, vaid otse masinkoodiks, mis samaaegselt kiirendab programmi ja raskendab oluliselt ka programmi dekompileerimist. *.NET Native*'i rakendamiseks peame selle seadistama ja kompileerimise ajal aktiveerima. Pärast seda hakkas rakendus visuaalselt kiiremini käima ja ka *dotPeek* ei saanud seda dekompileerida, näidates, et vormingut ei toetata.

*DotPeek*'i ekraanipilt pärast *.NET Native*'i rakendamist:




Joonis 9 dotPeek'i ekraanipilt pärast .NET Native'i rakendamist

### 4.6.2 Ajutised litsentsid

Kuna arvutitel, kus seda programmi kasutatakse, pole alati internetti, tuleb arendada lahendus, mis võimaldab mõne aja pärast programmi tööd piirata. Selleks, et programm oluliselt ei aeglustuks, otsustati luua lokaalsesse kausta mitu faili, kus hoitakse esimese käivitamise kuupäeva krüptitud kujul. Samuti salvestatakse programmi esimese käivitamise kuupäev vahemällu. Pärast kontrollib programm käivitamisel neid kuupäevi ja otsustab selles määratud olelusringi põhjal, kas ajutine litsents on aegunud või mitte. See meetod ei kaitse programmi täielikult, kuid takistab prooviversiooni langemist, installides programmi uuesti.

Rakendamiseks otsustati luua abstraktne klass *DateStoreFolder*, see kirjeldab failiga kausta, mis sisaldab krüptitud kuupäeva. See klass määratleb niisugused omadused nagu kausta *StorageFolder* tüüp, faili nimi, stringi tüüp ning meetodid faili initsialiseerimiseks, valideerimiseks, failist andmete lugemiseks, kausta loomiseks ja andmete faili kirjutamiseks. Seejärel luuakse otse konkreetsete kaustade klassid, mis pärinevad abstraktsest klassist ja kus määratakse failidele juurdepääsu parameetrid, samuti

rakendatakse abstraktseid meetodeid. Faili kirjutatava kuupäeva krüptimiseks valiti asümmeetriline krüptimine, kuna seda kasutatakse tavaliselt väikeste andmemahtude krüptimiseks. Krüptimiseks on lisatud klass *Ctyptography*, mis rakendab *EncryptString* ja *DecryptString* meetodeid, mis krüpteerivad stringi ja vastupidi. [21]

Seda loogikat kasutatakse programmi käivitamisel. Käivitamise ajal algväärtustab programm kaustad ja kontrollib nendes olevaid andmeid. Kui programm ei leia kaustadest ega vahemälust esimese käivitamise kuupäeva, krüpteerib see praeguse kuupäeva ja paigutab selle kaustadesse failidesse. Kui programm leiab kuupäeva, võrdleb see praegust kuupäeva esimese käivitamise kuupäevaga + määratud aeg. Kui praegune kuupäev on aegumiskuupäevast väiksem, jätkab programm töötamist. Vastasel juhul teatab programm prooviversiooni aegumisest ja soovitab pöörduda programmi pakkuja poole.

Allpool on prooviversiooni valideerimise näide:

```
public static async Task<bool> ValidateTrialVersion()
{
    var tempFolder = new DateStoreTempFolder();
    var windowsFolder = new DateStoreMicrosoftFolder();
    var virtualStoreFolder = new DateStoreVirtualStoreFolder();
    var arrayOfFolders = new DateStoreFolder[] { tempFolder, windowsFolder,
virtualStoreFolder };
    foreach (var item in arrayOfFolders)
    {
        await item.Initialize();
    }
    await CheckFilesInFoldersAsync(arrayOfFolders);
    var date = Cache.GetFirstLaunchDate().DateTime;
    if (date.Equals(DateTime.MinValue))
    {
        Cache.SetFirstLaunchDate(new DateTimeOffset(DateTime.Now));
        await CreateFoldersIfNeeded(arrayOfFolders);
        return true;
    }
    await UpdateFilesIfNeeded(arrayOfFolders,
Cache.GetFirstLaunchDate().DateTime);
    date = Cache.GetFirstLaunchDate().DateTime;
    var expirationDate =
date.AddMonths(trialVersionInMonths).AddDays(trialVersionInDays);
    return DateTime.Now < expirationDate;
}
```

## 4.7 Uued versioonid

Tulevikus on kavas lisada rakendusse uusi funktsioone, samuti parandada vead, kui neid on. Seetõttu on lahenduses vaja versiooni värskendamise võimalust. Samuti ei tohiks värskendamine toimuda ilma sellest kasutajale teatamata. Rakenduse värskendamiseks otsustati kasutada *UWP*-sse sisseehitatud mehhanismi, mis värskendab rakenduse versiooni uue programmiga paketi abil. Uuendamiseks peame kompileerima paketi uue versiooniga, määrates eelmisest kõrgema versiooni. Seejärel peame rakendusse kirjutama loogika, mis skaneerib kataloogis uue versiooni olekut. Kui uus versioon on saadaval, peaks ilmuma aken, mis annab teada, et on võimalik uuemale versioonile üle minna. [22]

Uue versiooni olemasolu kontrolliv kood on järgmine:

```
public async Task<string> CheckForUpdates()
{
    var package = Package.Current;
    PackageUpdateAvailabilityResult result = await
package.CheckUpdateAvailabilityAsync();
    string checkStatus = string.Empty;
    switch (result.Availability)
    {
        case PackageUpdateAvailability.Available:
            checkStatus = "Available";
            break;
        case PackageUpdateAvailability.Required:
            checkStatus = "Required";
            break;
        case PackageUpdateAvailability.NoUpdates:
            checkStatus = "No updates";
            break;
        case PackageUpdateAvailability.Unknown:
        default:
            // Log and ignore or whatever
            break;
    }
    return checkStatus;
}
```

Edasi peab kasutaja vajutama kahte nuppu, siis hakkab programm end värskendama ja taaskäivitub. Pärast taaskäivitamist on värskendatud rakendus töövalmis.

# 5 Tulemused

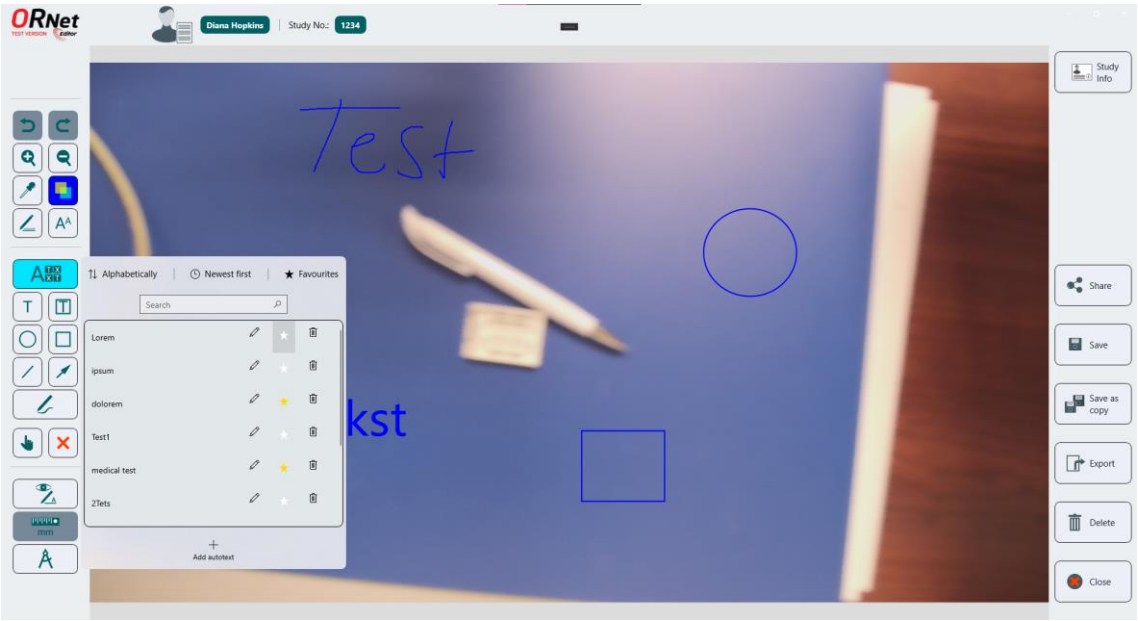
Ühiste pingutuste tulemusena valmis töötav rakendus, mis töötab kiiresti ja stabiilselt. Kõik rakenduse jaoks vajalikud funktsioonid töötavad korralikult ja neil on automaatsed testid, mille on loonud teine projekti arendamisel osalenu. Rakenduses kasutajaliides on samuti ümber kujundatud ning on nüüd välimuselt intuitiivne ja kaasaegne.

## 5.1 Loodud lahendus

Loodud rakendus vastab ootustele ja töötab ettenähtud viisil. Rakendus võimaldab juurdepääsu vajalikule infole ja annab võimaluse failidega toiminguid teha. Kõik autori tutvustatud funktsioonid läbivad nii automaatse testimise kui ka käsitsi testimise. Rakenduse logid võimaldavad jälgida rakenduse stabiilsust ja töö korrektsust.

Pärast ümberkujundamist on pildiredaktor kaasaegsem ja sellel on uued funktsioonid. Kasutajaliides on täielikult ümber tehtud ning seal, kus need ei töötanud õigesti, on meetodeid kontrollitud ja parandatud.

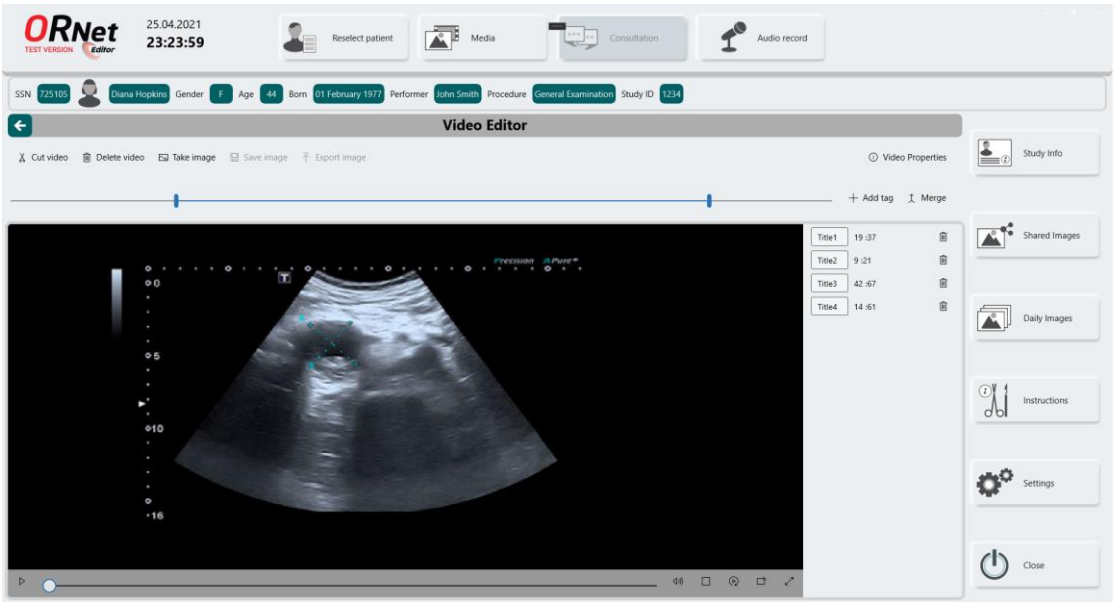
Uus pildiredaktor näeb välja selline:




Joonis 10 Uus pildiredaktor

Pildil on avatud aken automaattekstidega, mis arendati välja selle töö käigus. Videoredaktor sai samuti uue disaini ja uusi funktsioone. Videoredaktoris saab nüüd videot lõigata, valida videofragmente ja valitud fragmendid eraldi videoks liita. Kõik ülaltoodud funktsionaalsused on testitud ja toimivad täielikult.

Uuendatud videoredaktor näeb välja selline:




Joonis 11 Uus videoredaktor

Kaitsta õnnestus ka rakenduskoodi, muutes selle kompileerimistehnoloogiat kasutades otse masinkoodiks, mis omakorda kiirendas programmi tööd. Samuti lisati rakendusse ajutiste litsentside süsteem, mis võimaldas saata rakenduse testimiseks haiglatesse, kus seda tulevikus kasutatakse. Tänu sellele oli võimalik saada esimene tagasiside rakenduse kasutamise kogemuse kohta tulevastelt klientidelt.

## 5.2 Tagasiside ja projekti tulevik

Tööandjalt saadud tagasiside näitab, et projekt teostati hästi ja kõikidele nõuetele vastavalt. Tööandjal oli võimalus kogu arenduse käigus korrektuure teha ja arutada moodulite arendamist. Tööandja usub, et projekti võib nimetada valmistooteks ja turule tuua. Seetõttu saadeti rakendus esimestesse haiglatesse testimiseks, mis näitab, et rakendus on valmis.

Samuti saadi kasutajatelt esimene tagasiside. Tagasiside oli enamasti positiivne, suuri vigu ei leitud. Kliendid hindasid rakenduse mugavaks ja kaasaegseks, rakenduse funktsionaalsus kiirendab tööprotsessi. Samuti öeldi, et rakendus teatas pärast ajutise litsentsi lõppemist sellest ega käivitunud enam. Esimesed kasutajad on juba koostanud nimekirja funktsioonidest, mida nad sooviksid rakenduses tulevikus näha, mis näitab nende huvi ja soovi rakendust kasutada. See tähendab, et rakendus on jõudnud staadiumisse, kus seda saab tervishoiuasutustes müüa ja kasutada.

Rakendust saab kindlasti edasi arendada. Tulevikus on kavas välja töötada konsultatsioonimoodul, mis võimaldab luua videokommunikatsiooni ettevõtte mitme rakenduse vahel, teha videokommunikatsiooni ajal konsultatsioone ekraanikuva näitamise ja piltidele annotatsioonide loomisega. Samuti sai pärast esimese tagasiside uurimist selgeks, et kasutajad sooviksid rakendusest teha päringuid *DICOM*'i ja saada sealt arhiivitud failid. Lisaks on tööandjal muid ideid, mida saab rakendusse järgmistes versioonides lisada.

# 6 Kokkuvõte

Selle töö eesmärk oli viia lõpule rakenduse väljatöötamine funktsionaalse kihi seisukohalt. Töös oli vaja parandada olemasolevad vead, töötada välja uued funktsioonid ja muuta kood turvalisemaks. Selle tulemusena on rakendus muutunud igakülgselt paremaks ja on valmistoode, mida meditsiinitöötajad tulevikus kasutavad.

Töö teostamise käigus analüüsiti tehnoloogiate rakendamist meditsiinivaldkonnas ning kasutatud tehnoloogiaid, see võimaldas probleemidele optimaalseid lahendusi leida. Töö lubas autoril tutvuda uute tehnoloogiatega, mida ta polnud varem kasutanud. Selle käigus omandati palju oskusi nii arendamisel kui ka uue info uurimisel. Töö eeldas väga sageli ettevõtte sisemiste protsesside põhjalikku mõistmist ja ka arusaamist, kuidas kõik programmi komponendid koos töötavad. Tänu sellele suutis autor teha muudatusi ja uuendusi nii, et uute moodulite arendamine tulevikus ei nõua olemasoleva koodi ümberkirjutamist.

Projektil on edasiarendamise potentsiaali, kuna autoril ja tööandjal on ideid rakenduse edasiarendamiseks tulevikus.

# Kasutatud kirjandus


[1] „XAML overview - WPF .NET | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0. [Kasutatud 28 aprill 2021].

[2] D. Whelan, „Big data analytics, digital health platforms, and precision medicine tools: moving towards individualized and personalized care practices,“ American Journal of Medical Research, 2018.

[3] L. Savage, M. Gaynor and J. Adler-Milstein, "Digital Health Data and Information Sharing: A New Frontier for Health Care Competition?," [Online]. Available: https://www.researchgate.net/publication/332530889_Digital_Health_Data_and_Information_Sharing_A_New_Frontier_for_Health_Care_Competition. [Accessed 10 aprill 2021].

[4] M. Swindells, „Why using digital data improves care,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/236068967_Why_using_digital_data_improves_care. [Kasutatud 10 aprill 2021].

[5] S. Garg, N. L. Williams, A. Ip ja A. P. Dicker, „Clinical Integration of Digital Solutions in Health Care: An Overview of the Current Landscape of Digital Technologies in Cancer Care,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/326073315_Clinical_Integration_of_Digital_Solutions_in_Health_Care_An_Overview_of_the_Current_Landscape_of_Digital_Technologies_in_Cancer_Care. [Kasutatud 10 aprill 2021].

[6] N. Innovation, "Nordic Storytelling," [Online]. Available: https://norden.diva-portal.org/smash/get/diva2:1297054/FULLTEXT01.pdf. [Accessed 28 aprill 2021].

[7] „A Tour of C# - C# Guide | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Kasutatud 28 aprill 2021].

[8] „What's a Universal Windows Platform (UWP) app? - UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide. [Kasutatud 28 aprill 2021].

[9] „SQL Server 2019 | Microsoft,“ [Võrgumaterjal]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-2019. [Kasutatud 28 aprill 2021].

[10] „Cue sheet - Hydrogenaudio Knowledgebase,“ [Võrgumaterjal]. Available: https://wiki.hydrogenaud.io/index.php?title=Cuesheet. [Kasutatud 28 aprill 2021].

[11] „Compiling Apps with .NET Native | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/framework/net-native/. [Kasutatud 28 aprill 2021].

[12] „Media compositions and editing - UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-

us/windows/uwp/audio-video-camera/media-compositions-and-editing. [Kasutatud 28 aprill 2021].

[13] „Drag and drop - UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/uwp/design/input/drag-and-drop. [Kasutatud 28 aprill 2021].

[14] „UIElement.Drop Event (Microsoft.UI.Xaml) - Windows UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/winui/api/microsoft.ui.xaml.uielement.drop?view=winui-3.0. [Kasutatud 28 aprill 2021].

[15] „ApplicationData.LocalSettings Property (Windows.Storage) - Windows UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/uwp/api/windows.storage.applicationdata.localsettings?view=winrt-19041. [Kasutatud 28 aprill 2021].

[16] „ApplicationDataContainer Class (Windows.Storage) - Windows UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/uwp/api/windows.storage.applicationdatacontainer?view=winrt-19041. [Kasutatud 28 aprill 2021].

[17] „C# и .NET | Определение интерфейсов,“ [Võrgumaterjal]. Available: https://metanit.com/sharp/tutorial/3.9.php. [Kasutatud 28 aprill 2021].

[18] B. De Sutter, J. Van den Broeck ja B. Coppens, „(Obfuscated integration of software protections,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/340007593_Obfuscated_integration_of_software_protections. [Kasutatud 28 aprill 2021].

[19] N. Patel ja R. Patel, „A WAY TO PROTECT SOFTWARE SECRETS FROM REVERSE ENGINEERING USING CODE OBFUSCATION TECHNIQUES,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/269694315_A_WAY_TO_PROTECT_SOFTWARE_SECRETS_FROM_REVERSE_ENGINEERING_USING_CODE_OBFUSCATION_TECHNIQUES. [Kasutatud 28 aprill 2021].

[20] „dotPeek: Free .NET Decompiler & Assembly Browser by JetBrains,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/decompiler/. [Kasutatud 28 aprill 2021].

[21] „Encrypting data | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/dotnet/standard/security/encrypting-data. [Kasutatud 28 aprill 2021].

[22] „Package.CheckUpdateAvailabilityAsync Method (Windows.ApplicationModel) - Windows UWP applications | Microsoft Docs,“ [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/uwp/api/windows.applicationmodel.package.checkupdateavailabilityasync?view=winrt-19041. [Kasutatud 28 aprill 2021].

[23] M. Swindells, „Why using digital data improves care,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/236068967_Why_using_digital_data_improves_care. [Kasutatud 10 aprill 2021].

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

Mina, Kirill Juferev

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Funktsionaalsuse kihi arendamine ja refaktoreerimine meditsiinirakenduse näitel", mille juhendaja on Nadežda Furs
   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.

17.05.2021

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