TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Nikita Gerassimov 185962IADB

Kasutajaliidese 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: Nikita Gerassimov

17.05.2021

Annotatsioon

Selle meeskonnaprojekti eesmärk on viia lõpule Universal Windows Platform-i põhineva meditsiinirakenduse arendamine, et seda saaks turule viia.

Arendamisprotsessi käigus analüüsis autor meditsiinivaldkonda ja selle mõju tarkvaraarendusele ning sai teada, kuidas kaasaegset ja kasutajasõbralikku kasutajaliidest luua.

Selle eesmärgi saavutamiseks töötas autor koos kahe kaasõpilasega ning autori kohustus oli arendada uued kasutajaliidese elemendid ning refaktoreerida olemasolev kood, et muuta see rakendus tulevikukindlaks.

Selle tulemusena, viis autor koos meeskonnaliikmetega edukalt kõik vajalikud arendusosad lõpuni ning rakendus anti valitud haiglatele testimiseks ja kinnitamiseks. Autor analüüsis ettevõtte ja kasutaja tagasisidet ning koostas koos meeskonnaliikmetega edasise arengu plaani.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 27 leheküljel, 5 peatükki, 19 joonist, 0 tabelit.

Abstract

UI Layer Development and Refactoring Based on Example of Medical Application

In today's world, information technology and medicine are very closely linked. Huge amounts of data flow through hospitals in different formats and through different layers of the hospital information system. Modern doctors and laboratory workers need modern and easy to use software solutions.

The aim of this team project is to complete the development of a Universal Windows Platform based medical application, so that it can be released to market.

The company for which the application is being made already has two solutions based on Windows Foundation Platform and one Universal Windows Platform application that was developed by other group of TalTech students but because of the lack of time development was not fully completed. The UI of the application was not usable and architectural changes were needed to be made.

To achieve this goal author worked together with other 2 co-students and author's responsibility was to develop new UI elements and interactions and refactor existing code to make this application future-proof.

During the development process author analysed the medical field and its impact on software development and found out how to create modern and user-friendly UI.

As a result, the author and other team members successfully completed all required parts of development and the application was released to some hospital for testing and validation. Author analysed company and user feedback and together with team members created a plan for future development.

The thesis is in Estonian and contains 27 pages of text, 5 chapters, 19 figures, 0 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface* - rakendusprogrammiliides
back end	kasutajale nähtamatu töötlev, talletav, käitlev põhiosa
code-behind	kood mis on seotud XAML vaatega
DICOM	*Digital Imaging and Communications in Medicine* - digitaalne pildistamine ja kommunikatsioon meditsiinis
drag and drop	hiirega lohistama
front end	kliendipoolne, inimkasutajat või kasutavat süsteemi tagaosaga liidestav
LIS	*Laboratory Information System* - laborite infosüsteem
MVC	*Model–View–Controller*
MVVM	*Model-View-ViewModel*
PACS	*Picture archiving and communication system* - arhiveerimise ja kommunikatsiooni süsteem
PDF	*Portable Document Format* - porditav dokumendivorming
PIS	*Patient Information System* - patsiendi infosüsteem
UI	*User interface* - kasutajaliides
UWP	*Universal Windows Platform*
UX	*User experience* - kasutajakogemus
WPF	*Windows Presentation Foundation*
XAML	*Extensible Application Markup Language* - laiendatav rakenduse märgistuskeel

Sisukord

Jooniste loetelu

1 Sissejuhatus

Kaasaegses maailmas on infotehnoloogia ja meditsiin väga tihedalt seotud. Haiglate sees liigub tohutult palju andmeid erinevates vormingutes ja haiglate infosüsteemi eri kihtides. Seega integreeritud lahendused lihtsustavad ja kiirendavad meditsiinitöötajate tööd.

Bait Partner OÜ tegeleb riistvara ja tarkvara arendamisega, mis aitavad lahendada seda probleemi.

Bait Partner on olnud turul alates 2003. aastast. Alates 2009. aastast ettevõte arendab oma tooteperet, mis on mõeldud meditsiini valdkonna AV ja protsessihalduse töövoogudeks. Ettevõtte tegevuse põhirõhk on riistvara ja tarkvara lahenduste arendusel, mis on ette nähtud kasutamiseks patoloogia laborites ja operatsioonisaalides. [1]

Selleks on ettevõttel kaks suurt rakendust haiglatele mis on kirjutatud WPF graafilise kasutajaliidese platvormil. Kuna need tooted on üsna massiivsed ja mõeldud suurtele tööjaamadele, otsustas ettevõte 2019. aastal välja töötada uue portatiivse toote meediafailide annoteerimiseks ja redigeerimiseks, mis põhineb kaasaegsel UWP platvormil.

Antud bakalaureusetöö eesmärk on olemasoleva meditsiinirakenduse kasutajaliidese lõpuni arendamine, et rakendus turule viia.

1.1 Toote tähtsus turul

Bakalaureusetöö raames tehtud eesmärgi ja meie rakenduse täielikuks mõistmiseks uurime, kuidas tüüpiline töövoog patoloogia laboris välja näeb. Anatoomilise ja makropatoloogia valdkondades dokumenteeritakse protseduure tänapäeval digitaalselt.

Konteineris proov, mida patoloog saab uuringute jaoks, esiteks registreeritakse PIS’is, kuhu samuti salvestatakse patsiendi, uuringu ja arsti andmed ning proovi pannakse konteinerisse.

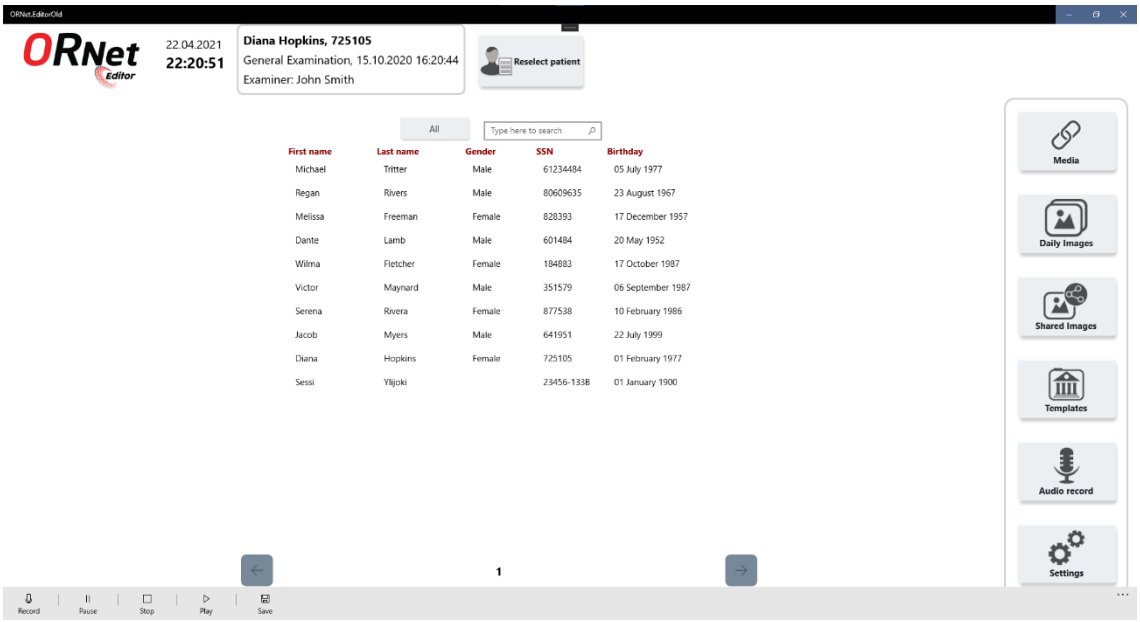
Kui prooviga konteiner laborisse sattub, registreeritakse see LIS'is. LIS integreeritakse ettevõtte toodetega API kaudu. Põhitöö tarkvaraga toimub otseselt patoloogia laboris, kus on tavaliselt paigaldatud 1–5 tööjaama, millega on ühendatud makrokaamerad. Nendes jaamades saab arst kasutada tarkvara, et luua ja vaadata patsiendiga seotud piltide-, video- ja helifaile ning lisada piltide annotatsiooni ja salvestada need keskarhiivi. Oluline on märkida, et meediafaile ei muudeta ja kõik annotatsioonid salvestatakse eraldi failina. Haiglas meediafailidega töötamisel on väga oluline, et kogu info dokumenteeritakse nii nagu ta on. [2]

Keskmiselt töödeldakse laboris 60–120 proovi, kuid kõiki neid ei saa lühikese aja jooksul diagnoosida, mõnikord on vajalik konsultatsioon või arstide nõupidamine. Järelikult, läheb meediafailidega töötamine mõnikord laborist välja.

Nende ülesannete täitmiseks arendati portatiivne UWP rakendus. See rakendus võimaldab luua ühenduse olemasoleva andmebaasiga, vaadata ja lisada meediafaile ning lisada annotatsioone väljaspool laborit, näiteks kontoris. Seejärel saab arst salvestada kõik muudatused PACS'is, mis on integreeritud DICOM-teenusega. See teenus pakib kõik patsiendi ja uuringuga seotud andmed ja failid DICOM-vormingus dokumenti. Sama teenus vastutab API kaudu teabe salvestamise eest LIS'is. [3]

1.2 Probleem

Praegusel hetkel on ettevõttel juba leitud UWP platvormi põhine lahendus, mida oli arendanud teine TalTechi üliõpilaste rühm 2019. aasta alguses. See lahendus ei sobi igapäevaseks kasutamiseks meditsiinivaldkonnas nii funktsionaalsuse, kui ka kasutajaliidese poolt, sest WPF rakenduse teisaldamise teostus jäi tudengite rühmal poolikuks liiga lühikese tähtaja tõttu. Olemasolev rakendus ei võta arvesse UWP platvormi rakenduse arendamise põhimõtteid ning ei vasta ettevõtte ootustele. Praeguse lahenduse peamine puudus on MVVM'i mustri vale rakendamine. See muster võimaldab eraldada rakenduse loogika visuaalsest esitlusest. Selle mustri õige rakendamise puudumise tõttu oli liidese loogika *code-behind*-is, mis raskendas kasutajaliidese suuremahuliste muudatuste rakendamist.



Joonis 1. Olemasolevate rakenduse avavaade

Selle lahenduse teine puudus on kiiruga loodud kasutajaliides. Kasutajaliides näeb välja aegunud ja lõpetamata ning ruumi ebaloogiline kasutamine ja elementide paigutus ainult raskendab kasutamist. Lisaks puudub värviskeemi valik, meediafailide nimekiri on korratu ning kasutajaliidese terviklikkus vajab ümbertegemist.

1.3 Lähtetingimused

Kuna see projekt on osa olemasolevate programmide perekonnast, siis autor pidi järgima teatud tingimusi, mis olid enne töö alustamist lepitud kokku Bait Partner OÜ-ga. Osa tingimustest lisandusi ja muutusid töö jooksul, sest UWP rakenduse arendus on alati seotud vastutulekukokkulepetega. Tingimuste eesmärk on luua kasutajasõbralik ja silmapaistev kasutajaliides.

Autori projekt on tehtud olemasolevate rakenduse põhjal ja seepärast:

- Rakendus peab olema tehtud UWP platvormi baasil ja toetama kõiki Windows 10 seadmeid
- Tuleks kasutada WinUI komponente
- Peab kasutama samu värvikoodi

Meditsiinivaldkonna tarkvara arendmise põhimõttest tulenevad tingimused:

- Rakendus peab vastama meditsiini-ja turvastandartidele
- Rakendusel peab olema lihtne ja turvaline paigaldus

Tingimused mis on seotud UX loogilise poolega:

- Elemendid ja tegevused peavad olema arusaadavad ilma võimalike topelttähendusteta
- Rakenduse navigatsioon peab olema kasutajale mugav ja arusaadav
- Esitleda rakenduse programmi nii heledas, kui ka tumedas värviskeemis
- Kasutaja jaoks ebaoluline teave peab olema peidetud, isikuandmeid kuvatakse ainult vajaduse korral
- Rakendus peab olema lokaliseeritud neljas keeles
- Meediafailide metaandmete kuvamine peab olema muudetav, et kasutaja saaks neid oma äranägemise järgi muuta

1.4 Panus

Selle rakenduse arendus on grupiprojekt, mis viidi ellu ettevõtteepraaktika läbimise käigus ja milles osaleb 3 inimest. Selle projekti autori ülesandeks oli rakenduse kasutajaliidese arendamine ehk *front end* pool ja koodi refaktoreerimine olemasoleva rakenduse baasil, loogilise ja mugava rakenduse navigatsiooni loomine, vaadete ja rakenduse üksikute elementide disaini muutmine ja stiilide korraldamine, MVVM'i mustri kasutusele võtmine ja rakenduse lokaliseerimine. Autor osales regulaarselt ka koosolekutel, kus arutati rakenduse funktsionaalseid ja visuaalseid aspekte.

Rakenduse loogilise ja funktsionaalse osa arendamisega ehk *back end* poolega tegeleb Kirill Juferev ja rakenduse testimisega ja automaattestide kirjutamisega tegeleb Jevgeni Sõritski. Ülesanded ja vastutus olid osalejate vahel selgelt jaotatud. Tööprotsessis otsustasime kasutada agiilset arendusmetoodikat, kuna meie plaanid ja eesmärgid polnud planeerimise etapis lõplikult paika pandud ja määratletud. Töö teostamist jälgisid alati pikaajalise töökogemusega ettevõtte töötajad, kellega sai vajadusel nõu pidada.

2 Analüüs

Enne tööle asumist oli vaja välja selgitada valmislahenduse seisund: kes on rakenduse kasutajate sihtgrupp ning millised vajadused ja ootused on meditsiinivaldkonna klientidel. Analüüsiti ka ettevõtte poolt pakutud prototüüpi.

2.1 Tehnoloogiad ja töövahendid

Rakendus kujutab endast UWP platvormi baasil tehtud töölauarakendust mis on kirjutatud C# keelega.

UWP on juhtiv ja kaasaegne platvorm Windows 10 rakenduste ja mängude jaoks. See on hästi kohandatav platvorm, mis kasutab XAML'i märgistust, et lahutada kasutajaliides (esitlus) koodist (äriloogika). UWP sobib töölauarakenduste jaoks, mis nõuavad keerukaid kasutajaliideseid, kohandamisstiile ja graafikamahukaid stsenaariume. UWP toetab automaatselt levinud sisestusmeetodeid, nagu käekiri, puutetundlikkus, mängupult, klaviatuur ja hiir.

Lisaks sellele, et on võimalik UWP'd Windows-i arvutitele töölauarakenduste loomiseks kasutada, on UWP ainus Xbox-i, HoloLens-i ja Surface Hub-i rakenduste poolt toetatud platvorm. [4]

Töö käigus kasutati ReSharperi laiendusega töökeskkonda Microsoft Visual Studio 2019. Kasutajaliidese elementide loomiseks ja muutmiseks kasutati Blend for Microsoft Visual Studio 2019.

Kuna Visual Studiosse sisseehitatud ressursihaldur pole eriti mugav, kasutati tõlgete lisamiseks ja kiiremaks ning mugavamaks kontrollimiseks Microsoft Multilingual App Toolkit tööriistakomplekti.

Koodi stiili säilitamiseks kasutati XAMLStyleri laiendust .

Tööülesannete jagamiseks, *user story* loomiseks ja *bug report*-i lisamiseks kasutati TargetProcess tarkvara.

Versioonikontrolliks ja tegevuste logimiseks kasutati Apache Subversion ja kliendina kasutati programmi TortoiseSVN.

2.2 Rakenduse sihtgrupi ja valdkonna analüüs

Rakenduse peamiseks sihtgrupiks arstid ja patoloogid kes töötavad patoloogia laboris koeproovidega ja nendega soetud meedifailidega, ning nende jaoks on mobiilne ja kiire töö meediafailidega ja näidiste dokumenteerimine ning verifitseerimine töövoo lahutamatu osa.

Pärast sihtgrupi analüüsimist sai selgeks, et meditsiinivaldkonna töötajate jaoks on väga oluline, et rakendus oleks kiire ja töökindel ning toetaks erinevaid ekraani suuruseid. Tihti tavalised ekraani suurused vahelduvad 13“ sülearvuti monitorist kontoris suure ekraaniga konverentsisaalis. Arstidele on vajalik ka erinevate sisestusmeetodite toetus, sest patoloogia laborites kasutatakse kõik-ühes lauaarvuteid puutetundliku ekraaniga.

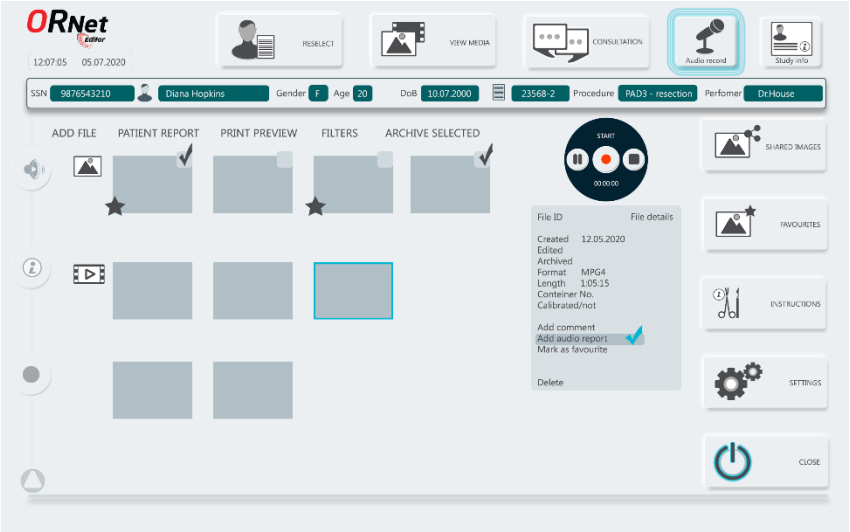
Samuti oli analüüsitud 2019. aastal läbi viidud uuring. Kasutusvigade vähendamine on meditsiiniseadmete kujundamisel ja tervise infotehnoloogiasüsteemide jaoks esmatähtis. Paljudel juhtudel on meditsiiniseadmetega seotud kasutusvead põhjustatud muude disainivigade kõrval ka kasutajaliide kujunduse või rakenduse vigadega. Tarkvara kasutajaliides, mis reguleerib seadme ja selle kasutaja vahelist suhtlemist, võib suurendada kasutusvigade tõenäosust ja kujutada endast ohtu patsiendi ohutuse jaoks, kui see ei halda nõuetekohaselt disaini ja rakendusvigu. Uuringus tuvastati, et USA meditsiinisektoris kokku 423 tarkvara tagasivõtmist oli osaliselt või täielikult põhjustatud tarkvara kasutajaliidese vigadega, mis moodustas 5,44% kõigist tagasivõtmistest ehk 46,33% kõigist tarkvaravigadega seotud tagasivõtmistest. Suurem osa (96,47%) tarkvara kasutajaliidese vigadest tingitud arvustustest olid klassist II, mis tähendab, et tagasikutsutud seadmete pärast oli suur tõenäosus vale diagnoosi panemine, kasutaja või patsiendi vigastamine. [5]

Seega on meditsiinisektori programmides informatsiooni kuvamise täpsus ning arusaadav ja reageerimisvõimeline kasutajaliides väga oluline, kuna sellest sõltub patsiendi diagnoosi täpsus ja tema järgnev ravi.

2.3 Prototüüp ettevõte poolt

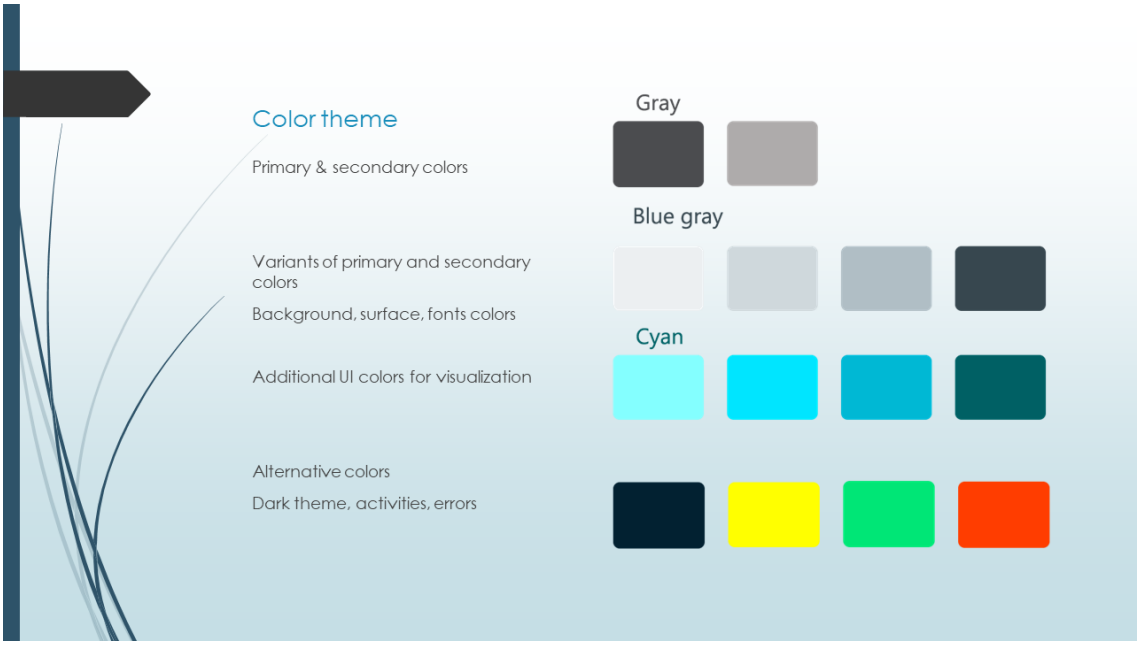
2020. aasta oktoobris valmistas ettevõte omalt poolt programmi kasutajaliidese prototüübi. Prototüübi väljatöötamisel võeti arvesse, et enamik kasutajaid on tuttavad ettevõtte muude toodetega, seetõttu peaks kasutajaliides ja kasutajakogemus olema selle

kasutajate rühma jaoks tuttav, kuid samas hõlpsasti mõistetav uute klientide jaoks. Peamiseks stiiliks oli valitud *neumprphism*.



Joonis 2. Meediaakna prototüüp

Uuringute ja klientide tagasiside põhjal koostati värvikoodi süsteem - süsteem, kus konkreetne värv määratakse ühele või mitmele terminile või omadusele.[6] Toodete vahelise ühtsuse ja järjepidevuse säilitamiseks rakendati ettevõtte põhitootes sama värvikoodi süsteemi.



Joonis 3. Rakenduse värviskeem

Kuna loodud programmis ja teiste ettevõtte toodetes on palju erinevaid elemente, töötati välja ettevõtte juhend - ettevõtte identiteedi elementide, näiteks fontide, ikoonide ja värvide, tehniline kirjeldus. Töökäigus tehtud programm võtab ettevõtte tarkvaraperes esimesena kasutusele uue korporatiivse identiteedi.

2.4 UWP disaini analüüs

Ühest kohustuslikust eeldusest oli kasutada WinUI komponente nii palju kui võimalik. WinUI on kasutajaliidese kiht, mis sisaldab kaasaegseid juhtelemente ja stiile UWP-rakenduste loomiseks. [7] Windowsi loomuliku kasutajaliidese kihina kehastab see Microsofti toodete kasutatav oma disaini süsteemi nimega Fluent Design. Informatsiooni korrastamiseks intuitiivsel ja instinktiivsel viisil kasutatakse valgust, varju, liikumist, sügavust ja tekstuuri. [8]

UWP pakub hulga ühiseid juhtelemente, mis toimivad garanteeritult kõigis Windowsi seadmetes ja on kooskõlas Fluent Design põhimõtetega. Need juhtelemendid hõlmavad kõike alates lihtsatest juhtelementidest, nagu nupud ja tekstielemendid.

Kasutades Microsofti rakenduse XAML Controls Gallery näidet, sai autor uurida UWP’sse sisseehitatud liideseelemente, samuti mõista sujuva disaini põhimõtet ja selle rakendamist nii, et toode näeks välja kaasaegselt, kuid mitte liiga keerulisena kasutamisel.

3 Rakenduse kasutajaliidese arendus

Rakenduse arendus algas 2020 septembris ja kestis aprillini 2021. Arendus jagunes kaheks etapiks: MVVM mustri kasutusele võtmine ning kasutajaliidese arendamine ja refaktoreerimine.

3.1 MVVM

MVVM-i muster võimaldab eraldada rakenduse loogika visuaalsest osast (vaatest). See muster on arhitektuuriline, mis tähendab, et see määratleb rakenduse üldise arhitektuuri.

Selle mustri tutvustas John Gossman 2005. aastal Presentation Model modifikatsioonina ja see oli algselt suunatud WPF'i rakenduste arendamisele. Kuigi nüüd on see muster läinud kaugemale WPF'ist ja seda kasutatakse mitmesugustes tehnoloogiates, sealhulgas Androidi, iOS'i arendamisel.[9]

MVVM koosneb kolmest komponendidest:

- *Model* - Mudel kirjeldab rakenduses kasutatud andmeid. Mudelid võivad sisaldada loogikat, mis on nende andmetega otseselt seotud, näiteks mudeli omaduste valideerimise loogikat.

- *View* - Vaade määratleb visuaalse liidese, mille kaudu kasutaja rakendusega suhtleb. UWP-vaates on see XAML'i kood, mis määratleb liidese nuppude, tekstikastide ja muude visuaalsete elementide kuju. Mõnikord võib koodi taga olla mõni loogika, mida on *ViewModel*-i MVVM mustri sees keeruline rakendada.

- *ViewModel* - Vaate mudel seob mudeli ja vaate andmete sidumismehhanismi kaudu. Kui mudelis atribuutide väärtused muutuvad, muutuvad mudelid INotifyPropertyChanged liidese juurutamisel vaates kuvatavad andmed automaatselt, kuigi mudel ja vaade pole otseselt seotud.

MVVM-i mustri juurutamiseks oli vaja eraldada liidese loogika kooditagusest. Töö käigus tuli märkimisväärne osa rakendusest ümber teha, sest olemasolevates rakendususes kasutati *ViewModel*-i klassi valet juurutamist. Olemasolevas lahenduses rakendati *ViewModel*-i klassi juurutamist MVC mustri ASP.NET raamistikust.

MVC-s pakub *ViewModel* kogu vaate (joonis 4) kuvatamiseks vajalikku teavet. Selles sisalduvad andmed on loodud mudelis määratletud andmete abil. Vaade loeb *ViewModel*-i ja kuvab tulemuse. Vaate sisend edastatakse kontrollerile, kes manipuleerib mudeliga, loob vastava vaate mudeli ja edastab selle vaate kuvamiseks.

```
public class ContainerViewModel
    {
        public int Id { get; set; }
        public int ParentId { get; set; }
        public int ContainerTypeId { get; set; }
        public string Name { get; set; }
        public bool IsOrdered { get; set; }
        public bool IsUsed { get; set; }
        public DateTime Created { get; set; }
        public DateTime Edited { get; set; }
    }
```

Joonis 4. *ViewModel*-i rakendamine praeguses lahenduses

MVVM'is täidab *ViewModel* sama funktsiooni nagu MVC's, kuid asendab ka osa MVC-kontrollerist, pakkudes käske, mis võimaldavad *View*-il mudeliga manipuleerida (joonis 5). UWP andmete sidumine (*binding*) haldab vaate värskendamist vastusena vaate mudeli muudatustele.

```
public class ContainerViewModel : ViewModelBase
    {
        public int Id { get; set; }
        public int ParentId { get; set; }
        public int ContainerTypeId { get; set; }
        public string Name { get; set; }
        public bool IsOrdered { get; set; }
        public bool IsUsed { get; set; }
        public DateTime Created { get; set; }
        public DateTime Edited { get; set; }
        public List<ContainerViewModel> CassettesList { get; set; }
        public List<Specimen> SpecimensList { get; set; }
        public INumberConverter ContainerConverter { get; set; }
        public INumberConverter CassetteConverter { get; set; }
        public int Index { get; set; }

        public string GetConvertedName()
        {
            switch (ContainerTypeId)
            {
                case 2:
                    if (CassetteConverter != null)
                    {
                        var canConvert = int.TryParse(Name, out var
containerName);
                        if (canConvert)
                        {
                            return CassetteConverter.Convert(containerName);
                        }
                    }
                    break;
                case 5:
                    if (ContainerConverter != null)
                    {
                        var canConvert = int.TryParse(Name, out var
containerName);
                        if (canConvert)
                        {
                            return ContainerConverter.Convert(containerName);
                        }
                    }
                    break;
            }
            return Name;
        }

        public bool SpecimenListNotEmpty => SpecimensList.Count > 0;

        public bool SpecimenAndCassetteNotEmpty => SpecimensList.Count +
CassettesList.Count > 0;
```

```
        public double EmptySpecimenOpacity => SpecimenAndCassetteNotEmpty ?
1.0 : 0.5;

        public int CassettesCount => CassettesList.Count;

        public string UsedOrderedData =>
            $"used {CassettesList.Count(x =>
x.IsUsed)}/{CassettesList.Count(x => x.IsOrdered)} ordered";
    }
```

Joonis 5. *ViewModel*-i korrektne rakendamine

Selle töö käigus autor tegid tihedat koostööd Kirill Jufereviga, kuna *ViewModel*-i muudatused mõjutasid oluliselt rakenduse funktsionaalset osa.

3.2 Navigatsioon

Rakenduses navigeerimine on kasutajakogemuse lähtepunkt ja ka see, kuidas kasutajad sisu ja funktsioone leiavad. Seetõttu oli navigatsiooni jõudluse ümberkujundamine ja parandamine üheks olulisimaks arengupunktiks.

Programmis navigeerimisloogika loomisel juhindus autor järgmistest põhimõtetest:

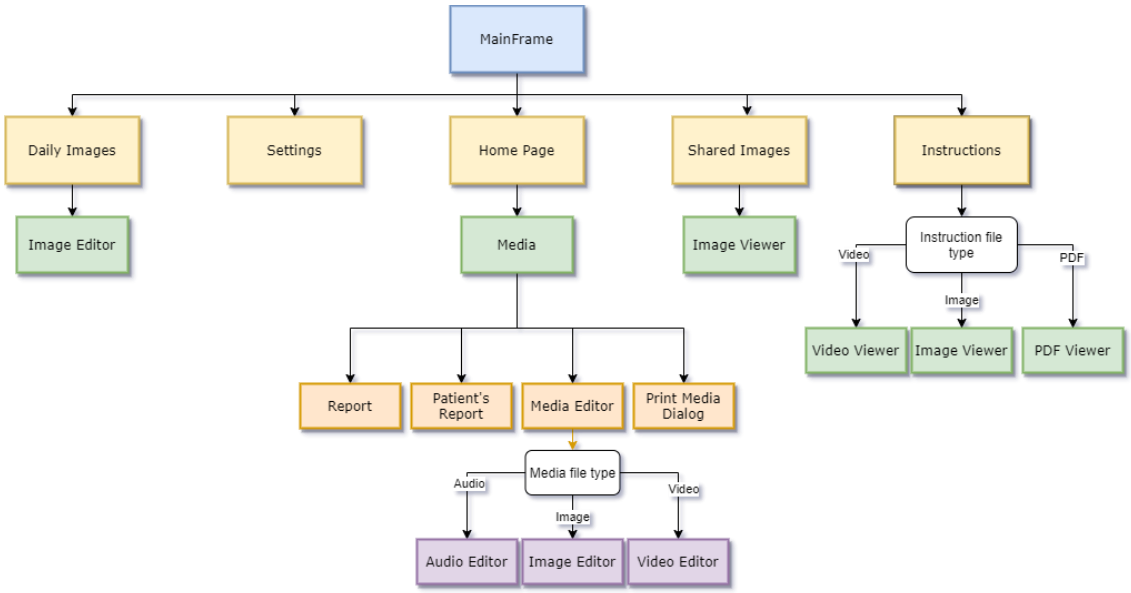
- Järjestus
- Lihtsus
- Selgus

Järjestus iseloomustab asjaolu, et navigeerimine peab vastama kasutajate ootustele. Standartsete juhtnuppude kasutamine ehk ikoonid, nende asukoht ja stiil, mis on juba tuttavad kasutajate jaoks, teevad navigatsiooni etteaimatavaks ning intuitiivseks. Kui mõtlete rakendust kui lehtede kogumit, kirjeldab navigeerimine liikumist lehe vahel ja selle sees.

Lihtsus aitab kasutajal programmi liidesega kiiresti harjuda. Vähem navigeerimiselemente muudab otsused kasutajate jaoks lihtsamaks. Olulistele sihtkohtadele lihtsa juurdepääsu tagamine ja vähem oluliste esemete peitmine aitab kasutajatel kiiremini jõuda sinna, kuhu nad tahavad.

Selgus avaldub kasutajate jaoks loogiliste navigeerimisahelate loomisel. Lehtede vahelised ilmsed ja selged navigeerimisvalikud peaksid takistama kasutajaid eksimast. [10]

Kõigi nende põhimõtete arvestamiseks loodi rakenduse navigatsiooni skeem (joonis 6).



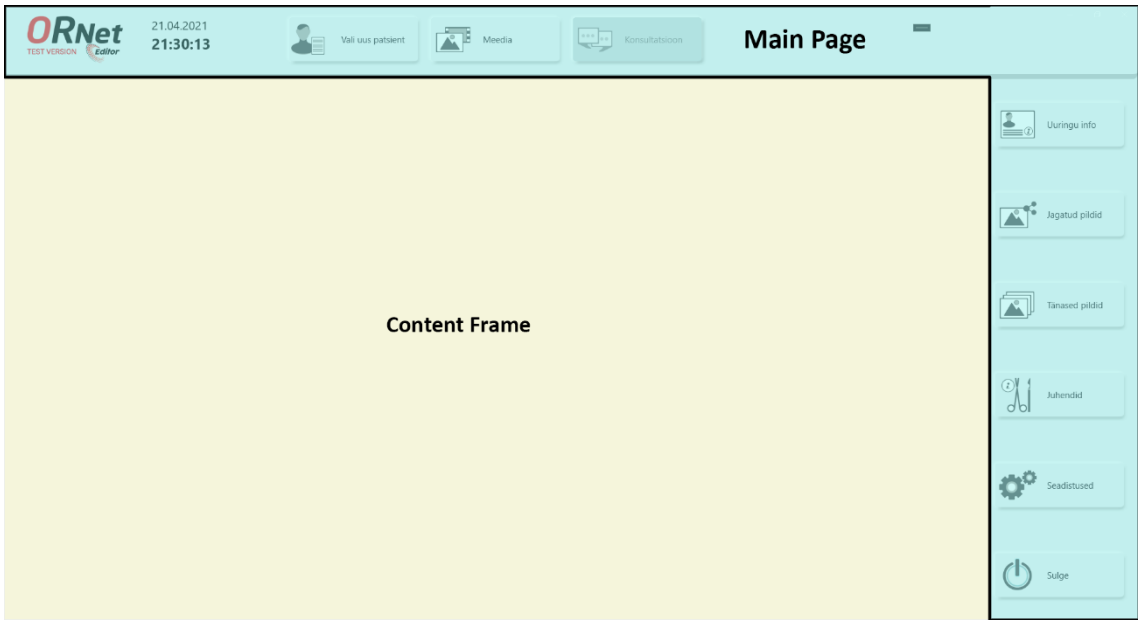
Joonis 6. Navigatsiooni skeem

Antud lahenduses autor kasutab rakenduste hierarhias navigeerimiseks ülalt alla navigeerimismeetodit. See tähendab, et kasutaja liigub edasi juurdepääsuks sisule, mis on sügavam vanemaekraanilt (see on hierarhia kõrgem tase) lapseekraanile (alumine tasand). [11]

Tagasi navigeerimine toimub siis, kui kasutaja vajutab universaalset tagasi nuppu, mis on olemas kõigis vaadetes, või kasutab Windowsi integreeritud tagasi nuppu tahvelarvuti režiimis.

UWP'l on järjepidev tagurpidi navigeerimissüsteem, et vaadata rakenduses kasutaja navigeerimisajalugu kronoloogilises järjekorras. Selleks kõigepealt luuakse rakenduse jaoks *rootFrame*-i rakenduses *App.OnLaunched* helistatav koodifaili *App.xaml* meetod. Raamiklass toetab erinevaid navigeerimismeetodeid, näiteks *Navigate*, *GoBack* ja *GoForward*. Selles raamis sisu kuvamiseks kasutatakse *Navigate* meetodit. Vaikimisi laadib see meetod *MainPage.xaml* vaadet. Lõpuks, kui leht on raamidesse laaditud, lisatakse leht *PageStackEntry* objektina *BackStack* või *ForwardStack*, mis meie puhul võimaldab navigeerimist tagasi.

Loogilise ja prognoositava liikumise jaoks lehtede kronoloogilises järjekorras keelati võimalus kaks korda samale lehele minna. Näiteks ei saa te „Juhendid“ lehelt „Juhendid“ lehele liikuda, muidu tekiks nõiaring, mis viib kasutaja segadusse.



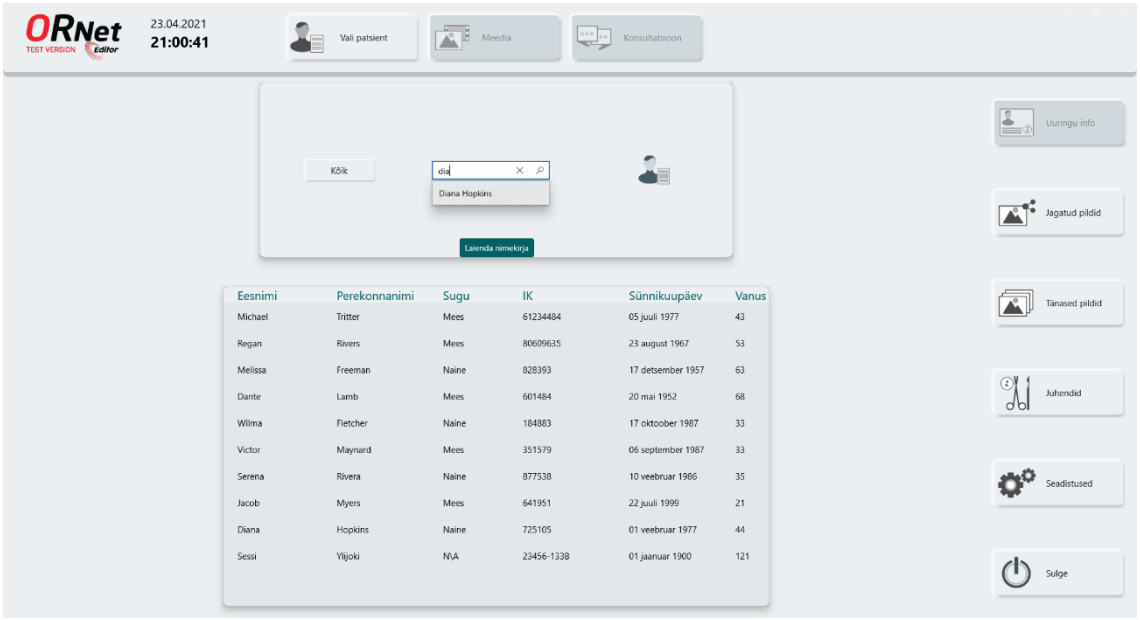
Joonis 7. Vaate ehitus

Joonisel 7 on võimalik näha, et iga leheraami sisu keskpunkt on *Main Page*, see leht vastutab rakenduse lehtede kaudu navigeerimise eest. Mõnel juhul, kui on vaja anda kasutajale rohkem tööruumi, asendatakse *Content Frame* (koodis *rootFrame*) vajaliku lehega. Näiteks juhtub see siis, kui avate pildiredaktori. Sellisel juhul peab kasutaja keskenduma pildiga töötamisele, mis laiendab sisu kogu ekraani täitmiseks ja peidab kõik muud kasutajaliidese elemendid.

3.3 Patsiendi ja uuringu valimine

Programmiga töötamine algab patsiendi valimisega. Esialgu oli kavas kuvada kõigi patsientide nimekiri, kuid autor tegi ettepaneku sellest ideest loobuda, kuna loetelu oleks paljude patsientide jaoks liiga pikk ja ebamugav. Enamikul arstidel on mugav otsida patsiente nende andmete järgi otsinguvälja kaudu. Sellel põhjusel tegi autor avalehe keskseks osaks otsingukasti ja nupu patsiendiloendi filtreerimiseks (joonis 8).

Lehel olev otsingumootor on väga paindlik ja patsienti on võimalik leida andmebaasist isikukoodi, ees- ja perekonnanime ning uuringukoodi järgi.



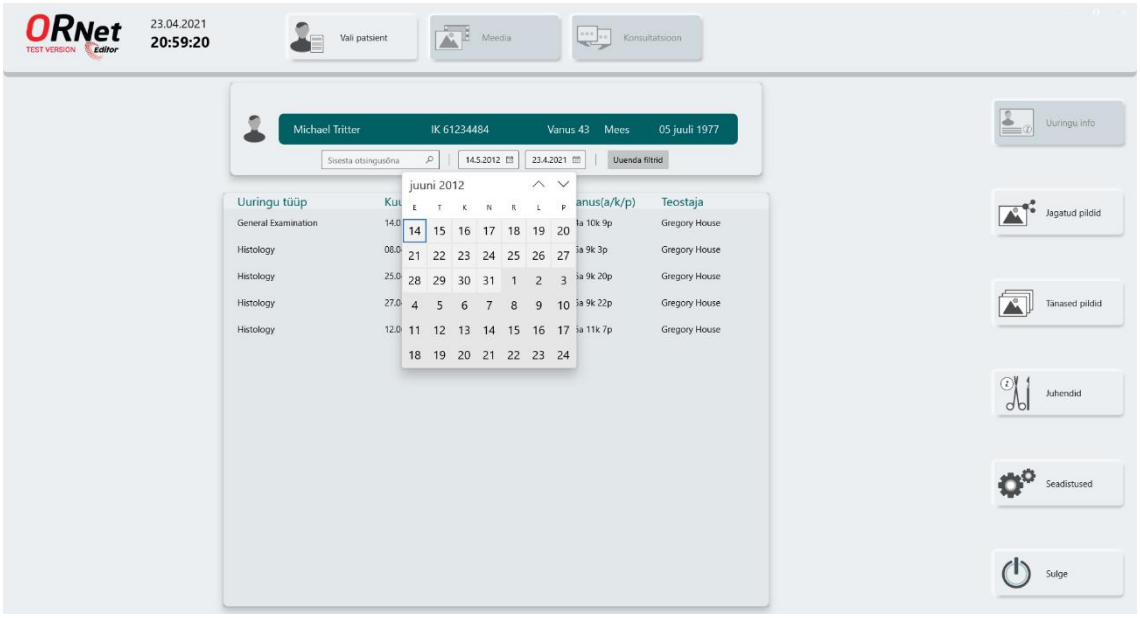
Joonis 8. Patsiendi valimine

Arst saab loendit filtreerida järgmiselt:

- Kõik – kõik patsiendid andmebaasist
- Tänased patsiendid – patsiendid, kelle uuringud toimusid täna
- See nädal – patsiendid, kelle uuringud toimusid sellel nädalal

Allpool on nupp "Laienda nimekirja", millega arst saab vajaduse korral kõigi patsientide nimekirja laiendada. Patsiendi nimekirja avamisega käib kaasas tore laiendusanimatsioon. Patsientide loetelu koosneb 6-st veerust, kus on põhiteave patsiendi kohta: nimi, perekonnanimi, sugu, isikukood, sünniaeg, vanus.

Pärast seda, kui arst on loendist või otsingutulemusest patsiendi valinud, suunatakse teda uuringu valiku lehele (joonis 9).



Joonis 9. Uuringu valimine

Selles vaates näeb arst kõiki patsiendiga seotud uuringuid, kui neid on rohkem kui 0. Kui patsiendiga ei ole mitte ühtegi uuring seotud, siis ilmub infoaken mis teavitab arsti uuringute puudumisest.

Lehe ülaosas on patsiendi andmed, nii et arst saaks veenduda, et valitud on õige patsient. Kui on valitud vale patsient, võib arst patsiendi valimise lehele naasmiseks vajutada nuppu „Vali patsient“. Allpool on otsinguvälja ja nupud uuringukuupäeva järgi filtreerimiseks. Otsida saab uuringukoodi ja uuringu tüübi järgi. Vajutades filtreerimisnuppu avaneb *CalendarView*, see liideseelement võimaldab arstile mugavalt ja kiiresti valida ajavahemikku uuringute otsimiseks. Filtrite lähtestamiseks ja algse loendi kuvamiseks klõpsake nuppu "Uuenda filtrid". Uuringute loetelu koosneb 5-st veerust: uuringu tüüp, kuupäev, uuringukood, patsiendi vanus uuringu ajal ja uuringu läbi viinud arsti nimi.

Pärast uuringu valimist suunatakse arst lehele „Meedia aken“, kus on kõik patsiendi ja uuringuga seotud meediafailid.

Antud lahenduses patsiente ega uuringuid pole võimalik lisada, muuta või kustutada. See on ettevõtte rakenduse skoobi kitsendamise piirang.

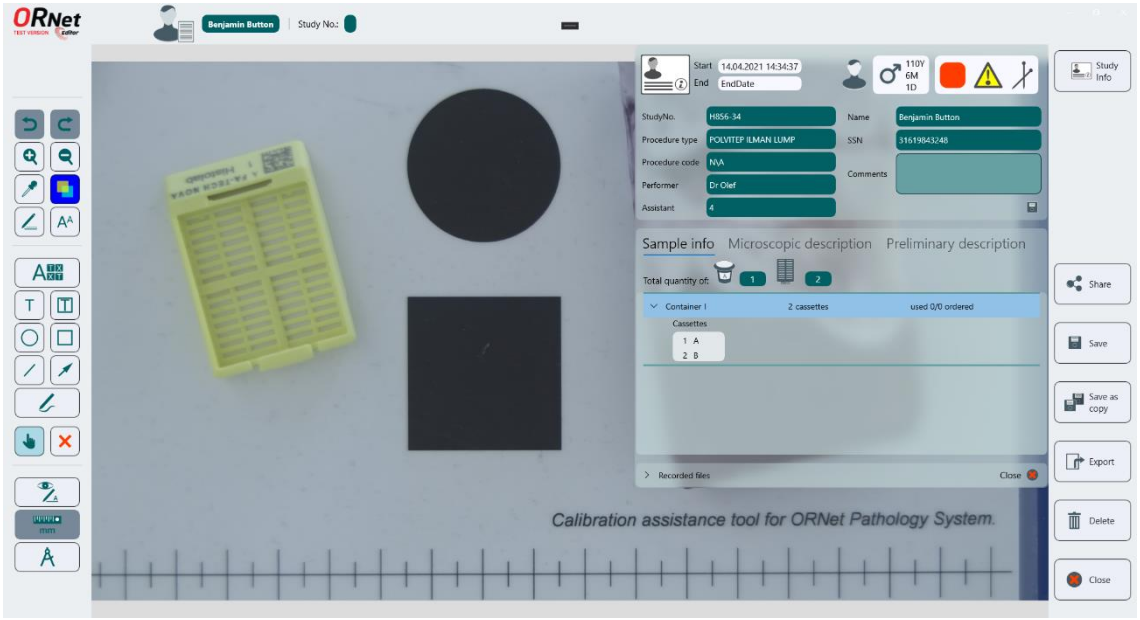
3.4 *User Control*

User Control on lihtne, disainerisõbralik lähenemine korduvkasutatava paigutuse loomisel. See on efektiivne strateegia rakenduskoodi vähendamiseks. Need kasutajaliidese elemendid pakuvad oma loogikat, järjepidevaid, korduvkasutatavaid liideseid ja kapseldatud *code-behind*. [12]

Selle elemendi loomisega on võimalik kaasata olemasolevate juhtelementide (nuppud ja paigutuselemendid) sündmusi ja meetodeid ning luua ka oma sündmusi. Kõigi *User Control* elementide loomisel autor kasutas MVVM mustrit.

3.4.1 Uuringu info

Üks autori loodud elementidest on *StudyInfoControl*, mis ilmub siis, kui arst valib "Uuringu info" menüüpunkti meediafailide nimekirja aknas ja *Image Editor* aknas. See element näeb välja kõikidel lehtedel ühesuguselt, asub *Popup* akna sees ja võimaldab arstil näha kogu informatsiooni patsiendi, uuringu ja sellega seotud proovide kohta. Arstil on võimalus patsiendi andmetele kommentaare või märkusi lisata.



Joonis 10. Uuringu info

See juhtelement kasutab palju erinevaid elemente, mis võimaldavad kompaktselt korraldada suurt hulka teavet. Näiteks võimaldab *Pivot* juhtelement navigeerida väikese sisusektsioonide komplekti vahel, antud juhul proovi andmete ja kirjelduse vahel. Kasutusel on ka *Expander* element mis on laiendatav konteiner Windows Community

Toolkit tööriistakomplekti elementide kogust, et mahutada kogu sisu, mida kasutaja saab teabele juurde pääsemiseks vajadusel avaldada või peita. [13]

Oluline on märkida, et konteinerid ja kassetid, milles proove hoitakse, nummerdatakse erinevates haiglates erinevalt. Praegu kasutatakse kolme tüüpi numeratsiooni: araabia numbrid, rooma numbrid ja tähestikulised koodid. Haiglatel on võimalus numeratsiooni peenhäälestada, näiteks jätta tähed tähestikulisest kodeeringust välja.

```
private static INumberConverter NumberToLetterConverter(Enums.NumberFormat
format, SystemSettings systemSettings, SystemSettingsBase systemSetting)
        {
            var converterSelector = new NameConverterSelector();
            var containerConverter = converterSelector.Select(format);
            if (containerConverter == null)
            {
                return null;
            }

            if (containerConverter.GetType() ==
typeof(NumberToRomanNumeralConverter))
            {
                return containerConverter;
            }

            INumberToLetterConverter numberToLetterConverter =
(NumberToLetterConverter) containerConverter;
            numberToLetterConverter.IsRomanNumeralsSymbolsAllowed =
                systemSettings.AllowRomanNumeralSymbolsInContainerName;


numberToLetterConverter.SetExcludeLetters(systemSetting.ContainerNamingOption
sEnum);

            return numberToLetterConverter;
        }
```

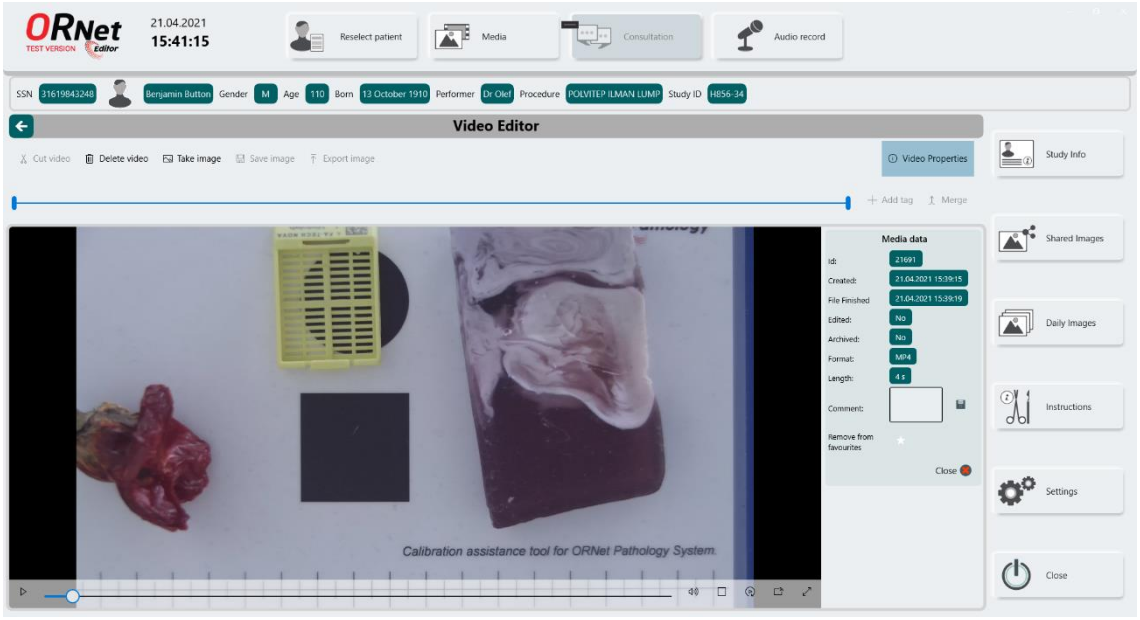
Joonis 11. Konverteri valimise loogika

Teave vormingu kohta pärineb andmebaasist ja läheb juhtelementi *ViewModel*-isse. *ViewModel* teisendusteenust (joonis 11) kasutades väljastab andmed vajalikus vormingus.

3.4.2 Meedia atribuudid

Hea näide selliste elementide iseseisvusest on *MediaPropertiesControl*. Seda UI elementi kasutatakse faili kohta teabe kuvamiseks meediafailide nimekirja aknas, *Video Editor* ja *Audio Editor* vaates. See element näeb kõigil lehtedel ühtemoodi, kuid kuvab vajaliku

teabe sõltuvalt failitüübist. Näiteks piltide puhul näitab see kalibreerimise olekut ning video- ja helifailide puhul salvestamise kestust. Selle elemendi abil saab kasutaja lisada tekstikommentaare või eemaldada fail lemmikute loendist.

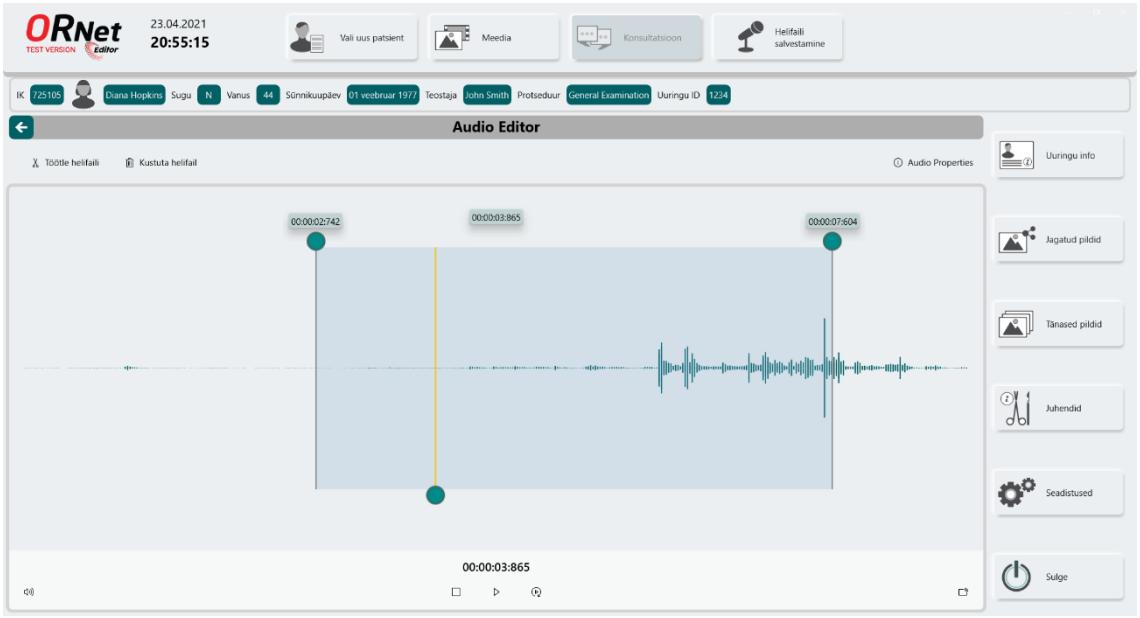


Joonis 12. Meedia atribuudid Video Editoris

Kogu selle juhtimise loogika toimub *MediaPropertiesControlViewModel*-i sees ja kasutab seega MVVM mustri põhimõtteid, vältides lehekoodi keerukaks muutmist ja vajadusel koodi dubleerimist. Kõik *User Control*-i tegevused töötavad *ICommand* liidese kaudu, sest nad pakuvad mehhanismi vaatele mudeli värskendamiseks MVVM'i arhitektuuris. Need võimaldavad otsida elementide puust käsuhaldurit.

3.4.3 Audio mängija

Üks keerukamaid elemente on *AudioPlayerControl*. See element võimaldab kuulata audiofaili, liikuda mööda helirada ja valida jaotisi ning faili lõigata. See sisaldab veel üht lisakontrolli *WaveformControl*, mis *Canvas* elemendi abil joonistab helifaili signaali kuju punkthaaval.

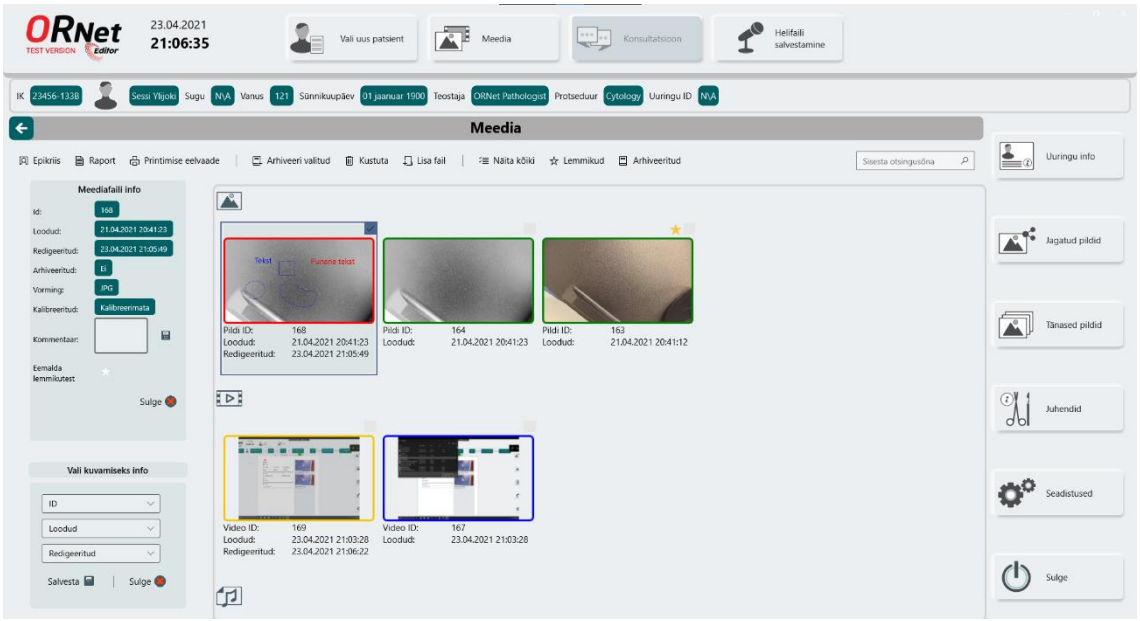


Joonis 13. Audio mängja

Mõlemad juhtelemendid kuulavad ja reageerivad üksteise sündmuste peale ning seetõttu töötavad nad väga stabiilselt ja kiiresti. Kõik see toimub ülejäänud lehekoodist sõltumatult ja hõlbustab nende elementide väljatöötamist tulevikus.

3.5 Meedia aken

Teine vajalik osa rakenduses on meediafailide lehekülg. Sellel leheküljel toimub peamine töö meediafailidega, seetõttu siin on palju erinevaid funktsionaalseid elemente. Autori eesmärgiks oli jagada kõik funktsionaalsed nupud ja filtrid gruppideks ning failide loetelu teha arusaadavaks, et võimaldada kiiremat otsingut ja koostööd.



Joonis 14. Meediaaken

Sellel lehel keskseks sisuks on meediafailide nimekiri. See on *GridView* tüüpi kogu, mis jaguneb meediafailide tüüpide põhjal kolme kategooriasse: pildid, video ja helifailid. Siin näete selgelt värvikodeerimise rakendamist, kus igale failitüübile ja redigeerimisolekutele on määratud erinev värv. Kogumik on täidetud autori loodud *MediaItemControl*-i elementidega, see on *User Control* element mis sisaldab meediafaili pisipilti (*thumbnail*) ja kõiki selle omadusi nii andmebaasist, kui ka failist. Kolm atribuuti, mille kasutaja kuvamiseks valib, on samuti osa *MediaItemControl*-ist. Ta saab neid valida kollektsioonist vasakul asuva menüüpunkti "*Media Data View*" kaudu. Nende meediafailide kuvamise seaded salvestatakse iga kasutaja jaoks eraldi. Samaaegseks kuvamiseks saab kasutaja valida järgmisest loendist kolm atribuuti:

- ID – meedia ID andmebaasist
- Loodud – näitab faili loomise aega
- Redigeeritud – näitab faili redigeerimise aega
- Arhiveerimise staatus – näitab kas pilt on PACS arhiivile saadetud või mitte
- Failivorming
- Faili pikkus

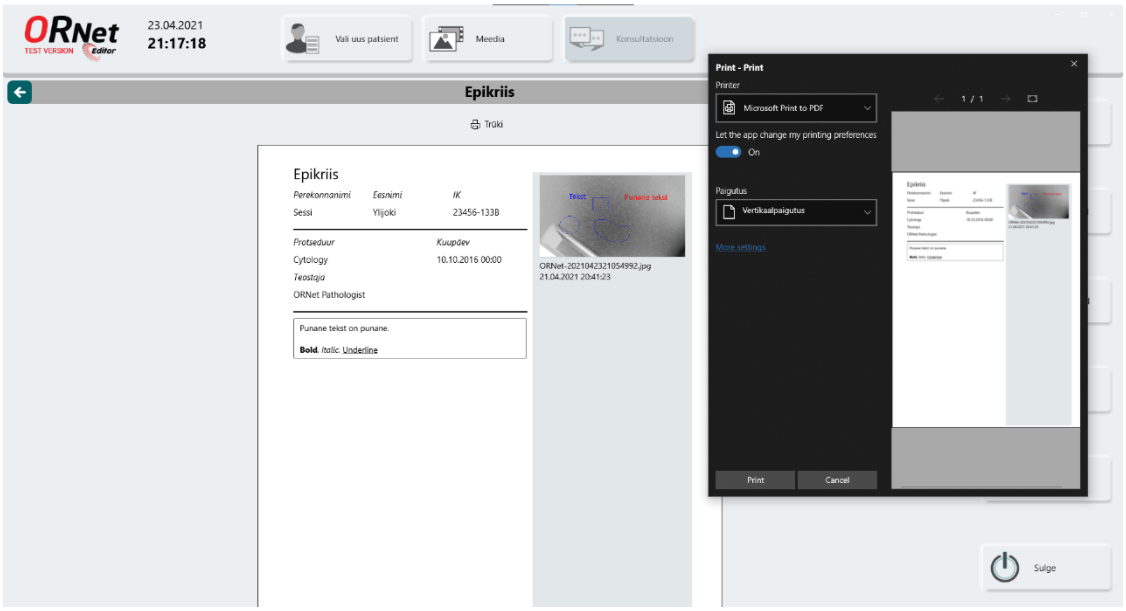
- Proovianuma number – konteineri number millega see fail on seotud
- Kalibratsiooni staatus – näitab kas pilt on kalibreeritud (käsitsi/automaatselt) või mitte.
- Kommentaar

Eespool on *MediaPropertiesControl* element, mis kuvab valitud faili atribuudid ja võimaldab kasutajal lisada seda oma lemmikute loendisse või lisada kommentaari.

Ülemine *Command Bar* element on jagatud kolme kategooriasse:

- Raportid
- Faili lisamine, eemaldamine ja arhiveerimine
- Filtrid

Aruanded võimaldavad mugavalt eksportida pilte soovitud vormingus. Aruande saab koostada patsiendiandmetega epikriisina (joonis 15), kuhu arst saab lisada oma kommentaarid ja seejärel printida ühendatud printeri abil või salvestada PDF-vormingus.



Joonis 15. Epikriis ja printimise dialoogaken

Kasutaja saab faile lisada, klõpsates nupul "Lisa fail". Sellisel juhul avaneb kõigile Windowsi operatsioonisüsteemi kasutajatele tuttav akna *File picker*, kus kasutaja saab

valida oma arvutisse, võrgukettale või välisele mäluseadmele salvestatud failid. Kasutaja saab faili nimekirjast eemaldada, klõpsates nupul "Kustuta". Sel juhul avaneb dialoogiaken, kus kasutaja peab kinnitama toimingud, et vältida olulise faili juhuslikku kustutamist. Nupp "Arhiveeri valitud" saadab valitud meediumifailid PACS arhiivi. Praegu toetatakse ainult piltide arhiiveerimist, kuid tulevikus võivad olla lisatud video- ja helifailide arhiveerimine ning juba arhiveeritud failide PACS'ist tagasivõtmise võimalus.

Andmeid on võimalik loendist leida ja filtreerida otsingu või filtrite abil. Kasutaja jaoks on saadaval järgmised filtreerimistüübid:

- Filtreerimine failitüübi järgi - kasutaja saab valida, kas kuvatakse kõik failitüübid, ainult pildid, ainult video või ainult helifailid
- Ainult lemmikud
- Ainult arhiveeritud

Filtrid saab kombineerida, näiteks saab loendis näidata ainult arhiveeritud lemmikuid pilte.

3.6 Teised tööd

Töö käigus olid tehtud ka väikesed tööd, mida autor ei saa mainimata jätta. Need rakenduse omadused otseselt mõjutavad UX ja UI adapteerimist.

3.6.1 Erinevate seadmete toetamine

Patoloogia laborites kasutatakse nii tavalisi personaalarvuteid kui ka sülearvuteid, samuti kasutatakse puutetundliku ekraaniga kõik-ühes arvuteid**.** Seetõttu võeti programmi kasutajaliidese arendamisel algusest peale arvesse vajadust muuta töö programmiga mugavaks nii arvutihiire kui ka puuteekraani või digitaalse pliiatsi kasutamisega.

Mõne programmi elemendi jaoks lisati *drag and drop* tegevus, et teksti *Popup* aknast pildile liikuda või loendis olevate elementide järjekorda muuta. See suhtlus on kasutajale mugav ja loomulik, sõltumata sisestusseadmest. Loodud rakenduses töötab *drag and drop* tegevus sujuvalt igas suunas, sealhulgas rakenduse-rakenduse ja töölaua-rakenduse vahel. Näiteks meediaaknas saab uusi meediafaili töölauast rakendusse lisada *drag and drop* tegevuse kaudu.

Mõned liidese elemendid tuli käsitsi kohandada. Selleks autor lõi välja *ExtendedControl* klassi mis tuvastab sisestusmeetodeid (joonis 16).

```
public class ExtendedControl : ContentControl
    {
        public static readonly DependencyProperty IsPointerOverProperty =
            DependencyProperty.Register("IsPointerOver", typeof(bool),
typeof(ExtendedControl),
                new PropertyMetadata(false));

        public bool IsPointerOver
        {
            get => (bool)GetValue(IsPointerOverProperty);
            protected set => SetValue(IsPointerOverProperty, value);
        }
        protected override void OnPointerEntered(PointerRoutedEventArgs e)
        {
            base.OnPointerEntered(e);
            if (e.Pointer.PointerDeviceType ==
Windows.Devices.Input.PointerDeviceType.Mouse ||
                e.Pointer.PointerDeviceType ==
Windows.Devices.Input.PointerDeviceType.Pen)
            {
                IsPointerOver = true;
            }
        }
        protected override void OnPointerCanceled(PointerRoutedEventArgs e)
        {
            base.OnPointerCanceled(e);
            IsPointerOver = false;
        }
        protected override void OnPointerExited(PointerRoutedEventArgs e)
        {
            base.OnPointerExited(e);
            IsPointerOver = false;
        }
    }
```

Joonis 16. Extended control klass sisaldab meetodeid, mis võimaldavad hiire või digitaalse pliiatsiga hõljudes andmeid kuvada

Näiteks on epikriisi loomise lehel kasutajal võimalus pilt loendist eemaldada. Hiire või digitaalse pliiatsi kasutamisel viib kasutaja lihtsalt kursori selle pildi kohale, mille ta soovib kustutada ja paremale ülemisse nurka ilmub rist, millele klõpsates pilt nimekirjast kustutatakse. Puutetundliku ekraani kasutamiseks on välja töödatud teistsugune suhtlusloogika. Kasutaja peab panema sõrme pildile, mida soovib kustutada, ja tuleb pühkida vasakule. Kui kasutaja vabastab libistatava üksuse enne künnisest üle liikumist,

sulgub menüü ja käsku ei täideta. Kui kasutaja libistab üksuse künnisest üle ja vabastab seejärel üksuse, käivitatakse käsk kohe.

Joonis 17. Pildi eemaldamine sõrmega (üleval) ja hiirega (all).

Paljudel sisendelementidel, näiteks *TextBox*-il, on sisseehitatud tugi mitte ainult tavalise klaviatuuriga või puuteklaviatuuriga kirjutamiseks, vaid ka digitaalse pliiatsiga läbi käsitsikirjutuspaani, mis teisendab käsitsi kirjutatud teksti automaatselt trükitekstiks.

3.6.2 Lokaliseerimine

Üheks arendustingimuseks oli rakenduste lokaliseerimine 4 keelde: inglise, eesti, soome, rootsi. UWP-rakenduses salvestatakse lokaliseeritud sõnad RESW-faili. Lokaliseerimisprotsessi optimeerimiseks soovitas autor kasutada Microsoft Multilingual App Toolkit tööriistakomplekti, mis võimaldab jälgida sõnade tõlgete muutusi, genereerida tõlkeid Microsofti toodete suure sõnastiku abil. Autor töötas välja ka väikese programmi, mis tõlkis rakenduse puuduvad osad ettevõtte Exceli arvutustabelite abil koos ettevõtte töötajate koostatud tõlgetega. Selle tulemusena loodi süsteem, tänu millele autor saab kõik tõlked rakendusse juurutada ühe tunni jooksul ja vajadusel tulevikus on

võimalik rakendus kiiresti teistesse keeltesse tõlkida. See võimaldab ettevõttel ja rakendusel laieneda uutele turgudele.

Rakenduse esmakordsel käivitamisel kasutab programm atribuuti *SystemInformation.Instance.Culture.Name* praegust Windows 10 süsteemikeelt. Pärast on kasutajal alati võimalus menüüs “Seadistused” programmi keelt muuta.

Lokaliseerimisel tuli arvestada ka sellega, et ühes keeles võivad olla sõnad mitu korda pikemad, kui teises. Paljud soomekeelse rakenduse menüüelemendid võtavad nüüd keskmiselt 11% rohkem ruumi ja rootsi keeles 14% rohkem ruumi (joonis 18).

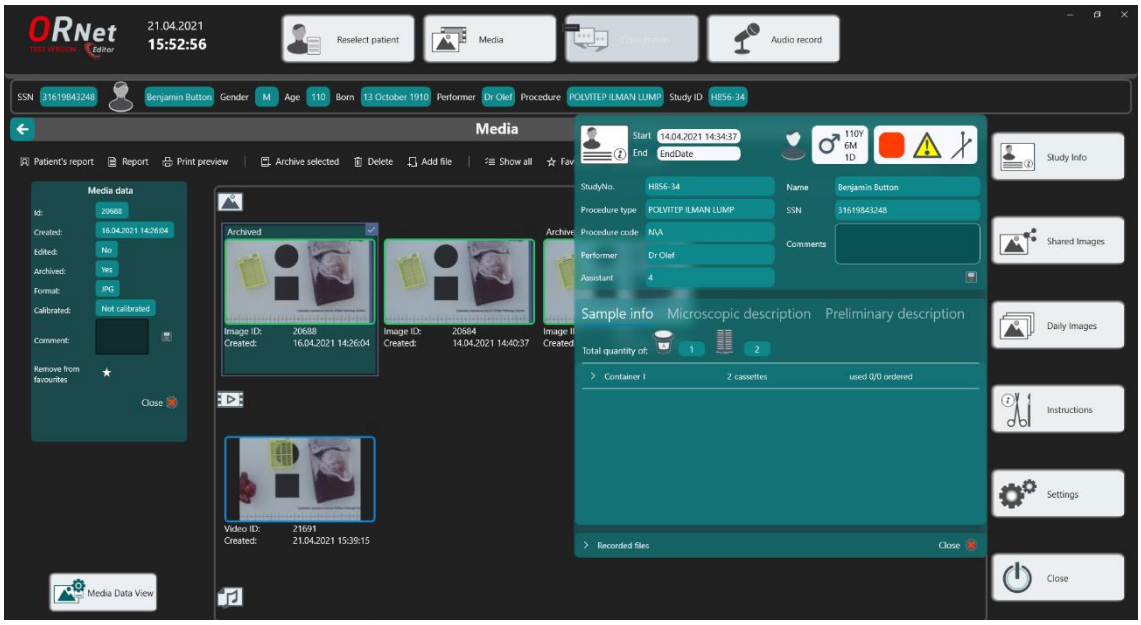
Joonis 18. Rootsi keel rakenduses

Neid tähelepanekuid võeti arvesse ja kasutajaliides muudeti pikkade ridade jaoks dünaamilisemaks, vajadusel kasutajaliidest pikendatakse ja read mähitakse tervete sõnadega, kasutades elemendi *TextBlock* atribuuti *TextWrapping* väärtust *WrapWholeWords*.

3.6.3 Tume värviskeem

Kuna operatsioonisüsteem Windows 10 toetab tumedat värviskeemit, peaks meie rakendus suutma soovi korral ka heleda teema tumedaks muuta. Üldiselt kipuvad valged heledad ja erksad värvid kasutaja tähelepanu hajutama, mistõttu kasutajale on keeruline ülesandeid täita ning tootlikkus väheneb. Tume kasutajaliides saab seda probleemi leevendada, suunates kasutaja fookuse mitte tööruumi taustale, vaid liidese sisualadele.

Selleks, et toetada rakenduses tumedat värviskeemi ressursisõnastikusse autor lisas kaks võtit *Light* (hele) ja *Dark* (tume), kus on määratud kõik värvid ja ikooni versioonid mis sõltuvad konkreetsest väärtusest.



Joonis 19. Tume värviskeem meediaaknas

Tumeda liidese väljatöötamisel valiti värvid nii, et elementide kontrastsus ei kannataks. Näiteks kasutati väga küllastunud kuvavärvide asemel pastelseid ja summutatuid värve.

Vaikimisi kasutatakse standardseks teemaks operatsioonisüsteemi praegust teemat, kuid vajadusel saab kasutaja seda programmi sätetes vahetada.

4 Tulemused

Tagasiside kogumine jagunes kaheks etapiks: ettevõttesisene testimine enne testversiooni turule viimist ja klientide tagasiside kogumine pärast testversiooni väljaandmist. Koostati ka edasine arenguplaan klientide soovide rahuldamiseks ja rakenduse funktsionaalsuse tõstmiseks.

4.1 Tagasiside ettevõte poolt

Kuna rakenduse kasutajaliides töötati välja ettevõtte pakutud prototüübi põhjal, oli väga oluline saada ettevõttelt tagasisidet, et mõista, kas lõplik projekt vastas nende ootustele.

Ettevõttesisesel testimises tõsiseid probleeme ei leitud, vaid tulid lisaarendused rakendusele, mis olid lisatud edasiarendus plaanile. Rakendus vastas ettevõtte ootustele ja aitas ette valmistada ka ettevõtte teiste toodete UI ja UX aluseid.

4.2 Tagasiside klientide poolt

Klientidelt tagasiside kogumiseks saavutati suurte haiglatega kokkulepe programmi testversiooni paigaldamiseks. Moodustatud on arstide rühm, kes kasutab loodud rakendust igapäevaselt. Töö esitamise ajaks saadi tagasisidet Joensuu haigla peapatoloogilt.

Saadud tagasiside on enamasti positiivne. Arst hindas rakenduse välimust, nimetades seda kaasaegseks, arusaadavaks ja silmale meeldivaks. Kiitis võimalust muuta meediaelementide all oleva info kuvamist, mugavat nimekirjade filtreerimist ja otsingu funktsionaalsust, ning võimalust lisada failidele kommentaare.

Samal ajal juhtis arst tähelepanu mõnele probleemile, mis on seotud kalibreerimisstaatuse kuvamisega pildiredaktoris. Rakenduse järgmistes versioonides soovib arst saada integratsiooni PACS’iga arhiveeritud piltide vastuvõtmiseks ning integratsiooni LIS’iga patsiendi otsimiseks ja valimiseks haigla infosüsteemist. Kõiki arsti kommentaare ja soove analüüsiti, ning lisati edasiarendus plaanile.

Kuna rakenduse kvaliteedi objektiivseks hindamiseks ei piisa ühe arsti ülevaatamisest, loodab autor lähitulevikus saada aruannet ülejäänud testimise osalejate poolt.

4.3 Projekti edasiarendus

Ühe suurema planeeritud omadusena mida kliendid vajavad on võimalus võtta pilte PACS'ist kasutaja arvutisse. Kuna tegemist on üsna komplitseeritud funktsionaalsusega, siis vajab palju läbimõtlemiset, eriti UI ja UX osas. Näiteks oleks vaja otsustada kuidas selline vaade rakendada, näiteks *Popup* akna kujul ning arstile on vaja anda võimalust valida pilte salvestamiseks.

Seepärast, et „Uuringu info" vaade oli lisatud rakendusse viimasena, seda on vaja lokaliseerida ja see vaade tuleb välja järgmise versiooniga.

Järgmisesse rakenduse versiooni ettevõte otsustas lisada ka konsultatsioonimooduli ja seepärast autori poolt on vaja luua sobivat vaadet ning vajadusel uusi kasutajaliidese elemente. Konsultatsioonimoodul võimaldab arstidel suhelda häälsuhtluse kaudu ja pilte ühiselt annoteerida.

5 Kokkuvõte

Selle bakalaureusetöö eesmärk oli viia lõpule meditsiinivaldkonna programmi kasutajaliidese arendamine. Kasutajaliidese disain kujundati täielikult ümber, muudeti olemasolevaid elemente ja lisati uusi. Samuti võttis olulise osa tööst üle üleminek täieõiguslikule MVVM mustrile ja loogika ülekandmine *code-behind*-ist ViewModel-isse.

Valdkonna ja kasutatud tehnoloogiate analüüs võimaldas autoril mõista, millised nõuded ja ootused sihtrühmal on ning millised on võimalused nende rakendamiseks kasutatavas platvormis. Ehkki lõplik kujundus erineb ettevõtte esitatud prototüübist, oli see ettevõtte visiooniga võrdlemisele jaoks suurepärane võrdluspunkt. Kõik prototüübile tehtud muudatused olid õigustatud nii disaini, kui ka funktsionaalsuse poolest. Arendusprotsessi jälgisid pidevalt ettevõtte töötajad ja vajadusel toimusid koosolekud, kus arutati vajalikke muudatusi.

Töö tulemusena saadi turule tipptasemel rakendus UWP, mis vastab ettevõtte ja kasutajate ootustele. Analüüsiti testimise käigus klientidelt saadud tagasisidet ja selle põhjal koostati edasiarendusplaan. Selle kõige põhjal võib väita, et kõik eesmärgid olid saavutatud ja töö õnnestus.

Töö käigus kasutas autor nii olemasolevaid teadmisi kui ka omandas palju uusi.

Kasutatud kirjandus

[1] „Bait Partner,“ [Võrgumaterjal]. Available: http://www.baitpartner.eu/home/. [Kasutatud 26 aprill 2021].

[2] „ORNet - Medical imaging and device management,“ [Võrgumaterjal]. Available: https://ornet.eu/. [Kasutatud 26 aprill 2021].

[3] O. S. Pianykh, Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide 2nd Edition, Berlin: Springer, 2011.

[4] „Choose your Windows app platform,“ Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/apps/desktop/choose-your-platform#uwp. [Kasutatud 14 aprill 2021].

[5] Y. Zhang, P. Masci, P. Jones ja H. Thimbleby, „User Interface Software Errors in Medical Devices: A Study of US Device Recall Data,“ [Võrgumaterjal]. Available: https://www.researchgate.net/publication/330713409_User_Interface_Software_Errors_in_Medical_Devices_A_Study_of_US_Device_Recall_Data. [Kasutatud 20 aprill 2021].

[6] D. Bila, „User Interface Design Best Practices for Medical and Healthcare Apps,“ [Võrgumaterjal]. Available: https://ugem.design/blog/ui-design-tips-for-healthcare-apps. [Kasutatud 26 aprill 2021].

[7] „WinUI,“ Microsoft, [Võrgumaterjal]. Available: https://microsoft.github.io/microsoft-ui-xaml/. [Kasutatud 26 aprill 2021].

[8] „Fluent Design System,“ Microsoft, [Võrgumaterjal]. Available: https://www.microsoft.com/design/fluent/#/. [Kasutatud 26 aprill 2021].

[9] „Определение паттерна MVVM,“ [Võrgumaterjal]. Available: https://metanit.com/sharp/wpf/22.1.php. [Kasutatud 14 aprill 2021].

[10] „UWP documentation: Navigation design basics for Windows apps,“ Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/uwp/design/basics/navigation-basics. [Kasutatud 26 aprill 2021].

[11] „Understanding navigation,“ Google, [Võrgumaterjal]. Available: https://material.io/design/navigation/understanding-navigation.html. [Kasutatud 26 aprill 2021].

[12] „XAML - Custom Controls,“ Tutorials Point, [Võrgumaterjal]. Available: https://www.tutorialspoint.com/xaml/xaml_custom_controls.htm. [Kasutatud 26 aprill 2021].

[13] „Windows Community Toolkit Documentation: Expander,“ Microsoft, [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/windows/communitytoolkit/controls/expander. [Kasutatud 14 aprill 2021].

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

Mina, Nikita Gerassimov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kasutajaliidese 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.