TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Ragnar Kramm 206591IADB

Interaktiivse esitlusvaate loomine Digitoimikule, kasutades WebSocket protokolli

Bakalaureusetöö

Juhendaja: Jaanus Pöial
PhD

Tallinn 2024

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: Ragnar Kramm

04.01.2024

Annotatsioon

Lõputöö eesmärk on luua kontseptsiooni tõestus reaalajas interaktiivse lisafunktsionaalsuse näol Digitoimiku rakendusele, mida saavad prokurörid ja kaitsjad tulevikus kasutada, et kohtuistungitel menetluse materjale esitada istungil viibijatele.

Lõputöö raames loodi lisafunktsionaalsus Digitoimiku rakendusele, mis annab menetluse osalistele võimaluse luua esitlusrežiimis esitlusjärjestuse ja seda modifitseerida. Esitlusvaatesse on võimalik saata esitlemiseks materjale, nende olekut kontrollida ning hoida vaatlejate tähelepanu materjali õige osa juures läbi sünkroniseeritud lehekülje kerimise. Esitlust on võimalik teha nii individuaalselt enda masina siseselt kui ka mitme kasutaja üleselt. Lisafunktsionaalsuse tagarakendus on loodud C# programmeerimiskeeles kasutades .NET raamistikku ning kliendi rakendus TypeScriptis kasutades React raamistikku, reaalajas andmeedastus on rakendatud kasutades WebSocketi tehnoloogial põhinevat SignalR teeki.

Töös kehtestatud nõuded on paika pandud Digitoimiku tiimijuhi, analüütiku ning prokuratuuri poolse tellija koostööl, millele tugines autori arendustöö. Enamus olulised nõuded suudeti kontseptsiooni tõestuse loomiseks rakendada, kuid mõned süsteemi jaoks vähem kriitilised funktsioonid jäid ajapuuduse tõttu rakendamata. Tulevikus on plaan jätkata loodud süsteemi edasiarendust, et terviklik süsteem valmis saada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 5 peatükki, 15 joonist, 3 tabelit.

Abstract

Interactive Presentation View for Digital File System Using WebSocket Protocol

The aim of the thesis is to create a proof of concept in the form of a real-time interactive add-on functionality for a Digital File system, which can be used in the future by prosecutors and defence lawyers to present procedural material to the audience at court hearings.

Within the framework of the thesis, an additional functionality was created for the Digital File system, which gives the participants of the proceedings the possibility to create and modify a presentation order in presentation mode. It is possible to send material for presentation, check its status and keep the observers' attention on the correct part of the material through synchronised page scrolling. Presentations can be presented individually within a single machine or across multiple users. The additional functionality has been developed in C# programming language using the .NET framework in the backend and the client application in TypeScript using the React framework, real-time data transfer has been implemented using the SignalR library based on WebSocket technology.

The requirements set out in the work have been established through a collaboration between the Digital File team leader, the analyst, and the client from the prosecution service, on which the author's development work was based. Most of the essential requirements were implemented to create the proof of concept, but some less critical features for the system were not implemented due to time constraints. In the future, the plan is to continue further development of the system designed to deliver a complete system.

The thesis is in Estonian and contains 23 pages of text, 5 chapters, 15 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusliides
E-ITS	Eesti infoturbestandard
Esitlusvaade	Digitoimiku lisafunktsionaalsus, mis võimaldab menetluse materjale esitada; vaate nimetus, millest esitluse jälgijad näevad materjale
Esitlusrežiim	Esitlusvaate funktsionaalsuse üks osa/vaade, millest esitaja juhib esitlust
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastuse protokoll
IETF	*Internet Engineering Task Force*, Interneti Tehniline Operatiivkogu
ISKE	Infosüsteemide kolmeastmeline etalonturbe süsteem
JSONP	*JavaScript Object Notation with Padding*, JavaScripti objektide notatsioon täidistusega, tehnika andmete laadimiseks kasutades <script> silti
RFC	*Request for Comments*, kommentaarikutse, liik tehnilisi dokumente, enamasti kasutatavad standardina
TCP	*Transmission Control Protocol*, edastusohje protokoll

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Digitoimik on rakendus, mis võimaldab digitaliseerida kohtutoimikuid, kriminaaltoimikuid ja mugavalt läbi viia menetlusega seonduvaid protsesse, millest peamised on: suure hulga erinevate failitüüpide kuvamine, metaandmete sidumine toimingutega, failide sisu märgendamine ja kommenteerimine, menetluse materjalide sorteerimine ning filtreerimine.

Digitaalne kriminaaltoimik on osa Digitoimikust, mille eesmärk on digitaliseerida menetlusprotsess, mille eesmärk on säästa menetlusosaliste ajakulu ning mugavdada nende tööprotsesse. Digitoimik on kõikide võimalike menetluses kasutavate failide kuvamiseks mõeldud rakendus, mis võimaldab mugavalt suure ulatuse erinevat tüüpi faile kuvada ning nendega tööd teha. Failide sisu on võimalik kuvada, mõjutada tööriistadega nagu vabakäe joonistamine, faili sisu kinni katmine või lehekülgede kaupa poolitamine ja lisada märgendeid ning kommentaare teiste menetlusgrupi liikmete jaoks. Oma olemuselt on Digitoimik nö lisarakendus, millel on klientsüsteemid, mida omakorda erinevad rakendused kasutavad, Digitoimik iseseisvalt ei ole kasutatav.

Kriminaalmenetluse lõpus on tarvis menetluse käigus kogutud materjale kohtuistungil esitada kohtule ja teistele osapooltele. Menetluse materjalid koosnevad tihti mitmetest eri tüüpi failidest ning metaandmetest, mille mõnda loenguprogrammi panemine on osutuda tüütu ja aega nõudev tegevus. Kriminaalmenetluse digitaliseerimise raames on oluline ka mugav ja kiire viis kogutud materjale kohtule esitada, muidu jääks üks osa aega nõudvast protsessist menetluse raamest katmata ning ajaline ja rahaline kasum ei ole võimalikult maksimaalne. Materjalide mugav ja kiire esitlemine teeb kõigi istungil viibijate elu kergemaks säästes nii aega kui ka vahendeid nt paberikulu pealt. Selle töö eesmärk on luua prototüüp süsteem antud probleemi lahendamiseks esialgu kontseptsiooni tõenduse tasemel. Antud töös kasutab autor metoodikana eksperimenti, mille käigus uurib võimalikke tehnoloogiaid, mida on varem sarnaste probleemide lahendamisel kasutatud ning selle informatsiooni abil loob probleemi lahendava süsteemi prototüübi.

Töö tulemusest saavad enim kasu pikas perspektiivis prokurörid, kelle igapäeva töö üht osa see oluliselt lihtsustab, lühemas vaates saab kasu Digitoimiku tiim, kes kasutavad töö tulemust, et täiendada Digitoimiku süsteemi ja viia see samm lähemale valmimisele.

2 Analüüs

Tööle vastavate tehnoloogiate valikut, nõuete püstitamist ja arendustegevuse otsuste tegemist on mõjutanud ja aidanud Digitoimiku arendusmeeskond. Nõuded on paika pandud vastavalt tellija soovile koostöös ärianalüütikuga ning tiimijuhiga. Töö raames kasutatavate programmeerimiskeelte, raamistike ja tehnoloogiate valikul oli oluliseks piirajaks Digitoimiku tehnoloogiavirn (*technology stack*).

Digitoimik on koodibaasi poolest üks rakendus, mis kuvab erinevat sisu ja erinevaid komponente vastavalt klientsüsteemile, millest Digitoimikusse siseneti. Kaks peamist ja sisult ka kõige enam eristuvat nö kujutamis viisi Digitoimikust on digitaalne kriminaaltoimik ja digitaalne kohtutoimik. Digitaalne kohtutoimik jääb selle töö kontekstist välja, kuid lühidalt on Digitoimiku abil kohtutoimikus kohtunikul võimalik kohtumenetlust puudutavate failidega tutvuda mugavalt ühes vaates samuti on võimalik ka kõigil teistel menetlusosalistel dokumente elektrooniliselt vaadata [1]. Digitaalses kriminaaltoimikus on hulganisti funktsionaalsust, mida kohtutoimikus ei ole vaja. Kriminaaltoimiku faile on Digitoimikus võimalik redigeerida, mis hõlmab endas näiteks sensitiivse info katmist ja lehekülgede välja lõikamist, lisada faili sisusse märksõnu ja adresseerida kommentaare teistele menetlusgrupi liikmetele. Nii nagu need funktsionaalsused on kriminaaltoimiku spetsiifilised samuti on ka materjalide esitlemine läbi esitlusvaate. Esitlusrežiim on osa esitlusvaatest, see on rakenduse vaade, mille läbi kontrollib esitluse esitaja, mis sisu näidatakse esitlusvaates.

Esitlusvaate eesmärk on võimaldada prokuröril ja kaitsjal kriminaalasjas kogutud materjale kohtule esitleda kohtuistungil Digitoimiku siseselt. Hetkel toimub kohtuistungitel materjalide esitamine füüsiliselt paberkujul materjalide osalistele jagamise ning ettelugemise läbi. Esitlusvaate eesmärk on mugavdada kohtuistungil materjalide jälgimist ning aitab hoida kuulajate fookust kõneleja poolt kirjeldatud materjalidel läbi sünkroniseeritud kerimise ja märgendamise. Esitlemine peab olema võimalik kasutaja enda arvuti kontekstis, ehk esitaja enda arvutis luuakse esitlusaken, milles on näha vaid esitletavat materjali ja märgendeid, mida ta on soovinud esitluse

kontekstis näidata. See aken on võimalik tõsta eraldi ekraanile, mida näevad istungil osalejad ning esitaja juhib esitlust enda arvutist Digitoimiku esitlusrežiimi vaatest. Teine variant esitlust läbi viia on nö otseülekande stiilis, kus esitaja kõik tegevused kajastuvad kõikidel menetlusosalistel esitluse jälgijana nende endi arvutites olles esitlusvaates.

Töö skoop piirdub vaid kriminaalmenetluses kasutatava osaga. Välja jääb skoobist kohtuniku osalus, mis hõlmab endas materjalide pealkirjade alusel otsustamist, kas esitletav materjal sobib või mitte. Selle töö eesmärk on luua prototüüp lahendus reaalajas informatsiooni saatmiseks mitmele menetlusgrupi liikmele korraga, et esitlust saaks läbi viia istungisaalis ilma kolmandat rakendust kasutamata sisu näitamiseks. Seega kohtu poole osalus antud kontseptsiooni tõestuseks ei ole vajalik.

Otseülekandena tehtav esitlus peab olema praktiliselt reaalajas toimuv, ehk kui esitluse esitaja kerib teise faili kolmanda leheküljeni peab esitluse vaatlejate esiltusvaate aken ka sama kaugele kerima koheselt, samuti kui esitaja soovib mõned read faili sisust esile tõsta nende taustavärvi muutes on see muudatus ka vajalik koheselt teistele esitluse jälgijatele edasi kanda. Sellest tulenevalt on vaja saata ühe kliendi poolt teistele klientidele reaalajas uuendusi, seega antud rakenduses ei piisa tavalise HTTP (*Hypertext Transfer Protocol*) põhise API (*Application Programming* Interface) suhtluse loomisest.

2.1 Nõuded

Siin peatükis on süsteemi tööpõhimõtted lahti selgitatud kasutades funktsionaalseid ja mittefunktsionaalseid tingimusi. Funktsionaalsete tingimuste aluseks on prokuratuuri poolse tellija, Digitoimiku tiimi analüütiku ja tiimijuhi koostööl tekkinud ärinõuded.

2.1.1 Funktsionaalsed nõuded

Esitlusjärjestuse koostamine on järjekorras esimene funktsioon, mida esitlusrežiimi saabunud kasutaja soovib teha, selle jaoks valib ta eksisteerivate menetluses asuvate materjalide seast soovitud materjalid ja lisab need esitlusjärjestusse. Esitlusjärjestusse lisatud materjale saab ümber järjestada või ka vajadusel sealt eemaldada.

Materjalide märgendamine ja kommenteerimine peab olema võimalik esitlusjärjestuses olevatel materjalidel samamoodi nagu mujal paiknevatel materjalidel, kuid lisaks on veel juures eksklusiivsed esitluste jaoks mõeldud kommentaarid ja märgendid, mis

eksisteerivad vaid esitluse kontekstis. Esitlusvaates on näha vaid neid eksklusiivseid kommentaare ja märgendeid, et vältida suure hulga andmete välja filtreerimist, mida ei soovita esitluse ajal näidata, näiteks menetluse ajal tehtud märkmed prokuröridel omavahel.

Järgmiseks sammuks on materjalide esitamine esitlusvaates, selle nõude alla kuulub sobiva materjali valimine ning selle edastamine esitlusvaatesse. Valmis süsteemis osaleb selles protsessis ka kohtunik, kellele kõigepealt esitatakse materjalide pealkirjad, mille järel kohtunik otsustab, kas antud materjal on relevantne ja sobib istungis ettekandmiseks või mitte, kui kohtunik on materjalid vastu võtnud on võimalik esitajal materjali kõigile istungil viibijatele näidata. Antud töö skoobist jääb kohtuniku osa välja, et töö maht ei kujuneks liiga suureks. Praeguse töö kontekstis on vastav tegevus täielikult esitaja kontrolli all ja esitlusvaatesse materjali edastamine ei nõua kolmanda poole nõusolekut.

Viimaks kui valitud materjal on esitlusvaatesse edastatud soovib esitaja teatava osa enda tegevustest üle kanda kõikidele istungil osalejatele, kes esitlust jälgivad. Peamiseks ülekantavaks informatsiooniks on esitaja lehe kerimise olek, et esitaja ja esitluse jälgija näeksid sama sisu samaaegselt. Olulisuselt järgmine informatsioon on märgendid, mida esitaja teeb, et jälgijate tähelepanu pöörata mõnele reale või lõigule materjalides.

2.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsetest nõuetest on olulisemad turvalisus ja jõudlus. Turvalisuse kohapealt peab olema garanteeritud, et kõrvalistel isikutel ei oleks võimalust näha esitlusi ega esitlusega seotud materjale, mille nägemisõigust neil pole. Tabelis 1 on välja toodud ISKE skaala läbi Digitoimiku turbeastmed, mis jäävad kehtima ka lisafunktsionaalsuse loomisel. Kasutatud on ISKE (infosüsteemide kolmeastmeline etalonturbe süsteem) skaalat, sest hetkel ollakse ülemineku etapis E-ITS (Eesti infoturbestandard) kasutamisele ning see ei ole veel täielikult valminud. [2]

Tabel 1. Digitoimiku turbetasemed ja seletused [3]

Digitoimikus rakendatavad turvaklassid	**Selgitus**
Käideldavus tase K2	Käideldavus suurem või võrdne kui 99% ja väiksem kui 99,9% aastas ning maksimaalne lubatud ühekordse katkestuse pikkus teenuse töö ajal kuni 4 tundi (st ühekordse katkestuse pikkus võib olla vahemikus väiksem või võrdne 4 tunniga ja suurem kui 1 tund)*;
Tervikluse tase T2	Info allikas, selle muutmise ja hävitamise fakt peavad olema tuvastatavad; vajalikud on perioodilised info õigsuse, täielikkuse ja ajakohasuse kontrollid;
Konfidentsiaalsuse tase S3	Ülisalajane info: info kasutamine lubatud ainult teatud kindlatele kasutajatele, juurdepääs teabele on lubatav juurdepääsu taotleva isiku õigustatud huvi korral.

Info liikumine esitaja ja jälgijate vahel peab olema piltlikult vahetu, seega võiks uuendused näiteks esitaja lehe kerimisest jõuda esitluse jälgijateni 500ms jooksul, et kogemus ei oleks liiga aeglane ning segaks tööd.

2.2 Võimalikud lahendused

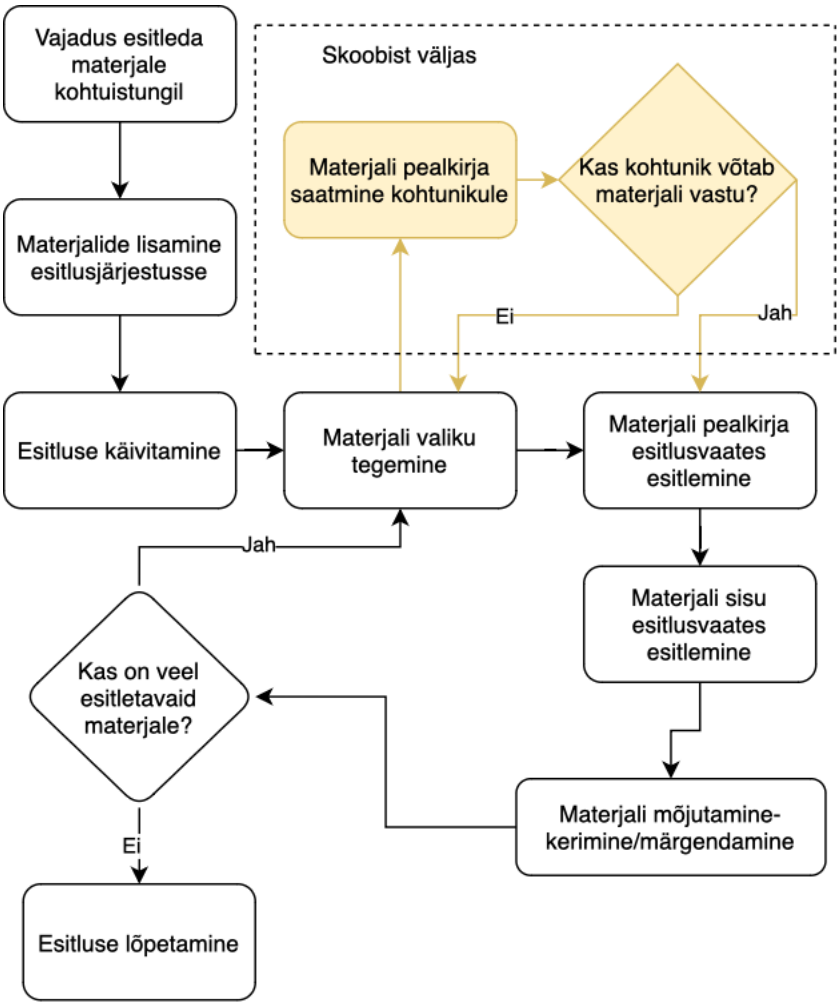
Selle süsteemi arenduse vajaduse tekkimisel soovitas Digitoimiku tiimi tehniline arhitekt Lauri Läänemets kasutada WebSocket tehnoloogiat, mis võimaldab just selliseid reaalajas saadetavaid uuendusi mitme kliendi vahel, nagu esitlusvaade nõuab. Ärinõuetes koorus välja ka kaks peamist andmeolemit, mis on esitlusvaates peamisteks informatsiooni kandjateks olem Esitlusjärjestus ja Esitlus. Kus Esitlusjärjestus omab informatsiooni, mis on individuaalne igale menetlusgrupi liikmele, kes soovib esitlusjärjestust koostada.

Sinna salvestatakse materjalid ja nendega seonduv info nagu materjalide järjekord, iga materjaliga seotud seisundid ja info esitaja kohta. Esitluse andmeolemis hoitakse informatsiooni, mis on ühe esitluse spetsiifiline ehk näiteks hetkel esitletav materjal, viidad nii kohtueelsele menetlusele kui ka kohtumenetlusele, millega seoses antud esitlust läbi viiakse ja ka esitluse erinevaid seisundeid.

WebSocket protokolli abil uuendatakse kõik esitlusega seotud andmed, mis peavad olema reaalajas uuenevad ja kõikidele klientidele samaaegselt saadetud. Kõik muud andmed, mis ei vaja kõikidele menetlusosalistele saatmist, uuendatakse tavapärase API liidese läbi.

2.3 Plokkskeem

Esitlusvaate töö pealiskaudset ülevaadet kirjeldab plokkskeem (vt Joonis 1), milles on kujutatud järjestikus olevad erinevad tegevused antud osarakenduses, tervikliku süsteemi osana on kujutletud ka kohtuniku poolne töö rakenduses, mis selle töö kontekstis jääb skoobist välja.



Joonis 1. Tegevuse plokkskeem

2.4 Reaalajas interaktiivse kasutajakogemuse loomise võimalused

Suurem osa internetist on üles ehitatud HTTP põhjal, mis tähendab seda, et enamus internetist on oma olemuselt olekuta (*stateless*). HTTP järgib nõue-vastus (*request-response*) mudelit, mille kohaselt klient pöördub serveri poole sooviga andmeid näha ja server vastab soovitud sisuga või annab keelduva vastuse. Kliendiks on tavaliselt kasutaja arvutis olev brauser või erandjuhtudel ka arendajate poolt kasutatavad, testimiseks mõeldud rakendused. Iga kliendi poolt algatatud päring on iseseisev ja server ei ole teadlik kliendi poolt eelnevalt tehtud päringutest, kuid kliendi poolt salvestatud brauserikookide (*cookies*) läbi on võimalik siiski pidada kasutaja sessiooni meeles ja seeläbi säilitada ühine kontekst kasutaja jaoks üle mitme päringu. [4]

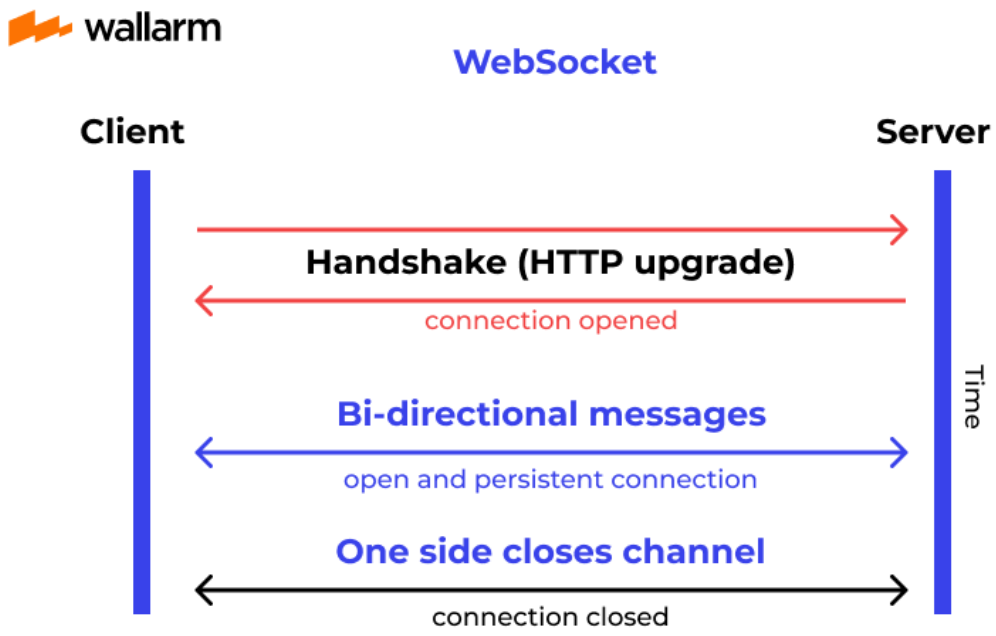
Nõude põhine suhtlus töötab hästi kui andmed, mida server väljastab, ei pea olema reaalajaliselt täpsed. Kuid reaalajas muutuva info kuvamiseks on tarvis stabiilset

kahesuunalist ühendust kliendi ja serveri vahel, mida võimaldab WebSocket protokoll. Lisaks sellele on kasutusel ka tehnoloogiad nagu long polling ja Server-Sent Events, mis võimaldavad sarnast tulemust saavutada, kuid jäävad WebSocket'ile alla mõningate näitajate poolest. [5]

2.4.1 WebSocket

WebSocket on arvutisideprotokoll, mis võimaldab kahepoolset suhtlust kliendi ja serveri vahel ühe TCP (*Transmission Control Protocol*) ühenduse kaudu. See standardiseeriti IETF (*Internet Engineering Task Force*) poolt RFC (*Request for Commenst*) 6455-s 2011. aastal. [6]

WebSocket kasutab kliendi ja serveri vahelise ühenduse loomiseks ja säilitamiseks kätlust (*handshake*), mille käigus saadab klient serverile HTTP päringu ja kui server toetab WebSocket protokolli, vastab ta staatuskoodiga 101, mis näitab, et ühendus on muudetud WebSocket-ühenduseks seda protsessi illustreerib joonis 2. Pärast edukat kätlemist saavad klient ja server saata üksteisele andmeid reaalajas. [6]



Joonis 2. WebSocket ühenduse loomine [7]

WebSocketit toetavad enamik kaasaegseid veebibrausereid ja -servereid. Samuti on loodud mitmeid WebSocket'i teeke eri programmeerimiskeelte jaoks, mis teeb WebSocket'i kasutamise oma rakendustes arendajatele lihtsaks.

2.4.2 Long polling

Long polling töötab nagu tavaline HTTP päring-vastus suhtlus, kuid server saadab vastuse vaid siis, kui kliendi poolt küsitavad andmed on muutunud või juurde tekkinud. Ehk kaob ära tavalise HTTP pollimise probleem, mis seisneb selles, et klient küsib serverilt andmete uuendust, teadmata kas andmetes on toimunud muutus või mitte. [7]

Kui server on lõpuks andmete uuendamisel saatnud kliendile vastuse, saadab klient uue päringu, mille järel protsess kordub- server saadab vastuse alles siis, kui andmetes on toimunud muudatus. [7]

2.4.3 Server-sent events

Server-sent events tehnoloogia erinevus tavalise HTTP suhtlusega seisneb selles, et server võib igal ajahetkel saata andmeid kliendile- klient ei pea andmete saamiseks päringut saatma. See võimaldab andmete uuenemisel kohe saata uued andmed kõikidele neist huvitatud klientidele. [8]

2.5 Kasutatavad raamistikud ja tööriistad

Kasutatavate tehnoloogiate valikut piiras Digitoimiku meeskonnas kasutatav tehnoloogiavirn (*technology stack*). Tagarakendus (*backend*) on kirjutatud C# programmeerimiskeeles kasutades .NET ja Entity Framework Core raamistikke. Kliendi poolne rakendus on kirjutatud TypeScripti programmeerimiskeeles ning React raamistiku abil, lisaks on kasutuses Redux teek rakenduse oleku jälgimise lihtsustamiseks ja Material UI teek visuaalsete komponentide jaoks. Andmebaasi haldamise süsteemina on kasutusel PostgreSQL. Lisaks kasutavad arendajad lokaalse arendustegevuse jaoks Dockerit, kus jooksutame andmebaasi, otsingumootorit Elasticsearch ja failide salvestamist haldavat Miniot. Dokumentatsioon ja ärinõuded on peamiselt Confluence viki süsteemis ning arendustegevust koordineerib tiimijuht Jira probleemide jälgimise tarkvaras Kanban projektihaldusmeetodil.

Raamistikke mis mugavdavad arendajatel reaalajas interaktiivseid rakendusi luua on mitmeid oma eeliste ja puudustega. (vt Tabel 2)

Tabel 2. Populaarsemate raamistike võrdlus [10]

Raamistik	**Eelised**	**Puudused**	**Toetatud keeled**
SockJS	Toetab varuvariante nagu Server-Sent Events, Long Polling ja JSONP (*JavaScript Object Notation with Padding*)-polling. Interaktiivne lihtsasti kasutatav API, mis teeb selle kasutamise alustamise lihtsaks.	Tänu oma lihtsusele puuduvad selles raamistikus oleku jälgimine, automaatne taasühendamine, sõnumite ajalugu. Lisaks SockJS ei garanteeri andmete terviklikkust- andmete järjekord ja kohaletoimetamine ei ole garanteeritud. Ühe domeeni kohta on võimalik avada vaid üks ühendus.	Ruby, Java, Scala, Python, Erlang, Rust
Rails ActionCable	Võimaldab paigaldada WebSocket otspunkti otse rakenduses, mis lihtsustab oluliselt juurutamise (*deploy*) protsessi ja võimaldab ka bruserikookide (*cookie*) jagatud kasutamist. Kliendi poolses rakenduses võimaldab ActionCable automaatseid uuesti ühendusi ja automaatset tellijate aboneerimist kanalitele.	Puuduvad varuvariandid kui WebSocket ei tööta. Puudub kviteerimine kui klient saadab sõnumeid serverile.	JavaScript, Ruby
SignalR	Kasutab WebSocketit seal kus võimalik ja omab varuvariante nagu Server-Sent Events ja HTTP Polling keskkondadeks, kus WebSocket ühendused ei ole lubatud. Laiuti arenduseks (*scale out*) on mitmeid valikuid: Redis, SQL Server ja Azure Service Bus. SignalR on .NET raamistikust, mis teeb selle kasutamise koos teiste .NET raamistiku erisustega (*feature*) nagu sõltuvuste süstimine (*dependency injection*), volitamine ja autentimine.	Sõnumite järjestus ja kohaletoimetamine ei ole garanteeritud.	C#, Java, Python, JavaScript

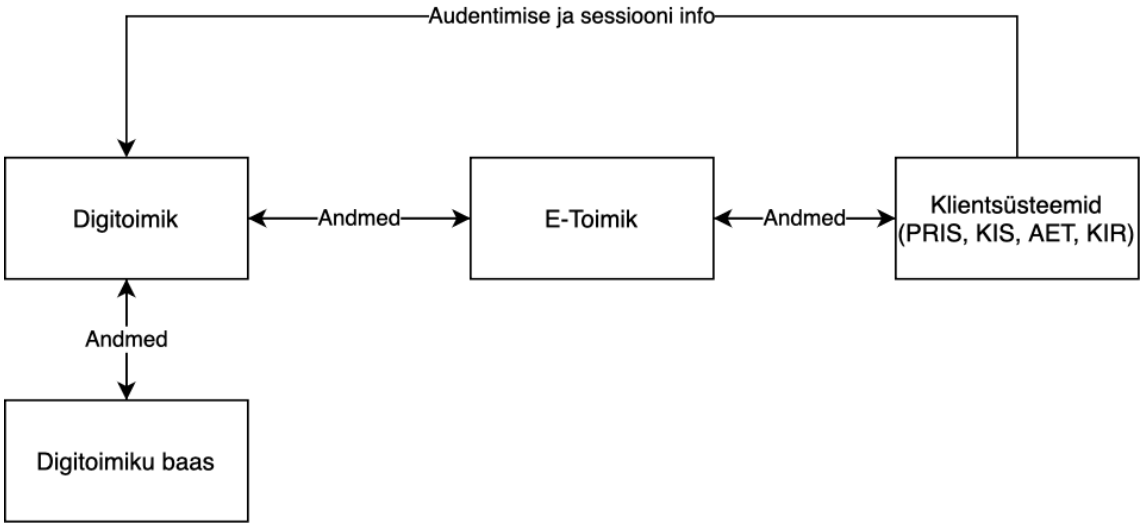
Eksisteerivate näidete põhjal on sarnaseid ülesandeid lahendatud just selliseid raamistikke kasutades, arvestades Digitoimiku tehnoloogiavirna on selle ülesande lahendamisel autoril valik piiritletud vaid SignalR raamistikule. Piiranguks on raamistike keeletoetus, millest vaid SignalR toetab C# programmeerimiskeelt, mida kasutab Digitoimiku tagarakendus.

3 Praktiline töö

Töö praktiliseks osaks on Digitoimiku rakendusse esitlusvaate ja esitlusrežiimi lisamine. Esitlusrežiim on osa kus esitluse koostaja loob esitlusjärjestuse, teeb märgendeid ja juhib esitlust. Esitlusvaade on eraldi vaade Digitoimikus, kus on võimalik esitlust jälgida, seda kaht võimaliku viisi- esiteks avades esitlusvaate enda brauseri aknas ning näidates seda akent teisel ekraanil jälgijatele või jälgija avab sama menetluse enda arvutis ning jälgib esitlust enda arvuti esitlusvaatest.

3.1 Baas

Andmebaasi mudeli täiendamisel võttis autor aluseks senise andmebaasi ja üldise andmete käsitlemise loogika, mida Digitoimiku tiimis jälgitakse. Informatsioon mis pärineb E-Toimikust seda Digitoimiku baasi kopeerida ei ole mõistlik, sest läbi selle võib tekkida mitmeid sünkroniseerimisega seonduvaid probleeme. Süsteemid, mis kasutavad E-Toimikut keskse andmekoguna on enamjaolt ülesehitatud sarnasel loogikal, süsteemis spetsiifilised andmed on salvestatud süsteemi enda andmebaasi, kuid erinevate süsteemide vahel jagatavad andmed on talletatud E-Toimikusse. (vt Joonis 3)



Joonis 3. E-Toimikut kasutavate süsteemide andmete liikumine

Andmebaasi mudel vajas täiendust kõige vähem, juurde tuli lisada kaks olemit- esitlusjärjestus ja esitlus. Esitlusjärjestus on kasutajapõhiselt individuaalne ning iga

kriminaalasjaga seoses saab eksisteerida vaid üks esitlusjärjestus. Baasi salvestatakse materjalide järjestus, erinevad tõeväärtus muutujad, mille läbi muudetakse materjalide seisundeid ja muud olulised väljad. Esitlus omab endas informatsiooni esitluse olekute kohta, mis kriminaalasjaga esitlus on seotud, kas ja millise kohtuasjaga antud esitlus on seotud, materjali kerimise kordajat ja esitluse esitaja informatsiooni. Baasis on meil kasutusel varjatud kustutamine kõikide andmete kohta, et andmed oleksid taastatavad, ka siis kui pahatahtlik kasutaja on midagi olulist ära kustutanud.

3.2 Tagarakendus

Tagarakendus koosneb peamiselt standardsetest osadest nagu kaardistaja (*mapper*) objektide viimiseks ühelt kujult teisele ja tagasi, *controller* mis on vahekihiks REST (*Representational state transfer*)-põhise liidese ja äriloogikat omava osa- *service*, vahel. Tagarakendus on Digitoimiku tiimis ülesehitatud enamasti Spring Boot rakendustes kasutatava Controller-Service-Repository mustri põhiselt, mis tuleneb autori hinnangul tiimi esimeste arendajate Java tasutast. Mõistagi peab lisama ka koodis kasutatavad objektid erinevate kihtide jaoks. Andmesaateobjekte on Digitoimiku rakenduses kahte tüüpi, esiteks objekti kuju millisena on see baasi salvestatud või kuju millisena see pärineb E-Toimikust, teiseks on objekti kuju, mida kasutatakse kliendirakenduses ja serveri poolsetes äriloogilistes tegevustes. Sellega oli tüüpsisu (*boilerplate*) loodud, et liigutada informatsiooni, mis ei vaja reaalajas sünkroniseerimist mitmete klientide vahel.

Esitlusjärjestuse objekti puhul kasutas autor pärinemist Toimik tüüpi objektist, sest oma olemuselt on need sarnased, Toimik objekt on loodud selleks, et kasutaja saaks juurde teha äriloogilises mõistes komplekte, kuhu on võimalik lisada materjale vastavalt enda soovile ning neid järjestada vastavalt vajadusele. Esitlusjärjestus nõudis sarnast funktsionaalsust väikeste lisadega, seega sai Esitlusjärjestuse objektile lisatud pärinemine Toimik objektist ning esitluse spetsiifilised väljad.

Sellele järgnes tagarakenduse SignalR osa teostamine, mille esimeseks sammuks oli veebirakenduse Program.cs teenuse registeerimine ja spetsiaalse Hub tüüpi lõpp-punkti defineerimine. (vt Joonis 4)

```
builder.Service.AddSingalR();
app.MapHup<DtHub>("/hub");
```

Joonis 4. Program.cs täiendused

Peale seda lisati Hub klassist pärinev klass DtHub, kuhu lisatakse vastava *hub* lõpp-punkti kõik avalikud meetodid, mille läbi informatsiooni mitmete klientide vahel sünkroniseeritakse. Näiteks on seal meetod, mis saadab esitluse esitaja kerimise oleku informatsiooni edasi kõikidele esitluse jälgijatele, et nad näeksid õiget lõiku materjalist, mida esitaja hetkel soovib näidata. (vt Joonis 5) Sellega on rakenduse tööks vajalikud andmeolemid ja failid lisatud tagarakendusse ning andmebaasi. [11]

```
public async Task SendKerimine(Contract.Dto.Esitlus esitlus)
{
    if (esitlus.KohtuAsiObjektId == null && esitlus.EsitlejaId !=
null)
    {
        await Clients.Group(esitlus.EsitlejaId)
            .SendAsync("esitlusUuendatud", esitlus);
    }
    await Clients.Group(esitlus.KohtueelneAsiObjektId)
        .SendAsync("esitlusUuendatud", esitlus);
}
```

Joonis 5. Kerimise info laiali saatmine tagarakenduses

SignalR tagarakenduse poolne töö on üsnagi lihtne, kogu loogika on ülesehitatud DtHub klassi siseselt, kus on avalikud meetodid, mida klient rakendused nimeliselt välja kutsuvad. Need meetodid saadavad selle peale vastu sarnaselt informatsiooni klientidele, kutsudes välja kliendi poolel nimeliselt toiminguid. Enne kui tagarakenduse toimingud saavad sünkroniseerivaid andmeid laiali saata on vaja kasutajad lisada jaotada gruppidesse, mille põhjal neile andmeid saadetakse. (vt Joonis 6)

```
public async Task LisaGruppidesse(string asiObjektId,
    string kasutajaIsikukood)
{
    await Groups.AddToGroupAsync(Context.ConnectionId, asiObjektId);
    await Groups
        .AddToGroupAsync(Context.ConnectionId, kasutajaIsikukood);
}
```

Joonis 6. Gruppidesse lisamine

Esitluse raames on iga kasutaja kahes grupis, tema enda unikaalse identifikaatoriga määratud grupp privaatseks esitlemiseks ning kohtueelse asja identifikaator, et läbi viia esitlust jälgijatega. Hub'is on võimalik kasutada ka erinevaid sõltuvusesüstimise läbi saadaval olevaid teenuseid mugavalt otse meetodite sees läbi meetodi parameetrite- *hub*

meetodeid inspekteeritakse vaikimisi, et leida sõltuvusesüstimisi [11]. Autori loodava rakenduse raames kasutatakse sõltuvusesüstimise läbi *hub*'i meetodis andmebaasiga suhtlevad *DataContext* klassi. (vt Joonis 7)

```
public async Task GetEsitlus(Contract.Dto.Esitlus esitlus,
    string kasutajaIsikukood, IDataContext dataContext)
{
    var dbEsitlus = dataContext
        .Query<Esitlus>()
        .SingleDefault(x =>
            x.KohtueelneAsiObjektId.ToString() ==
            esitlus.KohtueelneAsiObjektId &&
            x.EsitlejaId == kasutajaIsikukood);
    if (dbEsitlus != null)
    {
        esitlus = MapEsitlus(dbEsitlus);
    }

    await Clients
        .Group(esiltus.KohtueelneAsiObjektId)
        .SendAsync("esitlusUuendatud", esitlus);
}
```

Joonis 7. Sõltuvusesüstimine *hub*'i meetodisse

3.3 Kliendi rakendus

Kliendi poolne rakendus vajas eelpool mainitud vahetarkvara kasutuselevõtmist SignalR spetsiifika rakendamiseks. Läbi Reduxi toimingute (*action*) teostatakse kliendipoolne sünkroniseerivate tegevuste käivitamine, see tähendab et õige toimingu välja kutsumise pealt saadetakse vastava Hub lõpp-punkti vastu päring, mis omab endas informatsiooni, mida on soov teistele klientidele laiali saata. (vt Joonis 8)

```
if (action.type === 'ESITLUS_SAADA_KERIMISE_KORGUS') {
    if (esitlus.aktiivne) {
        sendEsitlus('sendKerimine',
        {
            …esitlus,
            kerimiseKorgus: action.kerimiseKorgus
        });
    }
}
```

Joonis 8. Kerimise *action* ja tagarakendusse info saatmine

Vahetarkvara osaks on ka SignalR ühenduse loomine kliendi ja serveri vahel ning ka selle sulgemine, mida samuti kutsutakse välja Reduxi toimingute läbi. (vt Joonis 9, 10)

```
try {
    return connection.start()
        .then(() => {
            const state = storeAPI.getState();
            return connection?.send('lisaGruppidesse',
                state.menetlused.menetlused[0].asjaObjektId.toString(),
                selectors.profiilid.selectIsikukood(state)
            )
        }).catch((err) => alert(err));
}
```

Joonis 9. Ühenduse loomine ja gruppidesse lisamine

```
      if (action.type === 'ESITLUS_LOPETA') {
          connection.stop().then(() => {
              store.dispatch(actions.session.signalrConnection(false));
          })
              .catch((err) => alert(err));
      }
```

Joonis 10. SignalR ühenduse lõpetamine

Sünkroniseeriva info vastu võtmine toimub samuti vahetarkvara siseselt, seda käivitab tagarakendus saates välja kindla nimega meetodile andmed. Kliendi rakendus võtab päringu vastu ning lisab saadud info Redux andmekogusse kutsudes välja vastava tegevuse. (vt Joonis 11)

```
      connection.on('esitlusUuendatud', (data: Esitlus) => {
          storeAPI.dispatch({type: 'ESITLUS_RECEIVE', esitlus: data});
      });
```

Joonis 11. Esitluse info vastuvõtmine kliendi rakenduses

3.3.1 Laisa laadimise loogika andmete laadimisel

Kliendirakenduses kasutatakse laisa laadimise (*lazy loading*) loogikat andmete laadimisel, ehk andmed laetakse alles siis kui neid on vaja, et säästa aega rakenduse esmase laadimise pealt. Selle rakendamiseks on Digitoimiku tiimis kasutusel laetavate andmeolemite peal staatuse väli, kuhu omandatakse väärtusi „*idle*“, „*queued*“, „*success*“ ja „*failed*“. Esimese haagi (*hook*), mida nimetatakse *use-hook*'iks, eesmärk on andmeid filtreerida ja sorteerida vastavalt vajadusele ning siis komponendile kasutada anda. (vt Joonis 12) See haak tagastab ka staatuse, mille abil kuvatakse Digitoimikus hetkel laadimisjärgus olevaid komponente.

```
const selector = createSelector([
        (state:RootState) => state.failid.failid,
        (state:RootState) => state.toimikud.toimikud,
        (state:RootState) => state.ui.rakendaFiltreid,
        (state:RootState) => state.esitlusjarjestus,
    ], (
        failid: Fail[],
        toimikud: Toimik[],
        rakendaFiltreid: boolean,
        esitlusjarjestuseToimik: Esitlusjarjestus
    ) => failid
        .filter(fail => filter(fail,
                               rakendaFiltreid,
                               esitlusjarjestuseToimik))
        .sort((a: Fail, b: Fail) => sort(a, b, esitlusjarjestuseToimik))
);
```

Joonis 12. useEsitlusjarjestuseFailid haagi sees andmete pärimine Redux andmekogust, filtreerimine ja sorteerimine

Teine haak, *use-load-hook,* hoolitseb andmete pärimise eest tagarakendusest, seal on peamiselt lasti (*payload*) ettevalmistamine õigete andmetega, andmete pärimise päringu saatmine ja vastusest tulnud andmete õigetele Reduxi tegevustele (*action*) õigel kujul edastamine. Staatus on andmeolemil esialgu olekus „*idle*“, kui mõni komponent pärib andmeid *use-hook*'i käest siis sätitakse staatus olekusse „*queued*“, mis käivitab *use-load-hook*'i ning seejärel muutub olemi staatus olekusse „*success*“, kui päring oli edukas ning „*failed*“, kui tekkis mõni viga. (vt Joonis 13)

```
const dispatch = useAppDispatch();
const fetch = useFetch();
const key = useSessionKey();

const toimikuteStaatus = useAppSelector(state => state.toimikud.staatus);
const staatus = useAppSelector(state => state.esitlusjarjestus.staatus);

useEffect(() => {
    if (toimikuteStaatus === 'succeeded' && staatus === 'queued') {
        dispatch(action.esitlusjarjestus.loading());
        fetch<Esitlsjarjestus>(´api/${key}/esitlusjarjestus´)
            .then(response => {
                if (response.ok && response.data !== undefined) {
                    dispatch(
                       actions.esitlusjarjestus.succeeded(response.data));
                } else {
                    dispatch(actions.esitlusjarjestus.failed());
                }
            });
    }
}, [dispatch, fetch, key, toimikuteStaatus, staatus]);
```

Joonis 13. useLoadEsitlusjarjestus haagi sisu, andmete laadimine ja andmekogusse salvestamise käivitamine

Andmete pärimise järel on järgmine suurem osa nende üle süsteemi kasutamine. Selle jaoks on Digitoimiku tiimis kasutusel Redux teek, mis võimaldab Flux arhitektuurilist mustrit kasutades mugavalt uuendada rakenduse ülest olekut vaid nendes komponentides, kus seda teha on vaja, selle läbi vajavad vaid need komponendid ka taas joonistamist, mis antud oleku muutusest on mõjutatud. Flux muster ise on üles ehitatud lihtsal kontseptsioonil- andmed liiguvad vaid ühesuunaliselt. Andmed läbivad sellist mustrit järgides läbi järgmised programmikoodi osad: andmekogu (*store*), dispetšer (*dispatcher*), vaated/komponendid (*views*/*components*) ja tegevused (*actions*). [12]

4 Töö tulemus

Töö praktiline osaga tegeles autor aktiivselt 2023 aasta veebruari kuust kuni sama aasta novembri kuuni ehk umbes 8 kuud. Selle projekti venitasid pikaks peamiselt kaks mõjutegurit, millega autor õppetöökäigus varasemalt kokku ei puutunud. Esiteks muutusid töö käigus mõningal määral ärinõuded, mis nõudis ka koodis muudatusi, mis omakorda osutusid üsnagi tüütuks. Teiseks selgus, et kerimise info, märgendite lisamine ning kerimise oleku näitamine sisukorras üle komplektide vajas täiendust, seega tuli ka need tööd autoril ette võtta, et esitlusvaate töödega jätkata. Selle täienduse tegemine võttis kokku umbes kaks kuud, sest ümber tuli teha unikaalsete identifikaatorite määramise süsteem, mis oli varasemalt loodud ilma teadmiseta, et komplektide raames võib ühes menetluses eksisteerida samu materjale dubleeritud kujul. Autorile olid mõningal määral ootamatud ärinõude muudatused peale arenduse alustamist ning ühilduvusprobleemid lisarakenduse ja rakenduse põhiosa andmemudelite vahel.

4.1 Töö vastavus püstitatud eesmärkidele autori hinnangul

Autor on töö seisuga pigem rahul, kontseptsiooni tõestus sai loodud, selle põhjal on tulevikus kindlasti lihtsam arendustööga jätkata, olenemata sellest kas tööga jätkab süsteemi arendust autor ise või mõni teine arendaja. Probleemid, mis arenduse vältel tekkisid on heaks õppetunniks sellele, milline võib arendaja töö olla ning kuidas aja hea planeerimine ei pruugi alati päästa üle tähtaegade minemisest, sest tiimis töötades ei sõltu arendaja töö kiirus vaid temast endast. Olgugi et osa nõuetes märgitud funktsionaalsusest jäi rakendamata on autor arvamusel, et töö täitis enda eesmärki ning tulemus on arvestatav.

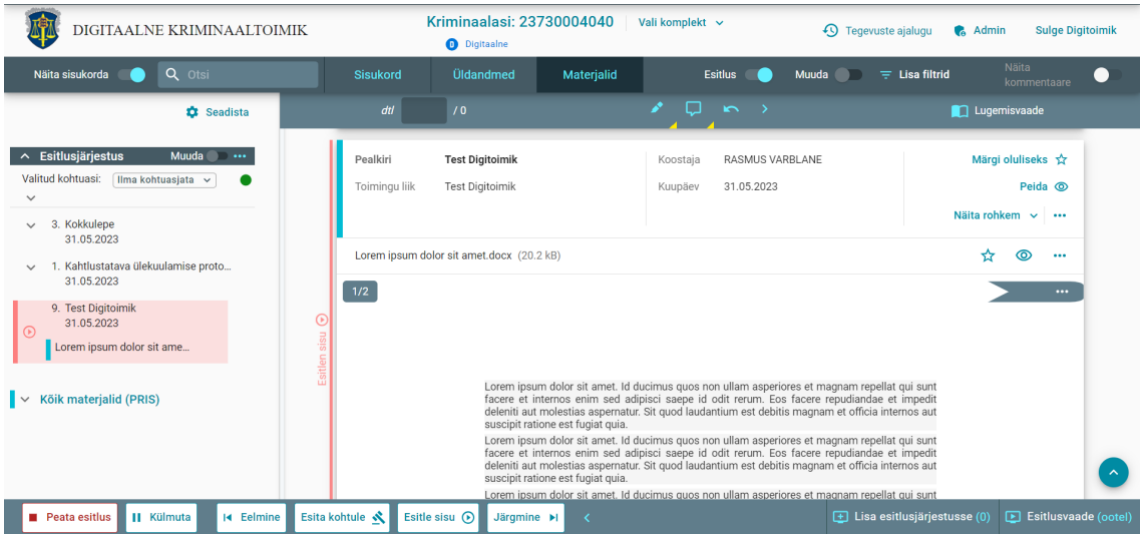
4.2 Testimise tagasiside

Töö testimisel oli autoril plaan kasutada reaalseid kasutajaid prokuratuurist, kes saaksid süsteemi kasutada ning läbi kogemuse anda tagasisidet, kui hästi sai probleem lahendatud. Kahjuks tulid prokuratuuri poolsel tellijal muud kohustused vahele ning töö autor leidis, et tiimi sisese testija poolt tagasiside küsimine on parem lahendus. Prokuröri enda kohustused ei kannata lisa töö tõttu ning tagasiside kvaliteet on samuti parem, kui seda ei pea kiirustades andma.

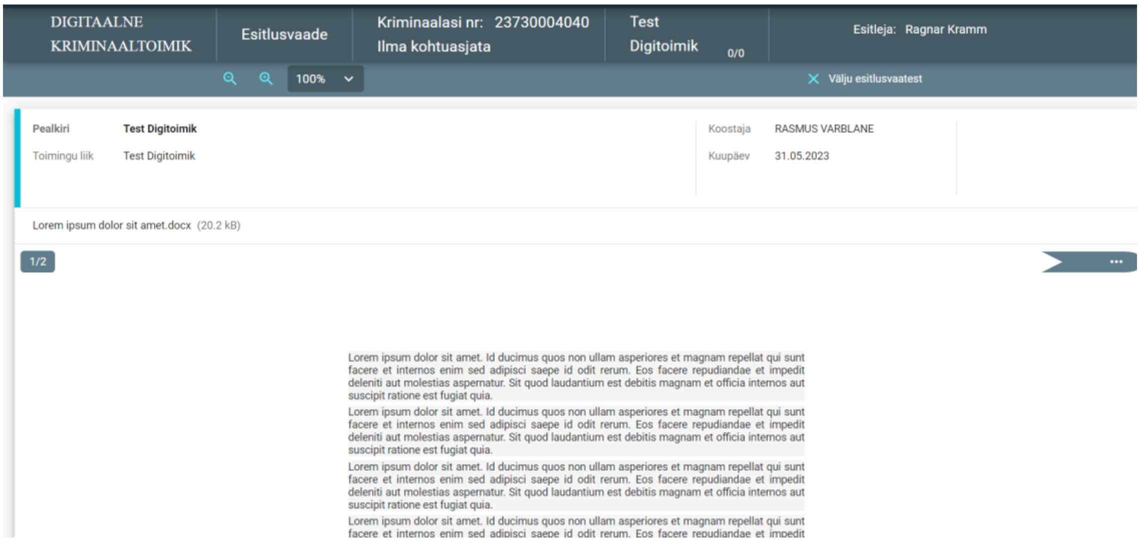
Testijale sai autori poolt antud lisaks nõuete kontrollimisele ka mõned küsimused, et saada tagasisidet kasutajakogemuse ning süsteemi mõistetavuse kohta. Need küsimused on välja toodud siin:

- Enne nõuetega tutvumist, kui esitlusvaade avada, kui hästi on aru saada, mida selles vaates teha saab?
- Kuidas hindad süsteemi osade disaini intuitiivsust?
- Kui vaadata süsteemi osade disaini, kuidas sobitub see varem loodud rakenduse osadega?
- Kui vaadeldes rakendust üldiselt, mis paistab silma kui üleliigne või ei leia potentsiaalselt kasutust?

Neile küsimustele saadud vastused võttis autor kokku järgmiselt. Süsteemi üldine töö on testijale teada, seega ei andnud esimene küsimus täielikult soovitud efekti, kuid erinevate rakenduse osade kohta leidis testija, et üldiselt on peale vaadates arusaadav, mis eesmärki iga komponent omab. Ainuke veidi segadust tekitav osa oli esitluse juhtimise nupud jaluses- mis vahe on peata, külmuta ja lõpeta nuppudel, see ei olnud rakendusele otsa vaadates koheselt arusaadav. Disain on mugav, kõik süsteemi peamised funktsionaalsed osad on eristatavad ning lihtsasti kasutatavad. Loodud rakenduse osa sobitub hästi üldise rakenduse disaini ja paigutusega, midagi häirivat või ebasobivalt silmatorkavat ei tähelda. (vt Joonis 14, 15)



Joonis 14. Esitlusrežiim laiendatud jaluse nuppudega



Joonis 15. Esitlusvaade

4.2.1 Funktsionaalsete nõuete tagasiside

Esitlusjärjestuses materjalide valimine, lisamine ja eemaldamine töötab nagu nõudes mainitud, materjalide ümberjärjestamise funktsioon ei ole töö testimise hetkel rakendatud.

Materjalide märgendamine ja kommenteerimine on võimalik sarnaselt materjalidele, mis ei ole lisatud esitlusjärjestusse. Esitlusvaates märgendeid ja kommentaare välja ei filtreerita, samuti ei ole loodud võimalust luua vaid esitlusvaate jaoks eksklusiivseid kommentaare ning märgendusi.

Materjalide esitamine esitlusvaates töötab prototüübina. Materjalide olekut on võimalik muuta ja see kajastub esitlusvaates, materjalide vahel navigeerimine on samuti esitlusvaatesse edastatud. Nagu nõudes märgitud, siis puudub kohtupoolne osa sellest protsessist, esitluse esitaja kontrollib materjalide olekuid ise.

Sünkroniseeriva informatsioonina on veel saadetud kerimise info, mis esitlusvaates kerib kasutaja vaate ka vastavale kaugusele, et materjal oleks täpselt sellises mahus näha nagu esitluse esitaja on ette näinud. Jälgijate tähelepanu juhtimiseks mõeldud märgendite märkimise võimalus ei ole rakendatud nagu eelnevalt mainitud, seega on ka see osa nõudest jäänud rakendamata.

4.2.2 Mittefunktsionaalsete nõuete tagasiside

Turvalisus on süsteemis tagatud läbi E-Toimiku, kust kõik andmed päritakse, kui kasutajal pole õigust näha esitluses olevaid materjale E-Toimiku kontekstis, siis ei ole neid võimalik näha ka Digitoimikus. Menetlusega mitte seotud isikutel ei ole võimalik Digitoimikusse siseneda, seega pole neil võimalik näha esitlusega seotud materjale. Lisafunktsionaalsus vastab ka Digitoimiku turbetasemetele.

Info liikumine esitluse esitaja ja vaatleja vahel on kasutaja jaoks piltlikult vahetu. Testimise käigus testiti kerimise info liikumise kiirust. Kerimise kordaja leidmisel lisati ajatempel ning kliendi juures kerimise teostamisel võrreldi salvestatud ajatemplit süsteemi hetke ajaga, et saada kerimise info liikumisele kulunud aeg. (vt Tabel 3)

Tabel 3. Kerimise andmete saatmise kiiruse testimise tulemused

Kerimise saatmise test nr	**Tulemus millisekundites**
1	489
2	511
3	478
4	494
5	482
6	503
7	496
Keskmine	493.3

4.3 Planeeritavad jätkutööd

Ühel hetkel, kui Digitoimiku tiimis saavad prioriteetsed tööd tehtud ning arendusressurssi on võimalik tiimisiseselt jaotada vastavalt soovidele, mitte vajadusele, siis on autoril plaan jätkata esitlusvaate arendamisega. Jätkutöödest esimene oleks tõenäoliselt kohtunike poolse osa välja töötamine, sest hetkel ei ole veel autoril head arusaama, kuidas seda täpsemalt luua ja milline oleks andmete liikumine täpne liikumine. Suurim erinevus selle uue osa lisamise juures eelnevatega on see, et seni ei ole sellist otse Digitoimiku baasi läbi erinevate menetluste vahelist suhtlust toimunud nagu kohtunike materjalide vastuvõtmine nõuaks. Seejärel on eesmärk seni ka selle töö raames puudulikud osad

juurde lisada nagu materjalide esitlusjärjestuses ümberjärjestamine ning eriotstarbelised märgendid ja kommentaarid lisada esitlusrežiimi tööriistaribale. Järgneva tegevusena plaanib autor ette võtta muude vähem funktsionaalsete täienduste tegemise, et terviklik esitlusvaade oleks valmis. Viimaks on plaanis üle vaadata süsteemi jõudlus ning teha optimeerivaid töid, kus see võimalik on.

5 Kokkuvõte

Käesoleva lõputöö eesmärk oli luua kontseptsiooni tõestus Digitoimiku materjalide esitamise süsteemile, kasutades reaalajas andmete sünkroniseerimise võimaldamiseks WebSocket protokolli tehnoloogiat.

Lõputöö raames loodi lisafunktsionaalsus Digitoimiku rakendusele, mis annab menetluse osalistele võimaluse luua esitlusrežiimis esitlusjärjestuse ja seda modifitseerida. Esitlusvaatesse on võimalik saata esitlemiseks materjale, nende olekut kontrollida ning hoida vaatlejate tähelepanu materjali õige osa juures läbi sünkroniseeritud lehekülje kerimise. Esitlust on võimalik teha nii individuaalselt enda masina siseselt kui ka mitme kasutaja üleselt. Lisafunktsionaalsuse tagarakendus on loodud C# programmeerimiskeeles kasutades .NET raamistikku ning kliendi rakendus TypeScriptis kasutades React raamistikku, reaalajas andmeedastus on rakendatud kasutades WebSocketi tehnoloogial põhinevat SignalR teeki.

Töös kehtestatud nõuded on paika pandud Digitoimiku tiimijuhi, analüütiku ning prokuratuuri poolse tellija koostööl, millele tugines autori arendustöö, samuti on lisafunktsionaalsuse disain loodud Digitoimiku analüütiku poolt. Enamus olulised nõuded suudeti kontseptsiooni tõestuse loomiseks rakendada, kuid mõned süsteemi jaoks vähem kriitilised funktsioonid jäid ajapuuduse tõttu rakendamata. Süsteemi on plaanis edasi arendada ning täiendada, et see vastaks tellija nõuetele ning oleks kasutajatele väärtust loov tööriist.

Kasutatud kirjandus

[1] Registrite ja Infosüsteemide Keskus, „Mis on digitoimik?“, [Võrgumaterjal]. Loetud aadressil: https://etoimik.rik.ee. [Kasutatud 01.10.2023].

[2] Riigi Infosüsteemi Amet, „ISKE“, [Võrgumaterjal]. Loetud aadressil: https://eits.ria.ee/et/avalehe-menueue/iske/. [Kasutatud 07.12.2023].

[3] Riigi Infosüsteemi Amet, „ISKE Rakendusjuhend“, 30.01.2017. [Võrgumaterjal]. Loetud aadressil: https://www.ria.ee/media/822/download. [Kasutatud 07.12.2023].

[4] MDN contributors, „An overview of HTTP“, Mozilla Foundation, 15 05 2023. [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. [Kasutatud 23.09.2023].

[5] M. Pekarsky, „WebSockets for fun and profit“, Stackoverflow Blog, 18.12.2019. [Võrgumaterjal]. Loetud aadressil: https://stackoverflow.blog/2019/12/18/websockets-for-fun-and-profit/. [Kasutatud 27.09.2023].

[6] I. Fette ja A. Melnikov, „The WebSocket Protocol“, Internet Engineering Task Force, 12.2011. [Võrgumaterjal]. Loetud aadressil: https://www.rfc-editor.org/rfc/rfc6455. [Kasutatud 20.09.2023].

[7] PubNub, „What is Long Polling?“, [Võrgumaterjal]. Loetud aadressil: https://www.pubnub.com/guides/long-polling/#h-0. [Kasutatud 27.09.2023].

[8] MDN contributors, „Server-sent events“, Mozilla Foundation, [Võrgumaterjal]. Loetud aadressil: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events. [Kasutatud 27.09.2023].

[9] Alby, „Socket.IO alternatives: Top 5 competitors to consider in 2023“, [Võrgumaterjal]. Loetud aadressil: https://ably.com/topic/socketio-alternatives. [Kasutatud 01.10.2023].

[10] R. Appel ja K. Griffin, „Use hubs in SignalR for ASP.NET Core“, 25.02.2023. [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/aspnet/core/signalr/hubs?view=aspnetcore-8.0. [Kasutatud 29.11.2023].

[11] L. Amadi, „The Flux Architectural Pattern Of Client-side Development“, Medium, 17.05.2020. [Võrgumaterjal]. Loetud aadressil: https://medium.com/@w3bh4ck/the-flux-architecture-pattern-for-frontend-development-1f2dae32b789. [Kasutatud 29.11.2023].

[12] M. Beschokov, „WebSocket Protocol“, Wallarm, 28.10.2021. [Võrgumaterjal]. Loetud aadressil: https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is. [Kasutatud 27.09.2023].

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

Mina, Ragnar Kramm

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Interaktiivse esitlusvaate loomine Digitoimikule, kasutades WebSocket protokolli", mille juhendaja on Jaanus Pöial
   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.

04.01.2024

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.