TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mikkel Paat 213223IADB

Rakendus otseülekannete videosalvestiste haldamiseks

Bakalaureusetöö

Juhendaja: Märt Kalmo
MSc

Tallinn 2025

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: Mikkel Paat

06.01.2025

Annotatsioon

Käesoleva lõputöö ajendiks on vajadus rakenduse järele, mis võimaldaks seostada otseülekannete videosalvestistega erinevaid metaandmeid ning salvestisi nende põhjal otsida. Töö eesmärgiks on analüüsi kaudu taolise rakenduse realiseerimiseks sobivaima lahenduse leidmine ning analüüsi praktiliselt valideerimiseks prototüüprakenduse arendamine.

Analüüsi käigus esmalt vaadeldakse olemasolevaid lahendusi ning fikseeritakse konkreetsed nõuded soovitavale lahendusele. Seejärel analüüsitakse erinevaid võimalusi metaandmete videofailidega seostamiseks ning kaalutakse, kas või millisel määral antud meetodeid saaks kasutada rakenduse peamise andmetalletusmeetodina. Järgnevalt valitakse välja rakenduse prototüübi arendamiseks kasutatavad tehnoloogiad ning kavandatakse rakenduse andmemudel. Teostuse peatükis kirjeldatakse arendusprotsessi olulisemaid aspekte ja otsuseid.

Lõputöö tulemuseks on videofailidega metaandmete salvestamise meetodite analüüs ning analüüsi põhjal arendatud veebirakendus, mis võimaldab videosalvestisi soovitud viisil hallata, kasutades analüüsi käigus leitud metaandmete salvestusmeetodeid muu tarkvaraga koostalitlusvõime parendamiseks ning failide taastuvastamiseks pärast nende ümbernimetamist või liigutamist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 7 peatükki, 4 joonist, 0 tabelit.

Abstract

Application for Managing Video Recordings of Livestreams

This thesis addresses a need for an application to associate metadata with video recordings of livestreams and filter them accordingly. The goal of the thesis is to determine the best approach to creating such an application and to validate this analysis by developing a prototype application.

The thesis begins by introducing the background of the problem at hand and describing the desired properties of a solution. Existing options for managing video files and their metadata are compared and determined unsuitable. Exact requirements are defined for the application.

Multiple methods for associating metadata with video files are investigated in order to determine whether or to what extent they would be usable as the main data persistence method of the application. None of the compared methods are deemed suitable to be the main storage method, but extended file attributes and the Matroska file format's metadata support are chosen as supplementary methods. An approach for synchronizing file system state with the application's database is planned, followed by choosing the technologies to be used in the application's prototype and designing a database model for it. After the analysis, the application's architecture and relevant aspects of the development process are described.

The results of the thesis are a thorough analysis of the different ways of associating metadata with video files and a web application prototype developed based on this analysis. The application uses supplementary file metadata persistence methods to identify moved or renamed files and allow some of the managed metadata to be compatible with other software.

The thesis is in Estonian and contains 40 pages of text, 7 chapters, 4 figures, 0 tables.

Lühendite ja mõistete sõnastik

Laiendatud failiatribuudid	*Extended File Attributes* – failisüsteemi toetus failiga nimi-väärtus paaride seostamiseks.
NTFS	*New Technology File System* – peamine failisüsteem Windows'i operatsioonisüsteemidel.
Alternatiivsed andmevood	*Alternative Data Streams* – laiendatud atribuutide alternatiiv NTFS failisüsteemis.
MP4	MPEG-4 standardi 14. osas defineeritud, tänapäeval levinuim videofailiformaat.
XMP	*Extensible Metadata Platform ehk* laiendatav metaandmete platvorm – Adobe loodud tehnoloogia failidesse metaandmete salvestamiseks.
ISO	*International Organization for Standardization* ehk rahvusvaheline standardite organisatsioon.
MKVToolNix	Matroska failide metaandmete haldamise utiliitide komplekt.
Sidecar failid	Metaandmete talletamiseks põhifaili kõrvale salvestatavad eraldi failid.
SQLite	Relatsiooniline andmebaasihaldussüsteem, mis talletab andmeid ühe faili sees ning on kasutatav otse rakendusest ilma eraldi andmebaasiserveri programmita.
ICU	*International Components for Unicode* – SQLite'i laiendus, mis lisab SQLite'i funktsioonidele rahvusvaheliste tähemärkide toetuse.
UUID	*Universally Unique IDentifier* ehk universaalselt unikaalne identifikaator.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Internetti videootseülekannete tegemise käigus tekib otseülekannetest palju salvestisi. Salvestistega kuulub kokku hulk metaandmeid, näiteks algusaeg, pealkiri, võtmesõnad, osalenud isikud või tehtud tegevused. Hetkel puudub rahuldav viis taoliste metaandmete videofailidega seostamiseks ning nende põhjal failide otsimiseks. See põhjustab ajakadu ning muudab salvestiste haldamise ja jagamise ebamugavaks. Näiteks salvestiste üleslaadimisel internetti tuleb informatsioon, mida soovitakse üleslaadimiseks kasutataval veebilehel kajastada, mitmest eri kohast kokku koguda ja käsitsi kasutatavale veebilehele sisestada, mis on piisavalt aeganõudev ja tülikas, et paljud salvestised jäävadki avalikustamata.

Antud probleemi lahendamiseks võiks leiduda rakendus, mis võimaldab otseülekannete autoril hallata tehtud otseülekannete videosalvestisi, täiendada neid vajalike metaandmetega ning kasutada talletatud andmeid salvestiste otsimisel. Sealjuures oleks eelistatav, et salvestatud metaandmed poleks mõistetavad ainult ühele rakendusele ning et nende seos failiga ei oleks habras faili liigutamise suhtes. Rakendus peaks ka olema kasutatav üle veebiliidese, et seda saaks kasutada paigaldatuna võrgumäluserverile.

Lõputöö eesmärk on analüüsida viise metaandmete seostamiseks videofailidega, töötada välja sobivaim viis soovitud metaandmeid haldava rakenduse arendamiseks ning analüüsi valideerimiseks selle põhjal rakenduse prototüüp arendada. Selleks antakse esmalt ülevaade probleemi taustast ja olemasolevatest lahendustest ning määratakse nõuded soovitava lahenduse jaoks. Analüüsi käigus valitakse sobivad meetodid, tehnoloogiad ja lähenemised lahenduse teostamiseks. Seejärel arendatakse analüüsi põhjal lahenduse prototüüp, kirjeldatakse arenduse olulisemaid aspekte teostuse peatükis ning viimaks kontrollitakse tulemuste vastavust eesmärgile ja nõuetele.

2 Probleemi kirjeldus ja töö eesmärk

Käesolevas peatükis tutvustatakse käsitletava probleemi tausta ning püsitatakse töö eesmärk.

2.1 Taust

Internetti otseülekannete tegemine on ajaviide, millega tegelevad tänapäeval paljud inimesed hobi või isegi töö korras. See hõlmab video- ja audiovoogude edastamist reaalajas otseülekande tegija arvutist või muust sobivast seadmest otseülekannete vaatamiseks mõeldud platvormile või platvormidele, mille kaudu huvilised ülekantavat materjali vaadata ja kuulata saavad. Populaarseimad selleks otstarbeks kasutatavad platvormid on Twitch ja YouTube, kuhu ülekande sisu edastamiseks kasutatakse enamasti kas OBS Studio või StreamLabs OBS tarkvara. Tavaliselt kestavad ülekanded mitu tundi. Ülekannete tegemise sagedus varieerub regulaarsetest igapäevastest ülekannetest mõne juhusliku ülekandeni kuus, olenevalt ülekannete autori eelistustest ja võimalustest. Ülekannetes tehtavad tegevused on mitmekesised, hõlmates näiteks videomängude mängimist, kunstiteose joonistamist või programmeerimist.

Huvi otseülekande sisu vastu ei kao ülekande lõpuga. Ülekande salvestist või selle osa võidakse soovida vaadata meelelahutuse või info saamise eesmärgil või kasutada montaažis. Samuti võib otseülekannete autor soovida näha ülevaadet tehtud otseülekannetest erinevate filtreerimis- ja rühmitamistingimuste põhjal. Salvestisi saab ülekande autor videofailidena alla laadida otseülekande platvormilt või otseülekande tegemise ajal kasutatavas tarkvaras otse enda seadmesse salvestada. Kui otseülekandeid teha piisavalt tihti, kasvab salvestiste hulk kiiresti. Näiteks käesoleva lõputöö autoril, kes suhteliselt harva otseülekandeid teeb, on aastatega kogunenud üle 350 salvestise, mille kestus on reeglina vahemikus 2 kuni 6 tundi.

2.2 Seosed ja metaandmed

Otseülekannete ja nende salvestistega kuulub kokku hulk metaandmeid ja seoseid, mille talletamine ja kasutamine filtreerimisel või salvestiste vaatamisel on soovitav. Mõned tüüpilised metaandmed on näiteks otseülekande pealkiri, algusaeg, kategooriad ning otseülekandes osalenud isikud.

Otseülekannete haldamisel on peamiseks huviobjektiks ülekannetes tehtud tegevused. Ühes otseülekandes võidakse tegeleda mitme erineva tegevusega ning samuti võidakse ühe tegevusega tegeleda mitme ülekande jooksul. Esimesel juhul võidakse soovida salvestada infot selle kohta, mis ajal tegevus kindlas otseülekandes lõppes või algas, kui ühe otseülekande jooksul vahetati tegevust. Nii otseülekandeid kui tegevusi oleks kasulik seostada võtmesõnadega, et hiljem nende põhjal ülekandeid otsida. Näiteks võivad võtmesõnadeks olla „videomäng" (üldine tegevuse kategooria), „Celeste" (konkreetse mängu nimi) ning „platvormimäng" (mängu žanr). Ka võidakse soovida otseülekandeid grupeerida neis tehtud tegevuse põhjal, eristades eri ajal eri kontekstis tehtud sarnaseid tegevusi. Näiteks kujutagem ette olukorda, kus otseülekannete autor mängis viie otseülekande jooksul läbi kindla videomängu. Millalgi hiljem mängis autor kolme otseülekande jooksul taas läbi sama videomängu. Sellisel juhul võib ta soovida liigitada need kaks läbimängimist eraldi tegevuste alla, kuigi mõlemas tehtud tegevus on täpselt samasugune.

Ühe otseülekandega võib olla seotud mitu salvestist. Tegemist võib olla näiteks eri kvaliteedis ja formaadis versioonidega kogu otseülekandest või siis võib otseülekanne olla jagatud mitmeks salvestiseks otseülekande jooksul toimunud tehnilise probleemi tõttu. Reeglina on soovitav metaandmeid seostada otseülekannetega, mitte salvestiste failide endiga.

Tihti laetakse otseülekannete videosalvestised üles YouTube'i või muule sarnasele videoplatvormile, et huvilised neid järgi saaksid vaadata. Siinkohal oleks kasulik, kui saaks otseülekannetele, tegevustele ja isikutele lisada märkmeid ja veebiaadresse, et neile ja ülejäänud otseülekandega seotud infole pääseks ühest kohast mugavalt ligi sihtveebiplatvormile info sisestamiseks.

2.3 Eesmärk

Otseülekannete videosalvestiste haldamine failidena tavalises failisüsteemis ei võimalda neid piisavalt detailselt kategoriseerida, grupeerida ja erinevate metaandmetega täiendada. Kuna võtmesõnade, seoste ja muude metaandmete salvestamine vajalikul määral pole võimalik, on piiratud ka võimalused videoid nende põhjal filtreerida ja sorteerida, mis põhjustab ajakadu videote otsimisel ja raskendab salvestiste kogust rahuldava ülevaate saamist.

Käesoleva lõputöö eesmärk on töötada välja optimaalne viis püstitatud probleemi lahendava tarkvaraprojekti arendamiseks, mis võimaldab otseülekannete autoril hallata tehtud otseülekannete videosalvestisi, täiendada neid vajalike metaandmetega ning kasutada talletatud andmeid salvestiste otsimisel ja filtreerimisel. Selleks leida sobivaim viis videofailide seostamiseks erinevate metaandmetega ning antud meetodist lähtudes kavandada sobivaim kombinatsioon tehnoloogiatest, andmemudelist ja muudest asjakohastest tehnilistest valikutest. Tulemuseks saadud teoreetilise aluse valideerimiseks arendada selle põhjal valmis rakenduse prototüüp.

3 Olemasolevad lahendused

Kuigi on olemas erinevaid faili- ja meediahalduslahendusi, ei kata need kirjeldatud probleemi kõiki aspekte, lahendavad veidi teistsugust probleemi või ei vasta nende kasutus- ja paigaldusvõimalused vajalikele tingimustele.

3.1 Pilvepõhised lahendused

Paljud videoressursside haldamise lahendused, nagu AdmiralCloud, Cloudinary või Canto, on mõeldud firmadele meediakogude haldamiseks ja levitamiseks. Nende puhul on tegemist teenustega, mis on mõeldud lahendama teistsugust probleemi teistsuguste sihtgruppide jaoks ning seega on need käesoleva probleemi lahendamiseks liiga keerukad lahendused, mille ebavajalikud lisafunktsionaalsused võivad osutuda segadust tekitavateks. Leidub ka rohkem tavakasutajatele suunatud failihaldusteenuseid, nagu Google Drive ja OneDrive, millel on vähem ebavajalikke aspekte.

Üldiselt võimaldavad sellised pilveteenused faile sildistada, filtreerida ja sortreerida ning mõnikord ka otse teistele videoplatvormidele üles laadida, mis võiks olla mingil määral adekvaatne lahendus käsitletavale probleemile. Samas puuduvad neil mugavused, mida konkreetselt otseülekannete videosalvestiste haldamiseks mõeldud rakendus võiks pakkuda, nagu näiteks videotes ajavahemike seostamine tegevustega, automaatne andmete pärimine otseülekannete platvormidelt või kasutajaliides, mis on loodud otseülekannete haldamisega seotud tegevusi silmas pidades.

Pilvepõhiste lahenduste peamine puudujääk tuleneb asjaolust, et otseülekannete salvestiste failid on mahukad. Näiteks käesoleva lõputöö autori salvestiste kogumaht hetkel on ligikaudu 4 terabaiti. Seetõttu läheks taoliste teenuste kasutamine salvestiste haldamiseks äärmiselt kulukaks, mis pole hobi korras otseülekannete tegijale aktsepteeritav. Mitmel pilveteenusel on ka mahupiirangud, mida mitmeterabaidine andmekogu ületaks, muutes nende kasutamise võimatuks või vajades eritingimuste kokkuleppimist. Lisaks tuleks kulutada aega salvestiste üleslaadimisele kasutatavale

pilveplatvormile ning seejärel sõltuks salvestiste kogu haldaja täielikult konkreetsest teenusepakkujast.

3.2 Lokaalsed lahendused

Pilvelahendustest praktilisemad kandidaadid on rakendused, mida kasutaja saab enda arvutis või serveris käivitada.

3.2.1 Nextcloud

Nextcloud on avatud lähtekoodiga rakendus, mis pakub Google Drive'i ja teiste pilvepõhiste failihalduslahendustega sarnast funktsionaalsust. Kuna Nextcloud'i on võimalik käitada kohtvõrgus salvestiste kogu haldaja serveril, ei kehti selle puhul pilvepõhiste lahenduste peamised puudujäägid: ei oleks vaja maksta suuri tellimustasusid kolmandatele osapooltele, failide serverisse toimetamise kiirus poleks piiratud internetikiirusega ning ei peaks muretsema mahupiirangute üle. Ülejäänud peatükis 3.1 väljatoodud puudujäägid kehtivad siiski ka Nextcloud'i kohta. Samuti, kuna Nextcloud on üldotstarbeline paljude funktsionaalsustega rakendus, võib selle haldamine ja käitamine osutuda tülikamaks, kui spetsiifiliselt otseülekannete salvestiste haldamiseks mõeldud lahenduse haldamine ja käitamine.

3.2.2 Fast Video Cataloger

Fast Video Cataloger on Windows'i operatsioonisüsteemile mõeldud rakendus, mis võimaldab kasutajal lisada ja muuta videofailide metaandmeid ning nende põhjal videoid otsida. Sellel on põhjalikud kategoriseerimisvõimalused, kaasa arvatud isikute videotega seostamise funktsionaalsus ning võtmesõnade seostamine kindlate stseenidega videotes.

Fast Video Cataloger on kõigist võrreldud lahendustest parim kandidaat, kuid see siiski ei võimalda piisava detailsuse ja paindlikkusega kirjeldada tegevuste, videote, isikute ja muude mõistete vahelisi seoseid. Rakenduse funktsionaalsust saab laiendada C#-skriptidega ning selle kasutajaliidest on võimalik ümber seadistada ja täiendada, mille kaudu saaks arvatavasti lisada rakendusele vähemalt osa puuduvast funktsionaalsusest. Ent soovitud funktsionaalsuse lisamine eeldaks peaaegu terve eraldi rakenduse võrra koodi kirjutamist Fast Video Catalogeri laienduste näol, nõudes samas Fast Video

Catalogeri andmemudeli piirangute ümber töötamist. Näiteks ei ühti Fast Video Catalogeri videofailidele orienteeritud ülesehitus eriti hästi ideega salvestiste kogu haldamisest ülekannete ja tegevuste seostamise näol.

Samuti on problemaatiline asjaolu, et Fast Video Cataloger toetab ainult Windows'i operatsioonisüsteemi ning see tuleks paigaldada igasse seadmesse, mille kaudu videote kataloogi hallata soovitakse. Lisaks, kuna Fast Video Cataloger on klientseadmetele mõeldud rakendus, mitte serveril jooksev teenus, piiraks selle kasutamine võimalusi rakendusele lisada pikaajalisi taustatöid vajavaid funktsioone, kuna eeldaks klientseadme töötamist terve protsessi jooksul. Üks funktsionaalsus, mille lisamist see takistaks, on salvestiste üleslaadimine erinevatele veebiplatvormidele. Ka on tegemist suletud lähtekoodiga ja tasulise rakendusega, mis pole küll otsustavad faktorid rakenduse sobivuse osas, kuid kujutavad endast siiski piiranguid, mida ideaalsel lahendusel ei tohiks olla.

4 Analüüs

Käesolevas peatükis fikseeritakse rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Seejärel analüüsitakse lähemalt nõuete täitmiseks vajalikke võimalikke lähenemisi ning tehakse asjakohased tehnilised valikud.

4.1 Nõuete määramine

Loodava lahenduse selgelt määratlemiseks fikseeritakse antud alapeatükis omadused, mis käesoleva lõputöö raames loodaval lahendusel peavad olema.

4.1.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded kirjeldavad tegevusi, mida süsteem töötamise käigus teeb või võimaldab kasutajal teha [1, lk 46–47]. Funktsionaalsete nõuetega defineeritakse äriloogiline funktsionaalsus, mida süsteem peab pakkuma, et sellest kasu oleks.

- Otseülekandeid peab saama seostada neis tehtud tegevustega. Valikuliselt peab olema võimalik määrata ajavahemik, millal otseülekandes tegevus toimus.
- Otseülekandeid ja tegevusi peab saama seostada neis osalenud isikutega.
- Otseülekandeid ja tegevusi peab saama seostada kategooriate ja võtmesõnadega.
- Tegevusi, otseülekandeid ja videofaile peab olema võimalik otsida järgmiste tingimuste alusel:
    - võtmesõnade kombinatsioonid
    - osalenud isikud
    - otseülekande toimumisaeg
- Tegevusi, otseülekandeid ja isikuid peab saama täiendada märkustega.

- Rakenduses peab saama seadistada failikataloogid, milles asuvaid videofaile saab rakenduse kaudu hallata.
- Videofaile peab saama seostada otseülekannetega.
- Kui faili asukohta muudetakse või faili nime muudetakse, peab rakendus liikunud faili võimalusel seostama eelmisega.
- Rakendus peab toetama .mp4 ja .mkv formaadis failide haldamist.

4.1.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad, kuidas tarkvara oma ülesandeid täitma peab. Mittefunktsionaalsed nõuded on näiteks nõuded tarkvara jõudluse, välisliidese, disaini ja kvaliteedi kohta. [2, lk 293]

- Rakendus peab olema kasutatav üle kohtvõrgu veebilehitseja kaudu.
- Kui võimalik, peaks rakendus andmed talletama viisil, mis ei ole habras failide rakenduse halduse alt väljumise suhtes ning võimaldab andmeid kasutada ka teistel programmidel.

4.2 Võimalused metaandmete seostamiseks failidega

Loodava süsteemi oluline funktsionaalsus on erinevate metaandmete seostamine failidega. Selleks on vajalik leida meetod ja asukoht metaandmete salvestamiseks. Failid ei pruugi olla ainult loodava süsteemi halduses ning seega võidakse neid liigutada teise asukohta, ümber nimetada, avada välistes programmides või täiesti süsteemi haldusest välja viia. Ideaalselt võiksid rakenduse salvestatavad andmed ja seosed olla failidega seotud viisil, mis püsib failiga koos rakendusest sõltumatult. Samuti oleks eelistatav, kui andmed oleksid failidega seostatud viisil, mida oskavad tõlgendada ja kuvada muud failidega tegelevad programmid.

Kui andmeid hoida vaid rakenduse poolt kasutatavas tüüpilises andmebaasis, viidates failidele näiteks faili pöördusteega, on andmete seos failiga habras ning teada vaid konkreetsele rakendusele. Kui oleks võimalik andmed salvestada otse failide külge või juurde, võiks kaaluda sellise meetodi kasutamist rakenduse andmete peamise

salvestusviisina. Olenemata konkreetsest meetodist oleks sellise lähenemise juures suurimaks puudujäägiks asjaolu, et sellega kaasnevad vältimatult tüüpilised normaliseerimata andmemudeli anomaaliad. Näiteks tuleks otseülekandes osalenud isiku andmed kirjutada iga temaga seotud faili juurde, misjuhul peaks isiku andmete muutmisel muutma isiku andmeid iga temaga seotud faili juures ning võib tekkida olukord, kus erinevate failide juures on sama isiku kohta erinevad andmed. Samuti oleks võimatu kirjeldada otseülekannet, mille jaoks salvestis puudub või isikut, kes pole ühegi failiga seotud. Siiski võib selline lähenemine olla väärt kirjeldatud anomaaliatega tegelemist, kui õnnestub leida meetod, mille ühilduvus teiste programmide ning andmete lähedane seotus failidega kaaluvad üles anomaaliad ja muud puudujäägid. [3]

Et andmete salvestamise meetod sobiks kasutamiseks rakenduse peamise salvestusviisina, peaks see vastama järgmistele tingimustele:

1. Antud meetodiga salvestatud metaandmed püsivad failiga koos ka pärast faili liigutamist, kopeerimist, teise failisüsteemi viimist või muud taolist tegevust.

2. Meetod on kasutatav kõigi failidega ja ei sõltu kindla videofailitüübi funktsionaalsusest.

3. Meetod võimaldab kirjutada suvalisi andmeid, mis pole kasutatava meetodi poolt ette määratud.

4. Meetod võimaldab andmeid kirjutada märkimisväärsete piiranguteta. Näiteks mahupiirang, kasutatavate tähemärkide piirang või korraga kirjutatavate andmete piirang.

5. Meetodiga andmete kirjutamine ei vaja kogu videofaili sisu ümber kirjutamist.

6. Meetodiga kirjutatud andmeid on võimalik faili juurest taas välja lugeda.

7. Meetod võiks võimaldada kirjutada andmeid kujul, mis ühildub ka teiste programmidega, mis faili võivad lugeda.

Kui pole võimalik leida kirjeldatud tingimustele vastavat meetodit, võib uuritavatest meetoditest siiski kasu olla, kuigi need ei sobi rakenduse peamiseks salvestusviisiks. Näiteks, kui meetod vastab tingimustele 1, 5 ja 7, saaks seda kasutada, et salvestada faili

juurde see alamhulk rakenduse andmetest, mida konkreetne meetod võimaldab, et vähemalt osa infost oleks kasutatav faili avamisel teiste programmidega. Või, kui meetod vastab tingimustele 3, 5 ja 6 ning vähemalt mingil määral ka tingimusele 1, saaks seda kasutades lisada rakenduse hallatavatele failidele unikaalse identifikaatori, mille järgi saab rakendus faili liigutamise korral tuvastada, et tegemist on sama failiga.

Järgnevalt analüüsitakse erinevaid meetodeid info sidumiseks failidega.

4.2.1 Laiendatud atribuudid

Failisüsteemi laiendatud atribuudid (ingl. k *extended attributes*) on failide ja failikataloogidega seotud nimi-väärtus paarid [4]. Neid kasutades on võimalik failiga seostada suvalist informatsiooni [5].

Tarkvara koostalitlusvõime eeskirjadega tegeleva organisatsiooni Freedesktop.org loodud ettepanekus on defineeritud mõned laiendatud atribuutide nimed, mida võiksid erinevad rakendused ühiselt kasutada [6]. Näiteks on defineeritud „user.xdg.comment" atribuut kasutaja kirjutatud kommentaari jaoks [6]. Defineeritud on ka 15 atribuuti, mis vastavad Dublin Core™ Metaandmete Initsiatiivi (*Dublin Core™ Metadata Initiative*) põhielementidele [6], [7]. Nende hulgas on näiteks „user.dublincore.creator" kirjeldatava faili autori jaoks ning „user.dublincore.source", kuhu võib panna viite ressursile, millel kirjeldatav fail põhineb [6], [7]. Samuti on mitme tarkvara poolt kasutusel „user.xdg.tags" atribuut võtmesõnade jaoks, kuigi see pole Freedesktop.org standardis defineeritud [5]. Lisaks on võimalik salvestada suvalise nimega atribuute, mis võimaldaks salvestada ka andmeid ja seoseid, millele sobivat olemasolevat atribuuti pole defineeritud. Et vältida kattumist teiste rakenduste poolt defineeritud atribuutidega, mis võib tekitada konflikte atribuudi tähenduse ja formaadi osas, soovitab Freedesktop.org kasutada rakenduse-spetsiifilist nimeruumi prefiksi atribuudi nimes, näiteks kujul „user.minurakendus.osaleja" [6].

Laiendatud atribuudid on failisüsteemi pakutav funktsionaalsus ning seega selle toetus sõltub kasutatavast failisüsteemist. Peamised Linuxi-põhistes operatsioonisüsteemides kasutatavad failisüsteemid, nagu Ext4, Btrfs, ZFS ja XFS, toetavad laiendatud atribuute [5]. NTFS (*New Technology File System*), mis on peamine failisüsteem Windowsi

operatsioonisüsteemidel, võimaldab kasutada alternatiivseid andmevooge (*Alternate Data Streams*) laiendatud atribuutide vastena [5], [8], [9].

Kuigi laiendatud atribuudid püsivad failiga koos faili liigutamisel sama failisüsteemi piires, on nende seos failiga siiski üsna habras. Erinevad failidega tegelevad tööriistad, kaasa arvatud Linuxil laialdaselt failide kopeerimiseks kasutatav „cp" käsk, vajavad eraldi seadistust laiendatud atribuutide säilitamiseks või ei toeta neid üldse [5]. Samuti kaovad atribuudid faili küljest, kui liigutada fail ühest failisüsteemist teise, näiteks Ext4 failisüsteemist NTFS failisüsteemi.

Laiendatud atribuutide suurim eelis on, et informatsioon salvestatakse failisüsteemi tasemel faili külge ilma faili ennast muutmata, see püsib failiga kaasas, kui faili liigutada sama failisüsteemi piires ning andmete salvestamise võimalikkus ja formaat ei sõltu faili enda formaadist. Ent nende kasulikkust piirab oluliselt tõenäosus, et need lähevad kopeerimise, teise failisüsteemi liigutamise või muude toimingute käigus kaduma ning asjaolu, et laiendatud atribuute kasutab väike arv programme. Samuti on neisse salvestatavate andmete maht tihti oluliselt piiratud. Peatüki 4.2 alguses kirjeldatud tingimustest täidavad laiendatud atribuudid tingimusi 2, 3, 5 ja 6 ning piiratud määral ka tingimusi 1 ja 7.

4.2.2 Konkreetsete failiformaatide metaandmed

Olenevalt failiformaadist võib olla võimalik metaandmeid salvestada otse faili sisse. See meetod oleks kindlaim viis tagamaks, et metaandmed püsivad failiga koos, kuna need on otseses mõttes osa failist – täites seega nõuet 1. Samuti, kui metaandmed on salvestatud kasutatava failiformaadi standardi kohaselt, on põhjust eeldada, et tarkvara, mis oskab antud failiformaati lugeda, oskab tõlgendada või vähemalt kuvada ka sellele failiformaadile omaseid metaandmeid, täites nõuet 7.

Antud liiki meetod sõltub täielikult kasutatava failiformaadi võimalustest ning vajaks iga erineva failiformaadi jaoks erinevat implementatsiooni, mistõttu ei saa selline meetod täita nõuet 2. Samuti, kuna metaandmete salvestamiseks tuleb muuta faili enda sisu, võib metaandmete muutmiseks olla vajalik kogu faili sisu ümber kirjutamine, mis võib suurte failide korral olla aeganõudev ning võib vigade esinemisel rikkuda ära kogu faili. Seega on võimalik, kuid mitte kindel, et taoline meetod ei pruugi täita nõuet 5.

Järgnevalt uuritakse lähemalt kahe erineva videofailiformaadi, MP4 ning Matroska metaandmete toetust [10], [11], [12]. Need konkreetsed failiformaadid valiti populaarsete failiformaatide seast, kuna neis formaatides on kõik otseülekannete salvestised käesoleva lõputöö autori arhiivis, mille haldamine on antud lõputöö peamine ajend.

MP4 failiformaadi metaandmed

MP4 failiformaat, mis on defineeritud MPEG-4 standardi 14. osas, on tänapäeval levinuim videofailiformaat [10], [13]. MP4 formaat põhineb firma Apple Computer Inc. loodud QuickTime® formaadil [14, lk 46]. MP4 failides on kõik andmed kapseldatud „kastide" (*box*) sisse, mida vanemates spetsifikatsioonides nimetati ka „aatomiteks" (*atom*) [14, lk 47], [15, lk 178]. Leidub mitmeid programme, mis võimaldavad erinevatesse metaandmete „kastidesse" väärtusi kirjutada. Mainimisväärseimad neist, mida on võimalik programmaatiliselt kasutada, on ExifTool [16], Bento4 mp4tag [17], MP4Box [18], AtomicParsley [19] ning ffmpeg [20]. Enamus metaandmete võtmeid, mille alla on võimalik väärtusi salvestada, on seotud Apple'i iTunes platvormi ja QuickTime® formaadiga [21], [22], [23]. Toetatud metaandmete hulka kuuluvad näiteks pealkiri, kirjeldus, kommentaar, autor, loomiskuupäev ja võtmesõnad [21], [22], [23]. Lisaks muudele andmetele võimaldab ffmpeg lisada videole peatükke, mida võiks näiteks kasutada tegevuste toimumisaegade salvestamiseks [24].

ExifTool võimaldab kirjutada ka struktureeritud informatsiooni XMP kujul [25]. XMP (*Extensible Metadata Platform* ehk laiendatav metaandmete platvorm) on Adobe loodud tehnoloogia failidesse metaandmete salvestamiseks, mis on alates 2012. aastast ka ISO (*International Organization for Standardization* ehk rahvusvaheline standardite organisatsioon [26]) standard [27], [28]. Kuna tegemist on struktureeritud formaadiga, võimaldab see salvestada andmeid komplekssemal kujul kui eelnevalt kirjeldatud võti-väärtus paaridena info salvestamine. Nagu nimigi vihjab, on XMP laiendatav formaat, mis tähendab, et selles defineeritud andmemudelit saab laiendada kolmandate osapoolte poolt defineeritud nimeruumidest pärit omaduste ja seostega või ka ise uusi omadusi ja seoseid defineerida [29, lk 5], [30, lk 20]. Tänu sellele on ExifTool'i kasutades võimalik MP4 failile lisada suvalisi struktureeritud metaandmeid [25], [31]. Ülejäänud vaadeldud programmidest toetab vaid AtomicParsley suvaliste metaandmete kirjutamist,

võimaldades defineerida uusi neljatähemärgilisi võtmeid, millega saab seostada tekstilise väärtuse või välise faili [32].

Kõik kirjeldatud programmid peale MP4Box'i peavad failile metaandmete lisamiseks terve faili läbi töötama ning ümber kopeerima. Suuremahuliste failide korral tähendab see, et iga väiksemagi metaandmete muudatuse tegemine võtab aega mitu minutit. Näiteks võttis autori katsetuse käigus 14 GB suuruse MP4 faili korral ffmpeg'iga metaandmete lisamine aega 2 minutit ja 11 sekundit ning ExifTool'iga 5 minutit ja 52 sekundit. Kuna kavandatav rakendus peab haldama sadu salvestisi, mis on reeglina vähemalt sama mahukad kui 14 GB, pole metaandmete sellisel viisil talletamine praktiline. Eriti problemaatiline oleks see juhul, kui rakenduses muudetakse näiteks isiku nime, kes on osalenud enamuses hallatavatest otseülekannetest – selline väike muudatus võtaks kokku aega vähemalt 5 tundi, eeldusel et metaandmete salvestamiseks kasutatakse ffmpeg'i, keskmine salvestise maht on 14 GB ning muutmist vajavad 150 salvestist. Lisaks vastuvõetamatule ajakulule ohustaks taoline lähenemine ka kasutatava arvuti massmälu eluiga.

Seega oleks ainuke aktsepteeritav variant metaandmete otse MP4 failidesse salvestamiseks MP4Box programmi kasutamine, mis muudab metaandmete kirjutamisel otse originaalfaili. Ent ka MP4Box peab mõnikord metaandmeid lisades kogu faili läbi töötlema. Käesoleva lõputöö autori hüpotees on, et see on vajalik juhul, kui metaandmete salvestamise piirkond eelneb faili alguses video sisule ning lisatavad metaandmed ei mahu sellesse piirkonda ära, mistõttu peab metaandmeid kirjutav tarkvara kogu faili sisu nihutama edasi. Kuna tegemist on pikema protsessiga, mille käigus muudab MP4Box originaalfaili otse, tekib siinkohal lisaks ajakulule oht, et protsess katkeb mingil põhjusel ning fail muutub kasutuskõlbmatuks. MP4Box'iga on ka muid probleeme. Autori katsetuste käigus põhjustas MP4Box'iga muudetud fail mõnikord erroreid nii ExifTool'is kui ka Windows'i failihalduris metaandmete muutmisel. Ka pole MP4Box'i käsurealt kasutades näiliselt võimalik lisada ühe käsklusega mitmele erinevale metaandmevõtmele väärtuseid ning kui defineerida kirjutatavad võti-väärtus paarid eraldi failis, mis MP4Box'ile ette antakse, ei tundu olevat võimalik kasutada mitme erineva võtme väärtustes reavahetusi.

Matroska failiformaadi metaandmed

Matroska on laiendatav, avatud lähtekoodi ja avatud standardiga multimeedia konteineriformaat, mida kasutatakse muuhulgas „.mkv" faililaiendiga videofailides [33]. Matroska formaat toetab metaandmete kirjutamist siltide (*tags*) kujul [34]. Matroska sildid võimaldavad kirjutada struktureeritud metaandmeid, asetades silte teineteise sisse [35]. Defineeritud on hulk ametlikke silte, näiteks sildid kirjelduse, võtmesõnade, salvestuskuupäeva, näitlejate (mida saaks kasutada näiteks otseülekandes osalenud isikute kirjeldamiseks) ja pealkirja jaoks [35]. Võimalik on ka kasutada mitteametlikke, rakenduse või kasutaja poolt defineeritud silte, mida Matroska juhised soovitavad eristada ametlikest siltidest, lisades sildi nime algusesse alakriipsu [35]. Lisaks saab Matroska failile lisada peatükke, mida saaks kasutada otseülekandes tehtud tegevuste seostamiseks ajavahemikega salvestises, et see info oleks kasutatav salvestise vaatamisel meediapleieris [34], [36]. Võimalik on isegi alapeatükkide ning erinevate peatükikomplektide loomine [36], mis võimaldaks paindlikult kirjeldada lisainfot, kui planeeritava rakenduse funktsionaalsust tulevikus laiendada.

Matroska failidesse siltide ja peatükkide kirjutamiseks ning nende lugemiseks saab kasutada MKVToolNix'i-nimelisse komplekti kuuluvaid käsurea programme [37]. Nii peatükke kui silte saab videofailile lisada kiirelt, korraga ning ilma kogu faili ümber kirjutamata. Andmed, mida kirjutada soovitakse, saab anda programmile ette XML-failina [38]. Tänu sellele saab metaandmete kirjutamiseks ettevalmistamisel kasutada olemasolevaid teeke ning ei pea muretsema erinevates operatsioonisüsteemides käskude käitamise iseärasuste üle, mis lihtsustab erimärkide kirjutamist ning vähendab rakenduses käsusüsti rünnaku võimalikkuse tõenäosust [39].

Matroska failide metaandmete toetus vastab peaaegu kõigile peatüki 4.2 sissejuhatuses kirjeldatud tingimustest. Kuid kuna selle kasutamine on piiratud vaid Matroska formaadis failidele, ei vasta see tingimusele 2 ning ei sobi rakenduse peamiseks salvestusviisiks. Siiski võiks antud meetodit kasutada, et kajastada vähemalt osa rakenduse hallatavast infost MKV failide küljes, võimaldamaks välistel programmidel seda infot kasutada.

4.2.3 *Sidecar* failid

Võimalik oleks ka metaandmeid salvestada eraldi failina olemasoleva videofaili kõrvale. Selliseid lisafaile nimetatakse *sidecar* failideks ning neid kasutatakse mõnikord näiteks eelnevalt mainitud XMP-metaandmete salvestamiseks, kui kasutatav failiformaat ei võimalda info otse peamisse faili salvestamist [40, lk 7]. Sel viisil metaandmete salvestamine ei sõltuks kasutatavast failisüsteemist ega videofaili formaadist, andmete kirjutamine poleks liialt piiratud mahu ega võimalike väärtuste poolest ning ei vajaks kogu meediafaili sisu ümber kirjutamist.

Ent info püsimine koos videofailiga eeldab seda, et kasutaja faile liigutades või ümber nimetades teeks vastava toimingu ka metaandmete failiga, kuna peamine meetod *sidecar* faili seostamiseks peamise failiga toimib tingimusel, et failid asuvad samas kataloogis ning nende nimed on faililaiendile eelnevalt teineteisega samasugused. Samuti näib, et muude programmide toetus *sidecar* failide kasutamiseks videofailide metaandmete allikana on väga piiratud. Kuna info seos videofailiga on siiski suhteliselt habras ning ühilduvus muu tarkvaraga üsna piiratud, ei oleks väärt *sidecar* failide kasutamine rakenduse peamise salvestusmeetodina ning ka lihtsalt rakenduse andmete dubleerimine *sidecar* failidesse poleks kuigi otstarbekas, kuigi seda võib tulevikus kaaluda valikulise funktsionaalsusena, mida rakendusele lisada.

4.2.4 Kasutatavate meetodite valik

Vaadeldud meetodite seast ei õnnestunud leida ühtegi, mis vastaks püstitatud tingimustele ning mille positiivsed aspektid ühilduvuse või andmete ja failide vahelise seose püsivuse kujul kaaluksid üles sellise meetodi poolt andmemudelile põhjustatavad piirangud. Seega on otstarbekas kasutada rakenduse andmete salvestamiseks siiski tüüpilist andmebaasi. Ent kirjeldatud meetodeid saab ikkagi teatud määral kasutada rakenduse töös.

Laiendatud atribuute ning Matroska failiformaadi metaandmeid saab kasutada, et faili juurde kirjutada unikaalne identifikaator. Seda saab rakendus kasutada abistava tunnusena, et rakenduse hallatavate kataloogide piires liigutatud või ümber nimetatud faile taas seostada andmebaasis vastava kirjega. Ka võiks neid samu meetodeid kasutades kirjutada failide juurde need metaandmed, mis on antud meetodite poolt defineeritud ning seega potentsiaalselt ühilduvad ka teiste programmidega.

4.3 Metaandmete sünkroniseerimine

Peatükis 4.2.4 sai otsustatud, et võimaluse korral tuleks osa rakenduse andmebaasis olevast infost salvestada ka otse hallatavate videofailide külge. Selleks on vajalik leida sobiv lähenemine andmete sünkroniseerimisele.

Ühele failile metaandmete kirjutamiseks võib leiduda mitmeid meetodeid, kuid mõne teise faili korral võivad meetodid hoopis puududa. Hetkel planeeritud meetodite korral saab meetodi kasutatavust tuvastada faililaiendi ning faili sisaldava failisüsteemi järgi. Ent on ka võimalik, et tulevikus tuleb rakendusele lisada metaandmete kirjutamise meetod, mille kasutatavuse üle otsustamiseks on vaja näiteks faili sisu lugeda. Seega loogika, kas meetod on kindla faili korral kasutatav, sõltub oluliselt meetodist endast. Mõistlik oleks luua üldine liides failile metaandmete kirjutamiseks ning konkreetsed metaandmete kirjutamise meetodid lisada rakendusse selle liidese implementatsioonidena, kusjuures nii meetodi kasutatavuse määramine kui meetodi rakendamine oleksid osa implementatsiooni loogikast.

Kuna erinevad metaandmete kirjutamise meetodid võimaldavad kirjutada erineval määral infot, võib teatud olukordades olla otstarbekas vältida metaandmete kirjutamist mõndade meetoditega. Näiteks, kui muudetakse otseülekandes tehtud tegevuse pealkirja, siis seda oleks asjakohane peegeldada Matroska failiformaadis vastava tegevusega seotud peatüki pealkirjas, kuid failisüsteemi laiendatud atribuutide hulgas ei leidu asjakohast atribuuti, kuhu seda muudatust kirjutada. Sellistes olukordades võiks rakendus tuvastada, et konkreetse meetodi poolt kirjutatavates andmetes pole tegelikku muudatust toimunud ning jätta ebavajaliku kirjutamise toimingu täitmata. Üks võimalus oleks iga muudatust kuidagi eraldi jälgida ning töötada välja süsteem nende muudatuste eraldi käsitlemiseks iga metaandmete kirjutamise korral. See oleks autori hinnangul ebavajalikult keerukas ning raskendaks oluliselt rakenduse edasiarendamist nii uute metaandmete kirjutamise meetodite lisamisel kui ka rakenduse andmemudeli laiendamisel. Oluliselt lihtsam alternatiiv oleks, et metaandmete kirjutamise meetod tõlgendab enda poolt kirjutatavad andmed deterministlikult kindlale kujule ning võrdleb enne andmete salvestamist praegu kirjutatavaid andmeid viimati kirjutatud andmetega. Kuna selle representatsiooni koostamisel kasutatakse ainult infot, mida konkreetne meetod võimaldab kirjutada, peaks see meetodi poolt ignoreeritavate andmete

muutumise korral muutumatu püsima. Kui see representatsioon salvestada andmebaasi antud faili kirje juurde seotuna antud meetodit tuvastava tunnusega, saab meetod järgmise kirjutamiskatse korral võrrelda failiga seotud andmetest tekkivat representatsiooni eelmisega ning jätta nende ühtimise korral metaandmed kirjutamata. Et mitte kasutada nende representatsioonidega andmebaasis liigselt mahtu, on arvatavasti otstarbekas salvestamisel ja võrdlemisel kasutada hoopis nendest arvutatud räsi.

Rakendusele võidakse edasiarenduste käigus lisada uusi metaandmete salvestamise meetodeid või täiendada olemasolevaid. Sellisel juhul on soovitav, et rakenduse uue versiooni esmakordsel käivitamisel rakendataks uusi või uuendatud meetodeid automaatselt kõigile hallatavatele failidele. See võib olla aeganõudev protsess ning võib seetõttu jääda pooleli juhul, kui rakendus seisatakse peagi pärast käivitamist. Samuti on võimalik, et rakenduse töö käigus mõne meetodi rakendamine ebaõnnestub esialgu, kuid hiljem tingimused muutuvad ning meetodi rakendamine uuesti proovimise korral õnnestuks. See võib olla aktuaalne näiteks juhul, kui Matroska failiformaadi metaandmete salvestamiseks vajalikud MKVToolNix'i programmid paigaldatakse või tehakse rakendusele kasutatavaks alles siis, kui rakendust on juba mõnda aega kasutatud. Mõlema olukorra jaoks oleks kasulik võimalus andmebaasist kas regulaarselt või rakenduse käivitamisel pärida failide kirjeid selle järgi, millised meetodid on neile rakendatud ning mis tulemusega. Meetodite uuendamise korral on oluline eristada ka sama meetodi erinevaid versioone. Seega tuleks andmebaasis faili kirjete juures meetodi tunnus seostada lisaks kirjutatud andmete deterministlikule representatsioonile ka meetodi viimati kasutatud versiooni ning selle rakendamise tulemusega.

Kui rakenduse andmebaasis muudetakse failiga seotud infot, võib see info olla failiga seotud vaid kaudselt – näiteks kui muudetakse isiku nime, võib see olla failiga seotud läbi tegevuse. Üks võimalik lähenemine andmebaasi seisu failide metaandmetega sünkroniseerimiseks oleks pärast iga muudatuse tegemist otsida üles mõjutatud failid ning käivitada neile metaandmete kirjutamine. Selline lähenemine lisaks aga rakendusele oluliselt keerukust, kuna iga andmebaasi muutva päringu korral tuleks mõjutatud videofailid veidi erineva loogika alusel leida. Praktilisem lähenemine oleks regulaarselt kõigile andmebaasis olevatele videofailidele proovida metaandmeid kirjutada. Tänu eelnevalt kirjeldatud kirjutatavate metaandmete räsi võrdlemisele ei

tähendaks ebavajalikult failidele metaandmete kirjutamist – kui faili metaandmed pole muutunud, jäetakse see fail lihtsalt vahele. Selle lähenemise korral küll ei peegelduks rakenduses tehtud muudatused failide juures kohe pärast muudatusi, kuid see pole ka tingimata vajalik funktsionaalsus ning kirjutamise toimumine viivitusega on täiesti aktsepteeritav. Juhuks, kui kasutaja siiski soovib viivitust ära ootamata mõne faili juurde metaandmeid kirjutada, saab lisada kasutajaliidesesse videofaili juurde nupu sellele failile metaandmete kirjutamise kohe käivitamiseks.

4.4 Andmebaasihaldussüsteemi valik

Käesolevas peatükis valitakse rakenduses kasutamiseks andmebaasihaldussüsteem.

4.4.1 Andmebaasihaldussüsteemi tüübi valik

Rakenduse andmete talletamiseks andmebaasihaldussüsteemi valimisel on esiteks vajalik otsustada, millist liiki andmebaasi kasutada. Üldjoontes eristatakse andmebaaside liigitamisel relatsioonilisi ja mitterelatsioonilisi andmebaase. Relatsioonilised andmebaasid said alguse juba 1970ndatel aastatel [41], [42] ning need on tänapäevani kasutatuim andmebaasitüüp [42], [43]. Tänu sellele on relatsiooniliste andmebaaside jaoks välja kujunenud üldtuntud töövõtted ning laialdane toetus erinevates tehnoloogiates ja arendajate seas on levinud kogemus nende kasutamiseks. Relatsioonilistes andmebaasides hoitakse andmeid kindlalt defineeritud struktuuriga tabelites ning kasutatakse standardiseeritud meetodeid tabelite vaheliste seoste kirjeldamiseks [42]. Mitterelatsioonilised andmebaasid seevastu võimaldavad andmeid salvestada paindlikumalt ilma range struktuurita ning sobivad oma kõrge horisontaalse skaleeritavuse tõttu hästi suurte andmemahtude ja paljude seadmete või kasutajatega hajussüsteemidele [44]. Kuna kavandatav rakendus on mõeldud kasutamiseks eraldiseisvate paigaldustena, mille kasutajate arv reeglina piirdub videokogu haldaja ja võib-olla mõne tema poolt volitatud isikuga, ei ole mitterelatsiooniliste andmebaaside skaleeritavus ja mitmetest allikatest pärinevate massiivsete andmekogude haldamise võimekus olulised faktorid andmebaasisüsteemi valikul. Seega peaks mitterelatsiooniline andmebaasisüsteem pakkuma mingisugust eelist rakenduse andmete kirjeldamisel ja kasutamisel, et oleks põhjust seda valida tüüpilise relatsioonilise andmebaasisüsteemi asemel.

Kavandatavas rakenduses on oluline, et oleks võimalik salvestada ja pärida andmeid olemite vaheliste mitu-mitmele tüüpi seoste kohta. Levinuimad mitterelatsiooniliste andmebaaside liigid on dokument-orienteeritud andmebaasid, võti-väärtus andmebaasid, veerupõhised andmebaasid ja graafandmebaasid [42], [44]. Dokument-orienteeritud ja võti-väärtus andmebaasid pole mõeldud taoliste seoste kirjeldamiseks ning raskendaksid sellest tulenevate takistuste tõttu soovitud andmete salvestamist ja kasutamist. Samuti poleks optimaalne kasutada veerupõhist andmebaasi, sest need on peamiselt mõeldud suurte andmekogude analüüsimiseks ning minetavad mitmed reapõhiste relatsiooniliste andmebaaside eelised [45]. Graafandmebaaside abil on võimalik kirjeldada komplekssseid erineva semantilise tähendusega seoseid suurtes andmekogudes [46]. Tänu võimalusele jooksvalt graafandmebaasis uusi seoseid defineerida ning nende põhjal keerukaid päringuid koostada, võiks graafandmebaasi kasutades planeeritavas rakenduses võimaldada kasutajal ise defineerida hallatavas meediakogus seoste täpseid semantilisi tähendusi ning kirjeldada nende abil metaandmeid detailsete mõistete ja seoste graafidena. Ent taoline lähenemine muudaks autori hinnangul nii rakenduse arendamisprotsessi kui ka rakenduse kasutuskogemuse märkimisväärselt keerukamaks lihtsama relatsioonilisel andmemudelil põhineva lähenemisega võrreldes. Kuna rakenduse eesmärk on salvestistekogu haldamise hõlbustamine, mitte otseülekannetes toimunu ning sellega seotud mõistete, seoste ja nende tähenduste kohta kõikehõlmava teadmistepagasi loomine, poleks sellise keerukuse lisamine mõistlik. Vaadeldud mitterelatsioonilised andmebaasid ei pakkunud piisavalt eeliseid, et õigustada nende kasutamist relatsioonilise andmebaasi asemel. Seega sobiks rakenduses siiski kasutada relatsioonilist andmebaasi.

4.4.2 Konkreetse relatsioonilise andmebaasihaldussüsteemi valik

Kavandatav rakendus peaks olema paigaldatav ja seadistatav ka tavakasutaja poolt, kellel ei pruugi olla põhjalikke infotehnoloogiaalaseid teadmisi ega kogemust. Seega võiks kasutatav andmebaasisüsteem mitte vajada rakendusest eraldi tarkvara allalaadimist, seadistamist, haldamist ja rakendusest eraldi käitamist. See kitsendab tuntud andmebaasisüsteemide valikuvõimaluste hulga koheselt ühe valikuni – SQLite. Erinevalt teistest populaarsetest andmebaasisüsteemidest nagu PostgreSQL, MySQL või Microsoft SQL Server, mis jooksevad eraldiseisvate protsessidena, on SQLite'i andmebaas lihtsalt fail, mis on täielikult hallatav rakenduse enda poolt [47]. See

võimaldaks soovi korral hoida andmebaasi koos hallatavate videofailidega ning laseks kasutajal andmetest varukoopiaid teha lihtsalt andmebaasifaili kopeerides. SQLite'i andmebaasifaili sisu on võimalik lugeda ja hallata paljude erinevate tööriistadega, mis aitab täita käesoleva töö eesmärki, et talletatavad andmed poleks kasutatavad ja tähenduslikud vaid loodava rakenduse jaoks [48]. SQLite'i veebilehe väitel on SQLite'i kasutatud edukalt paljude, sealhulgas ka sealhulgas ka meedia kataloogimiseks mõeldud rakenduste failiformaadina [48]. Lisaks paljudele teistele programmidele kasutavad SQLite'i oma peamise andmebaasimootorina näiteks ka isikliku audioraamatute ja taskuhäälingsaadete kogu haldamise tarkvara Audiobookshelf ning isikliku filmi- ja telesaadete kogu haldamise tarkvara Jellyfin [49], [50], [51]. Neid on asjakohane mainida, kuna sarnaselt käesolevas töös kavandatavale tarkvarale on tegemist isikliku meediakogu haldamiseks mõeldud rakendustega.

4.4.3 SQLite'i funktsionaalsuste kontroll

Enne SQLite'i kasuks lõpliku otsuse tegemist tuleb kontrollida, kas ja kuidas see suudab pakkuda funktsionaalsusi, mis rakenduse töös vajalikud või kasulikud oleksid.

Osaline tekstiotsing

Otseülekannete otsimine pealkirjades, võtmesõnades vms tekstis sisalduvate tekstilõikude järgi on planeeritavas rakenduses ülioluline funktsionaalsus. SQLite toetab LIKE operaatorit, millega saab muuhulgas otsida tekste nendes sisalduvate tekstilõikude järgi [52, Ptk 5]. Ka on SQLite'il olemas täisteksti otsingut võimaldav laiendus, millega saab filtreerida suurt kirjete hulka otsinguterminite sisaldumise järgi [53].

Tõstutundetu tekstiotsing

Et tekstiotsingut oleks praktiline kasutada, peab olema võimalik otsida teksti tõstutundetult ehk suur- ja väiketähtede erinevusi ignoreerides. Äsja kirjeldatud LIKE funktsiooniga saab küll teksti tõstutundetult võrrelda, kuid vaikimisi toimib see vaid ASCII tähemärkidega – avaldis `'a' LIKE 'A'` annab positiivse tulemuse, aga `'ä' LIKE 'Ä'` annab negatiivse tulemuse [52, Ptk 5]. Ent SQLite'i täisteksti otsingu laiendusega saab teha ka tõstutundetuid päringuid, millel see piirang puudub. Kui kasutada täisteksti otsingu seadistamisel „trigram" teksti tükeldajat, siis saab lisaks

sõnade ja fraaside otsingule kasutada täisteksti päringus ka täpse tekstilõigu sisaldumise otsingut [53, Ptk 4.3.4].

Kui aga peaks osutuma, et täisteksti otsingu kasutamine pole millegipärast praktiline, on võimalik kasutada SQLite'i ICU (*International Components for Unicode*) laiendust, mis lisab SQLite'ile Unicode'i toetusega versiooni LIKE operaatorist [52, Ptk 5], [54]. Kui ka see osutub ebaotstarbekaks, oleks viimane variant ise tekitada andmebaasi eraldi veerg, mis sisaldaks suurtähtedes koopiat originaaltekstist. Selle veeru vastu saaks teha tõstutundetuid päringuid, olles enne ka otsitava teksti suurtähtedesse teisendanud. Kuid ka SQLite'i suurtähtedesse teisendamise funktsioon vajab Unicode'i toetuse jaoks ICU laienduse kasutamist [55], mistõttu poleks antud alternatiivse lähenemise korral võimalik selle veeru väärtustamiseks kasutada andmebaasi trigerfunktsiooni. Seega sel juhul tuleks suurtähtedes veergu uuendada rakenduse koodist, mis tõstab veidi ohtu, et see jääb originaalveeruga sünkroonist välja.

Rekursiivsed päringud

Võib osutuda soovitavaks, et rakenduse organiseerimisvõimaluste hulka kuuluks otseülekannete või muude kirjete seostamine kategooriate või võtmesõnade hierarhiatega. Nende põhjal otsingute teostamiseks ning hierarhiate välja kuvamiseks võib olla kasulik, kui andmebaasisüsteem toetaks rekursiivseid päringuid. SQLite'is on vastav toetus olemas „*Recursive Common Table Expressions*" avaldiste kujul, millega saab koostada rekursiivseid graafi või puud läbivaid päringuid [56, Ptk 3].

4.4.4 Mõned SQLite'i puudujäägid

Lisaks on SQLite'il veel mõned puudujäägid, mida reeglina teistel andmebaasisüsteemidel pole.

Piiratud võimalused tabeli muutmiseks

„*ALTER TABLE*" käsklus, mida relatsioonilistes andmebaasides kasutatakse olemasoleva tabeli muutmiseks, on SQLite'is piiratud funktsionaalsusega. Tabelisse veeru lisamisel pole võimalik kasutada teatud vaikeväärtusi ning lisatav veerg peab vastama kindlatele tingimustele – näiteks ei saa sellele veerule rakendada unikaalsuse piirangut. Ka pole otseselt toetatud ükski tabeli andmemudeli muutmise käsk, mis ei

kuulu käskluste "*RENAME TABLE*" („muuda tabeli nime”), "*RENAME COLUMN*" („muuda veeru nime”), "*ADD COLUMN*" („lisa veerg”) või "*DROP COLUMN*" („kustuta veerg”) hulka. See tähendab, et „*ALTER TABLE*” käsuga pole võimalik näiteks veeru andmetüübi muutmine või veerule rakendatud nõuete (nt unikaalsus või nullväärtuste keelamine) muutmine. SQLite’is on siiski võimalik läbi viia keerukamaid andmemudeli muudatusi, kuid tegemist on tülikama protsessiga, mis hõlmab uue tabeli loomist koos soovitud muudatustega ning andmete vanast tabelist uude ümber kopeerimist. [57]

Andmebaasifaili fragmenteerumine

Suurte andmemahtude kustutamine võib jätta SQLite’i andmebaasifaili tühja ruumi, mille tagajärjel võib fail jääda mahukamaks, kui rangelt vajalik. Ka võib SQLite’i andmebaasifail ajaga fragmenteeruda, jättes ühe tabeli või indeksi sisu hajutatuna failis erinevatesse asukohtadesse. Fragmenteerumise ja tühja ruumi eemaldamiseks saab käitada „*VACUUM*” käsku, kuid sel ajal ei tohi andmebaasiga teha muid toiminguid. Samuti on võimalik kasutada „*auto-vacuum*” režiimi, mis eemaldab tühja ruumi automaatselt pärast kustutamistoiminguid, kuid ei puhasta andmebaasifaili nii korralikult kui „*VACUUM*” käsu käitamine. Võimalik, et antud puudujäägid ei osutu suuremaks probleemiks, kuid neid tasub siiski teadvustada, juhuks kui sellega tuleb hiljem arvestada – näiteks võib tulevikus olla mõistlik lisada rakendusele võimalus regulaarselt ja/või kasutaja sisendi korral andmebaasifaili peal „*VACUUM*” käsklust käitada. [58]

Andmetüüpide piiratud valik

SQLite toetab vaid viite erinevat andmetüüpi – nullväärtus, täisarv, ujukomaarv, tekst või kahendandmete bloob (ingl. k *blob – binary large object* ehk suur kahendobjekt [59]) [60, Ptk 2]. Teistes andmebaasisüsteemides, näiteks PostgreSQL’is [61] või Microsoft SQL Serveris [62] on palju ulatuslikum andmetüüpide toetus. Muuhulgas on SQLite’ist puudu näiteks ajahetkede, kuupäevade, püsikomaarvude ja universaalsete unikaalsete identifikaatorite (ingl. k U*niversally Unique IDentifier*, UUID [63]) jaoks mõeldud andmetüübid. See aga ei takista mittetoetatud andmetüüpide salvestamist SQLite andmebaasis, lihtsalt võib muuta nende kasutamise veidi ebamugavaks.

Andmetüüpide rangus

Erinevalt teistest andmebaasisüsteemidest kasutab SQLite dünaamilist andmetüüpide süsteemi, kus talletatava väärtuse andmetüüp on seotud väärtuse endaga, mitte veeruga, kus see väärtus asub [60]. Seega ei kindlusta SQLite, et tabelisse sisestatud andmed päriselt vastavad defineeritud andmemudelile. Näiteks, kui defineerida tabelis täisarvu andmetüübiga veerg, ei takistaks SQLite sellesse veergu hoopis teksti sisestamist. Alates 2021. aastast toetab SQLite ka nõndanimetatud „rangete tabelite" („*STRICT TABLES*") režiimi, mis nõuab veergudele nende defineerimisel andmetüübi määramist ning takistab kasutajat neisse muid andmetüüpe sisestamast [64].

Rangete tabelite režiim pole siiski täiuslik. SQLite toetab ainult üsna piiratud andmetüüpide hulka, ent vaikimisi võimaldab SQLite kasutada ka muid nimetusi veergudele andmetüüpide määramisel, mida see siis tõlgendab ühena toetatud andmetüüpidest, olenevalt kasutatud nimetusest [60]. Näiteks, kuigi SQLite ei toeta otseselt tõeväärtuse (BOOLEAN) ega ajahetke (DATETIME) andmetüüpe, saab neid nimetusi kasutada veeru andmetüübi määramisel ning SQLite tõlgendab neid täisarvu (INTEGER) andmetüübina [60]. Rangete tabelite režiim aga seda ei võimalda ning aktsepteerib ainult kindlaid toetatud andmetüüpide nimetusi [60], [64]. Ühest küljest on see hea, kuna nii vastavad veergudes kasutatud andmetüüpide nimetused täpselt andmetüüpidele, millena SQLite neid käsitleb ning ei saa tekitada segadust isikutes, kes pole tuttavad SQLite'i andmetüüpide süsteemiga ja võiksid ilma selle piiranguta arvata, et tegemist on päriselt andmetüüpidega, millele nimi viitab. Teisest küljest aga võib see muuta andmemudeli raskemini loetavaks, kui andmeid, mida rakendus tegelikult käsitleb erinevate andmetüüpidena, hoitakse andmebaasis sama nimega andmetüübi all. Autori hinnangul on rangete tabelite režiimi eelised seda väikest puudujääki väärt, kuna aitavad tagada rakenduse korrektset toimimist ning sellega kaotatud andmetüüpide nimetuste loetavus on suhteliselt ebaoluline.

Välisvõtmete toetus

Kui tabelis on defineeritud viide kirjele teises tabelis välisvõtme piiranguga, siis erinevalt teistest andmebaasisüsteemidest SQLite vaikimisi ei kindlusta viidatud kirjete olemasolu [65, Ptk 2]. Et piirangud jõustuksid ja nende kehtivust kontrollitaks, on vaja vastav toetus iga SQLite andmebaasiühenduse korral sisse lülitada [65, Ptk 2]. Kuna

piirangud saab sisse lülitada, pole otseselt tegemist puudujäägiga, kuid siiski on see üks SQLite'i veidrus, millest tuleks kindlasti teadlik olla enne SQLite'i kasutuselevõtmist.

4.4.5 Lõplik otsus

SQLite'i kasutamine on eelistatav teistele andmebaasisüsteemidele, mis muudaksid rakenduse paigaldamise ja haldamise lõppkasutaja jaoks oluliselt keerulisemaks. Kuigi SQLite'il on mitmeid puudujääke, leidub neist igaühele lahendus või kohandus, mis võib küll olla veidi ebamugav, kuid ei takista soovitud toimingute läbiviimist. Antud puudujäägid ei takista SQLite'i kasutamist rakenduse peamise andmebaasisüsteemina. Samuti on SQLite edukalt kasutusel paljudes teistes rakendustes.

Ka käesolevas töös kavandatava rakenduse peamiseks andmetalletusmehhanismiks sobib ja saab seega SQLite. Muidugi võiks rakenduse arendamisel abstraheerida andmebaasi kasutamise mehhanismi headele tavadele vastavalt, et rakendusele oleks võimalik vajadusel ka teiste andmebaasisüsteemide toetus lisada.

4.5 Tehnoloogiate valik

Käesoleva töö raames loodav prototüüp on veebirakendus, mis jaguneb kaheks osaks: serveril jooksev tagarakendus ja veebibrauserist ligipääsetav kasutajaliides ehk eesrakendus.

4.5.1 Tagarakenduse programmeerimiskeele valik

Aastatel 2023-2024 tagarakendustes enimkasutatud programmeerimiskeeled on JavaScript, TypeScript, Python, Java, C#, C++, C, PHP ja Go [43, Alajaot. Web frameworks and technologies], [43, Alajaot. Programming, scripting, and markup languages], [66], [67].

C, C++ ja Go on keeled, mida reeglina kasutatakse võimalikult suure jõudluse ja optimaalse mälukasutuse saavutamiseks. Kavandatav rakendus on mõeldud korraga kasutamiseks reeglina üksikule kasutajale või väikesele kasutajate grupile ning see ei pea ise töötlema suuri andmemahte, seega pole jõudlus programmeerimiskeele valikul hädavajalik faktor. PHP, Python ja JavaScript on dünaamilise tüübisüsteemiga programmeerimiskeeled, mis tähendab, et erinevalt staatilise tüübisüsteemiga

programmeerimiskeeltest ei kontrollita nende kasutamisel andmetüüpe, funktsioonide signatuure ja nende kasutamist staatiliselt enne programmi käivitamist. Staatilise tüübisüsteemiga keelte eelistena tuuakse tihti välja vigade varajane avastamine, koodibaasi parem hallatavus ja mõistetavus ning korralikum funktsionaalsuste valik integreeritud arenduskeskkondades [68]. Dünaamilise tüübisüsteemi eelistena mainitakse peamiselt kiiremat arendusprotsessi ning suuremat paindlikkust arenduse käigus [68]. Kavandatav rakendus on mõeldud olema kauakestev projekt, millele võib olla vaja ka aastaid hiljem jätkuarendusi teha. Pikaajalise hooldatavuse ja koodi selguse tagamiseks oleks mõistlik valida rangema tüübisüsteemiga keel ning seega jäävad PHP, Python ja JavaScript valikust välja.

Loodav rakendus peab suutma failihierarhiates muudatusi tuvastada ning laiendatud failiatribuute lugeda ja kirjutada. Valikusse jäänud keeled Java, TypeScript ja C# kõik võimaldavad nõutud funktsionaalsust realiseerida. Java'l (või täpsemalt Java virtuaalmasinal) on küll ainsana sisseehitatud liides laiendatud atribuudite kasutamiseks, kuid sama funktsionaalsuse lisamine TypeScript'is või C#'is kirjutatud projekti on triviaalselt lihtne ning seega pole tegemist otsustava põhjusega Java valimiseks.

TypeScript'i valimine tagarakenduse programmeerimiskeeleks võimaldaks ees- ja tagarakenduse vahel andmeedastusobjektide tüüpide lihtsat jagamist ning kogu rakenduses vaid ühe programmeerimiskeele kasutamist. Samas võib sellega kaasneda ees- ja tagarakenduse vahelise piiri hägustumine, mis võib muuta rakenduse toimimise arendajatele raskemini mõistetavaks ning põhjustada turvavigu. Mainitud eelised on pigem väikesed mugavused, mitte otsustavad faktorid. Samuti pole TypeScript'i kasutamine ainuke viis antud eeliste saavutamiseks – näiteks ka C#'i saab Blazor'i abil kasutada eesrakenduse kirjutamiseks.

TypeScript'i, Java ja C#'i seast kindla keele valimiseks tugevad tehnilised põhjused puuduvad. Antud juhul osutus autori eelistusel valituks C#.

4.5.2 Eesrakenduse tehnoloogia valik

Kavandatav rakendus ei pea töötama ilma ühenduseta eesrakenduse ja serveri vahel ning kasutajaliides ei pea olema eriti kompleksne, mistõttu võiks kaaluda eesrakenduse kirjutamist mitmeleherakendusena. Ent mitmeleherakenduse potentsiaalsetele eelistele

vaatamata on tõenäoline, et mugava kasutuskogemuse pakkumiseks osutub soovitavaks lokaalse oleku ning muude üheleherakenduse võimaluste kasutamine. Siin-seal HTML-sisule JavaScript'i koodi lisades on küll võimalik mitmeleherakenduse funktsionaalsust täiendada ning puudujääke leevendada, kuid sel viisil kahe erineva lähenemise segamine võib osutuda raskesti hallatavaks ning arvatavasti oleks mõistlikum siiski võtta kohe algusest kasutusele mõni tänapäevane üheleherakenduse raamistik, et vältida kasutajaliidese võimaluste piiramist.

Kasutades mõnda WebAssembly-põhist raamistikku saaks kirjutada ka eesrakenduse C# keeles, jagades tagarakendusega ühist koodi. Samas peab taolistes raamistikes brauseri erinevate funktsionaalsuste kasutamiseks ikkagi JavaScript'i kasutama ning kuigi WebAssembly ise on pika perspektiiviga standard, on sellel põhinevad raamistikud võrdlemisi uued ja ebakindlama tulevikuga. Tõenäoliselt oleks kindlam hetkel valida siiski JavaScript'il põhinev raamistik.

Erinevaid JavaScript'i-põhiseid eesrakenduse raamistikke on palju ning argumente ühe teistele eelistamiseks veelgi rohkem. Neist populaarseim on juba vähemalt 6-8 aastat olnud React [43, Ptk Web frameworks and technologies], [69], [70]. React'il on mitmeid puudujääke, sealhulgas ebaoptimaalne jõudlus [71], kuid võib väita, et tegemist on kompromissidega muude eeliste nimel [72]. Kuigi alternatiivsetel raamistikel nagu Svelte või SolidJS on paeluvaid aspekte, on React'il tänu selle laiale kasutajaskonnale korralikum toetus erinevate teekide poolt, suurem valik arendajaid ning ilmselt ka stabiilsem tulevik. Seega on praktiline ka käesoleva töö raames React'i kasutada.

4.6 Andmemudeli loomine

Enne arendama asumist kavandati loodava rakenduse andmemudel. Käesolevas alapeatükis kirjeldatakse olulisemaid andmemudeli kavandamisel arvestatud kaalutlusi.

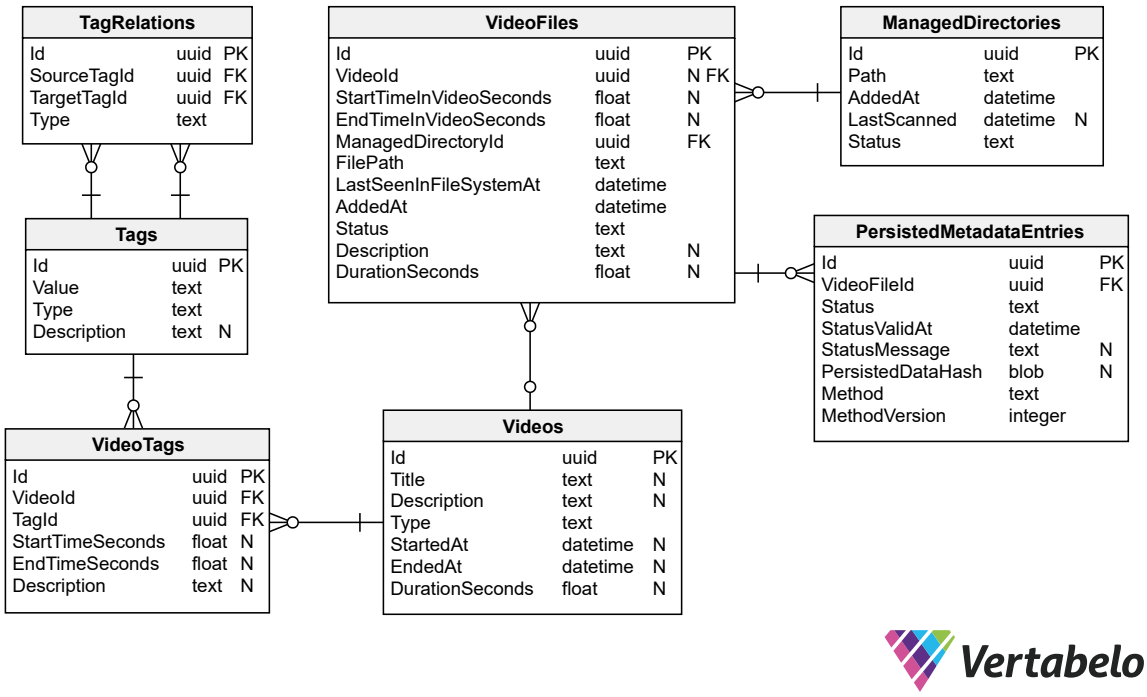
Ühe otseülekandega võib kokku kuuluda mitu erinevat videofaili, näiteks kui otseülekandest tehti mitmes vormingus salvestisi või kui otseülekanne on jaotatud kaheks lühemaks videofailiks, kuna otseülekande jooksul katkes salvestamine ajutiselt. Nagu peatükis 2.2 mainitud, on rakenduses failide haldamisel soovitav peamiselt käsitleda just otseülekandeid kui mõisteid, mitte nendega seotud konkreetseid videofaile – metaandmed nagu võtmesõnad, osalenud isikud või tehtud tegevused kuuluvad

tegelikult otseülekande mõiste juurde ning on videofailiga seotud vaid antud seose tõttu. Seega on andmemudelis otseülekannete kirjeldamiseks abstraktne video olem (*Videos* tabel), millega saab olla seotud mitu videofaili (*VideoFiles* tabel). Videofaili seostamisel videoga saab määrata ka ajavahemiku videos, mida antud videofail kajastab. Keerukamad ajalised seosed videote ja videofailide vahel pole antud töö skoobi raames vajalikud ning andmemudel neid kirjeldada ei võimalda.

Vastavalt peatükis 4.1.1 kirjeldatud nõuetele peab rakenduses olema võimalik otseülekandeid seostada neis osalenud isikute, tehtud tegevuste, kategooriate ja võtmesõnadega. Kõik need täidavad rakenduses sarnast rolli – need esinevad sarnastes seostes ning nende põhjal peab saada videoid filtreerida. Kuna antud mõistete käsitlemine erinevate olemitena pole antud rakenduse skoobis tingimata vajalik ning suurendaks oluliselt keerukust nii andmemudeli, rakenduse koodi kui ka rakenduse kasutamise tasemel, on otstarbekas need abstraheerida ühe üldise mõiste „silt" alla (*Tags* tabel). Iga sildi juures on diskriminaatorveeruga määratud, mis tüüpi sildiga on tegemist – näiteks isik, tegevus, tegevuse ilming, kategooria või võtmesõna. Siltide videotega seostamiseks on andmemudelis *VideoTags* vahetabel, mis võimaldab valikuliselt määrata ka ajahetke või ajavahemiku videost, millega silt seotud on. Silte saab *TagRelations* tabeli abil seostada ka teiste siltidega, võimaldades näiteks tegevust seostada kategooriatega või tegevuse ilminguid seostada nendes osalenud isikutega.

Peatükis 4.3 tehtud analüüsile vastavalt on andmebaasi seisu erinevate metaandmete kirjutamise meetoditega sünkroniseerimiseks vajalik talletada infot selle kohta, milliseid metaandmete kirjutamise meetodeid on faili peal rakendatud. Seda eesmärki täidab videofailide tabeliga üks-mitmele seoses olev tabel *PersistedMetadataEntries*, kuhu salvestatakse info metaandmete kirjutamise tulemuste kohta. Antud tabelis hoitakse infot õnnestunud, ebaõnnestunud ning ka järjekorras ootavate tulevikku ajastatud metaandmete kirjutamise katsete kohta.

Joonisel 1 on kujutatud kavandatud andmemudel olemi-suhte diagrammi kujul.



Joonis 1. Rakenduse jaoks kavandatud olemi-suhte diagramm.

4.6.1 Primaarvõtmete andmetüüp

Relatsioonilises andmebaasis hoitakse erinevate olemite andmeid eraldi tabelites, mis viitavad teineteises asuvatele kirjetele kirjet unikaalselt tuvastavate võtmete väärtuste kaudu. See võti on tihti täisarv, mida iga uue kirje lisamisel ühe võrra suurendatakse. Teine variant on juhuslikult genereeritud UUID, mida saab tekitada, arvestamata juba andmebaasis leiduvate võtmete seisuga.

Antud rakenduse kontekstis olulised UUID'de eelised on:

- Võimalus võtmeid genereerida ilma andmebaasiga suhtlemata [73]. See lihtsustab toiminguid nagu seotud kirjete korraga andmebaasi lisamine ja paljude kirjete sisestamine ühe INSERT käsuga, kui sisestatud kirjete võtmeid on vaja pärast sisestust kasutada.

- Asjaolu, et kahe samasuguse UUID tekkimine, isegi erinevates süsteemides, on statistiliselt võimatu. See aitab vältida võimalust, et ühe tabeli võtme ekslikult kasutamine teises tabelis leiab juhuslikult võtmele vaste. [73]

Mitmes andmebaasisüsteemis saab UUID'de eelistena mainitud omadusi saavutada ka täisarvuliste primaarvõtmetega, kuid selleks vajalike funktsionaalsuste toetus varieerub süsteemiti. SQLite näiteks ei toeta jadasid (ingl. k *sequence*), mida võiks mõnes teises andmebaasisüsteemis kasutada võtmete tabeliteülese unikaalsuse tagamiseks või kirjete korraga sisestamisel võtmete eelgenereerimiseks. Samuti nõuab antud omaduste saavutamine tihti veidi keerukamat loogikat andmebaasipäringute koostamisel. Andmebaasiga suhtlemiseks kasutatav teek võib selle lisakeerukuse küll sageli arendaja eest ära peita, kuid taolist keerukust vajava valiku tegemine on siiski küsitav, kui UUID'de kujul on olemas ilma sellise lisakeerukuseta variant.

UUID'de peamised puudujäägid täisarvudega võrreldes on loetavus (UUID'sid kuvatakse enamasti 36-tähemärgise sõnena) ning suurem mälu- ja mahukasutus (UUID on 16 baiti, võtmena kasutatav täisarv tavaliselt 8 baiti). UUID'de kasutamisel võtmetena on tihti ka halvem jõudlus, kuid seda peaks aitama oluliselt leevendada UUID 7nda versiooni kasutamine, kus UUID'd tekitatakse ajahetke põhjal ning on selle alusel järjestatavad. [73]

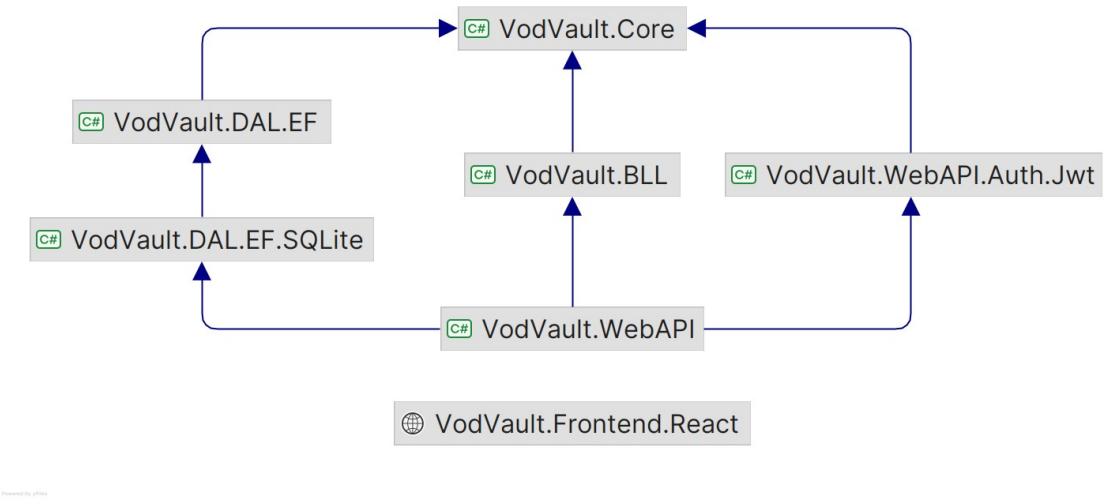
Autori hinnangul on UUID'de puudujäägid väheolulised eelistega võrreldes ning täisarvude kasutamine primaarvõtmetena põhjustaks UUID'dega võrreldes rakenduse koodile ebavajalikke tehnilisi piiranguid. Seega otsustati rakenduse andmemudelis kasutada primaarvõtmetena UUID'sid.

5 Teostus

Käesolevas peatükis kirjeldatakse rakenduse prototüübi arendamise protsessi olulisemaid aspekte.

5.1 Lahenduse arhitektuur

Rakenduse koodibaasi struktureerimisel lähtuti puhta arhitektuuri põhimõtetest, mille populariseeris Robert C. Martin [74, Ptk 22]. Joonisel 2 on kujutatud ülevaatlik diagramm rakendust moodustavatest projektidest ning nendevahelistest seostest.



Joonis 2: Rakenduse projektide sõltuvusdiagramm.

- Rakenduse „keskmes" (joonise ülaosas) asub tuumikprojekt (VodVault.Core). Seal defineeritakse rakenduse põhifunktsionaalsuseks olulised olemid, liidesed ja andmeedastusobjektid ning üldkasutatavad abifunktsioonid ja -klassid.

- Infrastruktuuri- ja teenustekihi projektides implementeeritakse tuumikprojektis defineeritud liideseid. Andmetalletuskihi projektides VodVault.DAL.EF ja VodVault.DAL.EF.SQLite projektides sisaldub otseselt andmebaasiga seonduv kood nagu olemite seadistamine, migratsioonid ja keerukamad päringud. Äriloogika projektis VodVault.BLL asuvad rakenduse funktsionaalsust

teostavad teenused nagu näiteks failide jälgimise ja metaandmete kirjutamise teenused.

- Esitluskihi projekt VodVault.WebAPI defineerib veebipõhise rakendusliidese, millega eesrakendus VodVault.Frontend.React üle võrgu suhtleb. VodVault.WebAPI on see projekt, mis serveris programmina käivitatakse, seega siin seadistatakse ka rakenduses kasutatavad teenused ning käivitatakse taustatööd. Rakenduse kasutaja autentimise loogika tagarakenduses on eraldatud VodVault.WebAPI.Auth.Jwt projekti.

5.2 Rakenduse arendamine

Vastavalt peatükis 4.5.1 otsustatule arendati tagarakendus C#-programmeerimiskeeles. Veebiliidese projektis kasutati C#-arenduses *de facto* standardina kasutatavat ASP.NET Core veebiraamistikku. Eesrakenduse arendamisel kasutati vastavalt peatükis 4.5.2 React'i ja TypeScript'i. Eesrakenduse arendamisel käesoleva töö teema kontekstis mainimisväärseid iseärasusi ei esinenud ning tüüpilist kasutajaliidese arendamise protsessi ei ole asjalik käesolevas töös kirjeldada. Küll aga kirjeldatakse järgnevalt olulisemaid aspekte tagarakenduse arendusprotsessist.

5.2.1 Suhtlus andmebaasiga

Andmebaasiga suhtlemiseks kasutati C#-rakendustes laialt kasutatavat Entity Framework Core teeki (edaspidi EF Core). Lisaks tüüpilistele objekt-relatsioonilise vastendaja (ingl. k *Object-Relational Mapper – ORM*) pakutavatele eelistele saab EF Core'iga ka genereerida ja hallata andmemudeli muudatusi ehk migratsioone. EF Core'i SQLite liides oskab käsitleda ka peatükis 4.4.4 kirjeldatud keerukamaid andmemudeli muudatusi, mis vajavad SQLite'i puudujääkide tõttu uue tabeli loomist. Ühe puudujäägina selgus, et EF Core'i SQLite liides ei toeta veel migratsioonide loomisel peatükis 4.4.4 mainitud „*STRICT TABLES*" režiimi. Ent kuna antud teek on piisavalt laiendatav, õnnestus autoril vastav toetus ise lisada.

Sageli abstraheeritakse rakendustes andmebaasi kasutus repositooriumite (ingl. k *repository*) taha, kus on defineeritud kindlad meetodid, mille kaudu teenustekihi klassid andmebaasiga suhtlevad. Seda tehakse tihti ka C#-rakendustes, kus on kasutusel EF

Core, kuigi EF Core tegelikult juba pakub repositooriumilaadset abstraktsiooni `DbSet` klassi kujul. Selleks kirjutatakse hulk liideseid ja klasse, mis abstraheerivad `DbSet`'i kasutamise teenustekihi eest ära veel eraldi repositooriumite taha. Taoline topeltabstraktsioon võib olla mõistlik, kui soovitakse teenustekihis viidata vaid rakenduse enda loodud abstraktsioonidele, kuid selle tagajärjel kannatab teenustekihis andmebaasiga suhtlemise paindlikkus. Näiteks sarnaste filtreerimistingimuste taaskasutamine erinevates päringutes või sama päringu tulemuste materialiseerimine erineval kujul muutub tülikaks ning nõuab repositooriumitesse iga kasutusjuhu jaoks eraldi meetodi lisamist või ebaefektiivsete päringutega leppimist. Autori hinnangul poleks selline topeltabstraktsioon antud rakenduses otstarbekas oma piiratuse ja ebavajaliku lisakeerukuse tõttu ning mõistlikum on kasutada EF Core'i `DbContext` ja `DbSet` klasse teenustes otse.

Keerukamate või konkreetsest andmebaasisüsteemist sõltuvate filtreerimistingimuste kasutamiseks andmebaasipäringutes kasutati rakenduses spetsifikatsiooni disainimustrist (ingl. k *specification pattern*) [75] inspireeritud lähenemist. Asjakohases klassis defineeritakse meetod, mis tagastab soovitud predikaatavaldise, kasutades C#'i `Expression<Func<T, bool>>` tüüpi. Tänu LinqKit teegile saab selliseid avaldisi kasutada ja kombineerida EF Core päringute koostamisel, võimaldades vältida koodikordust, hallata andmebaasisüsteemispetsiifilisi päringuid eraldi projektides ning mugavalt koostada teenuskihis konkreetses olukorras sobivaimaid päringuid. Näiteks `FilterByTagQueryBase` meetodi tagastatud avaldis filtreerib siltide kirjeid etteantud päringuobjekti põhjal – seda avaldist kasutatakse siltide otsingu meetodis, kuid ka osana keerukamast päringust videote otsingu meetodis, kus saab valikuliselt otsida videoid, millega seotud sildid vastavad antud päringuobjekti tingimustele. Kuna `FilterByTagQueryBase` meetod tagastab predikaatavaldise, mitte juba andmebaasist päritud kirjete tulemi (erinevalt tüüpilistest repositooriumimeetoditest), on selle filtri kasutajatel võimalik ise otsustada, kas pärida selle predikaadi põhjal üks kirje või mitu kirjet või töödelda päringut enne tulemuste andmebaasist tagastamist, näiteks vastuseid grupeerides või ainult kindlaid veerge pärides.

Nagu peatükis 4.4.3 mainitud, on SQLite'is tõstutundetu tekstiotsingu võimalused veidi piiratud. Analüüsis leitud esimene võimalik lahendus, täisteksti otsingu kasutamine vajab tekstiotsingu otstarbeks eraldi tabeli loomist, mille seostamine tavaliste tabelitega

ja päringutes kasutamine olnuks antud eesmärgi saavutamiseks liiga keerukas. Teine analüüsis toodud lahendus eeldab SQLite'ile ICU laiendi paigaldamist, mis samuti lisaks loodavale rakendusele märkimisväärselt keerukust. Seega otsustati kasutada kolmandat analüüsis pakutud lahendust – tõstutundetut tekstiotsingut vajavate veergude sisust suurtähtedes koopia talletamine eraldi veerus, millega tõstutundetu otsingu korral eelnevalt suurtähtedesse teisendatud päringut saab võrrelda. Kuna tegemist on SQLite'i-spetsiifilise teostusdetailiga, siis kogu lisaveergudega seotud loogika, kaasaarvatud nende lisamine andmemudelile, kirjutati VodVault.DAL.EF.SQLite projekti ning ülejäänud projektid ei ole lisaveerust teadlikud. Antud SQLite'i-spetsiifilises projektis lisatakse EF Core'i andmebaasikontekstile sekkumismehhanimid ehk nõndanimetatud *interceptor*'id, mis asjakohases tabelis kirje lisamisel või muutmisel automaatselt uuendavad lisaveergu vastavalt põhiveeru sisule. Antud veergude kasutamiseks päringute filtreerimisel on tuumikprojektis defineeritud asjakohased spetsifikatsiooniliidesed, mida SQLite'i-spetsiifilises infrastruktuuriprojektis implementeeritakse.

5.2.2 Metaandmete kirjutamine ja lugemine

Lähtudes peatükis 4.3 otsustatust loodi metaandmete kirjutamiseks liides `IFileMetadataProvider`, mille kaudu rakenduse üldteenused saavad erinevaid metaandmete kirjutamis- ja lugemismeetodeid kasutada. Erinevate metaandmete talletusmeetodite jaoks defineeritakse seda liidest teostavad klassid ning registreeritakse rakenduse teenustekonteineris. Vastavalt peatükis 4.2.4 otsustatule lisati rakenduse prototüübile toetus laiendatud failiatribuutide meetodi ja Matroska failivormingu metaandmete meetodi kasutamiseks. Laiendatud failiatribuutide alternatiivina NTFS failisüsteemis kasutati peatükis 4.2.1 mainitud alternatiivseid andmevooge. Prototüübi raames lisati meetoditele võimekus mõne olulisema metaandmete tüübi kirjutamiseks, sealhulgas kirjeldus ja võtmesõnad, ning lisaks ka rakenduse-spetsiifilise identifikaatori lugemise ja kirjutamise funktsionaalsus, et tuvastada videofail, kui seda rakenduseväliselt ümber nimetatakse või liigutatakse.

Matroska failivormingu metaandmete kirjutamiseks ja lugemiseks kasutati vastavalt „mkvpropedit" ja „mkvextract" utiliite MKVToolNix'i-nimelisest tarkvarakomplektist. Laiendatud failiatribuutide haldamiseks kasutati operatsioonisüsteemipõhise „xattr" teegi funktsioone „getxattr" ja „setxattr". Kuna tegemist on sõltuvustega, mis võivad

mõnes käituskeskkonnas puududa, siis metaandmete kirjutamisega tegelev taustatöö pärib enne konkreetse meetodi kasutamist sellelt, kas meetodi kasutamiseks vajalikud tingimused on täidetud ning kui ei ole, siis lükkab antud metaandmete kirjutamise ülesande täitmise edasi, ootamaks hetke, mil vajalikud sõltuvused paigaldatakse.

Videofaili esmakordsel rakendusse lisamisel käivitatakse sellele failile metaandmete kirjutamise taustatööd automaatselt, et kirjutada faili juurde identifikaator faili hilisemaks taastuvastamiseks failiraja muutuse korral. Edaspidi, vastavalt peatükis 4.3 otsustatule, proovib rakendus regulaarselt kirjutada metaandmeid iga hallatud videofaili külge, jättes kirjutamise vahele, kui kirjutatavad metaandmed pole muutunud. Lisaks saab kasutaja metaandmete kirjutamise käivitada käsitsi konkreetse videofaili jaoks.

5.2.3 Failisüsteemis muudatuste jälgimine

Failisüsteemi seisu sünkroniseerimiseks andmebaasiga loodi rakendusse taustatöö, mis regulaarselt skaneerib rakenduse haldusse lisatud failikatalooge, lisades uute videofailide kohta kirje andmebaasi *VideoFiles* tabelisse ning kirjutades juba lisatud videofailide korral vastavas kirjes *LastSeenInFileSystemAt* veergu uue ajahetke, millal antud fail viimati failisüsteemis tuvastati. Failisüsteemist leitud faili ja andmebaasis olemasolevate failide vastendamine toimub peamiselt failiraja põhjal, kuid kui failiraja põhjal vastet ei leita, siis proovib rakendus leida faili metaandmetest peatükis 5.2.2 mainitud identifikaatorit. Kui faili küljest õnnestub ühe metaandmete meetodi abil leida otsitav identifikaator, andmebaasis leidub sellele identifikaatorile vastav kirje ning selle kirjega seotud failirajal ei asu enam faili, siis seostab rakendus leitud faili antud kirjega.

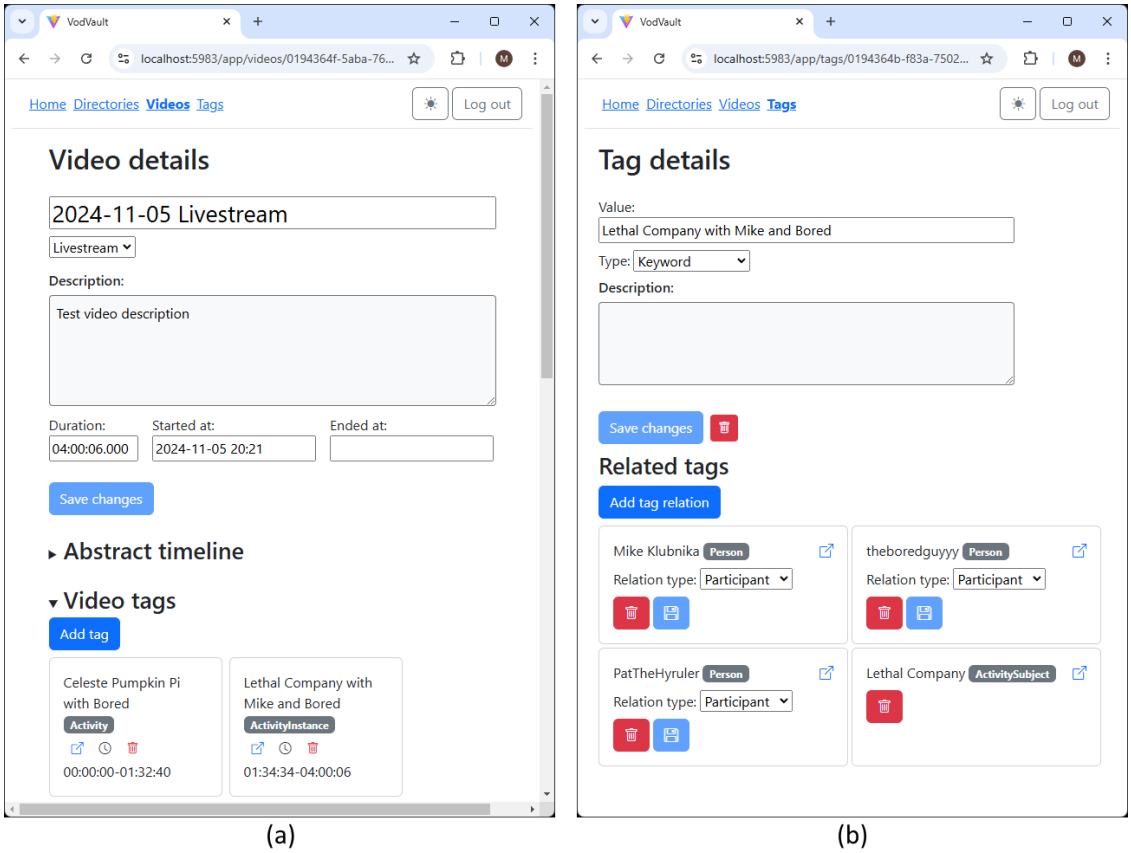
Erinevalt uute ja olemasolevate failide tuvastamisest, kus lähtutakse failisüsteemi sisust, tuleb kustutatud failide tuvastamiseks alustada andmebaasi kirjete töötlemisest ning kontrollida, kas need on jätkuvalt failisüsteemis alles. Kuna tegemist on põhimõtteliselt erineva tegevusega uute failide tuvastamisest, loodi rakenduses selle jaoks eraldi taustatöö. Et optimeerida selle käigus käsitletavate failide hulka, kontrollib see taustatöö enamasti vaid nende failide olemasolu, mida eelnevas lõigus kirjeldatud failisüsteemi skaneeriv taustatöö ei tuvastanud. Selleks võrreldakse videofailide tabeli kirjete viimati failisüsteemis leidumise ajahetke veergu (*LastSeenInFileSystemAt*) ja nendega seotud hallatavate failikataloogide tabeli kirje viimati skaneerimise hetke veergu (*LastScannedAt*).

Lisaks regulaarselt failisüsteemi vaatlevatele taustatöödele reageerib rakendus ka failisüsteemi poolt muudatuste korral saadetavatele teavitustele, kasutades .NET-põhiteekidesse kuuluvat `FileSystemWatcher` klassi. Tänu sellele on võimalik muudatusi tuvastada koheselt pärast nende toimumise hetke, selmet oodata regulaarse taustatöö graafiku järel. See meetod pole aga täielikult töökindel, kuna mitmed failisüsteemid ei toeta teavituste funktsionaalsust ning ka failisüsteemid, mis neid toetavad, ei pruugi garanteerida, et iga muudatuse kohta teavitus saadetakse või kätte saadakse. Seega saab seda lähenemist kasutada vaid failimuudatuste tuvastamise täiendamiseks, kuid mitte peamise failisüsteemi jälgimise meetodina.

6 Tulemused

Lõputöö tulemusteks on analüüs ja selle põhjal arendatud veebirakenduse prototüüp. Hindamaks, kas rakendus ka päriselt sobib käsitletava probleemi lahendamiseks, prooviti selles läbi mõned olulised näidisolukorrad.

Esiteks prooviti rakenduses kirjeldada otseülekannet, mille alguses mängiti ühte mängu ning lõpus mängiti teist mängu, kusjuures teise tegevusega liitus isik, kes esimeses tegevuses ei osalenud. Selleks loodi otseülekande jaoks video ning kummagi tegevuse jaoks silt, mis seostati vastava ajavahemikuga loodud videos. Tegevustele lisati seosed kummaski tegevuses osalenud isikutega. Loodi ka tegevuse subjekti tüüpi sildid mängitud mängude jaoks, millega tegevuse ilmingud seostati ning määrati otseülekande algusaeg. Joonisel 3 saab näha antud video ja teise videos tehtud tegevuse seadistusi rakenduse kasutajaliideses.

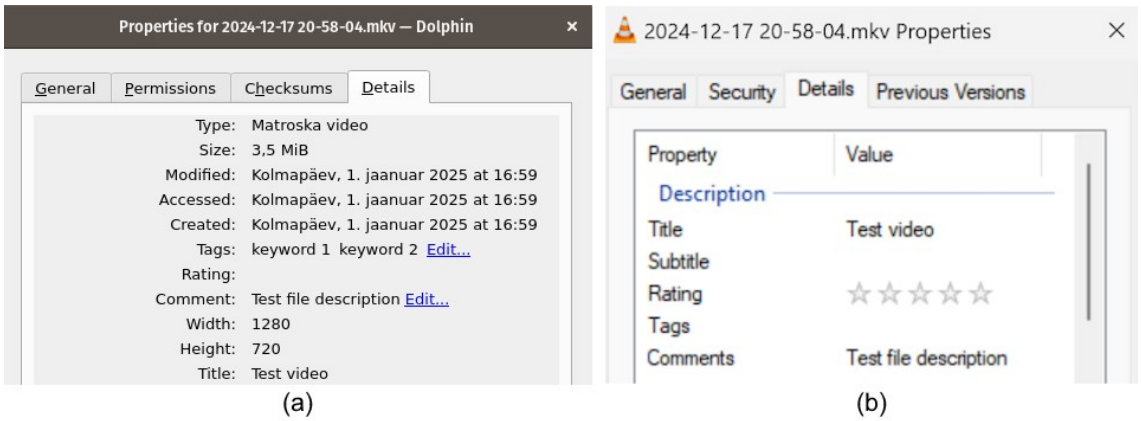


Joonis 3: Kuvatõmmised rakendusest: (a) video andmed, (b) tegevuse andmed.

Eelnevas lõigus kirjeldatud omaduste põhjal prooviti ka videoid otsida. Kontrolliti, et toimib siltidepõhine rekursiivne otsing, mis leiab kirjeldatud otseülekande üles sellega kaudselt seotud isikute järgi (Lisa 2). Lisas 3 on kujutatud videote filtreerimine nende algusaja kindlasse ajavahemikku kuuluvuse põhjal. Samuti saab videoid otsida pealkirja järgi, kasutades osalist tekstiotsingut või valikuliselt ka regulaaravaldisi.

Lisa 4 alumises osas saab näha videofailide videoga seostamise kasutajaliidest. Selles on võimalik otsida videotega veel seostamata videofaile ning määrata neid valitud video külge. Ühe videoga saab seostada mitu videofaili ning valikuliselt ka määrata ajavahemiku videos, millega videofail seotud on. Ka on videofaili juures nupp sellele metaandmete kirjutamise käivitamiseks.

Metaandmete kirjutamise ning kirjutatu muu tarkvaraga ühilduvuse testimiseks lisati videole kirjeldus ja mõned võtmesõnad ning seostati videoga videofail, misjärel käivitati videofailile metaandmete kirjutamine. Seejärel vaadati faili omadusi kahes faililehitsejas, kus ootuspäraselt olid rakenduses salvestatud metaandmed sobivates väljades kuvatud. Lisa 4 sisaldab kuvatõmmist antud video seadistusest rakenduse kasutajaliideses ning Joonis 4 esitab kuvatõmmised faililehitsejates kuvatud infost. Läbi prooviti ka rakenduse võimekus faili taastuvastamiseks pärast selle liigutamist või ümbernimetamist. Rakendus tuvastas katsetamise käigus tehtud muudatused otsekohe ning seostas uued failirajad vastavate kirjetega andmebaasis.



Joonis 4: Videofaili info: (a) Dolphin’is, (b) Windows File Explorer’is.

Loodud rakendus võimaldab kirjeldada peatükis 4.1.1 nõutud seoseid ning videoid nende põhjal otsida. Rakendus suudab liigutatud faile taastuvastada ning salvestab hallatavad metaandmed võimalusel ka kujul, mida saavad lugeda muud programmid. Lahendus vastab peatükis 4.1 määratud nõuetele ning töö eesmärgid said täidetud.

Samas oli töö skoop piiratud ning rakenduse edasi arendamiseks on mitmeid võimalusi. Neist peamine oleks videote omavahel seostamise funktsionaalsuse lisamine. Antud töö raames oli oluline otseülekannete haldamine, kuid oleks kasulik ka võimalus rakenduses hallata lühiklippe, montaaže ja muid sarnaseid videoid ning neid otseülekannete ja teineteisega seostada. Samuti võiks rakendus pakkuda salvestiste subtiitrite haldamise võimalust. Lisada saaks veel erinevaid mugavusi ja täiendusi, nagu näiteks detailsemalt kohandatav siltidepõhine otsing või kasutajaliideses faili mahukuse ja muutmiskuupäeva kuvamine. Parendada saaks ka metaandmete talletusmeetodite toetust, lisades uusi meetodeid või suurendades lisatud meetodite poolt kirjutatavate metaandmete hulka. Viimaks võiks tulevikus rakendusele lisada liidestusi väliste platvormidega nagu Twitch ja YouTube, võimaldamaks neilt otseülekannete info pärimist ja salvestiste neile üleslaadimist.

7 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli välja töötada optimaalne viis otseülekannete videosalvestiste haldamiseks, metaandmetega täiendamiseks ja metaandmete põhjal filtreerimiseks ning arendada analüüsi valideerimiseks valmis rakenduse prototüüp.

Esiteks tutvustati lahendatava probleemi tausta ning püstitati eesmärgid. Seejärel analüüsiti olemasolevaid lahendusi ning seati nõuded loodavale rakendusele. Analüüsi peatükis võrreldi põhjalikult erinevaid võimalusi metaandmete videofailide külge salvestamiseks. Võrdlusest selgus, et ükski võrreldud meetoditest ei sobi rakenduse peamiseks andmetalletusmeetodiks, kuid neid saab võimalusel kasutada rakenduse funktsionaalsuse täiendamiseks ning muu tarkvaraga koostalitlusvõime parendamiseks. Järgnevalt planeeriti rakenduse andmebaasi ja failisüsteemi sünkroniseerimist, misjärel valiti rakenduses kasutatav andmebaasisüsteem ning ees- ja tagarakenduse tehnoloogiad. Analüüsi lõpus kavandati rakenduse jaoks andmemudel. Viimaks arendati analüüsi põhjal rakenduse prototüüp ning kirjeldati selle arendamise olulisemaid aspekte teostuse peatükis.

Lõputöö tulemuseks on metaandmete videofailidega seostamise võimaluste analüüs ning selle põhjal arendatud otseülekannete videosalvestiste haldamise rakenduse prototüüp veebirakenduse kujul. Rakendus võimaldab hallata failikataloogides asuvaid videofaile, seostada nendega soovitud metaandmeid ning filtreerida salvestistega seotud videoid nende metaandmete põhjal. Kuigi võrreldud faililähedasemad metaandmete talletamise meetodid ei sobinud andmete peamiseks salvestusmeetodiks, õnnestus neid rakenduses edukalt kasutada liigutatud ja ümbernimetatud failide taas andmebaasi kirjetega seostamiseks ning metaandmete teistele programmidele loetavaks muutmiseks. Autori hinnangul said lõputöö alguses püstitatud eesmärgid täidetud.

Kasutatud kirjandus

[1] K. M. Adams, *Non-functional Requirements in Systems Analysis and Design*. Springer, 2015.

[2] *ISO/IEC/IEEE International Standard - Systems and software engineering--Vocabulary*. doi: 10.1109/IEEESTD.2017.8016712.

[3] „Database normalization“, *Wikipedia*. 5. aprill 2024. Vaadatud: 16. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://en.wikipedia.org/w/index.php?title=Database_normalization&oldid=1217305645

[4] „xattr(7) - Linux manual page“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://man7.org/linux/man-pages/man7/xattr.7.html

[5] „Extended attributes - ArchWiki“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://wiki.archlinux.org/title/Extended_attributes#Preserving_extended_attributes

[6] „CommonExtendedAttributes“, Freedesktop.org. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.freedesktop.org/wiki/CommonExtendedAttributes/

[7] Dublin Core Metadata Initiative, „DCMI Metadata Terms“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

[8] J. Gerend, „NTFS overview“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview

[9] Microsoft, „[MS-FSCC]: NTFS Streams“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/c54dec26-1551-4d3a-a0ea-4fa40f848eb3

[10] Adobe, „How to pick the best video file format“. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.adobe.com/creativecloud/video/discover/best-video-format.html

[11] V. Tejeda, „Video File Formats and Types: A Beginner’s Guide“, descript. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.descript.com/blog/article/video-file-types

[12] G. Maayan, „8 Best Video File Formats for 2020“, IEEE Computer Society. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.computer.org/publications/tech-news/trends/8-best-video-file-formats-for-2020/

[13] Moving Picture Experts Group, *MPEG-4: MP4 File Format*. Vaadatud: 12. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.mpeg.org/standards/MPEG-4/14/

[14] Moving Picture Experts Group ja R. Koenen, „MPEG-4 Overview - (V.21 – Jeju Version) - ISO/IEC JTC1/SC29/WG11 N4668“. märts 2002.

[15] *ISO/IEC 14496-12:2015: Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*, 2015.

[16] P. Harvey, „ExifTool“, Exiftool by Phil Harvey. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://exiftool.org/
[17] „mp4tag - Bento4“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.bento4.com/documentation/mp4tag/
[18] „Other Features: MP4Box tagging support - GPAC wiki“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://wiki.gpac.io/MP4Box/mp4box-other-opts/#tagging-support
[19] „AtomicParsley“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://atomicparsley.sourceforge.net/
[20] „ffmpeg Documentation“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://ffmpeg.org/ffmpeg.html
[21] Apple, „QuickTime metadata keys“, Apple Developer Documentation. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://developer.apple.com/documentation/quicktime-file-format/quicktime_metadata_keys
[22] „MPEG-4 files - AtomicParsley“, AtomicParsley. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://atomicparsley.sourceforge.net/mpeg-4files.html
[23] P. Harvey, „QuickTime Tags“, Exiftool by Phil Harvey. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://exiftool.org/TagNames/QuickTime.html
[24] K. Howells, „How to Add Chapters to MP4s with FFmpeg“, Kyle Howells Blog. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://ikyle.me/blog/2020/add-mp4-chapters-ffmpeg
[25] P. Harvey, „Structured Information“, Exiftool by Phil Harvey. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://exiftool.org/struct.html
[26] „ISO - International Organization for Standardization“, ISO. Vaadatud: 6. jaanuar 2025. [Võrgumaterjal]. Loetud aadressil: https://www.iso.org/home.html
[27] „File management, metadata integration | Adobe Extensible Metadata Platform (XMP)“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.adobe.com/products/xmp.html
[28] „XMP - Semantic Web Standards“, W3C. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://www.w3.org/2001/sw/wiki/XMP
[29] Adobe, „XMP SPECIFICATION PART 1: DATA MODEL, SERIALIZATION, AND CORE PROPERTIES“. Adobe Inc., aprill 2012. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://github.com/adobe/XMP-Toolkit-SDK/blob/9f779e85f15c92108f8b911930998a152b209491/docs/XMPSpecificationPart1.pdf
[30] Adobe, „XMP SPECIFICATION PART 2: ADDITIONAL PROPERTIES“. Adobe Inc., 2022. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://github.com/adobe/XMP-Toolkit-SDK/blob/9f779e85f15c92108f8b911930998a152b209491/docs/XMPSpecificationPart2.pdf
[31] P. Harvey, „ExifTool example config“, Exiftool by Phil Harvey. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://exiftool.org/config.html#xmp-xxx
[32] bram, „Atomic Parsley Manual : idoru.be/notes“. Vaadatud: 13. aprill 2024. [Võrgumaterjal]. Loetud aadressil: http://www.idoru.be/notes/atomic-parsley-manual/
[33] Matroska, „Matroska Media Container Homepage“, Matroska. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://matroska.org/index.html

[34] Matroska, „What is Matroska?“, Matroska. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://matroska.org/what_is_matroska.html
[35] Matroska, „Tags“, Matroska. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://matroska.org/technical/tagging.html
[36] Matroska, „Chapters“. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://matroska.org/technical/chapters.html
[37] Matroska, „MKVToolNix“, Matroska. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://matroska.org/downloads/mkvtoolnix.html
[38] M. Bunkus, „mkvpropedit -- Modify properties of existing Matroska files without a complete remux“. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://mkvtoolnix.download/doc/mkvpropedit.html
[39] W. Zhong, „Command Injection | OWASP Foundation“. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://owasp.org/www-community/attacks/Command_Injection
[40] Adobe, „XMP SPECIFICATION PART 3: STORAGE IN FILES“. jaanuar 2020. Vaadatud: 17. aprill 2024. [Võrgumaterjal]. Loetud aadressil: https://github.com/adobe/XMP-Toolkit-SDK/blob/9f779e85f15c92108f8b911930998a152b209491/docs/XMPSpecificationPart3.pdf
[41] „Types Of Databases“, MongoDB. Vaadatud: 4. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://www.mongodb.com/resources/basics/databases/types
[42] R. Sheldon, „How to choose between SQL and NoSQL databases“, Simple Talk. Vaadatud: 3. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://www.red-gate.com/simple-talk/databases/nosql/how-to-choose-between-sql-and-nosql-databases/
[43] „Stack Overflow Developer Survey 2023“, Stack Overflow. Vaadatud: 4. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://survey.stackoverflow.co/2023/
[44] A. Prakash, „Exploring Different Types of Databases: A Guide for Data Engineers | Airbyte“. Vaadatud: 3. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://airbyte.com/data-engineering-resources/types-of-database
[45] C. Archer, „What is a columnar database? Here are 35 examples.“ Vaadatud: 4. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://www.tinybird.co/blog-posts/what-is-a-columnar-database
[46] „What is a Graph Database?“, Oracle. Vaadatud: 4. mai 2024. [Võrgumaterjal]. Loetud aadressil: https://www.oracle.com/in/autonomous-database/what-is-graph-database/
[47] „About SQLite“, SQLite. Vaadatud: 7. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/about.html
[48] „SQLite As An Application File Format“, SQLite. Vaadatud: 29. september 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/appfileformat.html
[49] *jellyfin/jellyfin*. (13. oktoober 2024). C#. Jellyfin. Vaadatud: 13. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://github.com/jellyfin/jellyfin
[50] advplyr, *advplyr/audiobookshelf*. (13. oktoober 2024). JavaScript. Vaadatud: 13. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://github.com/advplyr/audiobookshelf
[51] „Well-Known Users Of SQLite“, SQLite. Vaadatud: 13. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://sqlite.org/famous.html
[52] „SQL Language Expressions“, SQLite. Vaadatud: 13. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/lang_expr.html#like

[53] „SQLite FTS5 Extension“, SQLite. Vaadatud: 13. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/fts5.html
[54] „SQLite: All Files in ext/icu/“, SQLite. Vaadatud: 16. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://sqlite.org/src/dir/ext/icu
[55] „Built-In Scalar SQL Functions“, SQLite. Vaadatud: 16. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://sqlite.org/lang_corefunc.html
[56] „The WITH Clause“, SQLite. Vaadatud: 17. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/lang_with.html
[57] „ALTER TABLE“, SQLite. Vaadatud: 12. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/lang_altertable.html
[58] „VACUUM“, SQLite. Vaadatud: 12. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/lang_vacuum.html
[59] „blob - AKIT“, Andmekaitse ja infoturbe portaal. Vaadatud: 12. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://akit.cyber.ee/term/5959
[60] „Datatypes In SQLite“, SQLite. Vaadatud: 12. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/datatype3.html
[61] „Chapter 8. Data Types“, PostgreSQL Documentation. Vaadatud: 17. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.postgresql.org/docs/17/datatype.html
[62] M. Ray ja Microsoft, „Data types (Transact-SQL) - SQL Server“. Vaadatud: 17. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16
[63] K. R. Davis, B. Peabody, ja P. Leach, „Universally Unique IDentifiers (UUIDs)“, Internet Engineering Task Force, Request for Comments RFC 9562, mai 2024. doi: 10.17487/RFC9562.
[64] „STRICT Tables“, SQLite. Vaadatud: 12. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/stricttables.html
[65] „SQLite Foreign Key Support“, SQLite. Vaadatud: 17. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.sqlite.org/foreignkeys.html
[66] „TIOBE Index“, TIOBE. Vaadatud: 21. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://www.tiobe.com/tiobe-index/
[67] „Programming languages used in most popular websites“, *Wikipedia*. 23. september 2024. Vaadatud: 21. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://en.wikipedia.org/w/index.php?title=Programming_languages_used_in_most_popular_websites&oldid=1247209278
[68] A. Bello, „Disadvantages of Dynamic Typing“. Vaadatud: 22. oktoober 2024. [Võrgumaterjal]. Loetud aadressil: https://amorserv.com/insights/disadvantages-of-dynamic-typing
[69] „React, Angular, Vue, Svelte, Solid JS - Explore“, Google Trends. Vaadatud: 12. november 2024. [Võrgumaterjal]. Loetud aadressil: https://trends.google.com/trends/explore?q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9,%2Fg%2F11bc69_ykv,%2Fg%2F11c5t00h04,%2Fg%2F11tdm_rdjb&date=2010-11-12%202024-11-12#TIMESERIES
[70] „React, Vue, Angular, Svelte trends“, Stack Overflow Tag Trends. Vaadatud: 12. november 2024. [Võrgumaterjal]. Loetud aadressil: https://trends.stackoverflow.co/?tags=reactjs,vue.js,angular,svelte,angularjs,vuejs3
[71] S. Krause, „JS Framework Benchmark 2024 - Chrome for Testing 124.0.6367.91“. Vaadatud: 12. november 2024. [Võrgumaterjal]. Loetud aadressil:

https://krausest.github.io/js-framework-benchmark/2024/table_chrome_124.0.6367.91.html
[72] D. Abramov, „Reply to ‘React vs Signals’“, DEV Community. Vaadatud: 12. november 2024. [Võrgumaterjal]. Loetud aadressil: https://dev.to/this-is-learning/react-vs-signals-10-years-later-3k71#comment-256g9
[73] Tianzhou, „UUID or Auto Increment Integer / Serial as the Database Primary Key?“, Bytebase. Vaadatud: 20. november 2024. [Võrgumaterjal]. Loetud aadressil: https://www.bytebase.com/blog/choose-primary-key-uuid-or-auto-increment/
[74] R. C. Martin, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Pearson. Vaadatud: 29. detsember 2024. [Võrgumaterjal]. Loetud aadressil: https://learning.oreilly.com/library/view/clean-architecture-a/9780134494272/
[75] M. Fowler ja E. Evans, „Specifications“. Vaadatud: 29. detsember 2024. [Võrgumaterjal]. Loetud aadressil: https://martinfowler.com/apsupp/spec.pdf

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

Mina, Mikkel Paat

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Rakendus otseülekannete videosalvestiste haldamiseks" mille juhendaja on Märt Kalmo

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.

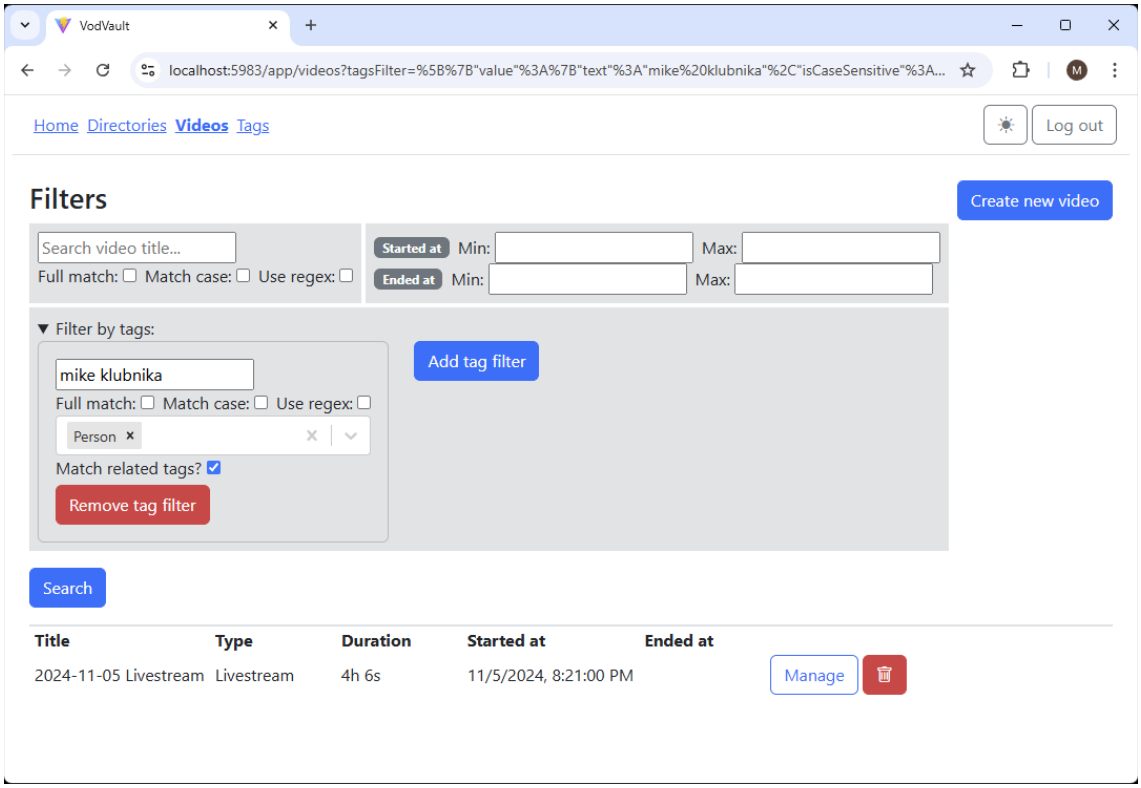
2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

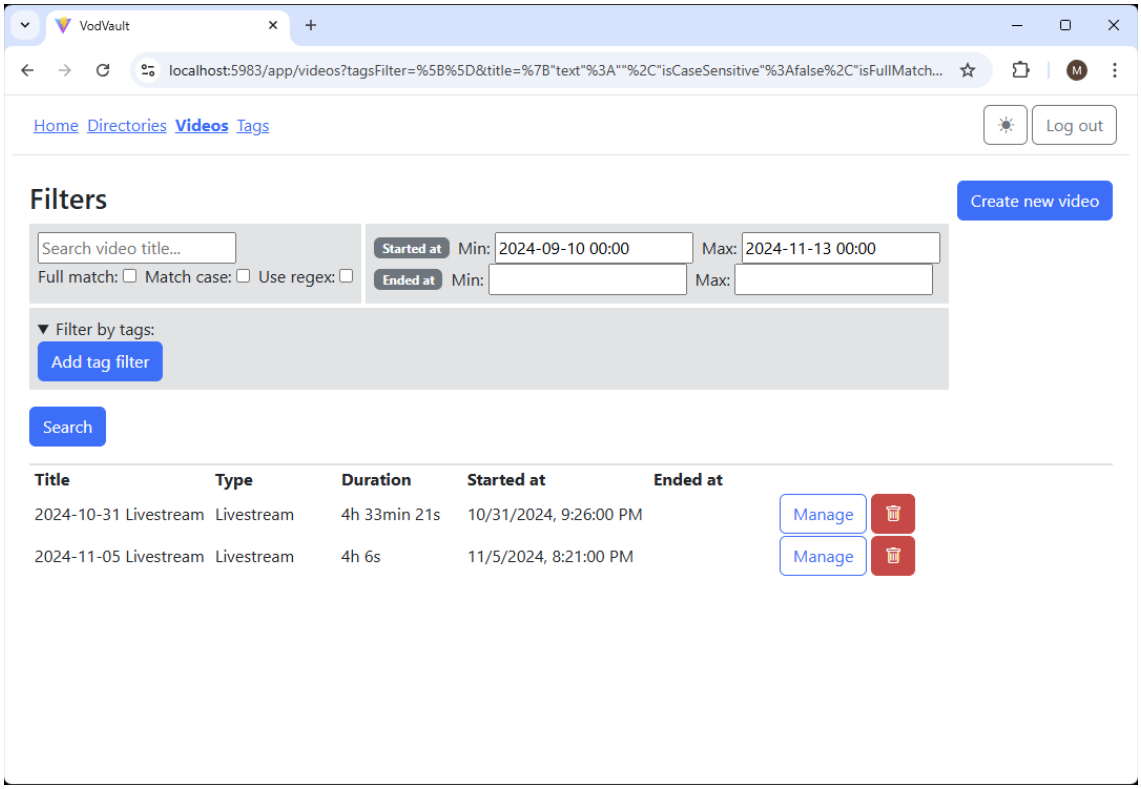
06.01.2025

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.

Lisa 2 – Kuvatõmmis videote otsingust kaudselt seotud isiku järgi



Lisa 3 – Kuvatõmmis videote otsingust alguskuupäeva järgi



Lisa 4 – Kuvatõmmis metaandmete kirjutamise katsetamiseks kasutatud video vaatest