TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Tõnis Täht 143144IAPB

GRAAFILISE KASUTAJALIIDESE ARENDAMINE YOUTUBE'I VIDEO ALLALAADIMISE PROGRAMMILE

Bakalaureusetöö

Juhendaja: Ago Luberg
MSc

Tallinn 2019

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: Tõnis Täht

21.05.2019

Annotatsioon

Antud töö raames luuakse graafiline kasutajaliides YouTube’i video allalaadimise programmile.

Hetkel on olemas mitmeid programme, millega YouTube’ist videoid alla on võimalik laadida, kuid paljud neist on tasulised, sisaldavad häirivaid reklaame või ei ole modernse välimusega.

Antud töö käigus antakse ülevaade kasutatud tehnoloogiatest, mille abil graafiline kasutajaliides loodi, ning tutvustatakse väliseid programme, mida liides sisemiselt kasutab. Ühtlasi tutvustatakse ehitamise protsessi, mille tulemusena on rakendust võimalik kasutada kolmel erineval operatsioonisüsteemil.

Lisaks tutvustatakse plaane, mida võiks tulevikus kasutajaliideses muuta ja millist funktsionaalsust juurde lisada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 21 leheküljel, 7 peatükki, 0 joonist ja 0 tabelit.

Abstract

Development of a Graphical User Interface for the YouTube Video Download Program

In this work, a graphical user interface is created for the YouTube video download program.

There is a number of programs currently available for downloading videos from YouTube, but many of them are not free, contain annoying advertisements or do not have modern appearance.

This work provides an overview of the technologies used to create a graphical user interface and introduces external programs that are used internally by the interface. Furthermore, the deployment process is introduced, resulting in the interface being usable in three different operating systems.

In addition, this work presents the plans for the future – what could be changed in the user interface and what functionality could be added.

The thesis is in Estonian and contains 21 pages of text, 7 chapters, 0 figures and 0 tables.

Lühendite ja mõistete sõnastik

API	*application programming interface*
EXE	*executable file format*
JSON	*JavaScript Object Notation*
OS	operatsioonisüsteem

Sisukord

1 Sissejuhatus

Tänapäeval on internetis saadaval palju programme, millega saab erinevatelt video sisu pakkuvatelt veebilehtedelt videoid alla laadida, kuid pahatihti on nende programmide funktsionaalsus piiratud, need sisaldavad häirivaid reklaame või on koguni tasulised, kui soovitakse kasutada funktsionaalsust täies ulatuses.

Hetkel kõige suurema funktsionaalsusega tasuta programm, millega saab youtube.com veebilehelt erinevaid videoid alla laadida, on käsurea programm nimega youtube-dl, mis on peamiselt mõeldud youtube.com veebilehelt videote allalaadimiseks, kuid võimaldab seda teha ka paljudelt muudelt populaarsetelt veebilehtedelt.

Käsurea programmi kasutamine on üldiselt ebamugav, mistõttu on töö eesmärgiks teha sellele graafiline kasutajaliides, mida oleks lihtne kasutada ja mis sisaldaks enim vajaminevat funktsionaalsust. Kuna youtube-dl programm on kirjutatud Pythoni keeles ja töötab tänu sellele erinevate operatsioonisüsteemide peal, siis peaks graafiline liides samuti toetama erinevaid operatsioonisüsteeme. Selle saavutamiseks on kasutatud raamistikku nimega Electron, mis võimaldab kirjutada *native* väljanägemisega programme erinevatele operatsioonisüsteemidele, kasutades selleks veebitehnoloogiaid (HTML, JavaScript, CSS).

2 Analüüs

Antud peatüki eesmärk on analüüsida olemasolevaid programme, mis võimaldavad youtube.com veebilehelt videoid alla laadida, ning tehnoloogia valikut ehk mida kasutati käsurea programmile graafilise kasutajaliidese loomiseks.

2.1 Olemasolevad video allalaadimise programmid

Hetkel on internetis saadaval üsna mitmeid programme (näiteks 4K Video Downloader[1], YTD Video Downloader[2], Ummy Video Downloader[3]), millega on võimalik youtube.com veebilehelt videoid alla laadida, kuid paljud neist on tasulised, kui soovida funktsionaalsust kasutada täies mahus, või sisaldavad häirivaid reklaame. Osad programmid on väga vähese funktsionaalsusega ning need, mis pakuvad kasutajale palju erinevaid võimalusi, on üldiselt tasulised. Suureks miinuseks on ka see, et enamus programme on mõeldud Windows operatsioonisüsteemile ja vähe on selliseid, mis töötavad erinevatel platvormidel. Üks parimaid programme, mis on hetkel saadaval, on programm nimega youtube-dl-gui. Tegemist on tasuta programmiga, mis töötab erinevatel platvormidel ja sisaldab palju funktsionaalsust, kuid selle rakenduse kasutajaliides on vanamoeline ning seetõttu üritatakse käesolevas töös luua rakendust, mis oleks modernse välimusega ja esialgne versioon sisaldaks minimaalset vajalikku funktsionaalsust. Tulevikus on plaanis rakendust täiendada selliselt, et oleks parem integratsioon erinevate operatsioonisüsteemidega ja laiem funktsionaalsuste valik.

2.2 Käsurea programm youtube-dl

Kui eelnevalt mainitud programmid olid kõik graafilise kasutajaliidesega, siis youtube-dl[4] on käsurea programm, millega on võimalik videoid alla laadida nii youtube.com

[1] 4K Video Downloaderi koduleht: https://www.4kdownload.com/products/product-videodownloader

[2] YTD Video Downloaderi koduleht: https://www.ytddownloader.com/

[3] Ummy Video Downloaderi koduleht: https://videodownloader.ummy.net/

[4] youtube-dl'i koduleht: https://ytdl-org.github.io/youtube-dl/index.html

veebilehelt kui ka paljudelt muudelt veebilehtedelt. Youtube-dl'i teeb eriliseks see, et see on kirjutatud programmeerimiskeeles Python ning tänu sellele töötab erinevatel operatsioonisüsteemidel. Windows OS-i jaoks on olemas käivitatav EXE fail, mis juba sisaldab endas Pythonit ning seetõttu ei ole programmi kasutamiseks vaja eraldi Pythonit installeerida. Erinevate Linux OS-i versioonide ja macOSi peal töötamiseks on vajalik eraldi Pythoni olemasolu, kuid enamjaolt on see vaikimisi kaasas OS-i installeerimisel.

Youtube-dl'i kasutamine on üldiselt väga lihtne: piisab vaid videole viitava veebiaadressi parameetrina kaasa panemisest ja üldjuhul suudab programm ise valida parima kvaliteediga video, juhul kui neid on mitu, ja selle alla laadida. Kuid graafilise liidese arendamiseks on vaja rohkem valikuvõimalusi ja paindlikkust. youtube-dl programmis on olemas palju erinevaid parameetreid, mida kasutades on võimalik saada erinevat infot allalaaditava video kohta ning graafilise liidese puhul peaks kasutajal olema võimalik valida, millise kvaliteediga video ta soovib alla laadida. Selle jaoks on võimalik kasutada parameetrit `--dump-single-json`, mis tagastab kogu vajaliku info vastava video kohta ja mille alusel saab kasutaja valida, millist kvaliteeti ta soovib. Kui kasutaja on oma valiku teinud, saab youtube-dl programmile parameetritega öelda, millise kvaliteediga video tuleks alla laadida.

2.3 FFmpeg

Kuna tänapäeval on muutunud väga tavaliseks see, et internetis saadaval olev video ei jõua kasutajani ühes tükis, vaid video ja heli saadetakse eraldi voogudena, siis youtube-dl'iga video allalaadimisel laaditakse tegelikult alla video ja heli eraldi failidena. Kuna kasutaja on huvitatud ühest failist, mis sisaldab nii videot kui ka heli, siis peale video ja heli alla laadimist tuleb need taas kokku panna üheks failiks. Selle saavutamiseks on üks lihtsamaid variante kasutada käsurea programmi nimega FFmpeg[1].

FFmpeg on multimeedia raamistik, millega on võimalik erinevaid video ja heli formaate konverteerida ühest formaadist teise. Kuna FFmpeg on programm, mis töötab samuti erinevatel operatsioonisüsteemidel, siis sobib see ideaalselt youtube-dl'iga. Üks FFmpeg

[1] FFMpeg'i koduleht: https://ffmpeg.org/

funktsionaalsus – erinevate meediavoogude panemine ühte konteinerisse – on heaks põhjuseks, miks kasutada seda raamistikku video ja heli failide uuesti ühendamiseks.

Kuna youtube-dl suudab ise FFmpeg'd kasutada, siis ei pea selleks eraldi midagi tegema, et video ja heli ühte faili saada ning piisab sellest, kui youtube-dl'ile parameetriga kaasa panna FFmpeg asukoht. Sellisel juhul pannakse pärast allalaadimise lõpetamist video ja heli ühte faili ning üleliigsed failid kustutatakse.

2.4 Erinevate raamistike valik graafilise liidese arendamiseks

Kuna käsurea programm youtube-dl töötab paljudel erinevatel operatsioonisüsteemidel, siis peaks seda tegema ka sellele kirjutatud graafiline kasutajaliides. Graafiliste liideste tegemiseks on mitmeid võimalusi, aga kui eesmärgiks on, et sama kood töötaks suhteliselt väikese vaevaga erinevatel operatsioonisüsteemidel, siis oleks mõistlik valida mõni raamistik, mis on just mõeldud sellise probleemi lahendamiseks. Üks sellised raamistikke on Electron[1].

Electron raamistik võimaldab teha *native* väljanägemisega rakendusi, kasutades selleks HTML-i JavaScripti ja CSS-i, kuna sisemiselt kasutab see Chromium veebilehitsejat kasutajale liidese kuvamiseks. Tänu erinevate veebitehnoloogiate kasutamisele on mugava ja hõlpsalt kasutatava liidese tegemine suhteliselt lihtne.

Võib öelda, et Electroniga loodud kasutajaliides on nagu tavaline veebileht ning et see oleks reaktiivne, siis oleks mõistlik kasutada mingisugust JavaScripti raamistikku. Reaktiivsete veebilehtede tegemiseks on mitmeid populaarseid raamistikke, nagu näiteks Angular, React või Vue.js. Kuna Angulari raamistikuga olen ise juba eelnevalt tuttav, siis et mitte aega kulutada uue raamistiku tundmaõppimisele, valisin just selle, et kasutajaliidese arendamist lihtsustada.

[1] Electroni koduleht: https://electronjs.org/

2.5 Nõuded kasutajaliidesele

Youtube-dl on peamiselt mõeldud YouTube'ist videote alla laadimiseks, aga võimaldab seda teha ka paljudelt teistelt veebilehtedelt. Kuna erinevatelt veebilehtedelt allalaadimise protsess võib varieeruda, siis loodav kasutajaliides peaks toetama vähemalt YouTube'i.

Siin on välja toodud minimaalsed nõuded, mis peavad olema rahuldatud esialgse rakenduse versiooni arendamisel:

- Kasutajal peab olema võimalus sisestada veebiaadressi, mis viitab videole või videote nimekirjale, mida soovitakse alla laadida.
- Kasutajal peab olema võimalik valida, millise kvaliteediga video soovitakse alla laadida.
- Kasutaja peab nägema alla laaditava video staatust ja progressi.
- Kasutajaliideses peab kuvama allalaadimise järjekorda või nimekirja, kuhu videod lisatakse ja alustatakse automaatselt nende allalaadimist.
- Kasutaja peab saama valida asukoha, kuhu tema arvutis alla laaditavad videod salvestatakse.
- Rakendus peab töötama vähemalt kolmel erineval operatsioonisüsteemil (Windows 10, Ubuntu 18.04 LST, macOS High Sierra).

3 Kasutatud tehnoloogiad

Antud peatükis kirjeldatakse põhilisi tarkvara raamistikke, mida käesoleva rakenduse arendamisel kasutati.

3.1 Electron

Electron on avatud lähtekoodiga tarkvara raamistik, mis võimaldab arendada graafilise kasutajaliidesega töölaua rakendusi, kasutades selleks veebitehnoloogiaid. Sisemiselt kasutab raamistik Chromium visualiseerimise mootorit ja Node.js *runtime*'i ning tänu sellele töötab raamistik nii Windowsi, Linuxi kui ka macOS operatsiooni-süsteemidel. Electron raamistikku on kasutanud ka paljud tuntud rakendused oma graafilise kasutajaliidese arendamiseks, näiteks Atom, Visual Studio Code ja Skype.

Arhitektuuriliselt koosnevad Electroni rakendused kahest protsessist: *browser* ja *renderer*. Esimeses protsessis asub rakenduse põhiloogika ja rakenduse erinevate akende haldamise loogika. See protsess saab omakorda käivitada mitu *renderer* protsessi, mis on aknad, mida kasutaja oma ekraanil näeb. Vastavate akende sisu kuvamiseks kasutab Electron HTML-i ja CSS-i. Erinevate protsesside omavaheliseks andmevahetuseks on olemas eraldi rakendusliides (API), läbi mille saavad protsessid saata andmeid üksteisele nii sünkroonselt kui ka asünkroonselt.

Electron sisaldab endas palju API-sid, läbi mille on võimalik operatsioonisüsteemiga suhelda. Vastavaid API-sid saab kasutada otse JavaScriptis ning tänu sellele ongi võimalik teha rakendusi, mis tunduvad nagu *native* rakendused, kasutades selleks ainult veebitehnoloogiaid.

Kuna Electroni rakendused on põhimõtteliselt nagu veebirakendused, mis jooksevad Chromiumi mootoris, siis võivad nad olla, sarnaselt tavalistele veebibrauseritele, haavatavad erinevatele rünnakutele, nagu näiteks *cross-site scripting* rünnak.

3.2 Angular

Angular on JavaScripti raamistik, mis on mõeldud reaktiivsete veebilehekülgede loomiseks. Angulariga arendamisel tuleb kasutada TypeScripti, mis erinevalt JavaScriptist kasutab muutuja tüüpe juba koodi kirjutamise käigus. Tänu sellele väheneb koodis leiduvate vigade tõenäosus ja enamik vigu tuleb välja koodi kompileerimisel. Kuna brauserid hetkel veel TypeScripti ei toeta, siis tuleb kood JavaScripti kompileerida. Angular raamistiku kaheks põhiliseks osaks on komponendid, mida kasutakse kasutajaliidese ülesehitamiseks, ja teenused, mida komponendid või teised teenused kasutavad.

4 Kasutajaliidese arendamine

Järgnevas peatükis on kirjeldatud projekti struktuuri ja kasutajaliidese arhitektuuri, komponente ja suhtlust välise käsurea programmiga youtube-dl.

4.1 Projekti struktuur

Projekti sõltuvused välistest teekidest, peale käsurea programmide, on hallatavad läbi Node Package Manageri ning on kirjeldatud failis `package.json`. Selles failis on täpselt kirjas, milliseid teeke ja versioone nendest projektis vaja läheb. Lisaks sisaldab see fail kõiki vajalikke skripte, mida projekti ehitamiseks ja tööle panemiseks vaja läheb, ja ka skripte, millega erinevate operatsioonisüsteemide jaoks neile sobivaid installeerimisfaile ehitada. Käsurea programmide sõltuvusi selles projektis automaatselt ära ei lahendata ja vajalikud programmid on lisatud projekti käsitsi.

Üldiselt koosneb projekt kahest osast. Esiteks on olemas Electroni osa rakendusest ja teiseks on olemas Angulari osa, kus asub enamik loogikast ja funktsionaalsusest.

4.2 Rakenduse üldine arhitektuur

Kuna kasutajaliides on kirjutatud Electron raamistiku peale, siis on rakenduses üks *browser* protsess ja üks *renderer* protsess. *Browser* protsess on koht, kust rakenduse eluiga alguse saab. Selle protsessi põhiliseks eesmärgiks on *renderer* protsesside ehk rakenduse akende tegemine ja haldamine ning rakenduse sulgemine, kui mingid kindlad tingimused on täidetud, nagu näiteks kasutaja poolt kõigi rakenduse akende sulgemine. *Browser* protsess on ühtlasi koht, kus saab vajadusel teha mingisugust eeltööd enne, kui rakenduse akent kuvada, näiteks lugeda konfiguratsiooni faili jms. Antud rakenduses sisaldab *browser* protsess minimaalset loogikat rakenduse toimimiseks, mis tähendab, et protsess sisaldab vaid loogikat selle kohta, kui suurelt ja millises kohas ekraanil tuleks aken kuvada. Kuna kasutajaliides koosneb ainult ühest aknast, siis on kasutusel ka ainult üks *renderer* protsess, mida võib võtta kui iseseisvat brauseri akent, kus sees kuvatakse Angulari rakendust. Vajadusel saavad *renderer* ja *browser* protsess omavahel suhelda ja üksteisele infot edastada, aga kuna käesolevas rakenduses asub kogu loogika Angulari poolses osas ja rakendus koosneb ainult ühest aknast, siis mingisugust suhtlust nende kahe protsessi vahel ei toimu. Seda funktsionaalsust on vaja kasutada sellisel juhul, kui

rakenduse ühest aknast on vaja saata andmeid teise. Sellisel juhul oleks *browser* protsess vahendaja rollis, edastades ühelt *renderer* protsessilt vastu võetud andmed teisele.

4.3 Angulari rakenduse arendamine

Angulari rakenduse poolne osa on üsna lihtne ja väike ning kuna tegemist on põhimõtteliselt veebilehega, siis kuvasid, kus kasutajale infot kuvatakse, on ainult üks. Kasutajaliidese peamiseks komponendiks on `home` komponent, mis sisaldab endas päise ja allalaaditavate videote nimekirja osa. Nimekiri ise kasutab `download-item` komponenti, mille põhiliseks eesmärgiks on allalaadimise progressi kuvamine. Lisaks on liideses veel kaks modaalakna komponenti, üks seadete kuvamiseks ja teine video kvaliteedi valiku tegemiseks. Rohkem eraldiseisvaid kasutajaliidese osasid rakendus ei sisalda ning kõik ülejäänud tööks vajalik loogika asub erinevates teenustes. Teenustest tuleb täpsemalt juttu järgmises peatükis.

Angular raamistiku üks suuri eeliseid on see, et kui kuskil andmemudelis mingi info muutub, siis uuendatakse ka vastavat kasutajaliidese osa, kus seda infot kuvati. Kuna käesolev rakendus suhtleb ka väliste programmidega, mis otseselt ei ole Angulariga seotud, siis sellisel juhul ei ole Angular teadlik, et mingid andmed muutusid (kuna muudatus ei toimunud läbi kasutajaliidese) ja kasutajale uuenenud infot ei kuvata. Selline olukord juhtub näiteks video allalaadimisel, kui youtube-dl tagastab uue allalaadimise progressi ning et uuenenud info ka liideses näha oleks, tuleb Angularile eraldi teada anda, et ta kontrolliks andemuudatusi ja vajadusel kasutajaliideses neid uuendaks.

4.4 Kasutajaliidese komponendid ja teenused

Liides on üldjoontes nagu tavaline veebileht, kus navigeerimist erinevate lehekülgede vahel ei toimu ja kogu info kuval muutub vastavalt kasutaja tegevustele reaktiivselt tänu Angular raamistikule. Järgnevalt on kirjeldatud rakenduses olevaid komponente ja teenuseid ning nende põhilisi ülesandeid. Kasutajaliidesest tehtud ekraanipildid on toodud töö lisades (Lisa 1 kuni Lisa 4).

4.4.1 Päis

Päise komponent sisaldab ühte teksti sisestusvälja, kuhu kasutaja saab kirjutada või kleepida veebiaadressi, millelt ta soovib video alla laadida. Veebiaadressi kinnitamiseks

eraldi nuppu ei ole lisatud ning kasutaja peab selleks kasutama ENTER klahvi. Lisaks on komponendis seadete nupp, millele vajutades saab kasutaja rakenduse seadeid muuta.

4.4.2 Allalaaditavate videote nimekiri

Komponent on mõeldud erinevate allalaaditavate videote kuvamiseks ja allalaadimiste järjekorra paika panemiseks.

4.4.3 Video nimekirja element

Nimekirja element näitab kasutajale video nime ja hetke olekut allalaaditavast videost. Lisaks näidatakse kasutajale mitu protsenti videost on juba alla laaditud, milline on allalaadimise kiirus, kui palju on allalaadimise lõpuni veel aega jäänud ja kui vastav info on olemas, siis ka allalaaditava faili suurust. Juhul, kui video koosneb mitmest osast ehk eraldi videost ja helist, kuvatakse kasutajale ka seda, mitmendat osa hetkel alla laetakse. Kui video kohta leidub pisipilt, siis näidatakse seda samuti komponendis. Peale video alla laadimist ilmub ikoon, millele vajutades avatakse kaust, kuhu video salvestati.

4.4.4 Kvaliteedi modaal

Kvaliteedi modaal avaneb peale seda, kui kasutaja on sisestanud veebiaadressi ja sellelt on leitud video, mis on saadaval erineva kvaliteediga. Sel juhul kuvatakse nimekirja võimalikest kvaliteedi näitajatest, millest kasutaja saab omale sobiva valiku teha. Vaikimisi on valitud parima kvaliteediga video.

4.4.5 Seadete modaal

Antud komponendis kuvatakse erinevaid seadeid, mida kasutajal on võimalik muuta. Valminud kasutajaliideses on valikus ainult üks seade, mida saab muuta, ning see on allalaaditavate videote asukoha valimine. Tulevikus on võimalik sinna lisada ka teisi kasutaja poolt muudetavaid seadeid.

4.4.6 Allalaadimiste haldamise teenus

Nimetatud teenus on üks tähtsamaid osasid sellest rakendusest. Nimelt selle teenuse ülesandeks on kasutaja poolt valitud videote allalaadimise järjekorda lisamine ja nende allalaadimine läbi käsurea programmi. Lisaks on selle teenuse ülesanne järgmise video allalaadimise alustamine, kui eelmine on lõpule jõudnud, juhul kui järjekorras on veel alla laadimata videoid.

4.4.7 Konfiguratsiooni teenus

Antud teenuse eesmärk on tegeleda rakenduse erinevate seadete ja konfiguratsioonidega ning vastavat infot pakkuda ka teistele komponentidele ja teenustele. Antud teenus vastutab ka kasutaja seadete sisse lugemise ja salvestamise eest. Näiteks kasutab seadete modaalaken seda teenust info saamiseks ja salvestamiseks ning igasugused väliste programmide asukohtade erisused, mis erinevates operatsioonisüsteemides võivad olla, lahendatakse vastava teenuse sees.

4.4.8 Electroni ja Node.js'i moodulitega suhtlemise teenus

See teenus on vajalik selleks, et saaks lihtsalt ja mugavalt otse Angulari keskkonnast kasutada erinevaid Electroni ja Node.js API-sid. Näiteks saab läbi selle teenuse pöörduda väliste programmide poole.

4.5 Suhtlus väliste käsurea programmidega

Rakendus kasutab kahte eraldiseisvat käsurea programmi, ühte neist otse ja teist kaudselt. FFmpeg on programm, mille poole kasutajaliides ise otse ei pöördu, vaid seda kasutab youtube-dl, et videod, mis allalaadimisel koosnesid mitmest osast, lõpuks ühte konteinerisse kokku panna. youtube-dl on ise võimeline selle programmi kasutaja arvutist üles leidma juhul, kui see on arvutis olemas. Kuna valminud graafiline liides peaks võimalikult vähe kasutajalt nõudma, et korrektselt töötada, siis vastav programm on installeerimisfailis kaasas ning selle asukoht antakse youtube-dl'ile parameetriga ette.

Käsurea programmiga suhtlemiseks kasutatakse Node.js `childProcess` API-s olevaid `spawn` ja `execFile` meetodeid, millega on võimalik väliseid programme käivitada. Need meetodid erinevad üksteisest selle poolest, et `spawn` meetodiga programmi käivitades tagastatakse standardväljundisse tulev info rea kaupa, aga `execFile` meetodi puhul tagastatakse kogu programmi väljund ühekorraga siis, kui see on oma töö lõpetanud.

Video allalaadimiseks käivitab kasutajaliides välist programmi kaks korda. Esimene päring on selleks, et kasutaja sisestatud veebiaadressil oleva video kohta infot saada. Saadav info on JSON formaadis ja sisaldab endas infot erinevate video kvaliteetide kohta ning üldist infot video kohta, näiteks pealkirja jms. Kuna JSON formaadis tekstist oleks mõistlik teha JavaScript objekt, siis oleks vaja kogu teksti korraga. Selle jaoks ongi

kasutatud `execFile` meetodit, mis teeb täpselt seda ning muudab JSON-ist objekti tegemise väga lihtsaks. Kuid sellel on ka mõni halb külg, nimelt kui kasutaja soovib korraga alla laadida suuri video nimekirjasid, siis käsurea programmi poolt tagastatav JSON muutub väga suureks, `execFile` meetodi puhver saab täis ja käsurea programmi poole pöördumine ebaõnnestub. Üks lahendus oleks puhvri mahtu suurendada, aga ka sellel oleks omad piirid, mistõttu oleks kõige õigem hoopis `execFile` asemel kasutada `spawn` meetodit, mis teeks loogika veidi keerukamaks, aga samas töökindlamaks. See probleem tuleks kindlasti rakenduse edasise arendamise käigus parandada ja käesoleva versiooni puhul ei ole garanteeritud, et nimekirjade, mis sisaldavad rohkem kui kakskümmend videot, allalaadimine õnnestub.

Teine päring, mis käsurea programmile tehakse, on selleks, et video reaalselt alla laadida. Selleks on kasutatud `spawn` meetodit, mille iga standardväljundi rida kasutatakse video allalaadimise progressi ja kiiruse andmete saamiseks. Kuna youtube-dl'i poolt väljastatav info on kindlas formaadis, siis kasutades regulaaravaldisi on vajaliku info kätte saamine üsnagi lihtne. Peale programmi väljundist iga uue rea saamist uuendatakse ka graafilises kasutajaliideses vastava video progressi.

5 Rakenduse ehitamine erinevatele operatsioonisüsteemidele

Järgnevalt on kirjeldatud protsessi, kuidas toimub rakenduse ehitamine erinevate operatsioonisüsteemide jaoks ning millised probleemid selle käigus esinesid.

5.1 Erinevad operatsioonisüsteemid

Electron raamistiku üks peamisi eesmärke on see, et oleks võimalik lihtsalt luua rakendusi, mis töötavad paljude erinevate operatsioonisüsteemide peal. Tänu sellele ei pea arenduse käigus väga palju mõtlema, millised erisused on erinevate platvormide vahel. Et rakenduse pakkimine oleks kergem, on mõistlik kasutada mingisugust tööriista, mis keerukama töö ära teeb ning ei nõua väga palju spetsiifilisi teadmisi sellest, kuidas see protsess täpselt välja näeb. Selliseid tööriistu, mille vahel valida, on mitmeid, kuid käesolevas töös on kasutatud Node.js paketti nimega electron-builder. Vajalik konfiguratsioon, mida selleks protsessiks vaja läheb, on kirjeldatud `electron-builder.json` failis ja üldjuhul rohkem midagi tegema ei pea.

5.2 Valitud operatsioonisüsteemid ja installeerijad

Käesolev rakendus on mõeldud töötama kolmel erineval operatsioonisüsteemil: Windows, Ubuntu ja macOS. Electron-builder paketiga on võimalik rakendust kokku ehitada erinevatesse formaatidesse, nagu näiteks nsis (Nullsoft Scriptable Install System), msi (Windows Installer) ning AppX. Viimane neist on mõeldud rakenduste jaoks, mida oleks võimalik levitada Microsoft Store'i kaudu ja eeldab, et süsteem, kus seda ehitatakse, oleks Windows. Linuxi puhul on toetatud põhimõtteliselt kõik erinevad distributsioonid ja nendes kasutatavad installeerijad või paketid. Kuna valminud kasutajaliidese jaoks valitud Linuxi distributsioon, kus see töötama peaks, oli Ubuntu, siis selle jaoks ehitatakse see kokku deb (Debian package) formaati. macOS jaoks kasutatud formaat on dmg (Apple Disk Image), mis on üks populaarseimaid formaate, millega rakendusi selles keskkonnas levitada.

5.3 Rakenduse ehitamine

Käesoleva rakenduse arendamisel kasutav operatsioonisüsteem oli Windows 10 ning selleks, et vastavas keskkonnas ehitada rakendust teistele operatsioonisüsteemidele, on vaja vastavate operatsioonisüsteemide spetsiifilisi sõltuvusi ja faile. Electron-builder teeb üldjuhul kõik vajaliku ära ja arendaja ei pea ise detailidega tegelema, kuid käesoleva töö arendamise hetkel seda tegema pidanud electron-builder paketi funktsionaalsus ei töötanud, sest serverid, kust vajalikke faile saada, ei olnud kättesaadavad. Seepärast on kasutajaliidese rakenduse erinevad versioonid kokku ehitatud just nendes operatsioonisüsteemides, kus rakendus on mõeldud töötama. Windowsi versioon on ehitatud Windows 10'es ja Linuxi versioon on ehitatud Ubuntu 18.04 LST'is. macOSi jaoks on kasutatud macOS High Sierrat, sest antud juhul ei ole muud võimalust, kui kasutada vastavat operatsioonisüsteemi, sest teistes OS-ides seda teha võimalik ei ole[1].

5.4 Välised käsurea programmid

Kuna valminud kasutajaliides vajab töötamiseks ka kahte teist programmi, siis selle probleemi lahendamiseks oleks kaks võimalust. Esimene võimalus oleks see, et kasutaja peaks ise selle eest hoolitsema, et kõik vajalikud programmid oleks tema arvutis olemas, kuid see ei oleks just kõige mugavam variant. Et kasutajale kasutajaliidese kasutamine võimalikult lihtne oleks, siis oleks mõistlik kõik vajalikud programmid installeerijasse kaasa panna. Selle jaoks on võimalik electon-builder konfigureerida nii, et olenevalt operatsioonisüsteemist pannakse õiged välised programmid installeerijasse ja need jõuavad kasutajaliidest installeerides automaatselt kasutaja arvutisse.

FFmpeg ja youtube-dl programmid on projekti lisatud käsitsi ja mingisugust automaatset sõltuvuste haldamist nende puhul kasutatud ei ole. Kui neist programmidest peaks ilmuma uuemad versioonid, siis nende uuendamine toimub käsitsi. Mõlemast programmist on olemas kolm erinevat versiooni, mis pannakse siis vastavalt õige operatsioonisüsteemi installeerijasse kaasa.

[1] https://www.electron.build/multi-platform-build

5.5 Valmis ehitatud rakendus

Enne kui electron-builder teeb installeerija valmis, ehitatakse rakendus valmis sellisena, millisena ta kasutaja arvutis peale installeerimist välja näeb. Üldiselt on valmis ehitatud rakendus erinevates operatsioonisüsteemides sama struktuuriga, erinevused on ainult platvormi spetsiifilistes failides, mida rakendus tööks vajab. Kõikide operatsioonisüsteemide korral on olemas external, resources ja locales nimelised kaustad (macOS puhul on üldiselt locales kaustas olevad failid hoopis kaustas resources). External kaustas asuvad välised programmid youtube-dl ja FFmpeg, locales kaustas on erinevad keelefailid, mida Electron kasutab ning resources kaustas on `app.asar` ja `electron.asar` failid. Asar tüüpi failid on põhimõtteliselt nagu tavalised arhiivi failid, kuid on loodud spetsiaalselt Electroni raamistiku jaoks. Asar arhiividest saab sisu lugeda ilma, et oleks vaja kogu arhiiv enne lahti pakkida ning Electron kasutab seda seepärast, et Windowsi keskkonnas ei tekiks pikkade failinimedega probleeme ja et rakenduse lähtekoodi tavakasutaja ees natuke peita. `app.asar` fail sisaldab kogu kirjutatud rakenduse koodi (HTML, CSS, JavaScript) ja `electron.asar` sisaldab Electron raamistiku tööks vajalikku JavaScripti koodi.

Installeerijate suurus jääb erinevatel platvormidel umbes 65 ja 80 MB-i vahele ning peale installeerimist on rakenduse suuruseks Windowsi ja macOSi platvormidel ligikaudu 200 MB ja Ubuntu operatsioonisüsteemil ligikaudu 290 MB.

5.6 Erinevate operatsioonisüsteemidega kaasnenud probleemid

Electron on raamistik, mis on mõeldud just selleks, et arendaja ei peaks mõtlema operatsioonisüsteemi erinevuste peale, kuid mõningaid probleeme võib ikkagi ette tulla. Üks probleem, mis arenduse käigus välja tuli, oli faili asukoha formaadi erinevus. Windowsis kasutatakse „\“, aga Linuxis ja macOSis „/“, et failide ja kaustade struktuuri kirjeldada. Probleemist üle saamiseks tuleks kasutada mingisugust teeki, mis olenevalt operatsioonisüsteemist selle probleemi ära lahendab. Kuna Electron on ehitatud Node.js'i peale, siis on võimalik kasutada Node.js teeki nimega `path`.

Kuna graafiline kasutajaliides kasutab oma töös väliseid käsurea programme ja käivitab neid taustal, siis on vajalik, et liides teaks täpselt, kus need programmid asuvad. Kui rakendus lokaalselt tööle panna, siis ei teki probleeme väliste programmide üles

leidmisega, sest olenemata operatsioonisüsteemist asuvad need programmid ühes kindlas kohas. Kuid kui rakendus installeerida ja seejärel rakendust kasutada, siis ei ole erinevates süsteemides välised programmid enam samades kohtades. Sellest probleemist üle saamiseks on antud rakendusse loodud üks Angulari teenus, kus asub loogika, mis siis olenevalt operatsioonisüsteemist oskab tagastada failide õige asukoha. Tänu sellele teenusele ei pea ülejäänud rakendus selle teenuse kasutamisel spetsiifilisest loogikast midagi teadma. See oli ka üks kõige suuremaid probleeme, mis tuli lahendada, et graafiline kasutajaliides töötaks erinevate platvormide peal.

5.7 Testimine

Käesolevas töös automaattestimist kasutatud ei ole, kuid tulevikus tuleks seda kindlasti teha. Kasutajaliidesele automaattestide kirjutamine ei sõltu operatsioonisüsteemist, kuid keerukust lisab väliste käsurea programmidega suhtluse testimine.

Kõik, mis puutub rakenduse testimisse, on tehtud manuaalselt. Testimine on läbi viidud kolmel erineval operatsioonisüsteemil ja näeb välja selline:

- Rakenduse installeerimine
    - Testimisel jälgiti, et rakenduse installeerimisel ei esineks tõrkeid ega probleeme. Installeerimist testiti mitme operatsioonisüsteemiga – Windows, Linux ja macOS.
    - Oodatav tulemus: rakendus on edukalt installeeritud.
- Veendumine, et rakendus käivitub
    - Testimisel jälgiti, et rakendus käivituks kiirelt ja tõrgeteta. Rakenduse käivitamist testiti kasutajaliidese loomise ja täiendamise protsessi jooksul korduvalt, st peaaegu pärast iga muudatuse tegemist.
    - Oodatav tulemus: rakendus on probleemivabalt käivitunud ja kasutajale kuvatakse kogu oodatav funktsionaalsus (päis ja tegevusnupud).
- Failide allalaadimise asukoha muutmine

- Testimisel jälgiti, et vastav funktsionaalsus seadetes avaneks, kasutaja saaks failide allalaadimise asukohta muuta ning et pärast muudatust failid salvestuksid uude asukohta. Lisaks testiti, et rakendus arvestaks uue asukohaga ka pärast rakenduse sulgemist ja taasavamist. Ühtlasi testiti failide allalaadimise asukoha muutmist sel ajal, kui video allalaadimine oli pooleli.

- Oodatav tulemus: kasutajal õnnestub failide allalaadimise asukoha muutmine, pärast mida salvestab rakendus uued failid uude asukohta ning rakenduse taasavamisel kuvab rakendus failide allalaadimise asukohana uut (viimati salvestatud) väärtust. Poolelioleva video allalaadimist failide allalaadimise asukoha muudatus ei mõjuta.

- Üksiku video ja nimekirjade kvaliteedi valiku testimine

  - Testimisel jälgiti, et kasutajale kuvatakse kvaliteedi valimise aken, parim kvaliteet oleks alati eelvalitud, kasutaja saaks kvaliteedi valikut muuta ning et rakendus arvestaks kasutaja poolt tehtud valikuga. Kvaliteedi valimist sai testitud mitu korda – näiteks madalaima, keskmise ja kõrgeima väärtusega. Ühtlasi sai testitud olukorda, kus kasutaja ei saa valikut muuta ehk videot on võimalik alla laadida vaid ühe kindla kvaliteediga.

  - Oodatav tulemus: rakendus on võtnud arvesse kasutaja tehtud valiku ja alla laadinud vastava kvaliteediga video.

- Faili allalaadimise protsessi testimine

  - Testimisel jälgiti, et fail laaditakse alla tõrgeteta.

  - Oodatav tulemus: rakendus on faili probleemivabalt alla laadinud.

- Kontrollimine, kas alla laaditud videod reaalselt eksisteerivad kasutaja arvutis ja töötavad laitmatult

- Testimisel jälgiti, et videofail oleks olemas, avaneks ja töötaks korrektselt (nii pilt kui heli). Videote korrektselt töötamist testiti mitme erineva kvaliteediga ja eri asukohtadesse laaditud failidega.
- Oodatav tulemus: allalaaditud video esitamine täispikkuses toimib tõrgeteta.

- Rakenduse desinstalleerimine
  - Testimisel jälgiti, et rakenduse desinstalleerimisel ei esineks tõrkeid ega probleeme. Desinstalleerimist testiti mitme operatsioonisüsteemiga – Windows, Linux ja macOS.
  - Oodatav tulemus: rakendus on edukalt kasutaja arvutist eemaldatud.

6 Tuleviku plaanid

Kuna antud töös valminud rakendus on suhteliselt väikese funktsionaalsusega, siis et antud rakendusest tulevikus rohkem kasu oleks, tuleks funktsionaalsust tunduvalt suurendada. Kuna käsurea programm youtube-dl toetab videote allalaadimist peale YouTube'i ka paljudelt muudelt veebilehekülgedelt, siis tuleks kasutajaliidest täiendada selliselt, et erinevad veebileheküljed oleks toetatud. youtube-dl väljastab infot üsna standardse formaadiga, seega on tegelikult võimalik juba hetkel valmis arendatud liidesega mõnelt muult veebisaidilt videoid alla laadida, kuid käsurea programmilt saadud info võib formaadi poolest siiski natuke erineda. Sellest tulenevalt ei pruugi kasutajaliides infot korralikult kuvada või üldse töötada.

Järgnevalt on toodud välja mõni funktsionaalsus, mida tulevikus võiks liidesele juurde arendada.

- Mitme video paralleelselt allalaadimine. Vastavat kogust peaks kasutaja saama seadetes muuta.
- Kasutajal võiks olla võimalus alla laadida ainult heli ning vajadusel seda endale sobivasse formaati konverteerida.
- Kasutajal peaks olema võimalik seadetes märkida, millise kvaliteediga video või heli ta üldjuhul soovib, et ei peaks iga video puhul kvaliteeti käsitsi määrama.
- Kasutajal võiks olla võimalik sisestada erinevate veebilehtede kontode kasutajanime ja parooli, et vajadusel alla laadida selliseid videoid, mis muidu võivad olla seotud vanusepiiranguga või lihtsalt ainult sisse logitud kasutajale kättesaadavad.
- Et kasutajal oleks veelgi mugavam rakendust kasutada, siis võiks olla erinevate veebibrauserite jaoks laiendused, mis kuvaksid allalaadimise nuppu vastava video juures ning millele vajutades hakkaks rakendus seda video automaatselt alla laadima.

- Kasutajale võiks kuvada operatsioonisüsteemi-põhiseid teateid selle kohta, kui mingi video allalaadimine jõudis lõpule või juhtus mingi muu sündmus, millest kasutajat võiks teavitada.

- Rakendus võiks kontrollida uuenduste olemasolu ja vastavalt võimalusele rakendust ise uuendada, kui kasutaja selleks loa annab.

7 Kokkuvõte

Antud töö peamiseks eesmärgiks oli kirjutada lihtne graafiline kasutajaliides käsurea programmile youtube-dl, läbi mille oleks kasutajal võimalik YouTube'ist erinevaid videoid ja videote nimekirjasid alla laadida.

Töö tulemusena valminud rakendus vastab kõikidele püstitatud eesmärkidele.

Rakendus on valmis kirjutatud Electron raamistiku abil ning töötab kolmel erineval operatsioonisüsteemil: Windows 10, Ubuntu ja macOS. Kasutajaliides on tehtud Angular raamistiku baasil, tänu millele on liides reaktiivne ning andmed muutuvad reaalajas.

Programmist on olemas versioonid erinevatele operatsioonisüsteemidele, mida on võimalik vastavates süsteemides installeerida ja vajadusel desinstalleerida. Linuxi ja macOSi versioonid vajavad töötamiseks Pythonit ning Windowsi versiooni puhul on vajalik, et kasutaja süsteemis oleks olemas Microsoft Visual C++ 2010 Redistributable Package. Kuna need tingimused on üldjuhul vastavates operatsioonisüsteemides täidetud ja ülejäänud vajaminevad programmid on installeerimisfailides olemas, saab töös valminud programmi lihtsalt erinevatesse operatsioonisüsteemidesse installeerida.

Kasutatud kirjandus

[1] youtube-dl programm [WWW] https://ytdl-org.github.io/youtube-dl/index.html

[2] Electron raamistik [WWW] https://en.wikipedia.org/wiki/Electron_(software_framework)

[3] Electron raamistik [WWW] https://electronjs.org/

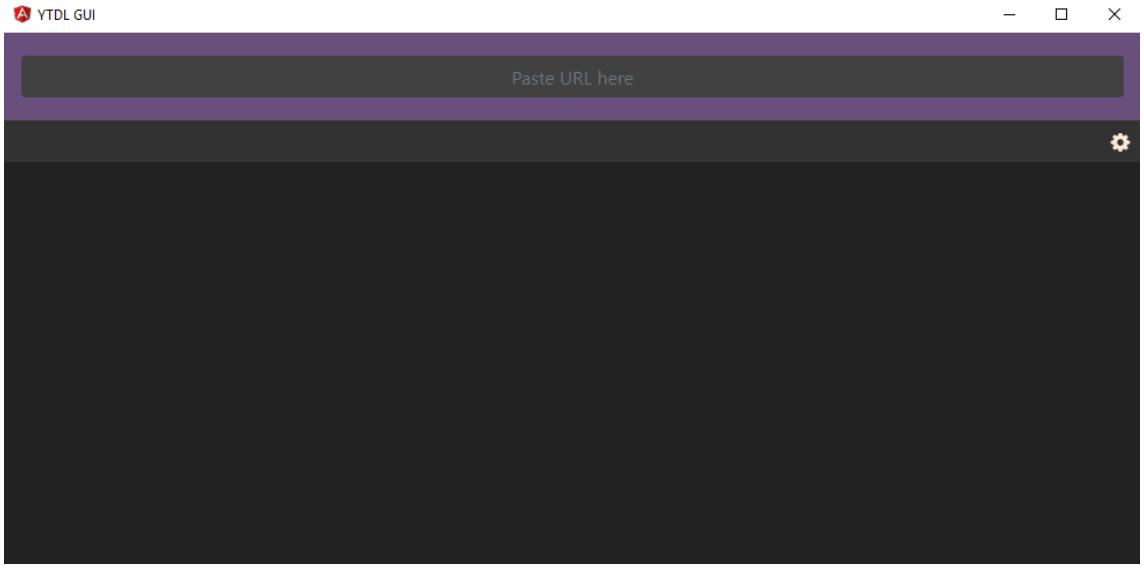
[4] Bootstrap teek [WWW] https://getbootstrap.com/

[5] Bootstrap komponentide teek [WWW] https://ng-bootstrap.github.io/

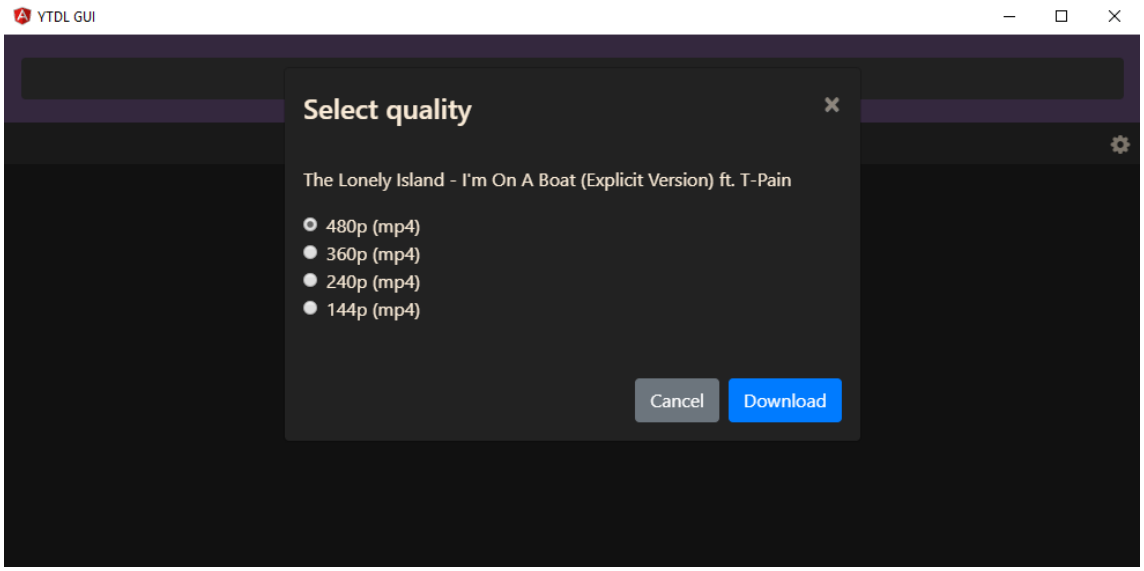
[6] Angular raamistik [WWW] https://angular.io/

[7] Alusprojekt [WWW] https://github.com/maximegris/angular-electron

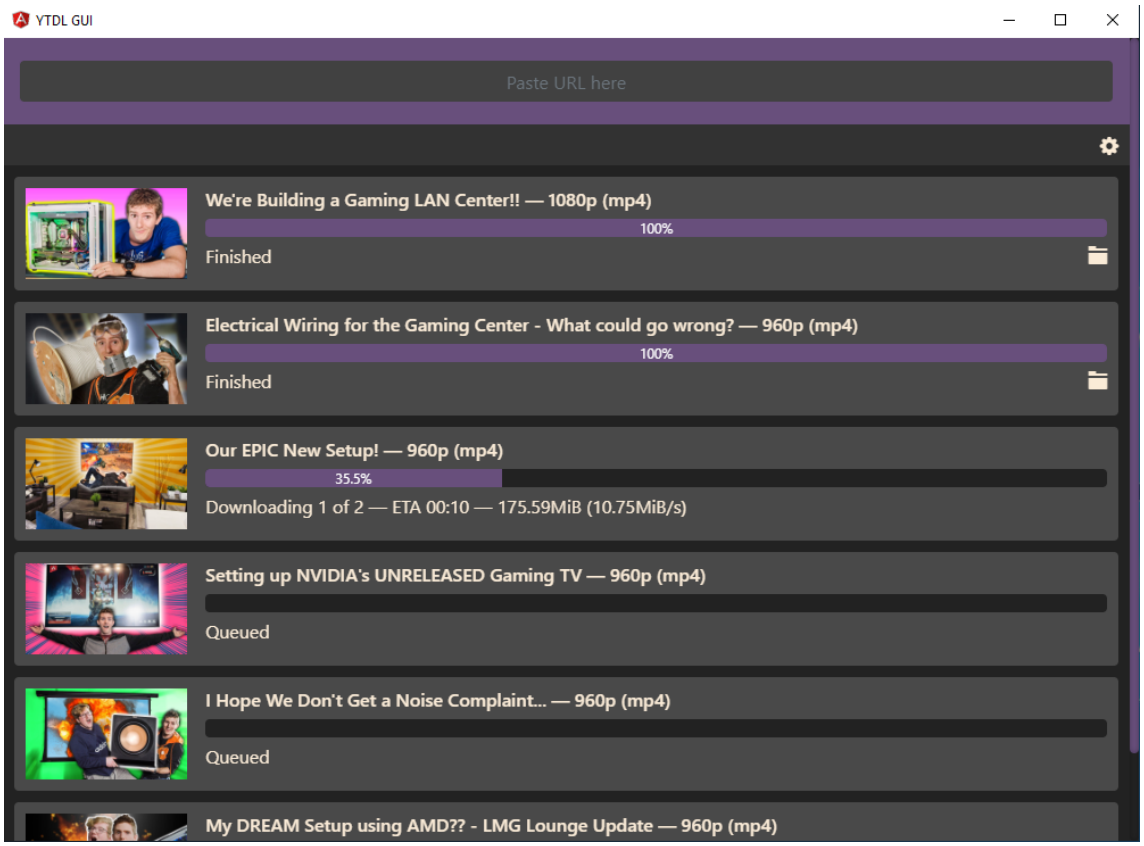
Lisa 1 – Kasutajaliidese tavavaade



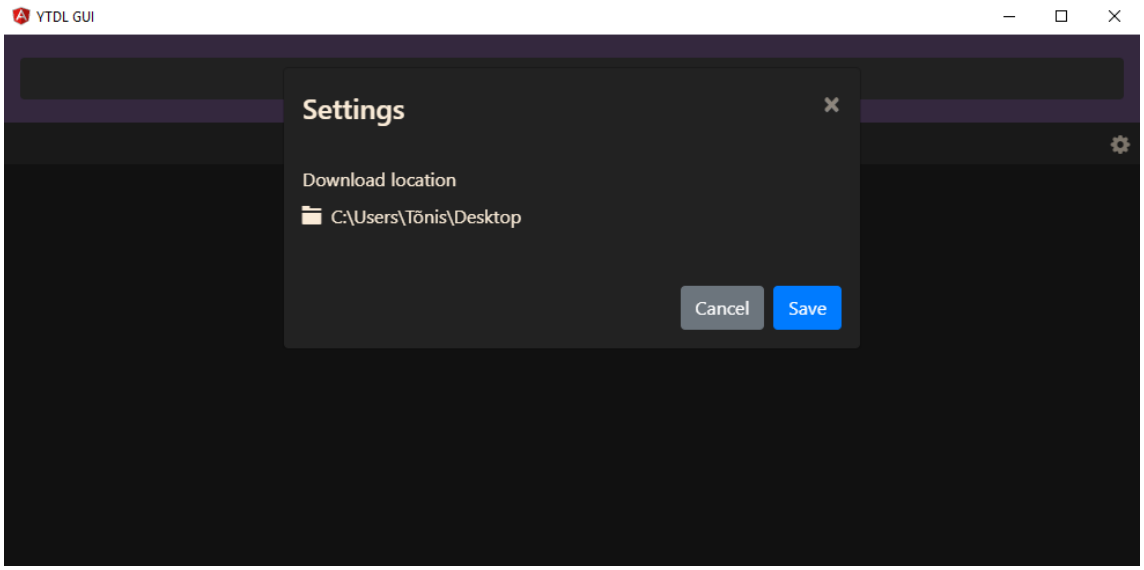
Lisa 2 – Kvaliteedi valiku modaal



Lisa 3 – Allalaaditavate videote nimekiri



Lisa 4 – Seadete modaal



Lisa 5 – Allalaadimise protsessi joonis