TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Henry Kaljulaid

Lõikelaua ajaloo jagamise rakendus

Bakalaureusetöö (9 EAP)

Juhendaja: Alo Peets

Tartu 2025

Lõikelaua ajaloo jagamise rakendus

Lühikokkuvõte:

Bakalaureusetöö eesmärk oli luua rakendus, mille abil saab lõikelaua ja selle ajaloo sünkroniseerida mitme arvuti vahel. Loodud rakendus jälgib aktiivselt lõikelaua sisu ning salvestab iga kopeerimise käigus sealt info andmebaasi. Nii saavad kõik arvutid, kuhu rakendus on paigaldatud ja mis on ühendatud sama kasutajaga ühise serveri külge, ligipääsu ühisele lõikelaua ajaloole. Eelnev võimaldab kasutajal hõlpsalt kasutada varasemalt kopeeritud teavet sõltumata kasutatavast seadmest.

Lõputöös tutvustatakse lõikelaua süsteemide toimimise põhimõtteid erinevates operatsioonisüsteemides, analüüsitakse olemasolevaid lahendusi ning tuuakse välja nende tugevused ja puudused. Töö käigus valmis lahendus, mis koosneb serverist ja klientrakendusest. Serveri API kiht on kirjutatud programmeerimiskeeles Go ning see salvestab andmed Postgres andmebaasi ja serveerib HTML faili veebilehitseja tarbeks. Klientrakenduse, mis suhtleb lõikelaua, kasutajaliidese ja serveriga, jaoks kasutati Go'd, C'd ja veebitehnoloogiaid. Dokumendis antakse ülevaade tehnoloogiatest, lahendustest ja nende valiku põhjustest.

Võtmesõnad:

Lõikelaud, rakendus, Linux, Windows, X11

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Clipboard History Sharing Application

Abstract:

The aim of this bachelor's thesis was to create an application that synchronizes the clipboard and its history between multiple computers. The app monitors the clipboard for changes and saves all available data to a database whenever something is copied. That way all the computers that have the application and are connected to the same server using the same account have access to the same clipboard history. This enables the user to use previously copied information regardless of the device.

The thesis explains the clipboard systems in different operating systems, analyzes existing solutions and their strengths and weaknesses. During the thesis, a solution was developed, consisting of a sever and a client application. The server API is written in the Go programming language, it stores data in a Postgres database and serves an HTML file for web browsers. The client application, which communicates with the clipboard, user interface, and server, uses Go, C and web technologies. In this document the used technologies and the reason behind the choices is given.

Keywords:

Clipboard, application, Linux, Windows, X11

CERCS: P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Arvuteid kasutades on lõikelaud laialdaselt kasutatud funktsionaalsus. Selle abil saab infot, olgu selleks tekst, pilt või midagi muud, liigutada erinevate programmide vahel või dubleerida samas programmis. Kui pooleli on suurem info kogumine, võib kopeerimise sagedus olla päris kõrge.

Selleks, et uue info kopeerimisel lõikelaual olev info kaotsi ei läheks, on välja mõeldud lõikelaua ajalugu. Lõikelaua ajaloo kasutamisel kirjutatakse kopeerimisel lõikelaual olev info üle, aga seal olnud andmed jäävad ajalukku alles, et neid ei peaks uuesti otsima ja kopeerima, vaid saab mõne nupuvajutusega eelmist infot kleepida.

Kui kasutusel on mitu arvutit ja nende vahel on vaja infokilde liigutada, saab seda teha mitmel viisil. Kõige mugavam on kasutada lõikelaua sünkroniseerimist. Näiteks on uuemates Windowsi versioonides võimalik lõikelaud siduda Microsofti kontoga ehk kopeerimisel saab teises arvutis, milles on sama kontoga sisse logitud, kleepida ilma lisa sammudeta. Sellel ja mitmel teisel lahendusel on aga mitmeid puudusi. Lahendus töötab kas ainult ühel operatsioonisüsteemil, ei edasta kogu võimalikku infot lõikelaualt või on sünkroniseerimine muul viisil puudulik.

Käesoleva töö eesmärk on luua lõikelaua ajaloo jagamise rakendus, millega saab ajaloo sünkroniseerida mitme arvuti vahel. Erinevate arvutite vahel sünkroniseerimiseks salvestatakse lõikelaua seisud andmebaasi, millele mitmest arvutist ligi pääseb. Suurema seadmete toe nimel töötab loodud rakendus nii Windowsi kui Linuxi operatsioonisüsteemidel. Lisades on aadress GitHubi repositooriumile, kust leiab kirjutatud koodi.

Töö koosneb viiest peatükist. Esimeses kahes peatükis kirjeldatakse täpsemalt erinevaid lõikelaua süsteeme ja analüüsitakse olemasolevaid lahendusi ning nende puudusi. Järgmises kahes peatükis kirjeldatakse valminud lahenduse ülesehitust ja kasutatud tehnoloogiaid. Viimases peatükis analüüsitakse tulemusi ning tuuakse välja mõned punktid, mida saaks tulevikus edasi arendada.

1. Lõikelaua süsteemid

Arvuti lõikelaud on mehhanism, mille abil saab ühest programmist ajutiselt infot kopeerida, et see samasse või teise programmi kleepida. Info võib olla mistahes formaadis. Kõige tihedamini kopeeritakse teksti ja pilte, aga iga programm võib ka enda formaadi defineerida ja selles formaadis infot lõikelauale panna. Lõikelaua funktsionaalsust kasutatakse tihti ka failide puhul. Failidega toimetamisel ei panda tavaliselt tervet faili sisu lõikelauale, vaid tehtud tegevus (kopeerimine või lõikamine) ja faili rada. Kleepimisel loeb vastuvõttev rakendus tegevuse ja faili raja ning vastutab failiga toimetamise eest.

Tavaliselt on operatsioonisüsteemis CTRL+C klahvikombinatsioon seadistatud kopeerimiseks ja CTRL+V kleepimiseks. Lisaks kasutatakse CTRL+X lõikamiseks ehk info kustutatakse algsest asukohast ära. Brad Myersi [1] sõnul said need kombinatsioonid alguse Apple'i esimesest graafilisest arvutist Lisa, kus valiti Z, X, C ja V vastavalt tühistamise (ingl *undo*), lõikamise (ingl *cut*), kopeerimise (ingl *copy*) ja kleepimise (ingl *paste*) nuppudeks nende järjestikkuse ja sümbolite tõttu. Z kujutab edasi-tagasi liikumist, sest alguses oli selle funktsiooniks tühistamine ja uuesti tegemine (*undo, redo*), nüüdseks aga lastakse mitu korda muudatusi tagasi võtta. X sümboliseeris mitmes kohas kustutamist. C on inglise keeles kopeerimise esimene täht. V tagurpidi näeb välja nagu tsirkumfleks (^), mis tähendab sisestamise käsku (ingl *insert*).

1.1 Windowsi lõikelaua tehniline kirjeldus

Windowsi süsteem on arendaja vaatepunktist suhteliselt lihtne. Peamiseks põhjuseks on hea dokumentatsioon. Järgnev ongi kirjutatud Win32 API dokumentatsiooni põhjal [2]. Windowsis on lõikelaud hallatud täielikult operatsioonisüsteemi poolt. Kui lõikelaud pole mõne teise programmi poolt juba avatud, saab rakendus selle avada, infot sealt lugeda ja sinna kirjutada ning peale töö lõpetamist peab lõikelaua uuesti sulgema, et teised programmid seda kasutada saaks.

Windowsis saab lõikelauale kirjutada andmeid erinevates formaatides. Win32 funktsioonides kasutatakse formaatide tuvastamiseks numbreid. Operatsioonisüsteemi poolt on hulk formaate ette defineeritud, nt CF_TEXT (väärtusega 1) teksti jaoks ja CF_WAVE (väärtusega 12) audio jaoks, aga programmid saavad ka oma formaate registreerida. Formaadi registreerimiseks peab ette andma formaadi nime, mille Windows ära registreerib ja annab vastu veel kasutamata

numbri. Kui teine programm tahab sama formaati kasutada, annab Windows sama nime kohta vastu sama numbri, et erinevad programmid saaks sama formaati kasutada tõrgeteta.

Win32 API-l on funktsioonid nagu `EnumClipboardFormats` kõikide hetkel lõikelaual olevate formaatide saamiseks, `SetClipboardData` lõikelauale info kirjutamiseks ja `GetClipboardFormatNameA` või `GetClipboardFormatNameW` formaadi nime saamiseks.

Mis Windowsis on selle töö jaoks hea funktsionaalsus, on võimalus registreerida lõikelaua kuulaja. Kui programm ennast lõikelaua kuulajaks registreerib, saadab Windows ise programmile signaali, kui lõikelaua sisu on muutunud. Selle eeliseks on efektiivsus. Programm ei pea kogu aeg pärima lõikelaua sisu ja seda eelnevaga võrdlema. Kui lõikelaua sisu muutub, annab Windows sellest teada ja programm saab lõikelaua lahti teha, sisu välja lugeda, lõikelaua sulgeda ja sisuga edasi toimetada.

1.2 Linuxi lõikelaua tehniline kirjeldus

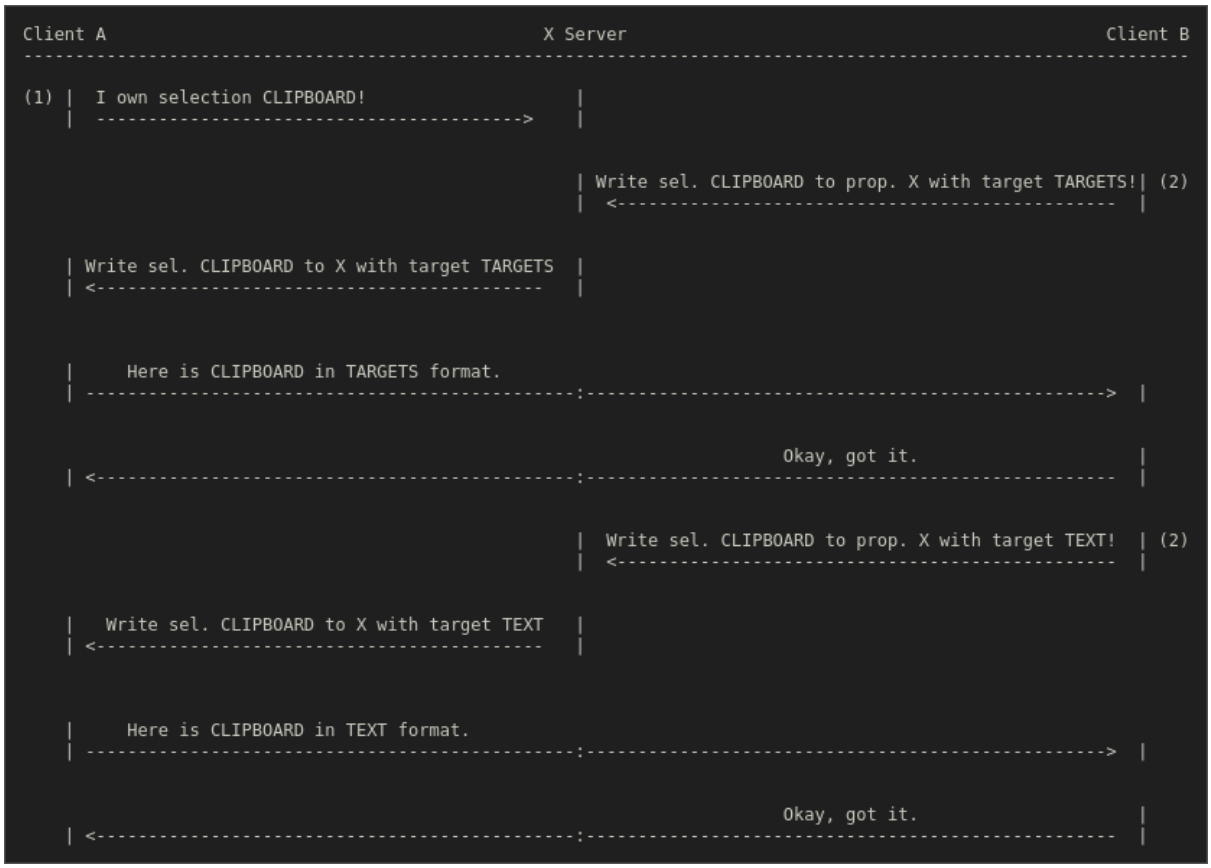
Linuxi masinad kasutavad peamiselt X akna süsteemi (ingl *X Window System*) versiooni 11. Enim kasutatud on X.Orgi[1] avatud lähtekoodiga implementatsioon. Lühidalt nimetatakse seda X11. Teine, uuem asendus sellele, on Wayland, mis loojate [3] sõnul üritab olla lihtsamini arendatav, laiendatav ja hooldatav. Käesolev töö keskendub X11-le autori arvuti seadistuse tõttu.

Järgnev ülevaade tugineb Ulrichi artiklil [4], milles kirjeldatakse X11 ülesehitust kliendi-serveri süsteemil. Server jookseb lokaalselt seadmes ja klientideks on arvutiprogrammid, mis ei pea tingimata samas masinas olema. X11 kasutab klientide vahel andmete jagamiseks “valikuid” (ingl *selection*), “sihtmärke” (ingl *target*) ja “atribuute” (ingl property). X akna süsteem on peamiselt ainult info vahendaja. Klient saab serverile ükskõik millal öelda, et see nüüd omab mingit valikut. Kui mingi programm tahab valiku sisu saada, küsitakse seda serverilt, mille peale server vaid edastab soovi valiku omanikule. Omavahel info saatmiseks saab üks klient kirjutada andmed teise kliendi atribuudile. Lõikelaua jaoks on X akna süsteemis defineeritud kolm standard valikut [5]: primary, secondary ja clipboard. Clipboard on kõige tavalisem lõikelaud, kuhu CTRL+C vajutamise peale pannakse valitud info. Primary on selline, kuhu tekst läheb kohe pärast selle aktiivseks tegemist (ei pea eraldi CTRL+C vajutama) ja kleepimiseks saab kasutada

1https://www.x.org/wiki/

keskmist hiire nuppu. Secondary oli kasutuses primary kõrval lisainfo jaoks, aga tänapäeval seda eriti ei kasutata, kuna primary ja clipboard valikud on enamus eesmärkideks piisav.

Peale valikute on veel olemas sihtmärgid. Sihtmärk tähendab siinses kontekstis justkui info formaati. Nt saab klient serverilt küsida valiku clipboard andmeid sihtmärgiga TEXT, mis tähistab lõikelaua infot teksti formaadis. Standardis on kirjeldatud ka sihtmärk TARGETS, mille küsimisel saadakse kõik toetatud formaadid, millest programm saab endale sobiva valida.



Joonis 1. X serveri suhtlus, kohandatud lõikelaua kontekstis [4].

Joonisel 1 on näha, kuidas teksti kopeerimisel ja kleepimisel info programmide vahel liigub. Programm A ütleb kopeerimisel serverile, et see nüüd omab valikut clipboard. Kleepimisel programmi B, ütleb see serverile “Kirjuta valik clipboard minu atribuudile X sihtmärgiga TARGETS”, et saada kõik lõikelaua formaadid. Selle päringu annab server kliendile A edasi, mille peale A kirjutab B atribuudile X kõik toetatud formaadid. Klient B saab enda atribuudilt X lugeda formaatide nimekirja ja nende hulgast valida endale kõige sobilikuma, nt TEXT. Peale

seda ütleb B serverile “Kirjuta valik clipboard minu atribuudile X sihtmärgiga TEXT”, et saada lõikelaua sisu tekstilisel kujul. Server edastab selle päringu jälle A-le ning A annab B-le vastava teksti.

Loodava rakendusega seoses on Linuxi süsteemide puhul halb kuulajate puudus. Seda seetõttu, et kui programm tahab teada saada lõikelaua muutustest, peab see iga mingi aja tagant küsima lõikelaua omanikult selle sisu ja võrdlema seda eelneva salvestatud sisuga. Kui seda teha liiga tihti, on see mõttetu arvutamine. Kui seda teha liiga harva, võivad mõned kiiremad muutused nägemata jääda. Nende (ja paljude muude) probleemide lahendamiseks on X11-le loodud laiendus XFIXES [6], millest selle töö jaoks kasutati valiku muutumisest teavitamise funktsionaalsust.

Kokkuvõtteks võib öelda, et kuigi kasutaja ei pea sellele mõtlema, erinevad Windowsi ja Linuxi lõikelaua implementatsioonid märgatavalt oma ülesehituse poolest. Üheks suureks erinevuseks on see, et need kaks süsteemi toimetavad andmetega eri formaatides ja sarnastel formaatidel on erinevad nimed. Sellest tuleb ka mõlemal platvormil töötava rakenduse keerukus. Järgmises peatükis analüüsitakse olemasolevaid lahendusi ja nende puuduseid.

2. Olemasolevad lahendused

Lõikelaua ajaloo lahendusi on olemas mitmeid. Windowsis on see funktsionaalsus sisse ehitatud, Linuxile on loodud programmid nagu CopyQ. Arvutite vahel lõikelaua sünkroniseerimiseks on loodud programmid nagu Ditto ja Clipsync. Nüüd natuke täpsemalt igast lahendusest.

2.1 Windowsi lõikelaua ajalugu

Windowsis, nagu enamuses operatsioonisüsteemides, käib kopeerimine ja kleepimine klahvikombinatsiooniga CTRL+C ja CTRL+V. Järgnev lõik põhineb Windowsi dokumentatsioonil [7]. Alates Windows 10 versioonist 1809, kui kleepimiseks kasutada WIN+V, avaneb kleepimise asemel hüpikaken, kus esimesel korral saab ajaloo sisse lülitada ning edaspidi on näha nimekiri eelnevalt kopeeritud andmetest, millest saab vajaliku valida. Kui eesmärk on vaid ühes arvutis kasutada lõikelaua ajalugu, on see väga hea ja mugav lahendus. Veel on võimalus lõikelauda siduda Microsofti kontoga. See võimaldab ühes arvutis kopeeritud teksti teises arvutis kleepida. Puuduseid on mitmeid: vajalik on Windowsi operatsioonisüsteem, sünkroniseeritakse vaid viimane kirje, mitte kogu ajalugu, sünkroniseeritakse ainult tekst, html ja bitmap formaadid, sünkroniseerimise limiit on 4 MB.

2.2 CopyQ

Linuxisse ei ole lõikelaua ajalugu sisse ehitatud. CopyQ [8] on hea lahendus, kus saab ajaloos olevaid andmeid ka sorteerida, filtreerida ja redigeerida. Dokumentatsiooni järgi salvestab see kõik lõikelaual olevad formaadid ning laseb sünkroniseerida ajaloo kettale, aga seda vaid kindlate formaatidega. Kettale salvestatud faile saab seejärel arvutite vahel jagada muude rakendustega.

2.3 Ditto

Ditto [9] on Windowsi rakendus, mis töötab sarnaselt sisseehitatud funktsionaalsusega. See hoiab lõikelaua ajalugu meeles ja sünkroniseerib seda mitme arvuti vahel. Erinevalt sisseehitatud funktsionaalsusest, sünkroniseerib see terve ajaloo (alates programmi käivitamisest, mitte tagantjärgi), mitte ainult viimase kirje, ning terve lõikelaua seisu, mitte ainult mõne lihtsama

formaadi. Veel on seal ka otsingu funktsioon, et ajaloost lihtsamini kindlat infot üles leida. Peamiseks puuduseks on jällegi vaid Windowsi tugi.

2.4 Clipsync

Clipsync [10] on vaid sünkroniseerimise rakendus Linuxi jaoks. See kuulab lõikelaua muutumist ning hoiab lõikelaua seisu samana mitmes arvutis. Selle puuduseks on asjaolu, et puudub ajaloo funktsionaalsus, sünkroniseeritakse ainult teksti ning programm töötab vaid Linuxi süsteemides. Suureks plussiks on võimalus server ise valida. Tavakasutaja saab kasutada avalikku serverit, kuhu külge arvutid ühendada, aga turvalisuse poole pealt on hea, et saab selle serveri ka ise püsti panna.

Kõik need on head lahendused üksikutele probleemidele, aga ükski neist ei lahenda kõiki probleeme. Peamiselt on puudused lõikelaua sünkroniseerimises või toetatud platvormides. Edastatakse kas mõni kindel formaat (Windows, CopyQ, Clipsync), mitte terve lõikelaua seis, või kui jagatakse kogu infot, töötab rakendus ainult ühel operatsioonisüsteemil (Ditto).

3. Rakenduse nõuded ja kirjeldus

Eelmise peatüki analüüs näitas, et kuigi ajaloo, sünkroniseerimise ja mitme platvormi implementatsioone on loodud, ei toeta ükski programm kõiki neid. Selles peatükis on välja toodud loodud rakenduse funktsionaalsed nõuded ning antakse ülevaade rakenduse osadest.

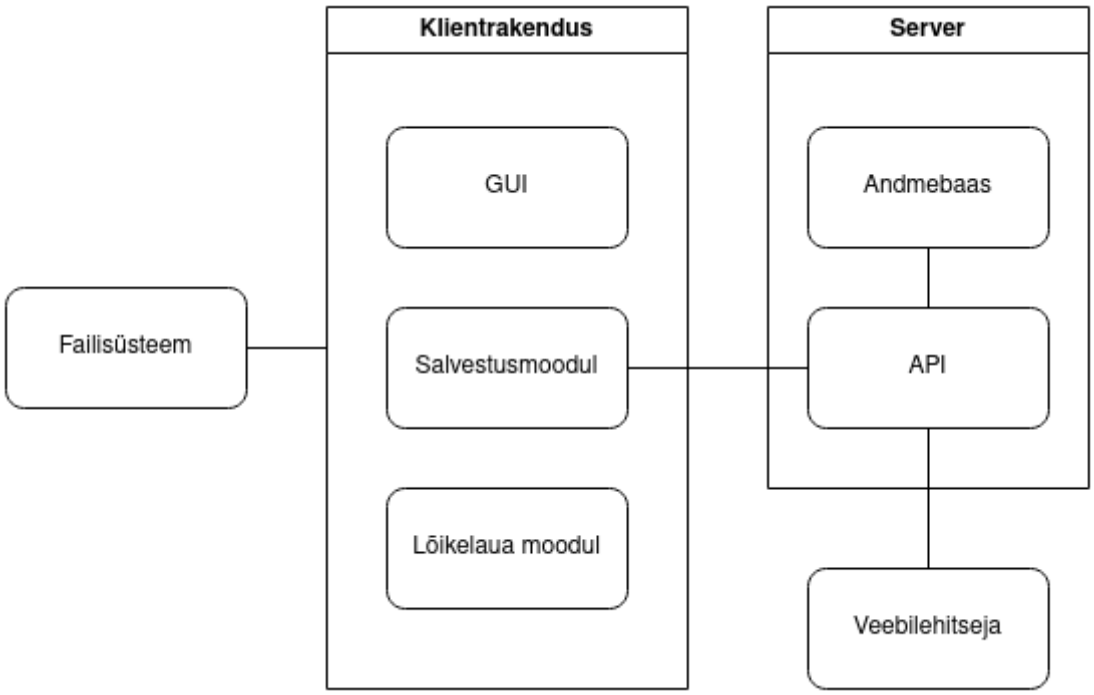
3.1 Funktsionaalsed nõuded

Tuginedes olemasolevatele lahendustele ja ilmnenud puudustele, püstitati rakendusele funktsionaalsed nõuded:

1. Rakendus ja kood on ingliskeelne;
2. Rakendus peab töötama nii Windowsis kui ka X11 kasutavas Linuxis;
3. Päringute tegemisel peab klientrakendus kasutama TLS-i;
4. Iga kord, kui kasutaja kopeerib midagi, lisab rakendus uue info lõikelaualt ajalukku;
5. Rakendus loeb lõikelaualt info kõikides võimalikes formaatides;
6. Erinevate platvormide vahel on toetatud vähemalt teksti formaat;
7. Rakendus salvestab ajaloo kirjed andmebaasi;
8. Andmebaasis on andmed seotud kindla kontoga;
9. Kasutaja pääseb ligi vaid enda kontoga seotud andmetele;
10. Rakendus peab piirama kontopõhiselt ajaloo andmemahtu andmebaasis;
11. Ühes arvutis uue info kopeerimisel värskendatakse ka teistes arvutites ajalugu;
12. Rakendusel on graafiline kasutajaliides;
13. Konto on serveripõhine kasutajanime ja parooli ühend (ei kasuta välist teenust);
14. Kasutaja saab UI-s konfigureerida kasutatavat serverit, kasutajanime ja parooli;
15. Kasutaja saab veebis konto luua;
16. Kasutaja saab veebis oma lõikelaua ajalugu teksti kujul näha ning kirjeid kopeerida.

3.2 Rakenduse osad

Loodud rakendus koosneb kahest suuremast osast: serverist ja kliendist. Klientrakenduse saab käima panna erinevates arvutites ning kui need ühenduvad sama serveriga sama kasutajaga, pääsevad arvutid ligi samale ajaloole. Serveri saab käima panna endale sobivas veebirakenduste majutuskohas, peaasi, et arvutid sellele ligi pääseksid ning andmeside on kaitstud TLS-protokolliga. Joonisel 2 on näha erinevad osad ning millised neist omavahel suhtlevad.



Joonis 2. Rakenduse skeem.

3.2.1 Server

Serveri poolel on Postgres andmebaas, kuhu andmed salvestatakse, veebileht konto registreerimiseks ja sellega seotud ajaloo nägemiseks ning nendega suhtlemiseks API kiht. API kihi tähtsamad lõpp-punktid on:

a) veebilehe saamiseks;

b) konto registreerimiseks;

c) konto kontrollimiseks (kas kasutajanimi ja parool on õiged);

d) kogu ajaloo saamiseks;

e) kindla kirje saamiseks ID järgi;

f) uue kirje lisamiseks;

g) teavituste jaoks kasutatava WebSocket ühenduse avamiseks.

Serveri programmi käivitamisel loetakse alguses vajalikud muutujate väärtused keskkonnast (nt andmebaasi ühenduse detailid ja konto mahulimiit). Peale seda migreeritakse andmebaas kasutades migratsiooni faile. See võimaldab uuendada serverit ilma, et andmebaasist andmed ära kaoks. Kui tõrkeid ei teki, seatakse üles lõpp-punktid ja käivitatakse HTTP server.

Veebilehele minnes näeb alguses vaid sisselogimis- ja registreerimisvormi. Kui kasutaja on sisse loginud, näeb seal salvestatud ajalugu teksti kujul. See võimaldab kasutajal oma andmetele ligi pääseda ka seadmetes, mille tugi puudub klientrakendusel, näiteks nuti- ja Apple'i seadmed.

Autentimiseks kasutatakse lihtsuse pärast HTTP *basic auth'i*[2] ehk päringule antakse kaasa kasutajanimi ja parool. Kasutaja autenditakse uue kirje lisamisel, ajaloost info pärimisel ning WebSocketi avamisel. Kui API saab uue kirje lisamise päringu ning kirje lisamine õnnestub, käivitatakse taustal peale seda kaks protsessi.

Üks, mis, alustades uusimast, liidab järjest kokku lisaja kontoga seotud ajaloo kirjete suurused ja kustutab kõik vanemad, mis ületavad määratud limiidi. Limiidi vaikeväärtuseks on 10MiB ehk kui kõikide kirjete suuruste summa ületab 10MiB, kustutatakse vanemaid kirjeid, kuniks järgmise kirje kustutamisel läheks summa alla 10MiB. See viimane kirje jäetakse alles, et vältida olukorda, kus lisatud kirje suurus ületab 10MiB ning see kustutatakse kohe peale lisamist ära.

Teine protsess käib üle lisaja kontoga seotud WebSocket ühendused ning teavitab neid ajaloo muutumisest. Kui server saab päringu WebSocket ühenduse avamiseks ning see õnnestub, seotakse avatud ühendus konto ID-ga ning jäetakse meelde. Kui ühendus sulgub, eemaldatakse see nimekirjast ehk kogu aeg hoitakse ülevaade lahtistest ühendustest.

WebSocket [11] ühendus võimaldab luua püsiva ja kahepoolse suhtluskanali kliendi ja serveri vahel. See tähendab, et mõlemad pooled saavad reaalajas teisele andmeid saata, mitte ei pea

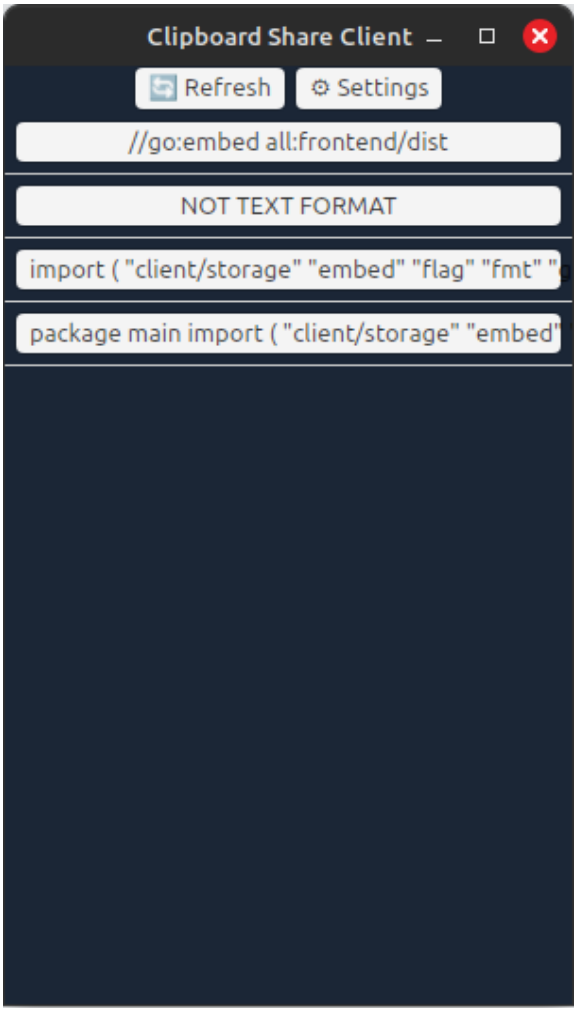
2https://datatracker.ietf.org/doc/html/rfc7617

klient korduvalt pärima, kas on uusi andmeid. Korduvalt pärimine (ingl *polling*) tekitab mõlemale osapoolele lisakoormust, kuna pidevalt peab haldama mitut ühendust.

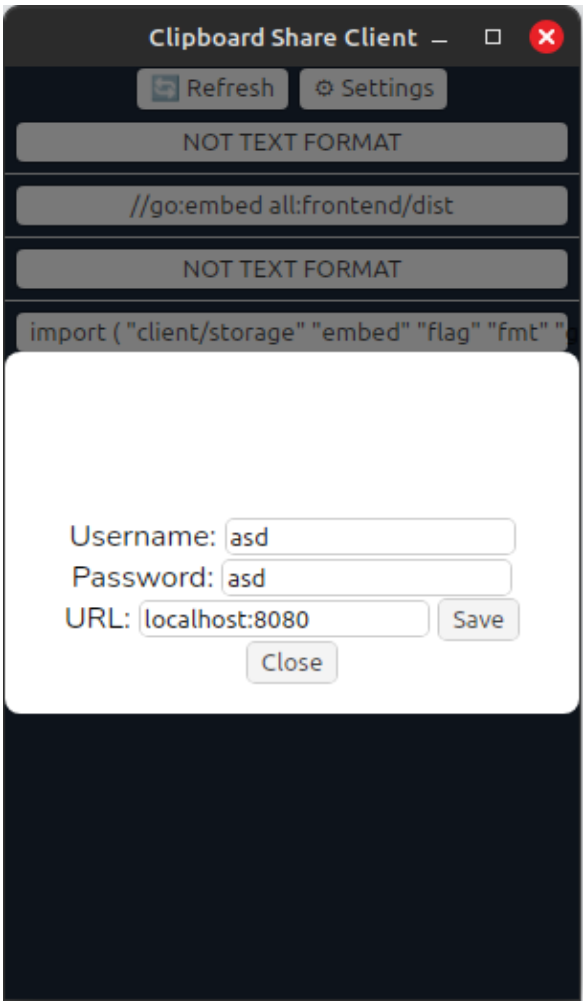
3.2.2 Klient

Klientrakendus on ehitatud moodulitena, et seda oleks tulevikus lihtsam edasi arendada või vajadusel osaliselt ümber kirjutada. Peamised moodulid on lõikelauaga liidestus, kasutajaliides ja salvestusmoodul, mis kasutab andmete salvestamiseks serverit.

Rakenduse käivitamisel loetakse failisüsteemist kasutaja konfiguratsiooni kaustast JSON kujul konfiguratsioonifail. Kui see puudub, siis see luuakse tühjade väärtustega. Kui kasutaja UI-s konfiguratsiooni muudab ja salvestab, kirjutatakse uued väärtused faili. Võimalik on seadistada kasutajanime, parooli ja kasutatavat serverit.



Joonis 3. Ajaloo kasutajaliides.



Joonis 4. Konfiguratsiooni kasutajaliides.

Joonisel 3 on kujutatud rakenduse kasutajaliides, kus on näha lõikelaua ajalugu. Kui kasutaja vajutab ajaloos kirjele, kirjutatakse see tervenisti lõikelauale. Joonisel 4 on näha seadistuse vaade, kus on võimalik seadeid muuta ning muudatus salvestada. Tõrgete puhul on võimalik ajalugu manuaalselt värskendada kasutades nuppu *Refresh* (värskenda).

Lõikelaua moodul annab kopeerimise peale uue info rakendusele ning see salvestatakse läbi salvestusmooduli ajalukku. Salvestusmoodul kasutab andmete salvestamiseks serverit. Selleks võtab moodul rakenduse konfiguratsioonist serveri aadressi, kasutajanime ja parooli, mida igal päringul vaja läheb. Salvestusmoodul vastutab ka uue info kohta teavitamise ja selleks vajaliku WebSocket ühenduse eest. Rakenduse käivitamisel, peale konfiguratsiooni sisse lugemist, avab salvestusmoodul serveriga WebSocket ühenduse, kust jäädakse ootama teavitusi. Kui mingi klient sama kontoga midagi kopeerib, saab rakenduse salvestusmoodul serveri ühenduselt teavituse ning värskendab UI-s ajalugu. Uus ühendus avatakse rakenduse käivitamisel, kui lahtine ühendus katkeb ning seadete muutmisel.

4. Kasutatud keeled, teegid ja APId

Serveri ja kliendi lähtekood on peamiselt kirjutatud keeles Go. Nagu Scalefocus [12] blogis välja on toonud, on Go Google'i poolt arendatud keel, mis on disainitud suuremate *backend* süsteemide jaoks. Selle eelisteks on väga hea standardteek, kõrge jõudlus, mitme tuumaga protsessoritele suunatud disain ja kiire prügikorjamine. Go süntaks on väga lihtne, tänu millele on koodi lugemine tunduvalt lihtsam kui paljude teiste keelte puhul. See kompileeritakse masinkoodiks, mis üldiselt toob endaga kaasa parema jõudluse kui virtuaalmasinate kasutamine. Go'sse sisse ehitatud *goroutine'ide* süsteem võimaldab programmil kasutada mitut protsessori tuuma efektiivselt, ilma et programmeerija sellega palju vaeva peaks nägema.

4.1 Server

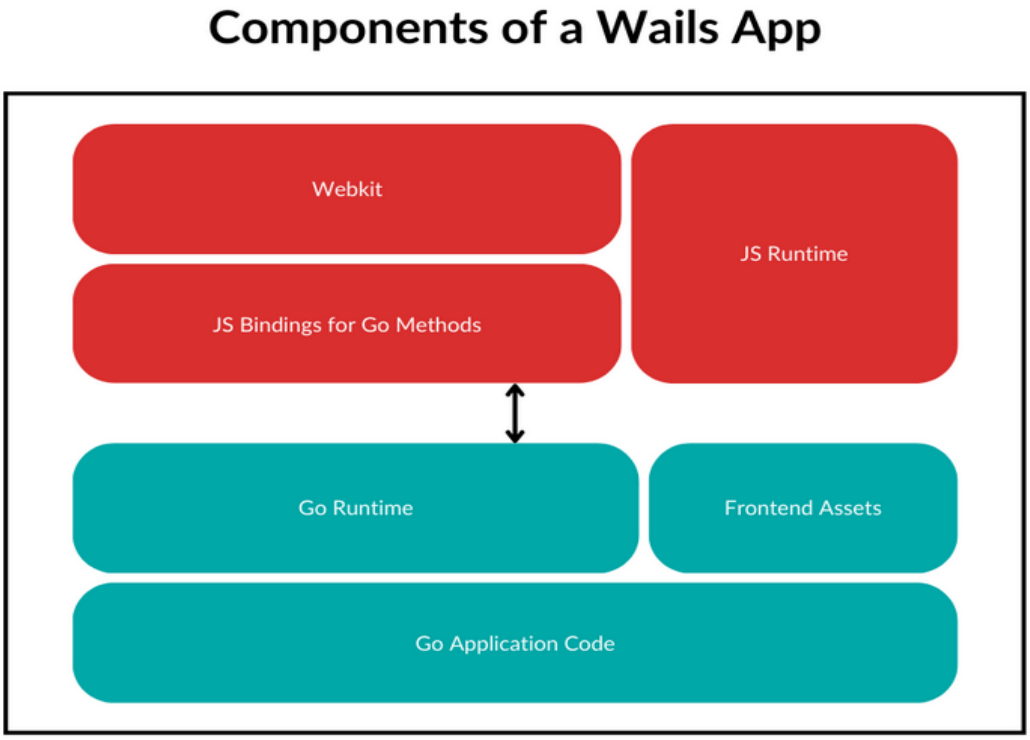
Serverirakenduse API kiht on kirjutatud keeles Go. Selle kõrval on kasutusel Postgres andmebaas selle populaarsuse ja lihtsuse pärast. Andmebaasi migreerimiseks kasutatakse migrate teeki[3]. Andmebaasiga suhtlemiseks kasutab API kiht Postgres draiveri teeki pgx[4]. Vähema koodi kirjutamise jaoks kasutati arenduse käigus tööriista sqlc[5], mis genereerib SQL päringutest Go failid vastavate funktsioonidega. Sqlc loeb ka migratsiooni faile ning kasutab seda infot Go tüüpide genereerimiseks. Veebileht on lihtsuse pärast tavaline HTML fail, mille API kiht kasutajale serveerib. WebSocket ühenduste jaoks kasutati gorilla projekti websocket teeki[6], sest seda soovitatakse kasutada Go arendajate poolt loodud, kuid nüüdseks aegunud, websocket teegi[7] dokumentatsioonis.

4.2 Klient

Klientrakenduse peamine loogika on kirjutatud keeles Go. Kasutajaliidese jaoks kasutab rakendus Wails arendusraamistikku, sest see võimaldab kasutada veebitehnoloogiaid töölauarakenduse loomiseks. Wails toetab natiivselt tavalise HTMLi ja JavaScripti kasutamist ning mitmeid veebiraamistikke, sealhulgas ka Svelte[8], millega autoril on varasem kogemus.

3 https://github.com/jackc/pgx
4 https://sqlc.dev/
5 https://svelte.dev/
6 https://pkg.go.dev/golang.org/x/sys/windows
7 https://github.com/golang-migrate/migrate
8 https://github.com/gorilla/websocket

Wails toetab ka kõikide valikute puhul TypeScripti ning rakenduse jaoks valiti Svelte koos TypeScriptiga. Joonisel 5 on näha, kuidas Wails seob Go ning JavaScripti funktsioonid.



Joonis 5. Wails rakenduse komponendid [13].

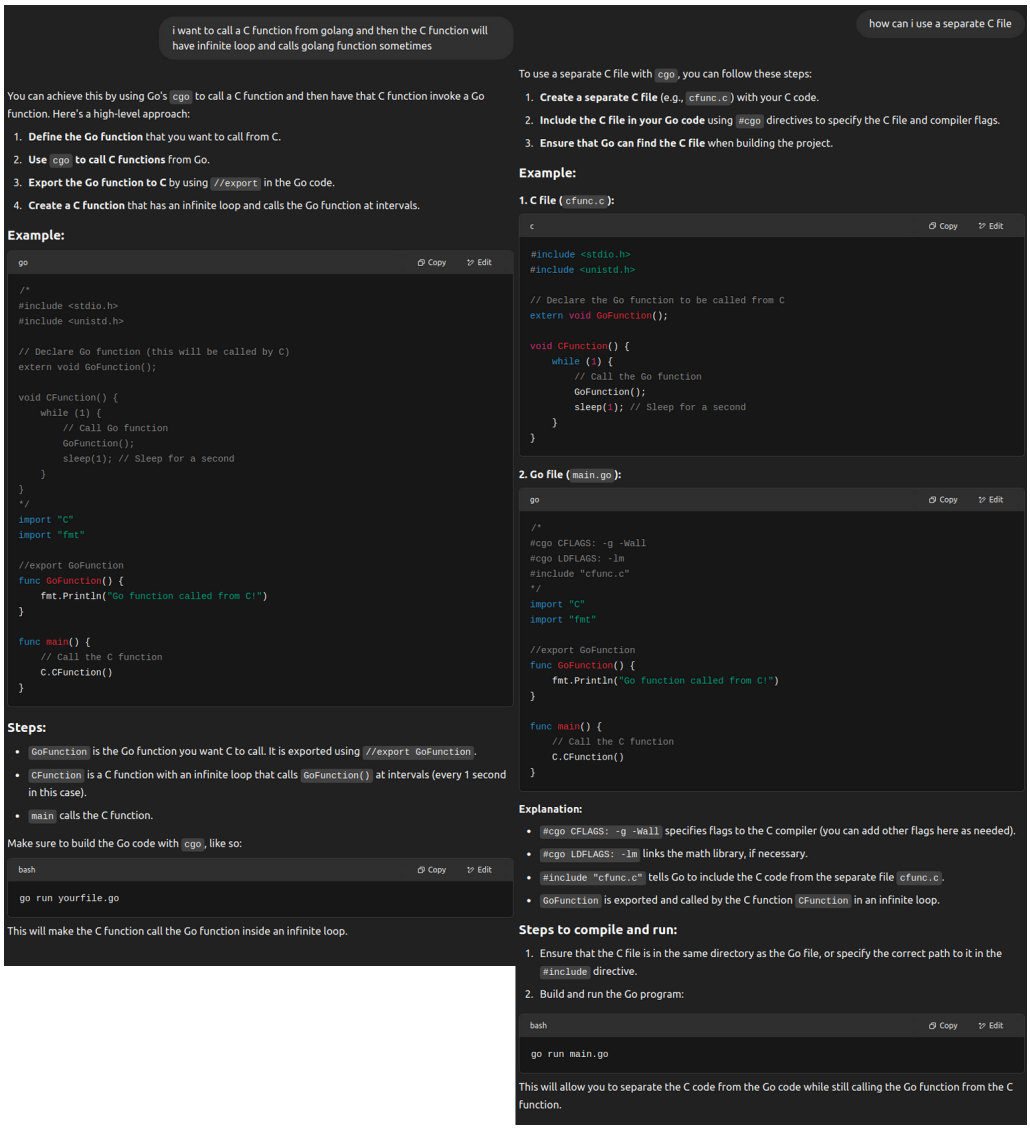
Lõikelaua moodul on Windowsi implementatsioonis tänu heale API-le ja dokumentatsioonile kirjutatud täielikult Go's kasutades teeki windows[9]. Selle kõrval X11 jaoks on kasutusel ka keel C, mis kasutab X11 API-ga suhtlemiseks Xlib-i ja selle laiendust XFixes. Go's kutsutakse C koodi ja vastupidi kasutades mehhanismi nimega cgo, mis võimaldab Go'l kutsuda C koodi ja vastupidi.

4.3 Tehisintellekti kasutamine

Enamus koodi on autori enda kirjutatud, aga vahel oli ka tehisintellekti abi vaja, peamiselt X11 lõikelauaga suhtlemiseks. Kasutatud TI-ks oli kasutamise hetkel OpenAI ChatGPT[10] vaikemudelid (GPT-4o ja GPT-4o mini). Sealt saadud kood pole puhtalt kopeeritud, vaid pigem

9 https://pkg.go.dev/golang.org/x/net/websocket
10 https://chatgpt.com

küsiti mingit üldisemat teemat. Genereeritud kood aitas ülesehitusest aru saada ning seda kasutati näitena rakenduse koodi kirjutamiseks.



Joonis 6. GPT päring.

Joonisel 6 on näha küsimus, kuidas kutsuda Go’s C koodi ja vastupidi. Genereeritud koodis on C kood kirjutatud Go faili, aga C koodi pidi kirjutama piisavalt palju, et selle võiks eraldi faili tõsta. Selleks küsiti jätkuna, kuidas C kood eraldi faili tõsta. Saadud vastuste kood on eeskujuks “client/clipboard/clipboardlinux/” kaustas olevate failide kirjutamiseks. Veel sai GPT abiga infot Xlib’i funktsioonide ja nende kasutamise kohta.

5. Tulemuste analüüs

Bakalaureusetöö eesmärgiks oli luua lahendus, mis sünkroniseerib mitme arvuti vahel lõikelaua ajaloo info kaota. See eesmärk sai täidetud. Järgnevad alapeatükid analüüsivad täpsemalt valminud lahenduse funktsionaalsust, funktsionaalsete nõuete täitmist ning tuuakse esile mõned arenduse ja testimise ajal ilmnenud edasiarendamist võimaldavad kohad.

5.1 Valminud rakendus

Valminud lahenduse lähtekood ja rohkem tutvustavat infot on GitHubi repositooriumis, mille aadress on lisas 1. Järgnevas analüüsis hinnatakse lahenduse vastavust eelnevalt esitatud funktsionaalsetele nõuetele. Punktid on samas järjekorras nagu nõuete loetelu.

1. Rakendus ja kood on ingliskeelne.
2. Rakendus töötab nii Windowsis kui ka X11 kasutavas Linuxis.
3. Päringute tegemisel kasutab rakendus HTTPS ja WSS protokolle, mis on vastavalt TLS-ga kaitstud HTTP ja WebSocket protokollid.
4. Töölauarakendus reageerib iga kopeerimise peale.
5. Andmebaasi salvestatakse info kõikides formaatides, mis on lõikelaualt loetavad.
6. Linuxist Windowsisse ja vastupidi saab kopeerida teksti.
7. Andmed salvestatakse andmebaasi, aga krüpteerimata.
8. Andmebaasis on kõik lõikelaua kirjed seotud kindla kontoga.
9. Igal andmete päringul on konto filter.
10. Peale iga kirje lisamist jooksutatakse mahu kontroll ja vajadusel kustutatakse vanemaid kirjeid.
11. Klientrakendus avab serveriga WebSocket ühenduse, kuhu kirjete lisamisel saadetakse teavitus.
12. Rakendusel on graafiline kasutajaliides loodud Wailsi ja Svelte’iga.
13. Kasutaja tuvastamiseks kasutatakse kasutajanime ja parooli, mitu kasutajat ei saa sama kasutajanime kasutada.
14. Kasutaja saab mugavalt rakenduse UI-s kõiki sätteid muuta.

15. Kasutaja saab veebilehel kontot luua ning saab vastava teavituse, kui kasutajanimi on võetud või midagi muud läheb valesti.
16. Kasutaja saab veebilehel sisse logida ning saab vastava teavituse, kui kasutajanimi või parool on vale või kui midagi muud läheb valesti; Sisselogituna näeb kasutaja oma lõikelaua ajalugu teksti kujul ning saab kirjeid ühe nupuga kopeerida.

Kõik nõuded said täidetud ning lahendus vastab ootustele. Mõned aspektid võiksid saada veel täiendusi, aga see ei mõjuta süsteemi üldist toimimist. Need on välja toodud järgmises alapeatükis.

5.2 Tuleviku võimalused

Kuigi töötav lahendus on olemas, tuli arendamise ja testimise ajal välja mõni koht, mida saaks veel edasi arendada:

1. Turvalisus - lõikelaua andmeid ja parooli võiks nii seadmes kui andmebaasis krüpteerituna hoida;
2. Rohkemate seadmete tugi - praegu toetab töölauarakendus vaid Windowsi ja X11-t kasutavaid Linuxi arvuteid, macOS-i ja mobiiliplatvormide tugi teeks kasutamise veel mugavamaks;
3. Rohkemate formaatide teisendamine platvormide vahel - praegu teisendatakse Windowsi ja Linuxi vahel vaid teksti;
4. Ajaloo haldamine - kirjete kustutamine, märkimine ja filtreerimine võivad kasutajakogemuse veel paremaks teha.

Nimetatud edasiarenduse võimalused annavad hea suuna rakenduse parandamiseks. Nende täiustuste elluviimine toetaks paremini erinevate kasutajate vajadusi ning tagaks, et rakendus püsiks jätkusuutlik ja konkurentsivõimeline ka tulevikus.

Kokkuvõte

Arvuti lõikelaud on laialdaselt kasutatav funktsionaalsus teksti, piltide, failide või muu info lõikamiseks, kopeerimiseks ja kleepimiseks. Lõikelaua mugavamaks kasutamiseks on välja mõeldud lõikelaua ajalugu, mis ei kaota uue info kopeerimisel eelnevat ära, vaid tekitab nimekirja kopeeritud andmetest. Mitme arvuti vahel info dubleerimiseks saab kasutada lõikelaua või selle ajaloo sünkroniseerimist, mis vastavalt kas ühes arvutis kopeerimisel kirjutab sama info ka teise arvuti lõikelauale või dubleerib lõikelaua ajalugu mitmes arvutis.

Bakalaureusetöö eesmärk oli luua lõikelaua ajaloo jagamise rakendus. Rakendus jälgib arvuti lõikelauda ja kopeerimise peale lisab kopeeritud info kõik kättesaadavad formaadid ajalukku, mis salvestatakse serveri andmebaasi. Ühendades mitu arvutit ühise konto ja serveriga, jagavad arvutid lõikelaua ajalugu. Tehes ühes arvutis CTRL+C, lisandub teistes arvutites kirje kohe ajaloo nimekirja. Kui teine seade pole Windowsi või Linuxi arvuti, saab ajaloole ligi läbi veebilehitseja. Vaid mõne hiire vajutusega liigub info teises arvutis lõikelauale ja selle saab CTRL+V-ga kleepida soovitud programmi.

Loodud lahendus täidab tööle seatud eesmärke ja püstitatud nõudeid. Võrreldes teiste sarnaste lahendustega, töötab töölauarakendus mitmes operatsioonisüsteemis ning lõikelaualt loetakse kogu võimalik info, mitte ainult mõni lihtsam formaat.

Lahenduse arendamiseks kasutati mitmeid programmeerimiskeeli ja tehnoloogiaid. Loodud lahendus koosneb serverist ja klientrakendusest. Serveri API kiht on kirjutatud programmeerimiskeeles Go ning see salvestab andmed Postgres andmebaasi ja serveerib HTML faili veebilehitsejate tarbeks. Klientrakenduse, mis suhtleb lõikelaua, kasutajaliidese ja serveriga, jaoks kasutati Go’d, C’d ja veebitehnoloogiaid.

Olulisemad edasiarenduse võimalused keskenduvad turvalisusele ja kasutaja mugavusele: andmete krüpteerimine, rohkemate seadmete tugi, rohkemate formaatide teisendamine erinevate platvormide vahel ning ajaloo detailsem haldamine.

Viidatud kirjandus

[1] Brad Myers. Past to Future: Various Undo Models, Interaction Histories, and Macro Recording Lecture 21. 2016 https://studylib.net/doc/16057899/past-to-future--various-undo-models--interaction-historie (08.12.2024)

[2] alvinashcraft, metathinker, mijacobs, msatranjr. Clipboard - Win32 apps | Microsoft Learn. 2020 https://learn.microsoft.com/en-us/windows/win32/dataxchg/clipboard (08.12.2024)

[3] Wayland. https://wayland.freedesktop.org/ (08.12.2024)

[4] Ulrich. X11: How does "the" clipboard work? 2017 https://www.uninformativ.de/blog/postings/2017-04-02/0/POSTING-en.html (08.12.2024)

[5] David Rosenthal, Stuart W. Marks. Peer-to-Peer Communication by Means of Selections. https://tronche.com/gui/x/icccm/sec-2.html (14.05.2025)

[6] Keith Packard. The XFIXES Extension, Version 5.0, Document Revision 1. 2010 https://www.x.org/releases/current/doc/fixesproto/fixesproto.txt (08.12.2024)

[7] Microsoft. Using the clipboard. https://support.microsoft.com/en-us/windows/using-the-clipboard-30375039-ce71-9fe4-5b30-21b7aab6b13f (14.05.2025)

[8] Lukas Holecek. CopyQ. https://hluk.github.io/CopyQ/ (14.05.2025)

[9] Ditto clipboard manager. https://ditto-cp.sourceforge.io/index.php (14.05.2025)

[10] Marco Paganini. Clipsync. https://github.com/marcopaganini/clipsync (14.05.2025)

[11] Alexey Melnikov, Ian Fette. The WebSocket Protocol. 2011 https://datatracker.ietf.org/doc/html/rfc6455 (15.05.2025)

[12] Scalefocus. Why Golang? https://www.scalefocus.com/blog/why-you-should-go-with-go-for-your-next-software-project (14.05.2025)

[13] Wails. How does it work? https://wails.io/docs/howdoesitwork (14.05.2025)

Lisad

1. GitHubi repositoorium

Töö käigus loodud lahenduse lähtekood on leitav GitHubi repositooriumis https://github.com/henrykalju/clipboard-share. Repositoorium sisaldab lähtekoodi ja lahendust tutvustavat “README.md” faili.

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina Henry Kaljulaid ,
, _____
(autori nimi)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Lõikelaua ajaloo jagamise rakendus ,
   (lõputöö pealkiri)

   mille juhendaja(d) on Alo Peets ,
   (juhendaja nimi)

   reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Henry Kaljulaid
15.05.2025