TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDN40LT

Gerda Põder 121059

# WEBRTC TARKVARA KASUTAV KLIENDITEENINDUSE VEEBIRAKENDUS

Bakalaureusetöö

Juhendajad: Innar Liiv
Doktorikraad
Dotsent

Janno Stern

Tallinn 2016

# 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: Gerda Põder

13.05.2016

# Annotatsioon

Järjest kiirenevas maailmas on oluline olla pidevalt kättesaadav. Nii tahavad suured ettevõtted leida võimalusi, kuidas oma klientidega võimalikult tihedas kontaktis olla. Samas soovitakse kulude optimeerimiseks kontoreid sulgeda. Selle probleemi lahendaks veebipõhine kanal klientidega suhtlemiseks, kus piisab paarist töötajast kõikide kliendibaasi murede lahendamiseks.

Töö eesmärgiks on esmalt analüüsida Eesti suurettevõtete pakutavaid klienditeeninduse võimalusi. Teiseks tehakse selgeks, millised peavad olema klienditeenindusrakenduse nõuded. Kolmandaks eesmärgiks on võimaldada veebipõhist suhtlust klientide ja klienditeenindajate vahel.

Töö tulemusena valmib veebirakendus, mis võimaldab turvalist suhtlust kahe osapoole vahel. Rakendus võimaldab:

- kasutajanime ja parooliga sisse logida
- pidada videokõnesid,
- saata tekstsõnumeid,
- saata faile.

Eesmärk on saada rakendus tööle veebibrauserites Mozilla Firefox, Opera, Google Chrome ning Google Chrome for mobile, sest need toetavad WebRTC protokolli ja kuuluvad kõige populaarsemate veebibrauserite hulka.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 5 peatükki, 7 joonist, 4 tabelit.

# Abstract

WebRTC based customer service web application

Nowadays it gets more and more important to be available all around the clock. That is why big companies are in search for new possibilities to connect with their customers. At the same time they want to cut down their opening hours or even close the offices to optimize expenses. This problem could be solved effectively by a web based channel where few customer consultants can deal with all of the problems that customers may have.

The purpose of this thesis is to first analyze possibilities that big companies offer for customer service, secondly find out the features that a proper customer service application needs and lastly realize the application for communication between customer service and clients.

As a result of the work there will be a web application that enables secure communication between two partners. The application contains the following features:

- logging in using username and password
- having video conferences
- sending text messages
- sending files

The application has to work in browsers like Mozilla Firefox, Opera, Google Chrome and Google Chrome for mobile as they support WebRTC and belong to the most popular web browsers.

The thesis is in Estonian and contains 41 pages of text, 5 chapters, 7 figures, 4 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| *API* | *Application Program Interface* – Rakendusliides.<br>Reeglistik, mille alusel kasutab rakendus teise rakenduse teenuseid [1]. |
| *BLOB* | *Binary Large Object* – Suur kahendobjekt.<br>Suur bitiplokk (nt pildi-, teksti- või helifail) [1]. |
| *CA* | *Certificate Authority* – Sertifitseerimisasutus.<br>Digitaalseid sertifikaate väljastav organisatsioon või firma, kelle ülesanne on garanteerida, et sertifikaadi saaja on see, kellena ta end esitleb [1]. |
| *CSR* | *Certificate Signing Request* – Sertifikaadinõue.<br>Sertifikaadi taotlemiseks moodustatud plokk krüpteeritud teksti, mis sisaldab domeeni ja taotleja andmeid [2]. |
| CSS3 | *Cascading Style Sheets* – Kaskaadlaadistik.<br>Veebilehtede kuvamise kirjeldamiseks mõeldud laadistiku viimane standard versioon [1]. |
| HTML5 | *HyperText Markup Language* – Hüpertekst-märgistuskeel.<br>Kodeerimissüsteem veebilehtede loomiseks [1]. |
| *ICE* | *Interactive Connectivity Establishment.*<br>Tehnika partnervõrkude loomiseks [3]. |
| *IoT* | *Internet of Things* – Asjade Internet.<br>Võrgustik, mille moodustavad omavahel suhtlevad füüsilised esemed, seadmed, sõidukid ja muu, mis sisaldab elektroonikat, tarkvara, sensoreid ning internetti ühendamise võimalust [4]. |
| JavaScript | Interaktiivsete veebisaitide loomiseks välja töötatud skriptikeel [1]. |
| JQuery | Võimalusterohke raamistik Javascripti kasutamise lihtsustamiseks [5]. |
| *JSON* | *JavaScript Object Notation.*<br>Lihtsalt loetav ja kirjutatav andmevahetusformaat [6]. |
| *NAT* | *Network Address Translation* – võrguaadresside tõlkimine.<br>Tehnika, mis võimaldab jagada välisvõrgu IP-aadressi kohtvõrkude vahel, et suurendada turvalisust ja vähendada välisvõrgu IP-aadresside arvu [7]. |

| | |
|---|---|
| Node.js | Tarkvaraplatvorm veebirakenduste loomisel serveri-poolseks programmeerimiseks [8]. |
| Partnervõrk | *Peer-to-peer connection.*<br>Sidevõrgu mudel, kus kasutajad saavad otse üksteisega faile vahetada [1]. |
| *plug-in* | Pistikprogramm.<br>Süsteemile lisafunktsionaalsust andev tarkvaramoodul [1], mis tuleb tavaliselt eraldi alla laadida. |
| *RTC* | *Real-Time Communication* – Reaalajaline kommunikatsioon.<br>Telekommunikatsioon, mille puhul andmete vahetamine käib ilma märgatava latentsuseta [9]. |
| *SDP* | *Session Description Protocol*<br>Formaat, mis kirjeldab voogedastatava meedia parameetreid (nt heli ja video formaat) ehk seda, kuidas toimub edaspidine suhtlus pärast võrguühenduse loomist [10]. |
| Signaalserver | *Signaling server.*<br>Server, mille vahendusel partnerid vahetavad kirjeldusi ning sõnumeid partnervõrgu loomiseks [11]. |
| Socket.IO | Node.js moodul serveri ja kliendipoolse rakenduse omavaheliseks suhtluseks, kasutades veebisokleid [12]. |
| *SQL* | *Structured Query Language* – Struktuurpäringukeel.<br>Reeglite kogum koostamaks päringuid andmebaasist andmete otsimiseks [1]. |
| *SSL* | *Secure Sockets Layer* – Turvasoklite kiht.<br>Üle interneti edastavate andmete turvalisuse tagamise protokoll, mis kasutab privaatse ja avaliku võtme süsteemi [1]. |
| *STUN* | *Simple Traversal of UDP through NATs.*<br>Server, mida kasutatakse välise IP-aadressi ja pordi teadasaamiseks [11]. |
| *TURN* | *Traversal Using Relays around NAT.*<br>Server, mida kasutatakse ebaõnnestunud partnervõrgu korral andmete vahetamiseks [11]. |
| *URL* | *Uniform Resource Locator* – Internetiaadress.<br>Ressursile viitav aadress [1]. |
| *UUID* | *Universally Unique Identifier.*<br>Juhuslikult genereeritud 128-bitine unikaalne identifikaator [13]. |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Tänapäeva maailmas mõtlevad kõik inimesed, kuidas olla efektiivsemad – kuidas hoida kokku raha, aega ja energiat. Elurütm on muutumas kiiremaks, ent samas mugavamaks. See kehtib nii ettevõtete kui ka nende klientide kohta.

Ettevõtete eesmärk on hoida kokku raha, mis kulub kontoritele, kus käib harva vähe kliente. See sunnib lühendama kontorite lahtiolekuaegu või neid sootuks sulgema. Ent siiski säilib vajadus olla ühenduses klientidega, kes soovivad abi või nõustamist.

Kontorite sulgemise probleem hõlmab tervet Eestit. Näiteks on riigiettevõte Eesti Energia sulgenud viimasel aastal 9 kontorit, jättes alles vaid 4 ning sedagi suurematesse linnadesse nagu Tallinn ja Tartu [14]. Ka Äripäeva artikkel „Pangakontorite sulgemise seiskab riigi sekkumine“ kinnitab kontorite sulgemise trendi. Artiklis kirjutatakse maapiirkondade ettevõtluse pärast muret tundes pankadest nii: „2013. aasta lõpu seisuga oli Eestis 140 kontorit, millest 96% asub linnades. See on 35% vähem kui 2009. aastal.“ [15].

Klientide elu on aga muutunud kiiremaks ning tihti on raske leida aega, mis sobiks kontorite lahtiolekuaegadega. Samas hinnatakse mugavust, et ei peaks lihtsate murede pärast tõusma püsti kodust diivani pealt ning aja kokkuhoidu, et mitte raisata seda kontorisse kohalemineku peale. Lisaks sellele elame ajastul, kus suur osa teenuseid kolibki internetti.

Näiteks on portaal Tarbija24 kirjutanud artikli If kindlustuse Kuressaare kontori sulgemisest. Nad põhjendavad seda sellega, et Eesti inimesed on harjunud tegema pea kõiki toiminguid internetis. Klientide harjumused olevat muutunud ning aastate eest eelistatuima kontori on välja vahetanud Ifi e-büroo ning klienditelefon [16].

Mujal maailmas on probleemiga juba tegeletud ja mitmeid lahendusi leitud. Üheks nendest on Barclays *video banking*, mis käivitati 2014 aasta lõpus. See on teenus, mis lubab Barclays panga klientidel saada näost näkku nõustamist, kasutades nutitelefoni, tahvel- või sülearvutit mistahes ajal ja kohas [17]. Lisaks sellele on olemas LiveBank, mis pakub samuti pankadele klienditeeninduste tarkvara. Seal on võimalik teha turvalisi

video- ja helikõnesid ning saata tekstsõnumeid. Lisaks pakub tarkvara erinevaid tööriistu nagu kalkulaator, vormid, ekraani jagamine ja failide saatmine [18].

Vaatamata edusammudele mujal maailmas ei ole Eestisse veel sarnased lahendused jõudnud. Käesolevas töös püütakse välja töötada alternatiiv, mis oleks mugavam ja kiirem kui tavapärane kliendi ja klienditeenindaja vaheline suhtlus kontoris.

## 1.1 Taust ja probleem

Suured ettevõtted seisavad silmitsi probleemiga, kuidas olla võimalikult suurel osal ajast klientidele kättesaadav, kuid samas mitte kulutada selleks üleliia raha. Kulutuste kokkuhoiuks suletakse kontoreid väiksemates piirkondades või piiratakse nende lahtiolekuaegu. Selle asemel on oluline mõelda välja muu moodus, kuidas kliendid saaksid vajalikel hetkedel ettevõttega ühendust.

Teisalt aga soovivad kliendid mugavust, privaatsust, vabadust ja aja kokkuhoidu. Kontorisse kohaleminek on tülikas, aeganõudev ning piirav, kuna on avatud vaid teatud kellaaegadel. Samas puudub seal tavaliselt privaatsus, sest kõrvallauas istuja võib vabalt kliendi ja klienditeenindaja juttu pealt kuulata.

Lõputöö tulemusena valminud rakendus on vajalik eelkõige suurtele mitmete kontoritega ettevõtetele, kes soovivad kärpida kulusid, ent olla siiski klientidele lihtsal ja mugaval moel kättesaadav. Samuti on rakendus oluline klientidele, kes hindavad privaatsust ja vabadust teha kõnesid neile sobival ajal ja kohas.

## 1.2 Ülesande püstitus

Töö esimene eesmärk on uurida Eesti suurettevõtete kodulehekülgedel pakutavaid klienditeeninduse lahendusi, ning neid analüüsides jõudes järeldusele, kas turul oleks vaja üheaegselt videokõnet, tekstsõnumite ja failide saatmist võimaldatavat rakendust. Samuti tuleb teha selgeks klienditeenindusrakenduse olulised nõuded mugavaks ja lihtsaks toimimiseks.

Töö teine eesmärk on valmistada prototüüp rakendusele, mis võimaldab video- ja tekstipõhist krüpteeritud suhtlust kahe osapoole vahel, kasutajate autentimist kasutajanime ja parooliga ning failide saatmist. Samuti peab rakendus töötama brauserites

Mozilla Firefox, Opera, Google Chrome ja Google Chrome for mobile, mis toetavad WebRTC protokolli ning kuuluvad populaarseimate veebibrauserite hulka. Neid kasutab 2016. aasta märtsi seisuga kokku 89% internetikasutajatest [19].

## 1.3 Metoodika

Eesmärkideni jõudmiseks analüüsitakse esmalt teisi Eesti turul leiduvaid sarnase eesmärgiga lahendusi ning otsustatakse, mis on rakenduse puhul tähtis. Seejärel arendatakse prototüüp.

Eesmärkide saavutamiseks kasutatakse erinevaid tehnoloogiaid ja vahendeid, mille hulka kuuluvad WebRTC, PostgreSQL andmebaas, Socket.IO, Node.js, Docker jm.

Tulemuste kontrollimiseks kasutatakse praktilist eksperimenteerimist ja testimist.

## 1.4 Ülevaade tööst

Töö koosneb neljast osast. Töö esimeses osas analüüsitakse erinevaid Eesti ettevõtete pakutavaid klienditeeninduse võimalusi. Lisaks sellele esitatakse rakenduse nõuded.

Töö teises osas kirjeldatakse rakenduse disaini. Peatükis tutvustatakse rakenduse arenduseks kasutatavaid tehnoloogiaid, süsteemi arhitektuuri ja kirjeldatakse rakenduse kasutamist.

Kolmandas osas testitakse rakendust ning neljandas tuuakse välja rakenduse võimalikud edasiarendused.

# 2 Analüüs

Kuna lõputöö eesmärk on arendada prototüüp klienditeenindusrakendusele, mida Eesti ettevõtetele pakkuda, püüdis autor selles peatükis võrrelda erinevaid Eesti turul olevaid klienditeeninduse lahendusi ning analüüsida, mis on arendatava rakenduse juures oluline. Lisaks sellele tuuakse välja analüüsi käigus selgunud rakenduse nõuded.

## 2.1 Olemasolevad lahendused

Lõputöö autor uuris Eesti suurettevõtete kodulehekülgedelt, milliseid klienditeeninduse lahendusi nad eraklientidele pakuvad. Erilist tähelepanu pöörati ettevõtetele, mille puhul pidas autor oluliseks klientide võimalust vajalikel hetkedel kiiret abi või nõustamist saada. Valitud ettevõtete hulka kuulusid pangad, kindlustus- ja laenuteenuseid pakkuvad ning elektrienergia, vee ja telekommunikatsiooniga tegelevad firmad.

Uurimise käigus selgus, et enim panustavad Eesti suurfirmad klienditeenindusse pealinnas Tallinnas, kus on avatud enim kontoreid. Samuti tegi autor kindlaks, et ühelgi ettevõttel ei ole kasutusel lahendust, mis hõlmaks üheaegselt tekstsõnumite ja failide saatmist, videokõne võimalust ega eeldaks mitte ühegi *plug-ini* või rakenduse allalaadimist. Sellega seoses tehti järeldus, et kuna väikestes kohtades on klienditeeninduse kättesaadavus vähene, mõnel puhul isegi puudulik, ja kuna ettevõtetel puudub vahetu üks-ühele ühenduses olemise võimalus nende kohtadega, võiks arendataval rakendusel turgu olla küll.

Edaspidi esitatakse eraldi peatükina ettevõtted, mis olid panustanud klienditeeninduse arendamise peale rohkem, kui telefoninumbri ja e-posti väljatoomine ning lihtsate täidetavate vormide asetamine kodulehele. Viimses peatükis „Muud ettevõtted“ tuuakse välja ülejäänud uuritud ettevõtted.

### 2.1.1 Swedbank

Swedbanki kodulehelt leiab info ettevõtte pangakontoritest üle Eesti. Selgus, et Swedbankil on 2016. aasta 2. mai seisuga üle Eesti kokku 36 vähemalt esmaspäevast reedeni avatud kontorit, kusjuures 13 neist asuvad Tallinnas. Lisaks sellele on ettevõttel vähemalt ühel päeval kuus kuni kolmel päeval nädalas lahtiolevaid kontoreid, mis pakuvad ka nõustamist. Neid on kokku üle Eesti 35 [20].

Lisaks kontoritele pakub Swedbank telefonipanga teenust ehk võimalust suhelda telleriga telefoni teel. Telefonipank pakub ülevaadet rahaasjadest ning operatiivset infot kõikide pangateenuste kohta. Telefonipanga eelis pangakontori ees on pikk tööaeg varahommikust hilisõhtuni, samuti laupäeviti ja pühapäeviti. Telefonipangas saab oma isikut tõendada paroolikaardi, PIN-kalkulaatori või mobiil-ID abiga [21].

Peale telefoni saab Swedbankiga ühendust võtta ka kodulehelt leitava e-posti teel ja läbi suhtlusrakenduse Skype [21].

### 2.1.2 LHV pank

LHV pangal on Eestis kontorit, mis asuvad Tallinnas ja Tartus. Peale kontorite saab klienditoe poole pöörduda telefoni ja meili teel. Lisaks neile võimalustele pakub LHV pank veebipõhist vestlust klienditoega. Vestlus klienditoega kujutab endast tekstsõnumite vahetamise rakendust, kus on võimalik saata ka faile. Klienditeenindajad vastavad esmaspäevast reedeni kell 9-19 ja laupäeval kell 10-17. Küll aga puudub rakenduses videokõne võimalus, mille lahenduseks on kodulehel välja toodud Skype konto [22].

### 2.1.3 Nordea pank

Nordea pangal on Eestis 14 kontorit, millest pooled asuvad Tallinnas. Lisaks sellele on võimalik Nordea panga klienditeenindajatega ühendust võtta klienditoe telefonil või meili teel [23].

Nordea pank pakub veel ka veebikohtumise lahendust. Selle abil on võimalik klientidel saada asjatundlikku nõustamist neile sobival kohas, ilma pangakontorit külastamata. Veebikohtumise aeg lepitakse kokku eelnevalt ning võib toimuda tööpäeviti kella üheksast hommikul kaheksani õhtul. Veebikohtumise miinus on aga see, et selle toimimise eelduseks peab kliendi arvutisse või telefoni olema allalaaditud programm Skype for Business ning Lync Web App *plug-in* [23].

### 2.1.4 Tele2

Telekommunikatsiooniteenuseid pakkuval ettevõttel Tele2 on Eestis 24 esinduskontorit, neist seitse asuvad Tallinnas. Tele2 erakliendiindusega saab kiiremate küsimuste puhul vastuseid iga päev kell 7.00 – 23.00, kuid on võimalik täita ka vorm ja oodata vastust e-postkasti. Tele2 kodulehelt leiab ka lingi nende Facebooki klienditeeninduse lehele. Seal on võimalik esitada küsimusi nii privaatselt kui ka avalikult ning küsimustele

vastatakse esmaspäevast reedeni kella 10 – 19. Seal saab lugeda ka eelnevalt avalikult postitatud küsimusi ning nende vastuseid, kuid puudub üks-ühele klienditeenindajaga vestlemise võimalus [24], [25].

### 2.1.5 Telia

Telekommunikatsioonieттevõte Telia omab üle Eesti 27 esindust ja nendest 7 asuvad Tallinnas [26]. Telia klienditeenindajatega saab suhelda helistades infotelefonil, saates e-kiri või täites kodulehel väljatoodud vorm oma andmete ja küsimusega ning oodates vastust e-posti aadressile. Lisaks sellele on kodulehel üles seatud ka võimalus suhelda klienditeenindajaga *online* tekstsõnumite vahendusel. Küll aga ei võimalda lahendus saata faile ega videokõnet pidada [27].

### 2.1.6 Seesam kindlustus

Seesam kindlustusel on esindusi kokku 11. Peale kontorite saab ühendust ettevõttega ka veel tagasiside vormi kaudu, kirjutades e-posti või helistades kodulehel väljatoodud telefoninumbritel. Lisaks nendele võimalustele on võimalik alustada vestlust klienditeenindajaga, mis hõlmab endast küll vaid tekstsõnumite saatmist [28].

### 2.1.7 Muud ettevõtted

Lisaks eelnevalt väljatoodud ettevõtetele uuriti ka SEB panga, Eesti Energia, Elektrilevi, aktsiaseltsi Tallinna Vesi, Salva ja If kindlustuse koduleheküлgi, kuid sealt ei leitud klienditeenindusega ühenduse võtmiseks muid võimalusi kui helistamine, e-posti kirjutamine või harvemal juhul vormi täitmine.

## 2.2 Rakenduse nõuded

Selles peatükis tutvustatakse projekti nõudeid, mis peavad arendatavas rakenduses realiseeritud olema. Nõuded on jaotatud kaheks: funktsionaalsed ja mittefunktsionaalsed. Funktsionaalseid nõudeid peetakse silmas rakenduse disaini käigus ning need kirjeldavad, kuidas süsteem peaks käituma kasutajapoolsete või teisest süsteemist pärinevate sisendite peale. Mittefunktsionaalseid nõudeid peetakse silmas rakenduse arhitektuuri juures ning need sisaldavad endas näiteks kasutatavust, käideldavust, toetavust ja jõudlust [29].

### 2.2.1 Rakenduse funktsionaalsed nõuded

Rakenduse funktsionaalsed nõuded on järgmised:

1. Rakendust käivitades on fookuses kasutajanimi väli ning vajutus klahvile Enter logib kasutaja sisse.
2. Sisseloginud kliendid näevad kontaktide menüüs kõiki *online* staatuses olevaid klienditeenindajaid ning saavad nendega ühendust võtta nimele klõpsates.
3. Sisseloginud klienditeenindajad näevad kontaktide menüüs kõiki *online* staatuses olevaid kliente ning saavad nendega ühendust võtta nimele klõpsates.
4. Kui kasutaja on ühendust võtnud teise kasutajaga, näeb ta nende vahel varem saadetud sõnumeid.
5. Kui kasutaja pole ise sõnumite loetelu kerinud, kuvatakse talle alati esimesena (ehk allpool) uued sõnumid.
6. Kasutaja saab saata teisele kasutajale tekstsõnumeid.
7. Tühja sõnumit ei saa saata.
8. Enterile vajutus saadab sõnumi automaatselt.
9. Kasutaja saab saata teisele kasutajale faile.
10. Kasutaja saab saata teisele kasutajale pakkumise alustada videokõnet.
11. Kui kasutaja on saanud kutse videokõnele, on tal võimalik sellele kas vastata või sellest keelduda.
12. Kasutajad saavad pidada kõnet, kasutades videot ja heli. Kasutajatel on võimalik igal hetkel videokõnest loobuda, vajutades kõne lõpetamise nuppu.

### 2.2.2 Rakenduse mittefunktsionaalsed nõuded

Rakenduse mittefunktsionaalsed nõuded on järgmised:

1. Kasutajad saavad sisse logida kasutades kasutajanime ja parooli.
2. Kasutajaliides peab olema lihtne ning sarnanema teiste sõnumirakendustega.
3. Kasutajaliideses peab saama kindlalt vahet teha enda ja teise kasutaja saadetud sõnumitel.
4. Kasutajaliideses peab olema näha info, millal on sõnum või fail saadetud.
5. Sisselogimata kasutaja peab tohtima näha vaid sisselogimise akent.

# 3 Rakenduse disain

Lõputöö käigus valmis WebRTC tehnoloogiat kasutav klienditeeninduse veebirakendus. Selles peatükis tutvustab autor rakenduse tehnilist teostust ning selle arenduses kasutatud vahendeid, tehnoloogiaid ja nende kasutust.

## 3.1 Kasutatavad vahendid ja tehnoloogiad

Klienditeenindusrakendus on veebipõhine rakendus, mis on kirjutatud kasutades HTML5, CSS3 tehnoloogiaid ning JavaScripti programmeerimiskeelt. Samuti kasutas autor JavaScripti raamistikku jQuery. Serveri arendamiseks kasutati tarkvaraplatvormi Node.js. Rakenduse ja serveri omavaheliseks suhtluseks ning signaalserveri toimimiseks kasutati JavaScripti raamistikku Socket.IO. Videokõne arendamiseks kasutati WebRTC tehnoloogiat.

### 3.1.1 WebRTC

WebRTC on tasuta vabavaraline projekt, mida standardiseeritakse W3C konsortsiumi poolt ning mis võimaldab veebibrauserites ja mobiilirakendustes reaalajalist suhtlust (ingl *Real-Time Communication* ehk *RTC*) läbi lihtsate JavaScripti API-de. WebRTC missioon on võimaldada arendada kõrgekvaliteedilisi RTC rakendusi veebibrauseritele, mobiiliplatvormidele ja *IoT* seadmetele ning lubada neil kõigil ühenduses olla läbi ühiste protokollide [30], [31].

WebRTC kasutab kolme JavaScripti API-t: MediaStream, RTCPeerConnection ja RTCDataChannel. Lõputöö käigus arendatud rakenduses kasutati neist kahte esimest: MediaStream API-t selleks, et saada ligipääs kasutaja arvuti kaamerale ja mikrofonile ning RTCPeerConnection API-t partnervõrgu (ingl *peer-to-peer connection*) loomiseks ning video ja audio voogedastamiseks. RTCDataChanneli, mis on mõeldud muude andmete (nt failid, tekstsõnumid) edastamiseks, asemel kasutati signaalserverit ja Socket.IO võimalusi, seda peamiselt selleks, et sõnumivahetus ei toimuks vaid läbi partnervõrgu, vaid oleks võimalik seda ka andmebaasi salvestada [32].

Kuigi WebRTC võimaldab luua ühenduse partnervõrgu abil, vajab see toimimiseks siiski serverite abi. Seda kahel põhjusel: esiteks, et vahetada metaandmeid ja koordineerida

suhtlus, mis toimub signaalserveri vahendusel, ning teiseks põhjusel, et tulla toime võrguaadresside tõlkimise ja tulemüüridega. Selle jaoks on STUN ja TURN serverid [11].

### 3.1.2 Socket.IO

Rakenduse ja serveri vaheliseks suhtluseks võttis autor kasutusele Node.js mooduli Socket.IO. Socket.IO võimaldab reaalaja kahesuunalist sündmuspõhist suhtlust kasutades veebisokleid. See on ideaalne moodul sõnumivahetuse rakenduste ja signaalserverite arendustes, kuna töötab igal platvormil, brauseris ja seadmes [12]. Samuti pakub Socket.IO võimalust luua gruppe, mis tagab sõnumite jõudmise ainult sinna kuuluvatele kasutajatele [33].

### 3.1.3 Signaalserver

Enne, kui on võimalik luua heli ja video edastamiseks RTCPeerConnection, peavad osapooled vahetama üksteise kohta infot omavahelise ühenduse loomiseks. See infovahetus toimub läbi signaalserveri (ingl *signaling server*). Läbi signaalserveri vahetatakse järgnevaid sõnumeid:

- Sõnumid suhtluse avamiseks, sulgemiseks ning veasõnumite saatmiseks.
- Meedia metaandmeid nagu näiteks ribalaius ja meedia tüübid, mida mõlema osapoole brauserid oskavad käsitleda.
- ICE kandidaadid infoga nagu IP-aadress ja port vastase leidmiseks ja ühenduse saamiseks.

Signaalserver ei ole WebRTC API-des endas implementeeritud ja vajab eraldi arendust [11], [32]. Klienditeenindusrakenduses kasutati signaalserveri arenduseks tarkvaraplatvormi Node.js ning serveri ja kliendi vahel sõnumite edastamiseks veebisokleid.

### 3.1.4 SDP

SDP (*Session Description Protocol*) on formaat, mis kirjeldab voogedastatava meedia parameetreid ehk seda, kuidas toimub suhtlus pärast võrguühenduse loomist [10].

Lisas 1 on toodud SDP näide, kus on kirjeldamise lihtsustamiseks read nummerdatud. SDP esimesel kuuel real on kirjeldatud SDP versioon, sessiooni identifikaator, versioon, tekstiline nimi ning algus- ja lõpuaeg, samuti võrgu ja IP-aadressi tüüp ning IP-aadress.

Ridadel 7.-35. on audio kirjeldus paljude erinevate parameetritega ning ridadel 36.-77. kirjeldatakse video atribuute ja parameetreid. Näiteks 7. ja 36. reas, mis algavad “m=...”, tuuakse ära meedia tüüp, sessioonis kasutatav transpordiprotokoll ja toetatav meedia formaat ning ridades 8 ja 37, mis algavad “c=...”, on informatsioon ühenduse kohta, ehk IP-aadress, kuhu saadetakse ja kust oodatakse sõnumeid [34].

### 3.1.5 ICE raamistik ja ICE kandidaadid

ICE (*Interactive Connectivity Establishment*) on tehnika, mida kasutatakse partnervõrkude loomisel võrguaadresside tõlkimiseks (ingl *NAT traversal*) veebirakendustes, mis kasutavad video- ja heliedastust, sõnumivahetust ja muud ning on loodud pakkumise-vastuse meetodil [3].

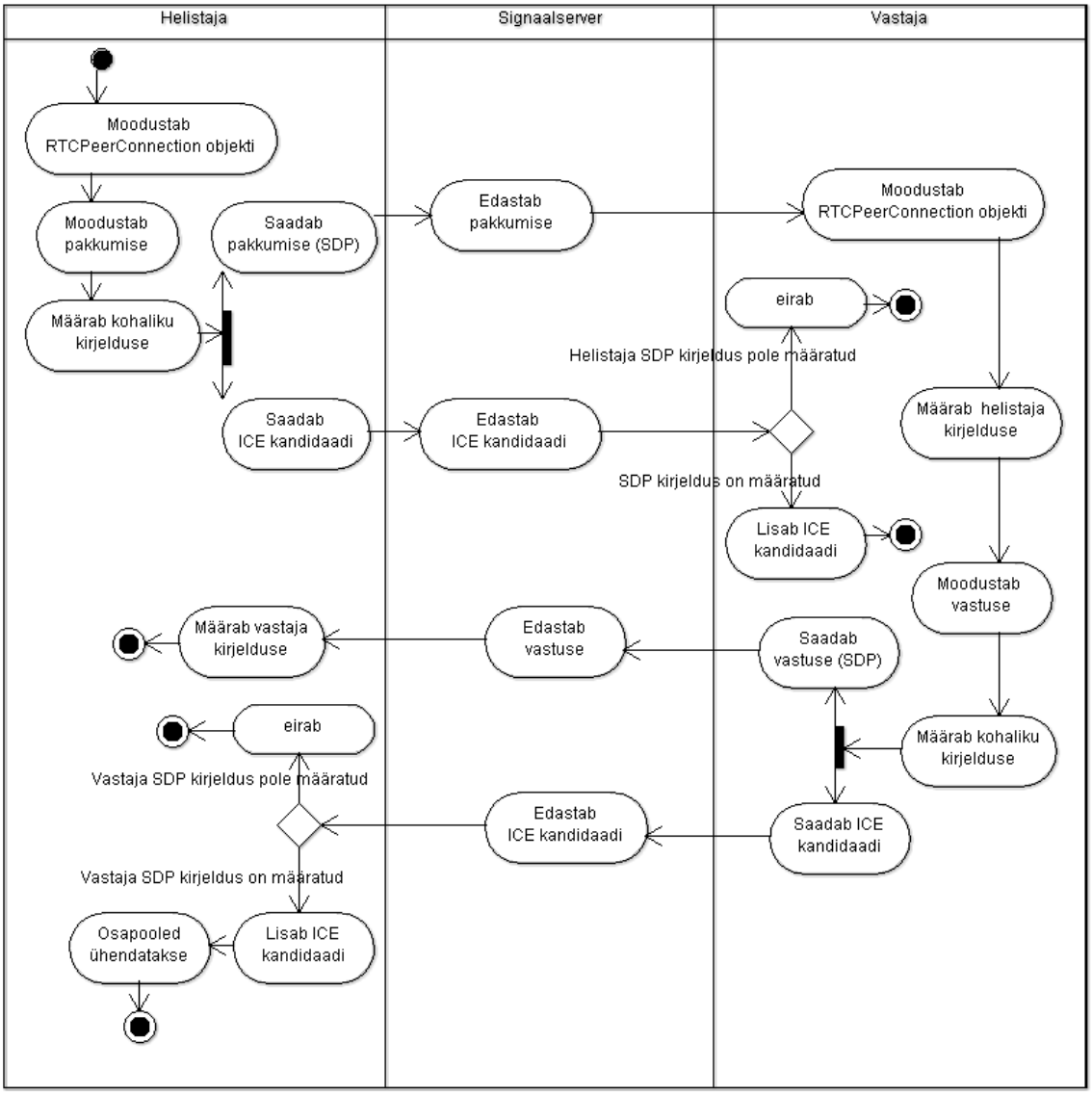
WebRTC rakendustes kasutatakse ICE raamistikku partnervõrkude loomiseks. Selleks lisatakse ICE serveri URL RTCPeerConnection objekti konfiguratsiooni. ICE proovib luua parima võimaliku teekonna kahe osapoole ühendamiseks, proovides paralleelselt kõiki võimalusi ning valides neist kõige efektiivsema. Esmalt proovitakse luua ühendus kasutades hosti aadressi, mis on saadud seadme operatsioonisüsteemist ja võrgukaardilt. Kui seadmed asuvad NAT „taga“, siis ühendus ebaõnnestub. Seejärel proovib ICE hankida STUN serverist seadme välise aadressi. Kui ka see ebaõnnestub, (ehk kogu otsese partnervõrgu loomine ebaõnnestub) suunatakse liiklus läbi TURN serveri [11].

ICE kandidaadid (ingl *candidates*), kujutavad endast lihtsalt öeldes kirjeldust, kuidas kliendiga ühenduda. ICE kandidaat sisaldab endas infot IP-aadressi ja pordi kohta, läbi mille infovahetus toimuma hakkab, samuti kasutatavat protokolli ja muutujat, mis näitab, kas kandidaat asub samas kohtvõrgus või tulemüüri taga. Selleks, et kaks kasutajat saaksid luua ühenduse, peavad nad oma ICE kandidaate teineteisega jagama [10].

### 3.1.6 RTCPeerConnection API

RTCPeerConnection on üks kolmest WebRTC API-st. Seda kasutatakse partnervõrgu loomiseks ning heli ja video edastamiseks. RTCPeerConnectioni esimene ülesanne on tuvastada osapoolte meedia omadused, nagu näiteks meedia tüüp (heli, video jne), transpordiprotokoll (RTP/UDP/IP, H.320 jne), meedia formaat (H.261 video, MPEG video jne) [35], mida kasutatakse pakkumiste (ingl *offer*) ja vastuste (ingl *answer*) moodustamisel. Teine ülesanne on leida kandidaadid ehk reaalsed võrguaadressid [11].

Kui kohalikud (ingl *local*) andmed on tuvastatud, vahetatakse need üle signaalserveri partneriga. RTCPeerconnectioni loomise tegevusdiagrammil (vt Joonis 1) on näidatud täpsemalt, kuidas toimub ühenduse loomine kahe osapoole vahel.




Joonis 1. RTCPeerConnectioni loomise tegevusdiagramm.

Esmalt moodustab helistaja RTCPeerConnection objekti ning selle meetodiga *createOffer()* pakkumise, mis salvestatakse ka kohalikuks kirjelduseks. Pakkumine saadetakse üle signaalserveri teisele osapoolele. Kohe pärast RTCPeerConnection objekti loomist hakkab see otsima ICE kandidaate. Kui kandidaat on leitud, saadetakse see üle signaalserveri teisele osapoolele [11].

Kui vastaja saab helistajalt pakkumise tema SDP kirjeldusega, loob ta esimesena samuti RTCPeerconnectioni objekti. Seejärel salvestab ta selle helistaja kirjelduseks. Pärast seda

moodustab ta vastuse ning salvestab selle ka enda kirjelduseks. Vastus saadetakse tagasi helistajale, mispeale helistaja selle partneri kirjeldusena salvestab [11].

Samal ajal kui on leitud ICE kandidaate, vahetatakse ka neid üksteisega, kuid kuna neid enne SDP kirjelduste saabumist kasutada ei saa, eiratakse neid hetkeni, kuni kirjeldused on salvestatud. Kui mõlemad kasutajad on kätte saanud teineteise ICE kandidaadid, luuakse nende vahel ühendus [11]. Kui ühendus on edukalt loodud, saab edasi andmeid saata otse osapoolte vahel või vajadusel läbi vahendava serveri. Voogedastuse eest vastutab edasi RTCPeerConnection [32].

### 3.1.7 STUN server

Kuna NAT tagatud kohtvõrgu IP-aadressid ei ole väliselt kasutatavad, ebaõnnestub tõenäoliselt esimene katse partnerite vahelise ühenduse loomiseks. Probleemi lahendamiseks on vajalik STUN (*Simple Traversal of UDP through NATs* [36]) serveri kasutamine. STUN serverid asuvad avalikus internetis ja neil on üks lihtne ülesanne: kontrollida sissetuleva päringu (ingl *request*) IP-aadressi ja porti ning saata see aadress vastusena (ingl *response*) tagasi. Teisisõnu: STUN serverit kasutatakse selleks, et saada teada oma väline IP-aadress ja port. Selle protsessi abil on võimalik ühel partneritest saada teada enda avalikult kättesaadav aadress ja saata see signaalserveri kaudu teisele osapoolele, et seejärel luua ühendus. Statistika kohaselt toimuvad 86% WebRTC kõnedest just tänu STUN serveritele [11].

### 3.1.8 TURN server

TURN (*Traversal Using Relays around NAT*) serverid võetakse kasutusele juhul, kui partnervõrgu loomine on ebaõnnestunud. Sellisel juhul suunatakse kogu osapoolte vaheline meediavahetus (video, heli, andmed) läbi TURN serveri. Selle tõttu on TURN serverid STUN serveritest palju võimekamad. TURN serveritel on avalikud aadressid, seega nendega saab ühendust ka siis, kui osapooled asuvad tulemüüride taga [11].

### 3.1.9 CSR ja SSL sertifikaat

SSL (*Secure Sockets Layer*) on protokoll, mida kasutatakse krüpteeritud ühenduse loomiseks veebiserveri ja brauseri vahel. See tagab, et brauseri ja serveri vahel saadetud andmed jäävad privaatseks. SSL kasutab avaliku ja privaatse võtme

krüpteerimissüsteemi. SSL ühenduse võimaldamiseks peab serverisse olema paigaldatud SSL sertifikaat [1], [37].

SSL sertifikaat on väike andmefail tõendamaks veebilehe omaniku identiteeti. See on oluline igal kodulehel, millel soovitakse tagada kasutajate turvatunne ning nende andmete turvalisus, eriti kui edastatakse privaatset infot. Sertifikaadi abil on võimalik kasutada HTTPS protokolli krüpteeritud andmete edastamiseks serveri ja külastaja vahel [38]. Lisaks sellele on SSL Sertifikaadi hankimine möödapääsmatu juhul, kui kodulehele plaanitakse arendada ID-kaardi teenuseid [39].

CSR (*Certificate Signing Request*) on plokk krüpteeritud teksti, mis moodustatakse esimese sammuna SSL sertifikaadi serverile taotlemiseks. CSR sisaldab endas infot nagu domeeni nimi, organisatsiooni nimi, alamorganisatsiooni nimi, riik, asukoht, e-posti aadress ja avalik võti. Samaaegselt CSR-ga genereeritakse ka privaatvõti, mida tuleb hoida saladuses [2].

Pärast CSR-i genereerimist edastatakse see CA-le (*Certificate Authority*), kes kasutab seda SSL sertifikaadi loomisel. SSL sertifikaat, mis on loodud mingist kindlast CSR-ist töötab vaid selle privaatvõtmega, millega see genereeriti [2].

Turvalisuse tagamiseks taotleti klienditeenindusrakendusele sertifikaat usaldusväärselt CA-lt Comodo CA Limited.

### 3.1.10 Amazon RDS

Amazon RDS (*Relational Database System*) on relatsioonilist andmebaasi pakkuv teenus, mis teeb andmebaaside pilve ülespanemise, haldamise ja vastavalt vajadusele nende suurendamise või vähendamise lihtsaks. Amazon RDS pakub kuut erinevat andmebaasi serverit. Lisaks sellele on võimalik teha Amazon RDS-is andmebaasi eksemplarist *Read Replica*, tänu millele saab tõhusalt andmebaasi koormust jagada ning õnnetuse korral andmebaasile tuleva liikluse sinna suunata [40].

Lõputöö käigus valminud klienditeeninduse rakendus kasutab andmete hoidmiseks PostgreSQL andmebaasi virtuaalserveri eksemplari.

### 3.1.11 Amazon EC2

Amazon EC2 (*Elastic Compute Cloud*) on teenus, mille abil on võimalik lihtsalt käivitada virtuaalserveri eksemplare vastavalt kasutaja vajadustele. Vaid mõne nupuvajutusega on võimalik muuta eksemplaride seadistusi vastavalt arvutusressursile, virtuaalserveri eksemplari mahule, mälu- ja CPU vajadusele. Amazon EC2 teenus on turvaline, sest järgib PCI DSS standardit, mis nõuab turvalise võrgu ehitamist, ülalpidamist, selle regulaarset testimist ja muid tegevusi, kaitsmaks kasutajate privaatseid andmeid [41]. Amazon EC2 eksemplarid asuvad privaatses isoleeritud virtuaalpilves ning see võimaldab ise otsustada, millised eksemplarid on privaatsed ning millised kättesaadavad. Lisaks sellele on võimalik lisada eksemplarid turvagruppidesse (*Security group*), ning siduda ligipääsu võimaldamiseks avalike IP-aadressidega. Amazon EC2 on disainitud kasutatama koos teiste Amazon Web teenustega, et pakkuda täielikku lahendust [42].

Klienditeenindusrakendus kasutab ühte Amazon EC2 virtuaalserveri eksemplari. Sellele on installitud Ubuntu 14.04 operatsioonisüsteem. Serveriga on seotud avalik IP-aadress, millele viitab ka A-kirje DNS-serveris. Samuti on serveriga seotud turvagrupp, kus määratakse ära, millistelt IP-aadressidelt saab sinna pöörduda. Rakenduse toimimiseks on avatud tulemüüris port 443 (HTTPS protokoll), mis on oluline turvalisuse tagamiseks.

### 3.1.12 Amazon S3

Amazon S3 (*Simple Storage Service*) on turvaline, vastupidav ja skaleeruv teenus failide hoiustamiseks pilves. Seal on mugav serveerida staatilisi veebilehti ja nende sisu [43]. Klienditeenindusrakenduse kliendi-poolse rakenduse serveerimiseks kasutati Amazon S3 teenust, kuhu paigaldati kõik HTML, CSS, JavaScripti ja pildifailid.

### 3.1.13 Amazon ELB

Amazon ELB (*Elastic Load Balancing*) on teenus, mis jaotab automaatselt rakendusele tuleva liikluse mitme Amazon EC2 virtuaalserveri eksemplari vahel pilves. See võimaldab saavutada rakendustele suurema veakindluse. Koormuse jaotamine (*Load Balancing*) tagab selle, et liiklust saaksid vaid töötavad virtuaalserveri eksemplarid. Elastne sealjuures tähendab seda, et skaleerutakse vastavalt liikluse muutusele ning selle eest on vastutav Auto Scaling [44].

### 3.1.14 Docker

Docker on töövahend, mis aitab luua, kokku panna ja paigaldada rakendusi, kasutades selleks konteinereid. Dockeri aluseks võetakse mingi tõmmis ja sinna hakatakse lisama erinevaid vajalikke konteinereid [45]. Docker võimaldab pakkida kõik rakenduse tööks vajalikud failid ühte konteinerisse ja garanteerib seega, et Docker jookseb samamoodi igal Linuxi masinal, vaatamata selle seadetele. Dockeri eelis näiteks virtuaalmasinate ees peitub võimaluses kasutada konteineritel sama Linuxi tuuma, optimeerides sellega muutmälu kasutamist ja vähendades rakenduse suurust, kuna lubab jagada samu faile rakenduste vahel. Lisaks sellele lisavad Dockeri konteinerid turvalisust, kuna on üksteisest ja muust infrastruktuurist isoleeritud [46].

Klienditeenindusrakenduses kasutatakse Dockerit kahe konteineriga: nginx ja api. Nginx server kuulab pordil 443 (HTTPS protokoll) sissetulevaid päringuid.

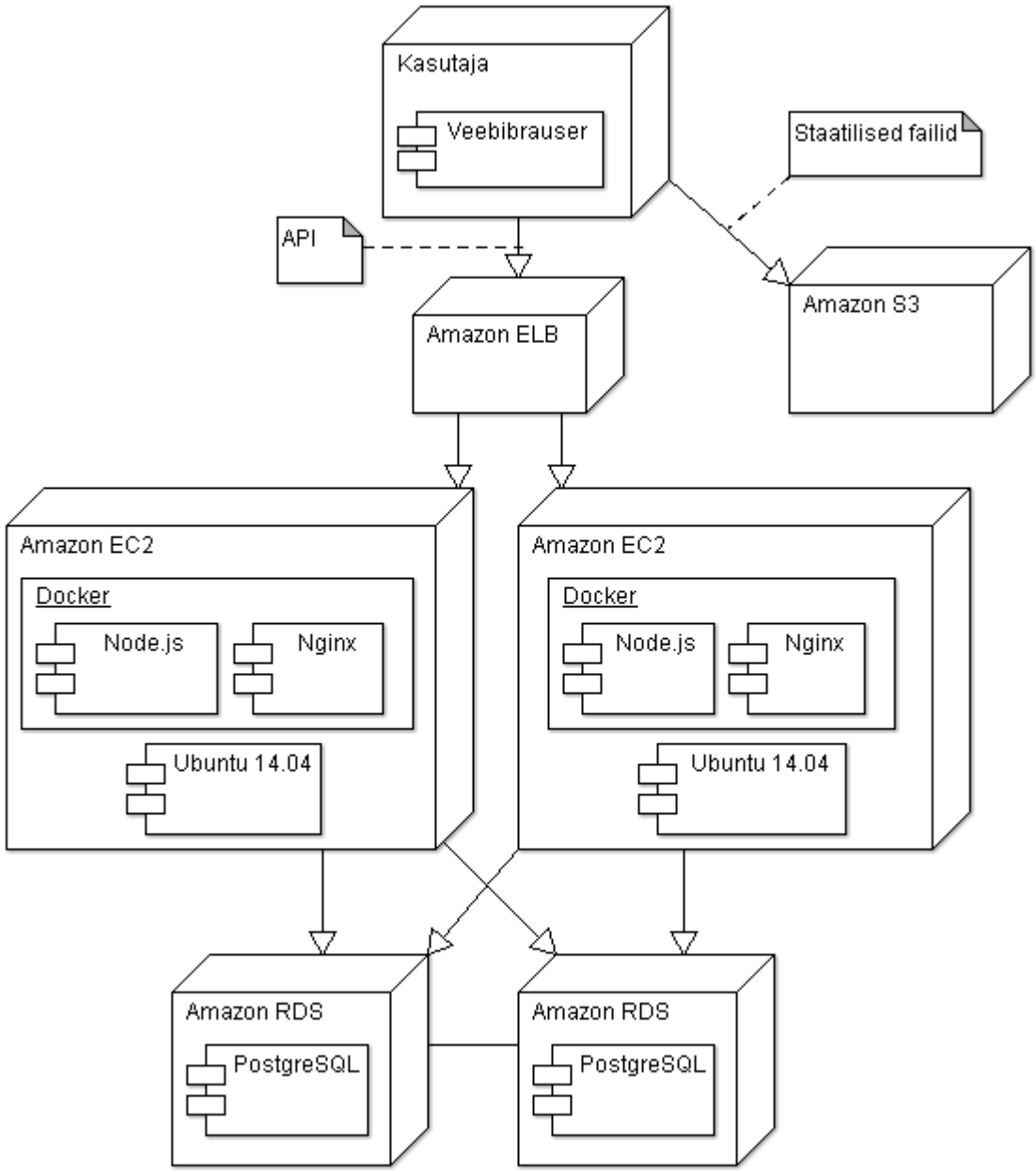
## 3.2 Süsteemi arhitektuur

Süsteemi arhitektuur on lihtne ning ülesehitatud nõnda, et serveritele langeks võimalikult väike koormus ja see suudaks teenindada korraga paljusid kasutajaid. Sellega seoses paigutati kõik staatilised failid Amazon S3 pilve, millega veebibrauser suhtleb HTTPS päringute abil.

Rakenduse serverid asuvad aga Amazon EC2 virtuaalserveri eksemplarides. Neid on mitu taaskord selleks, et jaotada koormust. Koormuse jaotamise ülesanne on Amazon ELB-il. Igale EC2 virtuaalserveri eksemplarile on installitud operatsioonisüsteem Ubuntu 14.04, millel omakorda jookseb Docker.

Rakendus kasutab andmete hoidmiseks ja lugemiseks Amazon RDS teenuse PostgreSQL andmebaasi virtuaalserveri eksemplari. Andmebaasi koormuse jaotamiseks kasutatakse ka selle *Read Replicat*, et muuta andmebaasi päringud kiiremaks ja töökindlamaks.

Rakenduse süsteemi arhitektuur on toodud kokkuvõtvalt joonisel 2.




Joonis 2. Rakenduse arhitektuur.

## 3.3 Andmebaasi tabelite kirjeldus

Rakenduse kasutatav andmebaas koosneb kolmest tabelist: *users*, *messages*, *calls*. Järgnevalt tuuakse välja tabelite veerud, nende tüübid, pikkused, võimalikud väärtused, vaikeväärtused ja kohustuslikkus.

Tabel 1. Andmebaasi tabeli *users* kirjeldus

| **Veeru nimi** | **Tüübi nimi** | **Pikkus** | **Võimalikud väärtused** | **Vaikeväärtus** | **Kohustuslik** |
|---|---|---|---|---|---|
| id | integer | | | nextval(’user_id’) | JAH |
| name | character varying | 40 | Nimi ei tohi olla tühi string või tühikutest koosnev string | | JAH |
| password | character varying | | Parool ei tohi olla tühi string või tühikutest koosnev string | | JAH |
| email | character varying | 40 | | | EI |
| online | boolean | | | | JAH |
| role | character varying | 10 | ’admin’ või ’client’ | | JAH |

Tabel 2. Andmebaasi tabeli *messages* kirjeldus

| **Veeru nimi** | **Tüübi nimi** | **Pikkus** | **Võimalikud väärtused** | **Vaikeväärtus** | **Kohustuslik** |
|---|---|---|---|---|---|
| id | integer | | | nextval(’message_id’) | JAH |
| from_id | integer | | | | JAH |
| to_id | integer | | | | JAH |
| message | text | | | | JAH |
| time | timestamp with time zone | | | current_timestamp | JAH |

Tabel 3. Andmebaasi tabeli *calls* kirjeldus

| Veeru nimi | Tüübi nimi | Pikkus | Võimalikud väärtused | Vaikeväärtus | Kohustuslik |
|---|---|---|---|---|---|
| uuid | character varying | | | | JAH |
| client_id | integer | | | | JAH |
| admin_id | integer | | | | JAH |
| start_time | timestamp with time zone | | | | EI |
| stop_time | timestamp with time zone | | | | EI |

## 3.4 Rakenduse kirjeldus

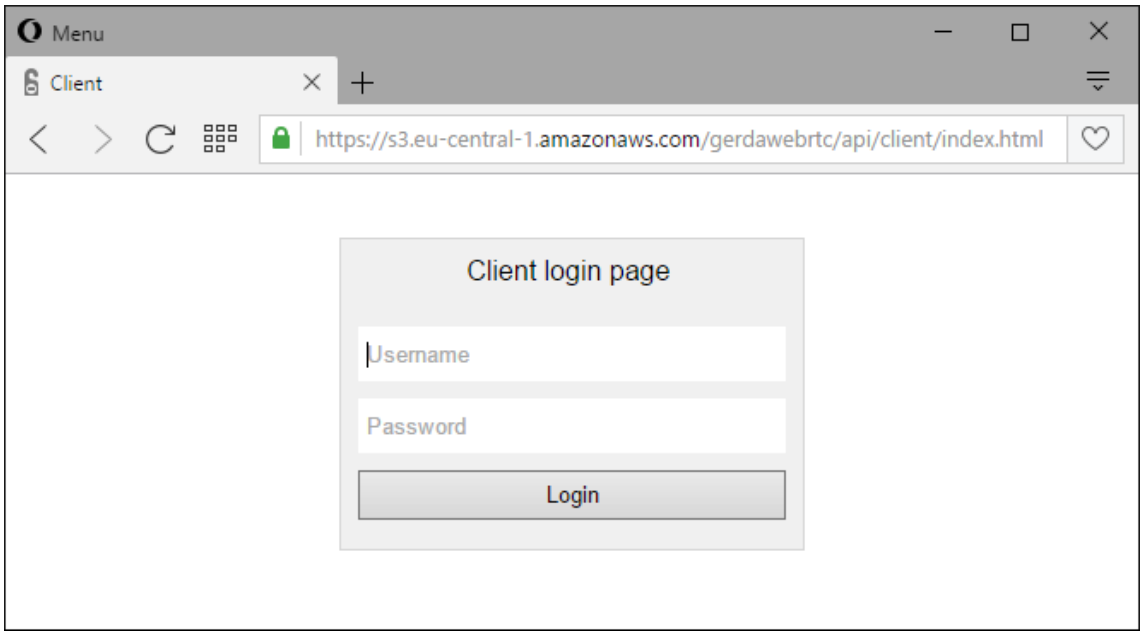
Töö autor arendas lõputöö käigus veebipõhise klienditeenindustarkvara, mis on mõeldud ettevõtetele, kes soovivad, et nende kliendid saaksid lihtsalt igal ajal ja kohas nende klienditeenindajatega ühendust võtta. Selle põhilised funktsioonid hõlmavad endast tekstsõnumite saatmist, videokõne pidamist ning failide saatmist.

Klienditeenindustarkvara kasutajaliides on tehtud võimalikult lihtsaks, matkides veidi teisi sõnumivahetuse rakendusi. See kõik on selleks, et kasutajal tekiks äratundmine ning programm oleks lihtsalt kasutatav.

### 3.4.1 Sisselogimine

Sisselogimise leht on nähtaval kõikidele tuvastamata kasutajatele. Sisselogimiseks peab kasutaja sisestama kasutajanime ja parooli. Selle peale käivitatakse sisselogimise funktsioon, mis kontrollib kasutajanime ja parooli vastavust ning nende sobivuse korral muudab andmebaasis kasutaja staatuse ning salvestab rakenduses kasutaja identifikaatori ja nime edaspidiseks kasutamiseks.

Joonisel 3 on näha rakenduse sisselogimise vaade. Sellel on kaks täidetavat välja – kasutajanimi (*Username*) ja parool (*Password*) ning nupp sisselogimise käivitamiseks.




Joonis 3. Sisselogimise vaade.

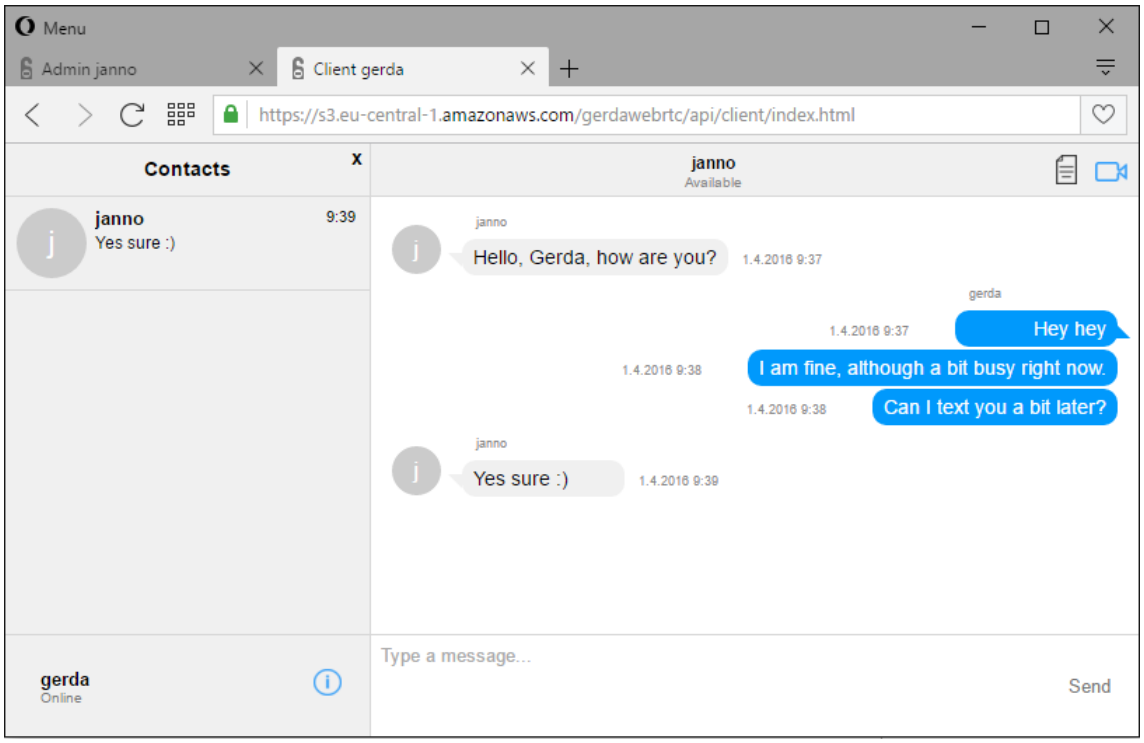
### 3.4.2 Kliendi või klienditeenindaja valimine

Kui sisseloginud kasutaja oli klient, kuvatakse talle pärast sisselogimist vasakus tulbas kõik *online* staatusega klienditeenindajad. Kui sisseloginud kasutaja oli klienditeenindaja, näeb ta vastupidi kõiki *online* staatusega kliente. Lisaks kasutaja nimele kuvatakse ka tema viimati saadetud sõnumit ning selle saatmise aega. Vasakust tulbast on võimalik valida kasutaja, kellega soovitakse suhtlust alustada ning selleks klõpsata tema nimel. Juhul, kui ühtegi kasutajat ei ole *online*, kuvatakse vasakus tulbas vastavasisuline teave.

*Online* kasutajad leiab server andmebaasist SQL lause abil, otsides tabelist *users* kõiki kasutajaid, kelle olek on *online* ja kelle roll (*role*) on vastavalt *client* või *admin*. Lisaks kasutajanimele leitakse ka veel tabelist *messages* selle kasutaja viimane sõnum ning selle saatmise aeg.

*Online* kasutajaid uuendatakse iga kord, kui keegi logib sisse, logib välja või saadab tekstsõnumi ning kogu nimekiri saadetakse vasaku tulba uuendamiseks kõikidele kasutajatele.

Joonisel 4 on näha vasakul asetsev kontaktide tulp ühe sisseloginud kasutaja ja tema viimati saadetud sõnumiga.




Joonis 4. Vasak kontaktide tulp ja tekstsõnumite vaade.

Kasutaja nimel klõpsates moodustatakse serveris uus Socket.IO grupp süntaksiga adminId_clientId. Näiteks, kui liituvad *admin*, kelle identifikaator on 1 ja *client*, kelle identifikaator on 2, moodustub grupp nimega 1_2. Peale grupi moodustamist liidetakse sokkel selle grupiga. Grupi moodustamine tagab, et sellega liitunud isikute vahel saadetud sõnumid jõuavad vaid nende kaheni. Teine kasutaja liitub grupiga juhul, kui ta klõpsab samuti selle sama kasutaja nimel vasakus tulbas, vastab videokõnele või failisaatmise pakkumisele.

### 3.4.3 Eelnevate sõnumite pärimine andmebaasist

Kui kasutaja on valinud vasakust kontaktide tulbast teise kasutaja, kellega ta soovib suhelda ja klõpsanud tema nimel, näeb ta eelnevalt nende vahel saadetud sõnumeid. Selle jaoks otsitakse SQL lausega andmebaasi tabelist *messages* kõik sõnumid (*message*), mille saatja (*from_id*) või saaja (*to_id*) on kas vastav klient või admin. Andmebaasist tulnud vastuse read saadetakse kasutajale.

Kasutajapoolne rakendus, saanud serverist JSON formaadis vastuse, moodustab sõnumitest JavaScripti objektid. Seejärel käiakse tsükliga läbi kõik sõnumid ja vastavalt sellele, kas tegu oli sisseloginud kasutaja või tema partneri saadetud sõnumiga,

omistatakse sellele CSS klass *mymessage* või *othermessage*, mis on vajalik sõnumite stiili ning paigutuse seisukohast.

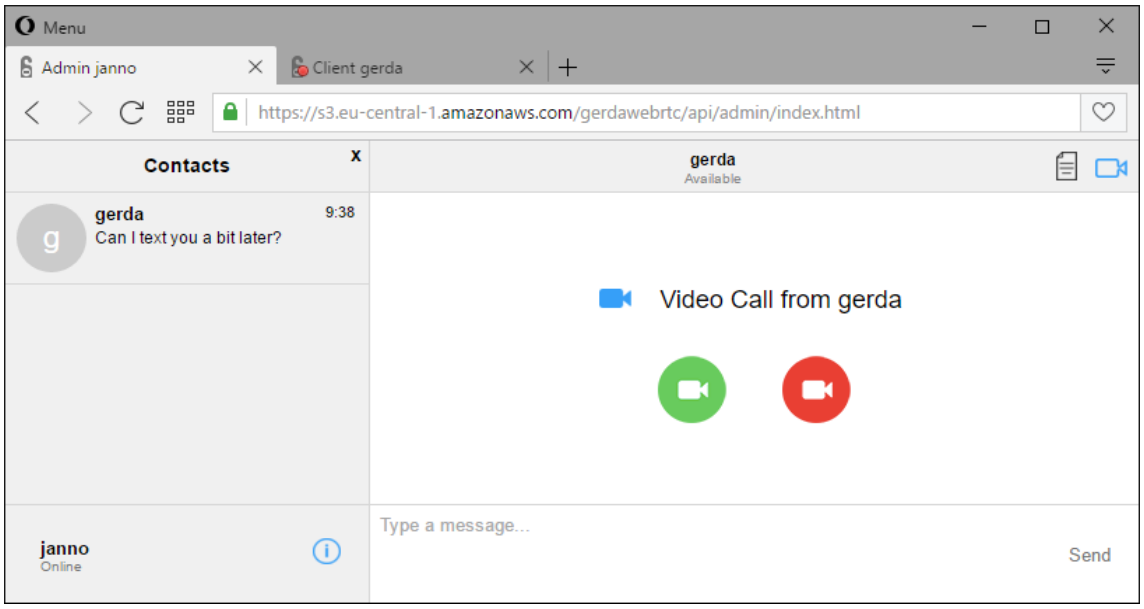
Joonisel 4 on näha kasutajate tekstsõnumite vahetamise vaade. Sellel on selgesti eristatavad sõnumid, kus sinise taustaga paremal pool asetsevad sõnumid on CSS klassiga *mymessage* ning kuuluvad sisseloginud kasutajale. Halli taustaga sõnumitel on aga CSS klass *othermessage* ning need kuuluvad teisele kasutajale. Lisaks sellele kuvatakse kõikide sõnumite juures saatja nime ja saatmise aega.

### 3.4.4 Videokõne

Kui sisseloginud kasutaja on valinud kontaktide tulbast teise kasutaja, on tal võimalik alustada videokõnet. Selleks tuleb vajutada videokõne alustamise nuppu rakenduse üleval paremas nurgas.

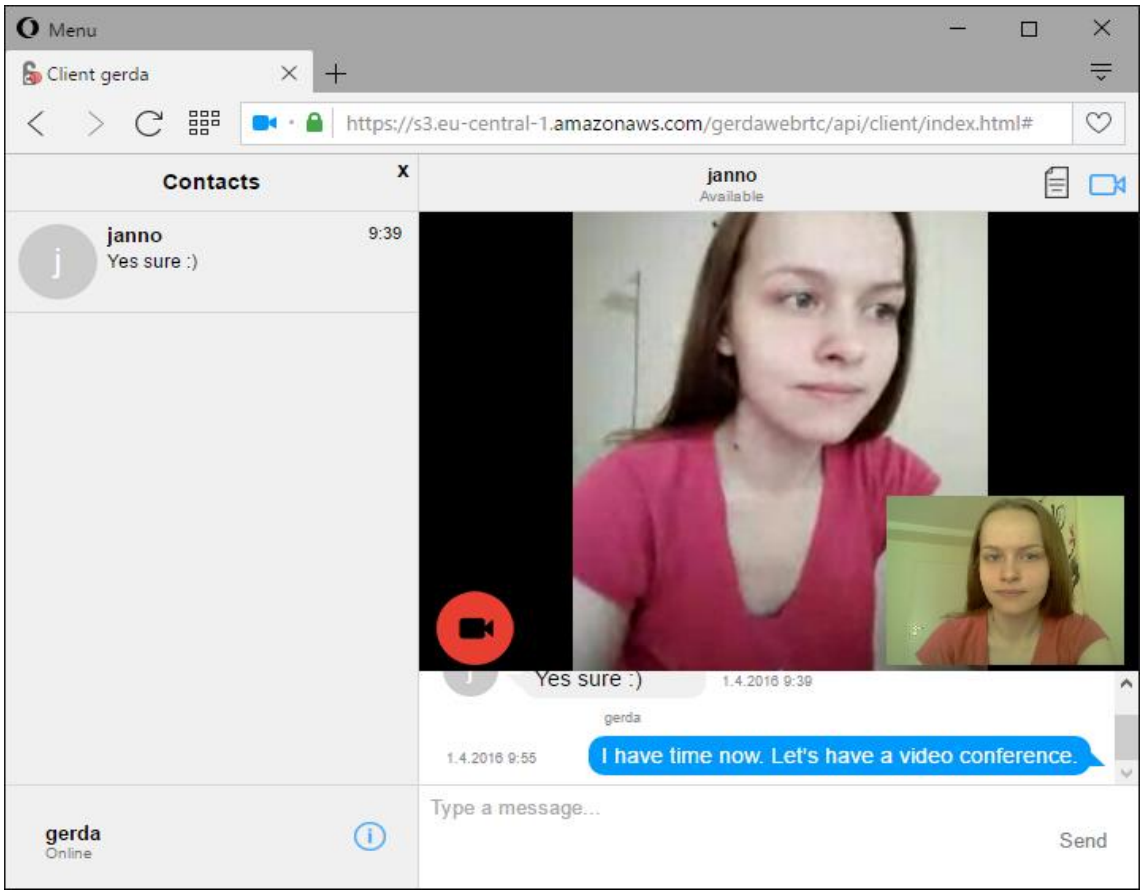
Videokõne alustamise nupule vajutades küsib brauser esmalt kasutajalt, kas ta soovib jagada oma kaamerat ja mikrofoni. Juhul, kui kasutaja arvutiga on ühendatud rohkem, kui üks audio- või videoseade, on tal võimalik nende hulgast valida sobivaim. Kui kasutaja on jagamisega nõus, kuvatakse talle videokõne vaade, millel on näha vaid tema video.

Videokõne vaate kuvamisega samaaegselt saadetakse üle signaalserveri videokõne kutsung teisele kasutajale (vt Joonis 5).Teisel kasutajal on võimalik kas kõnele vastata või vastamisest loobuda. Kõne vastuvõtmise nupule vajutades küsitakse ka sellelt kasutajalt brauseri poolt, kas ta soovib oma mikrofoni ja kaamerat jagada. Kui ka see kasutaja on lubanud meediaseadmete jagamise, kuvatakse ka talle videokõne vaade, luuakse RTCPeerConnection ning videokõne algab.




Joonis 5. Videokõnele vastamise vaade.

Joonisel 6 on näha rakenduse videokõne vaade. Selles vaates on võimalik näha enda ja partneri videot ning nende juures asuvat kõne lõpetamise nuppu. Peale selle on vaatel endiselt alles tekstsõnumite saatmise väli ja sõnumite loend, mida kuvatakse väiksemalt.




Joonis 6. Videokõne vaade.

Kui teine kasutaja on aktsepteerinud videokõne, lisatakse andmebaasi tabelisse *calls* uus rida. Tabelisse lisatakse *uuid, client_id, admin_id* ja *start_time*, kusjuures lisatav *uuid* moodustatakse serveris selleks ette nähtud funktsiooniga. Tabeli veerg *stop_time* jäetakse esialgu tühjaks.

Videokõne saab lõpetada vajutades videokõne lõpetamise nupule video all vasakus nurgas. Videokõne lõpetamisel leitakse SQL tabelist *calls* viimane rida, mis on seotud vastava *client_id* ja *admin_id*-ga ning mille veerg *stop_time* on tühi. Sinna lisatakse kõne lõppemise aeg.

### 3.4.5 Tekstsõnumi saatmine

Ka tekstsõnumi saatmiseks peab kasutaja olema esmalt võtnud ühendust teise kasutajaga. Seejärel saab ta rakenduse all olevasse sõnumikasti kirjutada oma sõnumi ning saatmiseks vajutada kas Enter või nuppu „Send“, kusjuures tühja sõnumit saata ei saa.

Sõnumi saatmisel ilmub see saadetud sõnumite loetellu ning saadetakse läbi sokli edasi ka serverile. Server annab käsu salvestada sõnum andmebaasi tabelisse *messages* ning edastab selle ka teisele kasutajale. Ka teine kasutaja näeb seepeale uut sõnumit oma sõnumite loetelus.

### 3.4.6 Faili saatmine

Faili saatmiseks teisele kasutajale tuleb vajutada faili saatmise nupule. Nupule vajutus avab faili otsimise akna ning lubab valida sealt ühe faili. Kui fail on valitud, luuakse kliendi ja serveri vahel voogedastus ning laetakse läbi selle fail serverisse.

Serveris salvestatakse fail kausta „files“. Voogedastuse lõppemisel saadetakse teisele kasutajale pakkumine, kas ta soovib nimetatud faili alla laadida. Klõpsates „Jah“, algab faili allalaadimine taas läbi voogedastuse, seekord serverist läbi sokli kliendile.

Kliendi poolel võetakse fail vastu tükkide kaupa ning lisatakse puhvrisse. Voogedastuse lõppedes liidetakse tükid kokku ning moodustatakse nendest BLOB (*Binary Large OBject*) ja samuti sellele viitav URL. Seejärel lisatakse moodustatud URL sõnumite loetellu lingina, millele klikkides on võimalik fail alla laadida. Lisaks sellele saadetakse ka sõnum faili nimega, et andmebaasi jääks märge saadetud faili kohta.

Faili on võimalik alla laadida vaid selle sessiooni käigus ning akna värskendamisel vastav link kaob. Uue sessiooni käigus näeb vanade sõnumite hulgas küll saadetud faili, kuid seda uuesti alla laadida ei saa.

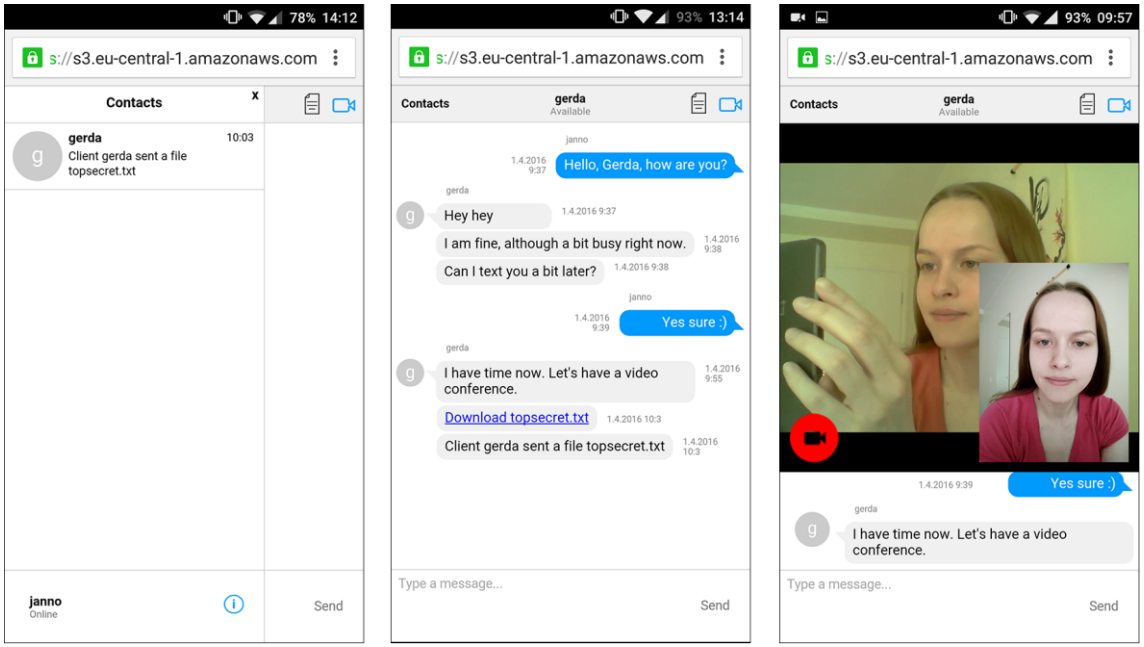
## 3.5 Rakenduse mobiiliversioon

Töö eesmärkidest tulenevalt oli oluline pöörata tähelepanu rakenduse kasutatavusele ja mugavusele mobiiltelefonidel. Joonisel 7 on välja toodud rakenduse mobiiliversiooni põhilised vaated.

Joonisel 7a on näha vasak menüü ja selle asetus. Vaikimisi on mobiiliversioonis menüü suletud, kuid selle saab avada vajutades rakenduse vasakul üleval asuvat linki „Contacts“. Samuti ei asu vasak menüü enam tekstsõnumite välja kõrval, vaid kuvatakse selle peal.

Joonisel 7b on näha rakendus, kui vasak menüü on suletud. Samuti näeb seal linki, millele vajutades on võimalik alla laadida kasutaja saadetud fail. Mobiilivaates on vähendatud sõnumite laiust, et mahutada ära kuupäev ja kellaaeg.

Joonisel 7c on videokõne vaade, mille on näha, et saadetud sõnumite kuvamise ala on muudetud väiksemaks ja video laius on määratud äärest ääreni.




Joonis 7. Rakenduse mobiiliversioon: (a) vasak menüü, (b) tekstsõnumite vaade ja faili allalaadimine, (c) videokõne vaade.

# 4 Testimine

Lõputöö tulemuste hindamiseks pidas autor vajalikuks testida rakenduse toimimist erinevates veebibrauserites. Testid saab jaotada selle alusel, millises veebibrauseris ja võrgus need toimusid (kas samas kohtvõrgus).

WebRTC MediaStream API on toetatud veebibrauseritel Chrome, Opera, Firefox ja Edge. RTCPeerConnectionit toetavad Chrome, Opera ja Firefox [32]. Kuna rakendus vajab toimimiseks neid mõlemat, jäeti Microsoft Edge brauseri testimine siinkohal välja.

Kuna olenevalt veebibrauserist võib rakendus nö hõivata mikrofoni ja kaamera ning seega mitte võimaldada enam teisel neid kasutada, viidi testid läbi korraga mitmel erineval seadmel. Rakenduse toimimist testiti Macbookil, Mac minil, kahel erineval Windows 10 operatsioonisüsteemiga personaalarvutil ja ühel Android 5.1.1 mobiiltelefonil.

Testiti rakenduse nelja põhilist funktsiooni:

- Kõne algatamine, mispeale pidi partnerile kuvatama kõnele vastamise vaade ja vastamine, mis pidi looma partnervõrgu ja edastama partnerite video ja heli.
- Kõne lõpetamine, mis pidi voogedastuse lõpetama.
- Tekstsõnumi saatmine, mille käigus pidi uus sõnum jõudma teise kasutaja ekraanile.
- Faili saatmine, mille käigus pidi teisele kasutajale jõudma link ja sellele klõpsates pidi alla laadima sama fail, mis saadeti.

Testide tulemused on toodud järgnevas tabelis:

Tabel 4. Testitulemused.

| **Kliendi veebibrauser** | **Admini veebibrauser** | **Võrk** | **Kõne algatamine** | **Kõne lõpetamine** | **Tekstsõnumi ja faili saatmine** |
|---|---|---|---|---|---|
| Google Chrome 50.0 | Google Chrome 50.0 | Sama kohtvõrk | toimis | toimis | toimis |
| Google Chrome 50.0 | Mozilla Firefox 45.0 | Sama kohtvõrk | toimis | toimis | toimis |
| Google Chrome 50.0 | Opera 37.0 | Sama kohtvõrk | toimis | toimis | toimis |
| Opera 37.0 | Chrome 50.0 for mobile | Sama kohtvõrk | toimis | toimis | toimis |
| Google Chrome 50.0 | Google Chrome 50.0 | Erinev kohtvõrk | toimis | toimis | toimis |
| Google Chrome 50.0 | Mozilla Firefox 45.0 | Erinev kohtvõrk | toimis | toimis | toimis |
| Google Chrome 50.0 | Opera 37.0 | Erinev kohtvõrk | toimis | toimis | toimis |
| Opera 37.0 | Chrome 50.0 for mobile | Erinev kohtvõrk | toimis | toimis | toimis |

# 5 Rakenduse võimalikud edasiarendused

Põhjusel, et lõputöö käigus valmis vaid esmane lihtne prototüüp klienditeenindusrakendusele, on autoril mitmeid ideid, kuidas seda edasi arendada.

Esimene võimalik edasiarendus võiks olla kasutajate autentimine ID-kaardiga ning failide digiallkirjastamine rakenduses. See võib olla oluline mõnedele ettevõtetele, kes tegelevad privaatse infoga ning kus on eriti oluline kontrollida kasutaja õigsust. See võimalus teeks allkirjastatud dokumentide jagamise oluliselt lihtsamaks, kuna siis ei peaks faili allkirjastamiseks avama selleks teist programmi.

Teisena võiks rakenduses olla võimalik uute kasutajate registreerimine. Kasutajal võiks olla võimalus end ühekordselt ID-kaardiga tuvastada ning seejärel endale uus kasutajakonto tekitada. Nii on võimalik järgmisel korral sisse logida vaid kasutajanime ja parooli kasutades. Mõnel puhul ei pruugi aga ID-kaardiga kasutaja tuvastamine üldse oluline olla, näiteks kui tarkvara soovib endale mõni veebipood, kus klienditeenindus hõlmab endast vaid toodete kohta käiva info andmises. Sellisel puhul võiks kasutajate registreerimine sisaldada andmete sisestamist ja kinnitamist näiteks e-posti vahendusel.

Kolmandana arendaks autor rakendusse juurde profiili funktsionaalsuse. See sisaldaks endast kasutaja andmete lisamist ja muutmist ning samuti profiilipildi lisamist. See funktsionaalsus on autori arust oluline autoriteedi lisamiseks, sest kui kliendil on võimalus näha, millise klienditeenindajaga ta suhtleb, võid see lisada turvatunnet ja lugupidamist klienditeenindaja ja kogu ettevõtte suhtes.

# Kokkuvõte

Käesolevas töös analüüsiti Eesti suurettevõtete klienditeeninduse kättesaadavust ning pakutavaid lahendusi. Kodulehekülgi uurides selgus, et enim panustavad suurettevõtted klienditeenindusse suurtes linnades, jättes maakohad tihti tähelepanuta. Enamus ettevõtteid pakuvad kontorite kõrval ka võimalust suhelda klienditeenindajatega interneti või telefoni teel, tuues kodulehel välja telefoninumbri ja meiliaadressi. Mõned uuritud ettevõtetest pakkusid ka rakendusi, mille abil saaksid kliendid kasutajatoega otse suhelda, kuid eeldasid rakenduste või pistikprogrammide allalaadimist.

Töö tulemusena valmis analüüs, töötati välja tehniline lahendus ja arendati prototüüp rakendusele, mis aitab ettevõtetel olla ühenduses oma klientidega lihtsal, turvalisel ja mugaval moel üle interneti. Ühelt poolt aitab see ettevõtetel hoida kokku raha, mis kulub kontorite avamisele ja tööshoidmisele ning teisalt muudab klientide elu mugavamaks. Prototüüprakendus võimaldab kasutajatel ja klienditeenindajatel sisse logida ja üksteisega turvaliselt üle HTTPS protokolli ühendust võtta. Selle abil on võimalik pidada videokõnesid, saata tekstsõnumeid ning faile. Veebirakendusele valmis ka mobiiliversioon, mis teeb selle kasutamise mugavaks mobiilsetel seadetel.

Rakenduse töötamist testiti WebRTC protokolli toetavatel brauseritel nagu Mozilla Firefox, Opera, Google Chrome ja Google Chrome for mobile. Testide käigus prooviti tekstsõnumite saatmist, videokõne alustamist ja lõpetamist ning failide saatmist. Tulemused olid positiivsed ning rakendus töötas nii samas kohtvõrgus kui ka erinevates kohtvõrkudes asuvate seadete vahel. Sellega võib lugeda töö eesmärgid täidetuks.

# Kasutatud kirjandus


[1] Vallaste, H. – *e-Teatmik*, 2016. [WWW] http://www.vallaste.ee/index.asp (01.05.2016)

[2] What is a CSR (Certificate Signing Request)? – *SSL Shopper*. [WWW] https://www.sslshopper.com/what-is-a-csr-certificate-signing-request.html (26.04. 2016)

[3] Rosenberg, J. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols – *Internet Engineering Task Force (IETF)*, 2010. [WWW] https://tools.ietf.org/html/rfc5245 (26.04.2016)

[4] Kiriland, A. IoT- Asjade Internet – *Küberfüüsikaline süsteemitehnika*, 2016. [WWW] http://kfst.ee/iot-asjade-internet/ (26.04.2016).

[5] jQuery – *The jQuery Foundation.* [WWW] https://jquery.com/. (01.05.2016)

[6] Introducing JSON. [WWW] http://www.json.org/ (01.05.2016)

[7] What is Network Address Translation? – *What Is My IP Address.* [WWW]. http://whatismyipaddress.com/nat (01.05.2016)

[8] About Node.js – *Node.js Foundation.* [WWW] https://nodejs.org/en/about/ (01.05.2016)

[9] Rouse, M. Real-time communications (RTC) – *TechTarget*, 2008. [WWW]. http://searchunifiedcommunications.techtarget.com/definition/real-time-communications (01.05.2016)

[10] Tully, S. A Dead Simple WebRTC Example, 2014. [WWW] https://shanetully.com/2014/09/a-dead-simple-webrtc-example (26.04.2016)

[11] Dutton, S. WebRTC in the real world: STUN, TURN and signaling – *HTML5 Rocks*, 2013. [WWW] http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/ (25.04.2016)

[12] Socket.IO. [WWW] http://socket.io/ (26.04.2016)

[13] Rouse, M. UUID (Universal Unique Identifier) – *TechTarget*, 2005. [WWW]. http://searchsoa.techtarget.com/definition/UUID (01.05.2016)

[14] Männik, J. Keskerakond pärib aru Eesti Energia kontorite sulgemise kohta – *Riigikogu*, 2015. [WWW] http://www.riigikogu.ee/fraktsioonide-teated/eesti-keskerakonna-fraktsioon/keskerakond-parib-aru-eesti-energia-kontorite-sulgemise-kohta/ (24.04.2016)

[15] Kald, I. Pangakontorite sulgemise seiskab riigi sekkumine – *Äripäev*, 2014. [WWW] http://www.aripaev.ee/uudised/2014/06/11/palo-turg-pangakontorite-sulgemise-probleemi-ei-lahenda (24.04.2016)

[16] Rudi, H. Interneti võidukäik sunnib kindlustust kontorit sulgema – *Tarbija24*, 2016. [WWW] http://tarbija24.postimees.ee/3646381/interneti-voidukaik-sunnib-kindlustust-kontorit-sulgema (24.04.2016)

[17] Jones, R. Barclays rolls out face-to-face video banking – *The Guardian*, 2014. [WWW] http://www.theguardian.com/business/2014/nov/30/barclays-roll-out-face-to-face-video-banking (24.04.2016)

[18] LiveBank – *Ailleron SA*, 2015. [WWW] http://livebank24.com/ (24.04.2016)

[19] Browser Statistics – *w3schools.com*. [WWW] http://www.w3schools.com/browsers/browsers_stats.asp (11.05.2016)

[20] Esindused ja pangakanalid – *Swedbank*. [WWW] https://www.swedbank.ee/private/home/more/channels?language=EST (02.05.2016)

[21] Telefonipank – *Swedbank*. [WWW] https://www.swedbank.ee/private/home/more/channels/phone_bank (24.04.2016)

[22] Kontakt – *LHV*, 2016. [WWW] https://www.lhv.ee/ettevottest/kontakt/ (24.04.2016)

[23] Kontaktandmed – *Nordea Bank* [WWW] http://www.nordea.ee/Nordea+pangast/Nordea+pangast/Kontaktandmed/66522.html#1b9541d8-0273-4a42-bd65-edfa1dc9171a (24.04.2016)

[24] Eraklienditeenindus – Tele2 Eesti. [WWW] https://tele2.ee/kontaktid/eraklienditeenindus (24.04.2016)

[25] Tele2 Eesti – *Facebook*, 2016. [WWW] https://www.facebook.com/tele2eesti/app/495773287160178 (24.04.2016)

[26] Kontaktinfo – *Telia Eesti AS*. [WWW] https://www.telia.ee/ettevottest/kontaktid#kaardid (24.04.2016)

[27] Anna tagasisidet – Telia Eesti AS, [WWW] https://www.telia.ee/abi/feedback (24.04.2016)

[28] Seesami E-pood. Klienditugi – *Seesam Insurance AS*. [WWW] https://pood.seesam.ee/et (24.04.2016)

[29] Potter, P. Süsteemi nõuete esiletoomine ja analüüs, 2014. [WWW] http://maurus.ttu.ee/sts/wp-content/uploads/2011/10/Süsteemi-nõuete-esiletoomine-ja-analüüs.pdf (24.04.2016)

[30] WebRTC. [WWW] https://webrtc.org/faq/#what-is-webrtc (26.04.2016)

[31] Frequent Questions. What is WebRTC? – *WebRTC.* [WWW] https://webrtc.org/faq/#what-is-webrtc (11.05.2016)

[32] Dutton, S. Getting Started with WebRTC – *HTML5 Rocks*, 2012. [WWW] http://www.html5rocks.com/en/tutorials/webrtc/basics (26.04.2016)

[33] Rooms and Namespaces – *Socket.IO*. [WWW] http://socket.io/docs/rooms-and-namespaces/ (26.04.2016)

[34] Anatomy of a WebRTC SDP – *webrtcHacks*. [WWW] https://webrtchacks.com/sdp-anatomy/ (11.05.2016)

[35] Handley, M., Jacobson, V., Perkins, C. SDP: Session Description Protocol – *Network Working Group*, 2006. [WWW] https://tools.ietf.org/html/rfc4566 (25.04.2016)

[36] STUN – *Voip-info.org*, 2016. [WWW] http://www.voip-info.org/wiki/view/STUN (30.04.2016)

[37] What is SSL? – *SSL.com.* [WWW] http://info.ssl.com/article.aspx?id=10241 (11.05.2016)

[38] SSL sertifikaadid – *Virtuaal.com*. [WWW] https://www.virtuaal.com/teenused/ssl (26.04.2016)

[39] Kasutaja tuvastamine veebis – *eID*, 2014. [WWW] https://eid.eesti.ee/index.php/Kasutaja_tuvastamine_veebis (26.04.2016)

[40] Amazon RDS – *Amazon Web Services, Inc.* [WWW] https://aws.amazon.com/rds/ (26.04.2016)

[41] PCI Compliance – *Amazon Web Services, Inc.* [WWW] https://aws.amazon.com/compliance/pci-dss-level-1-faqs/ (11.05.2016)

[42] Amazon EC2 - Virtual Server Hosting – *Amazon Web Services, Inc.* [WWW] https://aws.amazon.com/ec2 (27.04.2016)

[43] Amazon S3 – *Amazon Web Services, Inc.* [WWW] https://aws.amazon.com/s3/ (30.04.2016)

[44] Elastic Load Balancing – *Amazon Web Services, Inc.* [WWW] https://aws.amazon.com/elasticloadbalancing/ (27.04.2016)

[45] Seerme, Ü. Docker – *EIK wiki*, 2015. [WWW] https://wiki.itcollege.ee/index.php/Docker (26.04.2016)

[46] What is Docker? – *Docker*. [WWW] https://www.docker.com/what-docker (26.04.2016)

# Lisa 1 – SDP näide

1. v=0
2. o=- 4511119904138919186 2 IN IP4 127.0.0.1
3. s=-
4. t=0 0
5. a=group:BUNDLE audio video
6. a=msid-semantic: WMS zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc
7. m=audio 9 UDP/TLS/RTP/SAVPF 111 103 104 9 0 8 106 105 13 126
8. c=IN IP4 0.0.0.0
9. a=rtcp:9 IN IP4 0.0.0.0
10. a=ice-ufrag:aycfxxNME9g+A6XA
11. a=ice-pwd:JNLxEyT38HqRsN9nB9gN3vLP
12. a=fingerprint:sha-256
    B1:E8:D0:D0:90:BE:8C:15:C3:8F:0A:EA:45:4F:46:BF:41:E6:E3:B4:47:F5:E4:
    11:C1:60:D8:0D:22:FC:EA:DA
13. a=setup:active
14. a=mid:audio
15. a=extmap:1 urn:ietf:params:rtp-hdrext:ssrc-audio-level
16. a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
17. a=sendrecv
18. a=rtcp-mux
19. a=rtpmap:111 opus/48000/2
20. a=rtcp-fb:111 transport-cc
21. a=fmtp:111 minptime=10; useinbandfec=1
22. a=rtpmap:103 ISAC/16000
23. a=rtpmap:104 ISAC/32000
24. a=rtpmap:9 G722/8000
25. a=rtpmap:0 PCMU/8000
26. a=rtpmap:8 PCMA/8000
27. a=rtpmap:106 CN/32000

28. a=rtpmap:105 CN/16000
29. a=rtpmap:13 CN/8000
30. a=rtpmap:126 telephone-event/8000
31. a=maxptime:60
32. a=ssrc:1213173530 cname:Pb/ApbsEka/tvvAP
33. a=ssrc:1213173530 msid:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc 7ad16938-a731-45fd-8bbe-6b9a94af5042
34. a=ssrc:1213173530 mslabel:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc
35. a=ssrc:1213173530 label:7ad16938-a731-45fd-8bbe-6b9a94af5042
36. m=video 9 UDP/TLS/RTP/SAVPF 100 101 116 117 96 97 98
37. c=IN IP4 0.0.0.0
38. a=rtcp:9 IN IP4 0.0.0.0
39. a=ice-ufrag:aycfxxNME9g+A6XA
40. a=ice-pwd:JNLxEyT38HqRsN9nB9gN3vLP
41. a=fingerprint:sha-256 B1:E8:D0:D0:90:BE:8C:15:C3:8F:0A:EA:45:4F:46:BF:41:E6:E3:B4:47:F5:E4:11:C1:60:D8:0D:22:FC:EA:DA
42. a=setup:active
43. a=mid:video
44. a=extmap:2 urn:ietf:params:rtp-hdrext:toffset
45. a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
46. a=extmap:4 urn:3gpp:video-orientation
47. a=sendrecv
48. a=rtcp-mux
49. a=rtpmap:100 VP8/90000
50. a=rtcp-fb:100 ccm fir
51. a=rtcp-fb:100 nack
52. a=rtcp-fb:100 nack pli
53. a=rtcp-fb:100 goog-remb
54. a=rtcp-fb:100 transport-cc
55. a=rtpmap:101 VP9/90000
56. a=rtcp-fb:101 ccm fir
57. a=rtcp-fb:101 nack
58. a=rtcp-fb:101 nack pli

59. a=rtcp-fb:101 goog-remb
60. a=rtcp-fb:101 transport-cc
61. a=rtpmap:116 red/90000
62. a=rtpmap:117 ulpfec/90000
63. a=rtpmap:96 rtx/90000
64. a=fmtp:96 apt=100
65. a=rtpmap:97 rtx/90000
66. a=fmtp:97 apt=101
67. a=rtpmap:98 rtx/90000
68. a=fmtp:98 apt=116
69. a=ssrc-group:FID 3467039524 2340372642
70. a=ssrc:3467039524 cname:Pb/ApbsEka/tvvAP
71. a=ssrc:3467039524 msid:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc d3cbaae6-6a6d-4a57-bfee-cdda9bb60321
72. a=ssrc:3467039524 mslabel:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc
73. a=ssrc:3467039524 label:d3cbaae6-6a6d-4a57-bfee-cdda9bb60321
74. a=ssrc:2340372642 cname:Pb/ApbsEka/tvvAP
75. a=ssrc:2340372642 msid:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc d3cbaae6-6a6d-4a57-bfee-cdda9bb60321
76. a=ssrc:2340372642 mslabel:zE3Tx8TVoF6WntxMV75AAPzdlNPoNlB5fBBc
77. a=ssrc:2340372642 label:d3cbaae6-6a6d-4a57-bfee-cdda9bb60321