TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

Kaugjuhitavliides Kantaris meediamängijale

Bakalaureusetöö

Üliõpilane: Jaak-Alo Lukanenok
Üliõpilaskood: 103797
Juhendaja: Evelin Halling

Tallinn
2014

Autorideklaratsioon

Deklareerin, et käesolev lõputöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud.

…….. ……
(kuupäev) (lõputöö kaitsja allkiri)

Annotatsioon

Käesoleva lõputöö eesmärgiks on muuta vabavaralise Kantaris Mediaplayer'i lähtekoodi nõnda, et Kantaris Mediaplayer'it oleks võimalik kaugjuhtida töö käigus loodud veebirakenduse abil.

Töö tulemusena on võimalik Kantaris Mediaplayer'is mängivaid meediafaile panna pausi, taasjätkata, muuta helitugevust või sulgeda Kantaris Mediaplayer'it teise interneti ühendatud seadme abil, mis asub samas kohtvõrgus.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 42 leheküljel, 8 peatükki ja 1 joonist.

Abstract

Aim of the thesis is to change freeware open-source Kantaris Mediaplayer sourcecode in the way that it could possible to operate remotely Kantaris Mediaplayer with created web application. As a result of thesis, it is possible to play, pause, change volume of opened mediafile or close Kantaris Mediaplayer remotly with another device that is connected to internet.

The thesis is in Estonian and contains 42 pages of text, 8 chapters and 1 figure.

1 Sisukord

2 Kasutatud lühendite loetelu ja definitsioonid

HTTP - Hypertext Transfer Protocol (ing. k), protokoll, mida kasutatakse info vahetamiseks arvutivõrkudest
IDE - Integrated Development Environment (ing. k), integreeritud arenduskeskkond on keskkond, mida kasutatakse programmide ja rakenduste arendamiseks
JSON - Javascript Object Notation (ing. k), standard info vahetamiseks veebirakenduste vahel
QR - Quick Response Code (ing. k), maatriks ribakood
WCF - Windows Communication Foundation (ing. k), rakendusraamistik, millega ehitada teenus-orienteeritud rakendusi
MVC - Model-View-Controller (ing. k), Mudel-Vaade-Kontroller on tarkvara disaini muster
SQL - Structured Query Language (ingl. k). Struktueeritud päringu keel, kasutatakse andmebaaside ja andmebaasis olevate andmete manipuleerimiseks

3 Sissejuhatus

3.1 Ülesande püstitus.

Käesoleva lõputöö eesmärgiks on vabavaralise Kantaris Mediaplayer'i (1) lähtekoodi muutmine nõnda, et Kantaris Mediaplayer'it oleks võimalik juhtida läbi töökäigus loodud veebirakenduse. Lõputöö realiseeritakse kasutades Microsoft arendusvahendeid. Kantaris Mediaplayer on meediamängija, mille abil on võimalik avada ja esitada laiemalt levinuimaid meediafaile.

3.2 Lahenduse nõuded.

Kuna on lahendusi, mis realiseerivad sama funktsionaalsuse, on käesolevale lahendusele püstitatud nõudede, mis eristavad teda olemasolevatest lahendustest ja muudavad unikaalseks. Nende nõuete täitmisega välditakse olemasolevate lahenduste miinuseid, mistõttu on antud lahenduse kasutamine lõppkasutajale mugavam ning lihtsam. Kolm tähtsaimat nõuet:

1. Puldi rakendus peab olema installivaba. Kasutaja ei pea tõmbama lahenduse kasutamiseks seadmesse uusi rakendusi/programme.

2. Rakendust peab saama kasutada kõikide platvormide peal. Kui kasutajal on olemas iPad tahvelarvuti ja Android nutitelefon, peab ta saama mõlemaga rakendust kasutada.

3. Lahenduse tööle saamine ja kasutamine peab olema lihtne ning kiire. Lahenduse kasutamiseks ei pea kasutaja intertnetist infot juurde otsima, kuidas lahendust seadistada või mida peaks tegema. Lahenduse seadistamine peab olema nii lihtne kui võimalik, ideaalis peaks kasutaja intuitiivselt ise aru saama, mida peab tegema lahenduse kasutama hakkamiseks tegema. Juhend peab olema väga lihtsasti leitav ja konkreetne. Rõhk on sisul ja informatsiooni kiirel ning efektiivsel edastamisel.

4 Ülevaade tööst

4.1 Töö kirjeldus

Kaugjuhitava meediamängija ideeks on anda kasutajale võimalus juhtida eemalt meediamängijat mõne teise seadmega. Põhimõttelt on idee sarnane teleri ja teleripuldile - puldi abil on võimalik meediamängija helitugevust muuta, sulgeda, teada saada, mis fail hetkel mängib ja panna mängitav fail pausi või taasjätkata.

Kuna arvuteid on võimalik ühendada teleriga on inimesed hakanud kasutama telerit selleks, et kuvada arvutis mängivat videofaili. Tehtud töö annab võimaluse mugavalt kaugjuhtida meediamängijat, mistõttu pole vaja iga meediamängija oleku muutuse jaoks minna füüsiliselt arvuti juurde.

4.2 Ülevaade valdkonnast

Kaugjuhitavaid meediamängijaid ja meediamängijatele mõeldud pistikprogramme, mis muudavad meediamängijad kaugjuhitavaks on tehtud varemgi. Muudel lahendustel on aga mitmeid miinuseid erinevatele, mida on üritatud käesoleva lahendusega vältida. Toon siinkohal konkreetse töö eelised erinevatest vaatenurkadest.

Eelised lõppkasutajale:

1. Puudub vajadus tõmmata puldiks olevasse seadmesse uus rakendus
2. Lihtne ja intuitiivne kasutaliides, selge õpetus
3. Kasutatavad seadmed ei pea omama sinihamba või infrapuna andmesideliideste olemasolu

Miinused lõppkasutajale:

1. Kasutatavad seadmed peavad olema pidevas interneti ühenduses
2. Kasutatavad seadmed peavad asuma samas kohalikus võrgus

Eelised arendajale:

1. Pult töötab mitmel platvormil korraga, mis teeb rakenduse haldamise lihtsamaks
2. Puudub vajadus programmeerida eraldi rakendusi erinevate platvormide jaoks
3. Puudub vajadus rakendust üles laadida erinevatesse rakenduste poodidesse (App store, Play Store jne)
4. Ei ole vaja maksta rakenduste poodidele lisa- ja/või vahendustasusid

Miinused arendajale:

1. Raske silma paista väljaspool ametlikku rakenduste turgu

4.3 Olemasolevad lahendused

Kantaris Mediaplayer'ile ei ole tehtud töö kirjutamise hetkel kaugjuhitavat liidest. Käesolevas peatükis kirjeldan olemasolevaid lahendusi, milles on realiseeritud analoogne funktsionaalsus teiste meediamängijate juhtimiseks. Valisin kirjeldamiseks kaks populaarset kaugjuhtimise rakendust.

4.3.1 VLC mediaplayer liidesed

VLC Mediaplayer on väga populaarne mitmeplatvormiline meediamängija, mis avab sisuliselt kõiki laiemalt tulnud meediafaili formaate. Kuna VLC mediaplayer'isse on sisseehitatud http tugi, on nüüdseks tehtud nii iOS'ile, Windows Phone'ile kui ka Androidile rakendused, millega on võimalik juhtida VLC mediaplayerit nutitelefoni rakenduste abil. Kuna puudub ligipääs iOS ja Windows Phone seadmetele kirjeldan järgnevalt Androidi põhist **Remote for VLC** (2) rakendust.

4.3.1.1 Kuidas töötab

Remote for VLC töötab samuti üle kohaliku võrgu, kusjuures ei pult ega meediamängija vaja funktsioneerimiseks ning omavaheliseks suhtluseks internetiühendust.

4.3.1.2 Vajalik tarkvara

Remote for VLC kasutamiseks on vajalik arvutisse alla laadida VLC Mediaplayer ning nutitelefoni rakendus Remote for VLC. Mõlema rakenduse alla tõmbamine ning paigaldamine on lihtne.

4.3.1.3 Algseadistamine keerukus

Kuigi Remote for VLC rakenduse poe kodulehel on link URL'ile, kus on selgitus, kuidas rakendust hakata kasutama, pole lingi taga asuv juhend tavakasutajale piisavalt selge. Reaalsuses tuleb internetist otsida, kuidas VLC mediaplayerit seadistada nii, et seda oleks võimalik rakenduste abil juhtida. Lisaks, pannes rakenduse nutitelefonis käima on reeglina vaja sisestada ka VLC meediamängijat jooksutava masina kohalik IP-aadress, mis tuleb kasutajal ise välja uurida. Kokkuvõttes on algseadistamise keerukus tavakasutaja jaoks liiga keeruline ja raske.

4.3.1.4 Tehnilised piirangud

Nii nutitelefon kui ka arvuti peavad asuma samas kohalikus võrgus.

4.3.1.5 Võimalused

Puldi abil on võimalik muuta mängiva faili helitugevust, panna pausi, stoppi, panna mängima taasesitusloendis eelnev/järgnev fail, sulgeda rakendus ning valida hetke, mis kohast soovitakse meediafaili esitada. Lisaks saab valida taasesitusloendis olevate failide vahel ja on veel palju muid võimalusi.

4.3.2 BSRemote

BSPlayer on samuti üks laialt levinumaid meediamängijaid. BSPlayeri kaugjuhtimiseks on Androidi rakenduste turul olemas rakendus **BSRemote** (3)**.**

4.3.2.1 Kuidas töötab

BSRemote töötab samuti üle kohaliku võrgu.

4.3.2.2 Vajalik tarkvara

BSRemote kasutamiseks on vajalikud rakendused on BSPlayer, BSRemote Server ning nutitelefoni rakendus BSPRemote. Kõikide vajalike komponentide allalaadimine ja paigaldamine on lihtne.

4.3.2.3 Algseadistamise keerukus

BSRemote lahenduse kodulehel on olemas BSRemote server ja BSRemote, lisaks on vaja tõmmata BSPlayer. Rakenduse kodulehel puudub juhend, kuidas rakendust hakata kasutama. Kasutamiseks tuleb avada installitud BSRemote Server, seejärel avada nutitelefonis BSRemote rakendus ning peale seda avada meediafail BSPlayer'iga. BSRemote rakendus on kohe kasutamiseks valmis. Algseadistamine on lihtne, kuid ilma selge ja konkreetse juhendita.

4.3.2.4 Tehnilised piirangud

Mõlemad seadmed peavad asuma samas kohalikus võrgus.

4.3.2.5 Võimalused

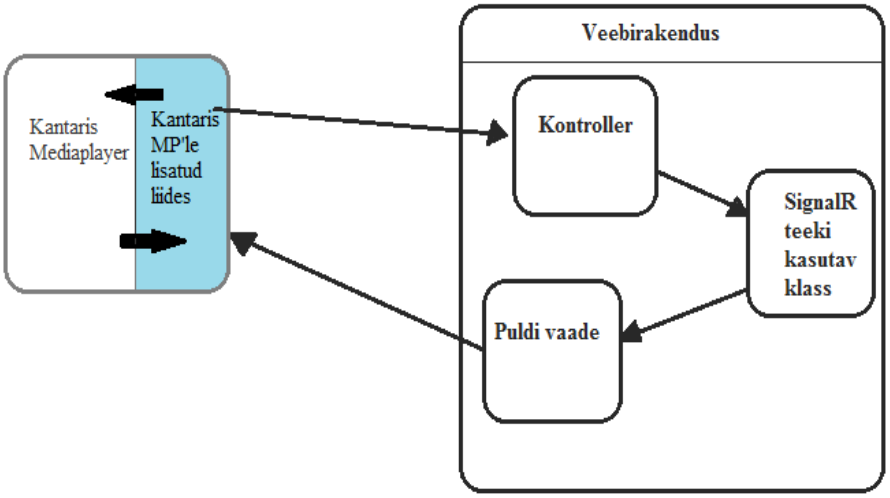
Põhifunktsionaalsus kattub Remote for VLC võimalustega.

4.4 Tehtud töö

Töö käigus tehti kaks üksteisest selgelt eristatavat osa:

1. Veebirakendus, mis realiseerib puldi funktsionaalsuse
2. Lisati Kantaris Mediaplayer'ile liides, mille abil suudab Kantaris Mediaplayer suhelda eelmises punktis loodud veebirakendusega

Joonis1 kirjeldab Kantaris Mediaplayer'i lisatud liidese (märgitud siniseks) ja veebirakenduse ning veebirakenduse komponentide vahelist suhtlust.



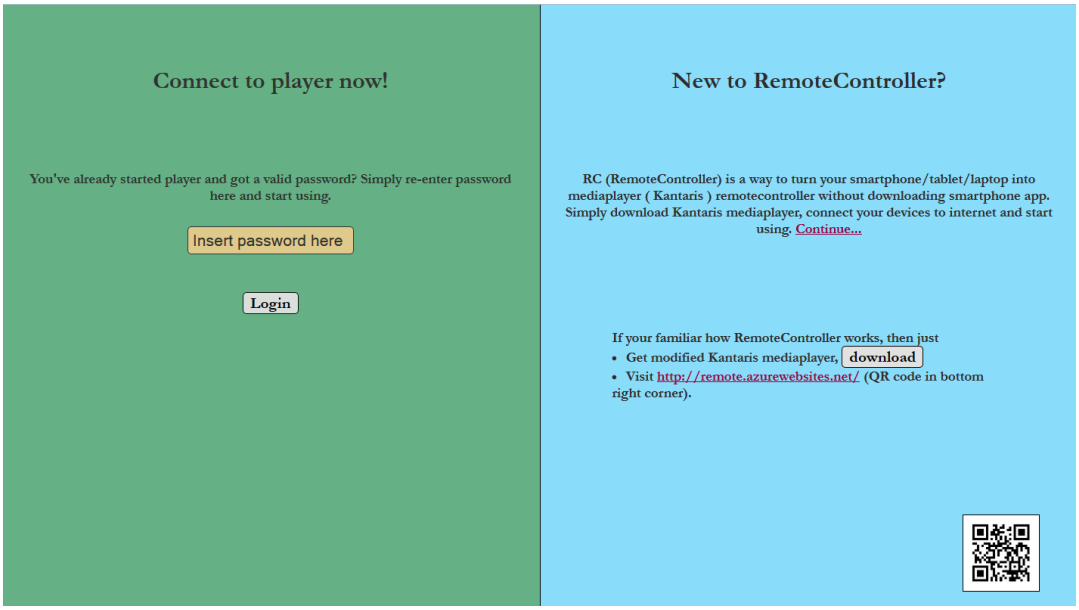
Joonis 1. Komponentide vahelised seosed

Kuna tehtud töö saab jagada selgelt piiritletavalt kaheks kirjeldan neid ka eraldi.

4.4.1 Veebirakendus

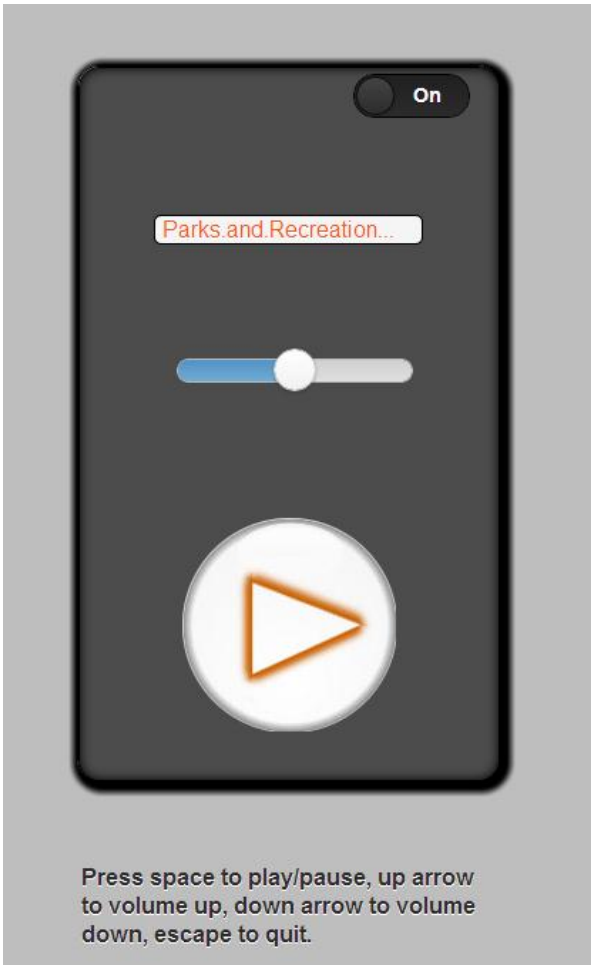
Veebirakenduse tööks on salvestada kasutajate paroole ja kohalikke IP-aadresse ning kuvada kasutajale pulti meenutavad vaated, kus saab muuta Kantaris Mediaplayer'i olekut. Veebirakendust luues oli läbivalt fookus kasutajakogemusel. Veebirakenduse loomise käigus realiseeriti järgnevad vaated ja funktsionaalsus:

1) Lauaarvuti peavaade.
   Vaade jaguneb kaheks – vasak pool, kuhu saavad kogenud kasutajad sisestada parooli ning hakata kohe rakendust kasutama ning parem pool, kus on rakendust lühidalt tutvustav info. Et parandada intuatiivsust ja kasutajamugavust on järgnevad elemendid interaktiivsed:
   i) Nii vasak kui parem pool muutub tooni võrra heledamaks kui kursor nende peale viia
   ii) Parooli sisestamise väljale on vaikimisi kuvatud juhendav tekst
   iii) Parooli sisestamise väljale vajutades muutub välja välimus
   iv) Viies kursori nupule muudab nupp tooni
   v) Vajutades nupul, muutub nupu välimus
   i) On lisatud QR-kood, mis suunab veebirakenduse pealehele. See võimaldab kasutajal ilma trükkimata veebirakenduse aadressi hakata kohe rakendust kasutama
   ii) Lingid on kergesti eristatavad tavalisest tekstist

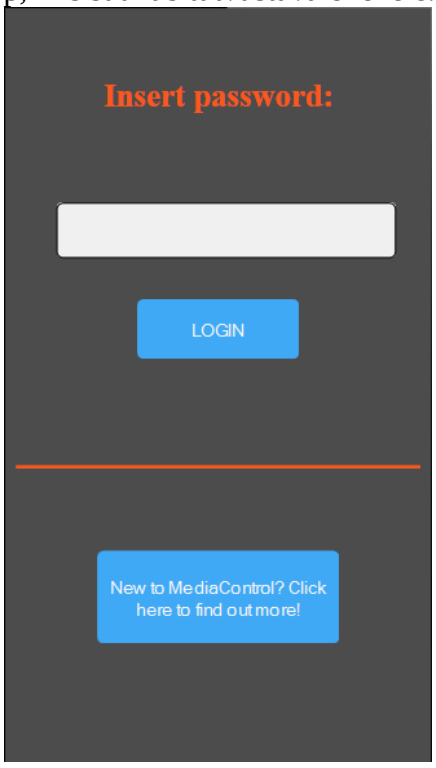


2) Lauaarvuti puldivaade
   Puldi vaatel kuvatakse järgnevad elemendid liidese kasutamiseks:
   a) nupp, mis kuvab klassikalist „mängi“/“paus“ nuppu meenutavat pilti vastavalt, kas meediafail hetkel on pandud pausi või mitte
   b) heli volüümi muutmise riba
   c) hetkel mängiva faili nimi
   d) liidese sulgemiseks mõeldud nupp

   Kasutajakogemuse parandamiseks on võimalik pulti juhtida ka arvuti klaviatuuri abil. Nupud, millega on võimalik juhtida on laialt tuntud nende funktsioonide poolest ka teistest suurtematest meediamängijatest. Lisaks kuvatakse nuppudega juhtimiseks vajalikku õpetust puldi all.



3) Mobiili peavaade
Mobiili peavaade sarnaselt lauaarvuti peavaatega jaguneb kaheks - ülemine pool, kuhu saavad kogenud kasutajad sisestada parooli ning hakata kohe rakendust kasutama ning alumine pool, kus on nupp, mis suunab tutvustavale lehele.



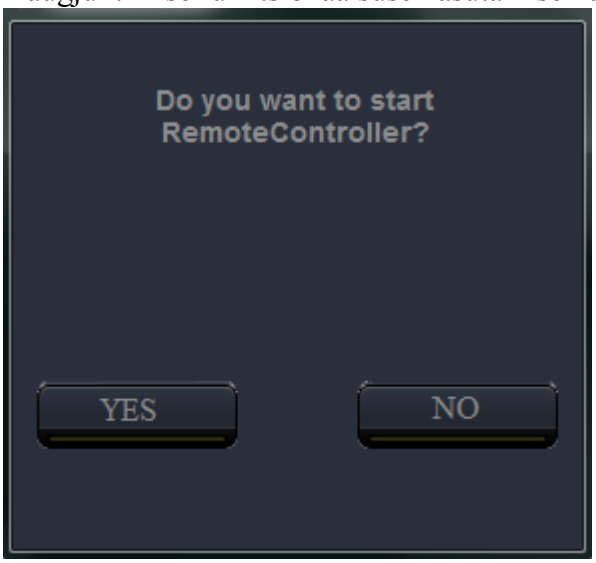
4) Mobiili puldivaates sarnaselt arvuti puldivaatega kuvatakse meediamängija sulgemise nupp, heli valjuduse muutmise nupp ja play/pause nupp.

5) Rakendust tutvustav vaade
Tutvustava vaate eesmärk on anda kasutajale võimalikult lihtsad ja selged juhised rakenduse kiireks tööle saamiseks ja kasutama hakkamiseks. Seal kuvatakse piltidega juhist, kuidas rakendust kasutada, kõik pildid on suurendatavad. Nagu ka muus rakenduses, on siin interkatiivsed nupud ja pildid ning esile tükkivad lingid.

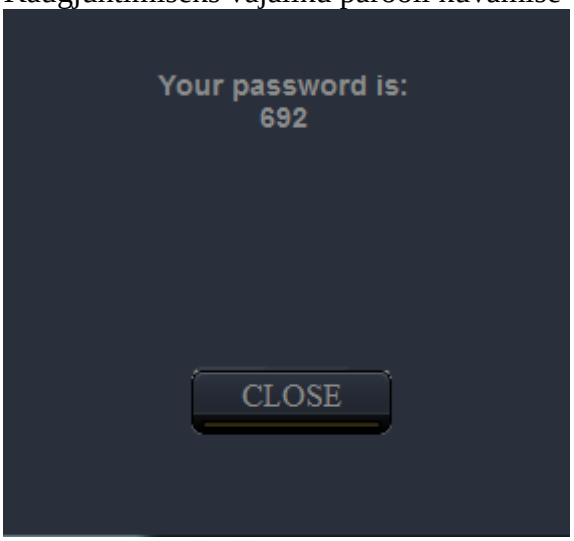
4.4.2 Kantaris Mediaplayer'i muudatused

Lõputöö teemaks on kaugjuhitava meediamängijale liidese tegemine, mistõttu ei programmeeritud töö käigus uut meediamängijat vaid kasutati olemasolevat vabavaralist avatud lähtekoodiga Kantaris Mediaplayer'it. Kantaris Mediaplayer sai valitud, sest tegu on ühe populaarseima vabavaralise, avatud lähtekoodiga meediamängijaga, mis on realiseeritud kasutades Microsofti tehnoloogiaid ning programmeeritud keeles C#. Tehnoloogia oli kriitilise tähtsusega, sest töö autor tunneb suurimat huvi just antud keele ja tehnoloogiate vastu.
Töö käigus muudeti Kantaris Mediaplayer'i lähtekoodi nõnda, et Kantaris Mediaplayer küsib avanedes, kas kasutaja soovib hakata kasutama kaugjuhtimisfunktsionaalsust. Kui kasutaja vastab jaatavalt, teeb Kantaris Mediaplayer'ile lisatud liides registreerimispäringu veebirakendusse ning registreerib ennast. Kaugjuhtimisfunktsionaalsuse kasutama hakkamiseks tehti ka kaks mainitud vormi:

1. Kaugjuhtimise funktsionaalsuse kasutamise küsimise vorm:



2. Kaugjuhtimiseks vajaliku parooli kuvamise vorm



4.5 Valitud tehnoloogia

Lahenduse realiseerimisel kasutati võimalikult palju Microsofti arendusvahendeid.

Veebirakenduse loomisel kasutati MVC disainimustrit, seda järgnevatel põhjustel:

1. Vaated, mudeli ja kontrolleri sai eraldada selgelt eristatavateks osadeks, mistõttu on hiljem lihtsam viia sisse muudatusi
2. Vaated, mudeli ja kontrolleri sai eraldada selgelt eristatavateks osadeks, mistõttu on kood hõlpsamalt hallatav ja arusaadavam
3. Erinevad vaated saavad kasutada sama koodi, seetõttu ei pea sama koodi mitmesse kohta kirjutama

Veebilehed on tehtud kasutades Razor Viewengine'it, Razor Viewnengine'it kasutati, sest:

1. Väga laialdaselt kasutatav, mistõttu on internetis palju materjale selle kohta
2. Väga lihtsalt kasutatav

3. Annab võimaluse .cshtml lehe sisse programmeerida, selle abil on võimalik erinevatele klientidele programmeerida erinevaid muutujaid HTML'i sisse

Veebirakenduses kasutati Microsoft .NET 4.5 platvormi, sest töö alustamise hetkel oli see uusim Microsofti .NET platvorm.

Loomisel kasutati Entity Frameworki, kasutamise põhjused:

1. Eemaldab vajaduse kirjutada käsitsi SQL skripte rakendusse
2. Eemaldab vajaduse kirjutada käsitsi SQL skripte andmebaasi loomiseks
3. Annab võimaluse C# keeles manipuleerida andmebaase ja relatsioone.

SignalR (4) teeki otsustati kasutada järgnevatel põhjustel:

1. Andis võimaluse muuta puldi vaate reaalaja vaateks, st. ilma lehe taaslaadimiseta muutub vaate välimus, kui selleks on vajadus
2. Väga laialt levinud teek reaalaja veebirakenduste loomiseks, mistõttu on internetis palju materjale antud teegi kohta
3. Väga lihtsalt kasutatav teek

Javascript koodi kirjutamise hõlbustamiseks ja puldi vaadetel olevate elementide loomiseks kasutati Javascript teeke jQuery 1.8.2 ja jQuery-ui.1.8.2 (5). Käesolevad teegid valiti järgnevatel põhjustel:

1. jQuery on väga laialt kasutatav Javascripti teek, mistõttu on internetis ka väga palju materjale antud teegi kasutamise kohta
2. jQuery kasutamine on väga lihtne
3. jQuery on tuntud töökindluse poolest, elemendid ja koodid töötavad veatult ka erinevates brauserites
4. Versioon 1.8.2 valiti, sest töö alustamise hetkel oli see uusim jQuery teek.

Kantaris Mediaplayer'i liides, mis kuulab pordil 8080 sissetulevaid sõnumeid on sisuliselt veebiteenus, mis kuulab signaale ning tagastab vastuse.Signaale kuulava veebiteenuse tegemiseks kasutati Windows Communication Foundation'i, sest:

1. Sisse ehitatud sõnumite kuulamise funktsionaalsus
2. Paljude võimalustega ja lihtsalt konfigureeritav
3. Laialt kasutatav, mistõttu on internetis WCF'i kohta palju materjale.

5 Veebirakenduse tehnilise kirjeldus

Veebirakendus on MVC-tüüpi rakendus, mis on tehtud kasutades .NET 4.5 platvormi. Lisaks kasutati SignalR teeki, mille abil on võimalik saata kontrollerist sõnumeid kliendi poolt laetud veebilehele kasutades Javascripti. Samuti kasutati nii arendamise lihtsustamiseks, kui ka kasutajakogemuse parandamiseks Javascripti jQuery teeki.Veebirakendust majutatakse Microsofti Azure keskkonnas aadressil http://Remote.Azurewebsites.net. Veebirakendus kasutab ka Entity Frameworki, mis on objekt-relatsiooniline vastendamise raamistik, mille abil saab programmi koodiga manipuleerida relatsioone ja andmebaase. Entity Frameworki kasutades puudub vajadus kirjutada käsitsi SQL skripte. Veebirakendus on lihtne failide kogum, mis koosneb:

1. Kontrollerist, mis on lihtne C# keeles programmeeritud klass, mis kuulab veebirakendusele saadetud päringuid ning käitub vastavalt päringule
2. .cshtml failidest, mis genereerivad kasutajale HTML vaate. Cshtml failid jagunevad üldiselt kaheks – ühed mis genereerivad HTML'i mobiilsetele seadmetele ja teised, mis genereerivad HTML'i mitte-mobiilsetele seadmetele.
3. Ühest mudelist, milles hoitakse veebirakenduse töötamiseks vajalikke andmeid (kasutaja unikaalset parooli, registreerimise kuupäeva ja Kantaris Mediaplayer'it jooksutava masina kohalik IP-aadress)
4. Erinevad konfiguratsiooni ja seadistamise failid. Nende abil konfigureeritakse rakendus nii, et mobiilsed seadmed suunatakse mobiilsetele seadmetele mõeldud vaadetele ning initsialiseeritakse vajalikud muutujad rakenduse tööks

5.1 Kontroller.

Kontroller kontrollib rakendust. Kui tehakse päring veebirakendusse, siis kontroller kuulab päringut ja vastavalt päringule ning küsitud infole annab vastuse. Veebirakendus on konfigureeritud nii, et meetodi poole saab pöördumine toimub järgmise URL skeemi järgi:

{veebirakenduse URL } / { kontrolleri nimi } / { meetodi nimi }

Kui soovitakse meetodile kaasa anda argumente, on skeem järgnev:

{veebirakenduse URL } / { kontrolleri nimi } / { meetodi nimi } ?
{param1}={väärtus1}&{param2}={väärtus2}

Antud veebirakenduses on ainult üks kontroller ning seetõttu määrati vaikimisi kontrolleriks RemoteController.cs ning kontrolleri poole pöördudes pole vaja URL'i skeemi lisada kontrolleri nime.

Kirjeldan järgnevalt kontrolleri meetodeid:

a) Veebirakenduse ja kontrolleri poole pöördudes on vaikimisi vastuseks pealeht.

```
public ActionResult Index()
{
    return View();
}
```

b) Meetod RegisterNewPassword on kontrolleri kõige tähtsam meetod, millega registreeritakse uus klient. Argumendina antakse kaasa aadress, sellele aadressile genereeritakse unikaalne

parool ning see salvestatakse andmebaasi. Tagastatakse parool avatekstina, mis lõpuks kuvatakse Kantaris Mediaplayeri kasutajaliidesel.

```
        public string RegisterNewPassword(string aadress)
        {
            try
            {
                string password = "";
                while (true)
                {
                    int passwN = new Random().Next(10000);
                    password = passwN.ToString();
                    if (!db.Aadresses.Any(k => k.password == password))
                        break;
                }

                Aadress aadressToAdd = new Aadress()
                {
                    aadress = aadress,
                    password = password
                };
                db.Aadresses.Add(aadressToAdd);
                db.SaveChanges();
                return "aadress registered@" + password;
            }
            catch (Exception e)
            {
                return "aadress not registered "+e.InnerException.ToString()+"
"+e.ToString();
            }
        }
```

c) Järgnev meetod kustutakse välja meediamängija liidese poolt kui meediamängija olekut muudetakse. Argumendina antakse kaasa parool ning oleku muutus.

```
        public string StateChanged(string password, string state)
        {

            MessageReciever.StateChanged(password,state);
            return "message sent";
        }
```

d) Järgnev meetod suunab kasutaja tutvustavale vaatel

```
        public ActionResult About()
        {
            return View("About");
        }
```

e) Järgnev meetod kutsutakse välja kui pealehele sisestatakse parool ja vajutatakse „Connect „ nuppu. Argumendina nõuab meetod parooli, samas kui parooli kaasa ei anta siis rakendus kokku ei jookse. Meetod kontrollib, kas andmebaasis on olemas vastava parooliga aadress, kui on, siis suunatakse puldi vaatesse, kusjuures vaatesse peidetakse varjatud kujul ka sisestatud parool ja aadress. Kui vastava parooliga kirjet andmebaasist ei leitud, siis suunatakse tagasi pealehele ning kuvatakse sõbralikus stiilis veateade.

```
        public ActionResult ConnectToPC(string password)
         {

            Aadress pcAadress = db.Aadresses.Where(t => t.password ==
password).FirstOrDefault();
            if (pcAadress != null)
            {
                TempData["password"] = pcAadress.password;
                TempData["aadress"] = pcAadress.aadress+":8090/hosting/";
            }
            else
            {
                TempData["error"] = "sorri vastavat kasutajat ei leitud :(";
                return View("Index");
            }

            return View("Remoter");

         }
```

5.2 Puldi vaadete poolt kasutatav Javascript fail

Puldi vaate loogika realiseerib Javascripti fail Remoter.js. Seda faili kasutavad nii mobiili kui tava puldi vaade.

1) Javascript koodi töötamiseks vajalike muutujate deklaratsioon

```
var isPlaying = true;
var clientAadress = getClientAadress();
var sendToPlayer = true;
var messageReciever = $.connection.messageReciever;
var volumeLevel = 50;
var name;
var reDirectEndpoint = 'http://rmcr.azurewebsites.net';
```

2) Meetod registreerib puldi tema parooli alusel SignalR teeki. Peale seda on SignalR teegi abil on võimalik kontrollerist saata signaale veebilehele.

```
$.connection.hub.start().done(function () {

        messageReciever.server.register(getPassword());
    });
```

3) Kasutades järgnevat jQuery funktsiooni tehakse ajax päring meediamängijasse, kust küsitakse meediamängija seisundit ning kuvatakse saadud info puldi vaatele. Meetod kutsutakse välja

puldi avanemisel.

```
    $.ajax({
        type: "GET",
        url: NameAadress,
        dataType: "jsonp",
        contentType: "application/json;charset=utf-8",
        success: function(data) {
            var bufferData = data.toString();
            var jsonInfo = $.parseJSON(bufferData);
            name = GetNameToShow(jsonInfo.name);
            isPlaying = jsonInfo.playing;
            volumeLevel = jsonInfo.volumeLvl;
            if (!isPlaying) {
                $('#playBtn').attr('src', '/Content/images/PlayBtn.png');

            } else {
                $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
            }

            $('.heading').html(name);
            $('#volumeSlider').val(volumeLevel);
            $('#volumeSlider').slider("refresh");
        },
        error: function(xhr, ajaxOptions, thrownError) {
            ConnectionLost();
```

4) Järgnev meetod kutsutakse välja, kui helitugevuse liuguri väärtus muutub. Kõigepealt kontrollitakse, kas see info tuleks saata meediamängijasse, et vältida signaalide kasutut edasi-tagasi saatmist. Kui signaal on vaja meediamängijasse saata, siis tehakse ajax päring. Meetodit kasutatakse, et helitugevuse muutmisel puldist jõuaks info ka meediamängijasse.

```
    $("#sliderDiv").change(function () {
        if (sendToPlayer) {
            var aadress = "http://" + getClientAadress() + "Volume/" +
$('#volumeSlider').val();
            $.ajax({
                type: "GET",
                url: aadress,
                timeOut: 2000,
                success: function (data) {
                },
                error: function (xhr, ajaxOptions, thrownError) {
                }
            });
        }
    });
```

5) Järgnevaid meetodid kutsutakse välja SignalR teegi poolt. Neid kasutatakse, et kuvada oleku muutusi, mis on läbi viidud kasutajaliidesel.

a) Muudab helitugevuse liuguri väärtust:

```
        messageReciever.client.volumeChanged = function (changed) {
            sendToPlayer = false;
            $('#volumeSlider').val(changed);
            $('#volumeSlider').slider("refresh");
            sendToPlayer = true;
        };
```

b) Kuulab kontrollerist signaali, mis teatab, et mängitav fail taaskäivitati ja muudab vastavalt puldi välimust.

```
messageReciever.client.play = function () {
    $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
    isPlaying = true;
};
```

c) Kuulab kontrollerist signaali, mis teatab, et mängitav fail peatati ja muudab vastavalt puldi välimust.

```
messageReciever.client.pause = function () {
    isPlaying = false;
    $('#playBtn').attr('src', '/Content/images/PlayBtn.png');
};
```

d) Kuulab kontrollerist signaali, mis teatab, et meediamängija suleti ja suunab tagasi pealehele.

```
messageReciever.client.close = function () {
    $('#onOffFlip').val('off');
    PlayerClosed();
};
```

e) Kuulab kontrollerist signaali, mis teatab, et mängitav fail on muutunud ja kuvab uue faili nime vaatele.

```
messageReciever.client.sendFilename = function (name) {
    $('.heading').html(GetNameToShow(name));
};
```

2) Lühendab failinime, kui see ei mahu ära mängitava faili nime kuvavale väljale.

```
function GetNameToShow(name) {
    if (name.length > 14)
        return name.substring(0, 20) + '...';
    else {
        return name;
    }
}
```

3) Sulgeb mängiva meedimängija ning suunab brauseri veebirakenduse pealehele.

```
$(".onoff").change(function () {
        var aadress = "http://" + getClientAadress() + "Close";
        $.ajax({
            type: "GET",
            url: aadress,
            timeOut: 5000,
        });
        PlayerClosed();
});
```

4) Meetod kutsutakse välja kui vajutatakse mängi/peata nuppu. Meetod kontrollib kõigepealt, kas hetkel meediamängija mängib ning vastavalt sellele saadab vajaliku signaali. Peale signaali

saatmise õnnestumist muudetakse kohalik mängimist indikeeriva muutuja väärtus vastupidiseks.

```
    $('#playBtn').click(function () {
        var aadress;
        if (isPlaying == true) {
            aadress = "http://" + getClientAadress() + "Pause";
            $('#playBtn').attr('src', '/Content/images/PlayBtn.png');

        } else {
            var aadress = "http://" + getClientAadress() + "Play";
            $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
        }
        $.ajax({
            type: "GET",
            url: aadress,
            timeOut: 5000,
            dataType: "jsonp",
            contentType: "application/json;charset=utf-8",
            success: function (data) {
                if (isPlaying)
                    isPlaying = false;
                else {
                    isPlaying = true;
                }
            },
            error: function (xhr, ajaxOptions, thrownError) {
                ConnectionLost();
            }
        });
    })
```

5) Meetodit kasutatakse, et teavitada kasutajat ühenduse katkemisest ning kasutaja ümber suunata pealehele

```
function ConnectionLost() {
        $.msg({
            bgPath: '/content/images/',
            content: "Connection has been lost...you're being redirected to
mainpage"
        });

        messageReciever.server.close(getPassword());
        setTimeout(function () {
            window.location.href = "http://remote.azurewebsites.net";
        }, 2000);
    }
```

6) Meetodit kasutatakse, et kasutajale teada anda, et meediamängija sulgeti Kantaris meediamängija kasutajaliideselt. Meetod kutsutakse välja kontrollerist.

```
function PlayerClosed() {
        $.msg({
            bgPath: '/content/images/',
            content: "Player has been closed...you're being redirected to
mainpage"
        });
        messageReciever.server.close(getPassword());
        setTimeout(function () {
            window.location.href = "http://remote.azurewebsites.net";
        }, 2000);
    }
```

7) Tagastab kliendi IP aadressi, mis on salvestatud peidetud vormiväljale

```
function getClientAadress() {
    var retval = $('#aadress').val();
    return retval;
}
```

8) Tagastab kliendi parooli, mis on salvestatud peidetud vormiväljale:

```
function getPassword() {
    var retval = $('#password').val();
    return retval;
}
```

5.3 Puldi vaate ja veebirakenduse vahelise suhtluse realisatsioon

Kui kasutaja sisestab veebirakenduse pealehel parooli jas avaldab soovi jätkamiseks, suunatakse ta edasi puldi vaatesse. Selle vaate tõmbab kliendi veebilehitseja kliendi masinasse. Selleks, et oleks võimalik ilma lehe taaslaadimiseta kuvada väljaspol vaadet toimunud oleku muutusi kasutakse SignalR teeki. SignalR baasklassist on võimalik saata sõnumeid vaatele. Järgnevalt kirjeldan SingalR baasklassi implementeeriva klassi meetodeid.

1. Meetodit eesmärgiks on saata vaatele oleku muutust kirjeldav signaal ning saata signaal õigele kliendile. Meetod kutsutakse välja kontrolleris

```
public static void StateChanged(string password, string action)
{

    var context =
GlobalHost.ConnectionManager.GetHubContext<MessageReciever>();
    if (action.StartsWith("Play"))
        context.Clients.Group(password).Play();
    if (action.StartsWith("Pause"))
        context.Clients.Group(password).Pause();
    if (action.StartsWith("VolumeChanged"))
    {
        string volumeLevel = action.Split(':')[1];
        context.Clients.Group(password).volumeChanged(volumeLevel);

    }
    if (action.StartsWith("Close"))
        context.Clients.Group(password).Close();
    if (action.StartsWith("Filename"))
    {
        string filename = action.Split(':')[1];
        context.Clients.Group(password).SendFilename(filename);
    }
}
```

2. Meetodit kasutatakse, et registreerida uus klient ja salvestada talle omane unikaalne ühenduse võti, mille abil on võimalik hiljem kliendiga ühendust võtta. Meetod kutsutakse välja Javascripti poolt, kui klient suunatakse puldi vaatele.

```
public void Register(string password)
{
    Groups.Add(Context.ConnectionId, password);
    return;
}
```

5.4 Muu

Muudest veebirakenduse failidest tasuks esile tuua rakenduse tööd seadistavat klassi Global.Asax.cs. Antud klassis on meetod, mis käivitatakse, kui veebirakendus alustab tööd. Järgneva meetodiga seadistatakse rakendus nõnda, et mobiilsete seadmete brauserid suunatakse mobiili vaadetele.

```
            DisplayModeProvider.Instance.Modes.Insert(0,
             new DefaultDisplayMode("Mobile")
             {
                 ContextCondition = (ctx => (
                     (ctx.GetOverriddenUserAgent() != null) &&
ctx.Request.Browser.IsMobileDevice
             ))
             });
```

6 Kantaris Mediaplayer'ile lisatud liidese tehnilise kirjeldus

Kantaris Mediaplayer on vabavaraline meediamängija, mis on tehtud kasutades Microsoft .NET raamistikku. Töös kasutatakse Kantaris Mediaplayer'it meediamängijana, millele lisat vajalik liidestus, et oleks võimalik meedijamängijat kaugjuhtida. Kantaris Mediaplayer'i lähtekoodile lisati juurde klass, mille tööks on:

1) Teavitab veebirakendust Kantaris Mediaplayer'i oleku muutustest
2) Kuulab veebirakenduse poolt saadetud signaale Kantaris Mediaplayer'i soovitavatest olekutest
3) Rakenduse tööks vajalike portide avamine/sulgemine

Liidestus on realiseeritud ühe Singleton klassina, et muudatused oleksid võimalikud lihtsalt hallatavad ning, et vältida arhitektuurilist segadust. Veebirakenduse poolt saadetud sõnumite kuulamiseks kasutatakse WCF teenust. Peale veebirakendusest tulnud käskluse täitmist tagastatakse JSON sõnumina õnnestumist indikeeriv teade.

1) Klassi Singleton disaini-mustri realisatsioon:

```
private static MovieService instance;
private MovieService() { }
public static MovieService Instance
{
    get
    {
        if (instance == null)
        {
            instance = new MovieService();
        }
        return instance;
    }
}
```

2) Liidestuse tööks vajalike muutujate deklaratsioon:

```
Kantaris mainForm;
private string password;

JavaScriptSerializer jsS = new JavaScriptSerializer();
private string myLocalIp;
protected string host = "http://remote.azurewebsites.net/remote/";
```

3) Kuulab veebirakenduselt tulnud Pause käsklust,peatab mängiva meediafaili ja tagastab vastuseks „paused".

```
[WebGet(UriTemplate = "/Pause",
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string Pause()
{

    mainForm.pauseForService();
    return jsS.Serialize("paused");
}
```

4) Kuulab veebirakenduselt tulnud Play käsklust, jätkab peatatud meediafaili mängimist ja tagastab vastuseks „playing“.

```
[WebGet(UriTemplate = "/Play",
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string Play()
{

    mainForm.playForService();
    return jsS.Serialize("playing");
}
```

5) Kuulab veebirakenduselt tulnud Volume käsklust, muudab vastavalt soovitule mängitava faili helitugevust ning tagastab vastuseks „Volume changed“

```
[WebGet(UriTemplate = "/Volume/{volumeLevel}"),
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string Volume(string volumeLevel)
{
    mainForm.volumeChangeForService(volumeLevel);
    return jsS.Serialize("Volume changed");
}
```

6) Kuulab veebirakenduselt tulnud „Close“ käsklust, sulgeb avatud pordid, sulgeb mängija ning tagastab vastuseks „closed“.

```
[WebGet(UriTemplate = "/Close"),
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string Close()
{
    RemoveFirewallException();
    mainForm.Close();
    return jsS.Serialize("closed");
}
```

7) Kuulab veebirakenduselt tulnud käsklust „Data“ ning tagastab Kantaris Mediaplayer olekut kirjeldava vastuse.

```
[WebGet(UriTemplate = "/Data",
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string GetInitalData()
{
    string name="no movie selected";
    int volumeLvl = 50;
    bool playing = false;
    try
    {
        playing = mainForm.isPlaying();
        volumeLvl = mainForm.getVolumeLevel();
        name = mainForm.getPlayList();
    }
    catch (Exception e)
    {

    }
    var response = new
    {
        name,
        volumeLvl,
        playing

    };
    string jsonResponse = jsS.Serialize(response);
    return jsonResponse;
}
```

8) Meetod kutsutakse välja Kantaris Mediaplayer’i poolt kui mängija käivitatakse ning selles väärtustatakse rakenduse tööks vajalikud muutujad.

```
public void Initialize(Kantaris mainForm, string password)
{
    PasswordForm passwordForm = new PasswordForm();
    passwordForm.Show();
    this.mainForm = mainForm;
}
```

9) Meetod kustutakse välja Kantaris Mediaplayer’i poolt, kui Kantaris Mediaplayer’i olekut muudetakse kasutajaliidesel. Meetod teeb HTTP päringu veebirakendusse andes parameetriteks kaasa parooli ja oleku muutust kirjeldava teksti.

```
public void ActionChanged(string action)
{
    if (password != null)
    {
        HttpWebRequest httpWReq =
(HttpWebRequest)WebRequest.Create(String.Format(host +
"StateChanged?password={0}&state={1}", password, action));
        Task.Factory.StartNew(() =>
        {
            httpWReq.GetResponse();
        });
    }
}
```

10) Meetodit kasutatakse, et registreerida veebirakendusse konkreetse masina kohaliku võrgu IP-aadress. Veebirakendus tagastab õnnestumise korral registreerimisega seotud info, mis sisaldab parooli, mida hakatakse hoidma kohalikus muutujas. Vea korral tagastab veebirakendus vastava

veateate. Meetod tagastab veebirakendusest tulnud vastuse.

```
        public string RegisterNewPassword(string aadress)
        {
            HttpWebRequest httpWReq =
(HttpWebRequest)HttpWebRequest.Create(String.Format(host +
"RegisterNewPassword?aadress={0}",aadress));
            WebResponse response = httpWReq.GetResponse();
            Stream stream = response.GetResponseStream();
            string result = String.Empty;
            using (StreamReader sr = new StreamReader(stream))
            {
                result = sr.ReadToEnd();
            }
            AddFirewallException();
            password = result.Split('@')[1];
            return result;
        }
```

11) Meetod kutsutakse välja Kantaris Mediaplayer'i poolt, kui mängitav fail muutub. Meetod teeb veebirakendusse päringu, et anda teada faili nime muutusest.

```
        public void SendFileName()
        {
            HttpWebRequest httpWReq =
(HttpWebRequest)WebRequest.Create(String.Format(host +
"Remote/StateChanged?password=test&Filname={1}", password, mainForm.getPlayList()));
            httpWReq.GetResponse();
        }
```

12) Meetod salvestab liidese klassi muutujasse parooli.

```
        public void SavePassword(string password)
        {
            this.password = password;
        }
```

13) Meetod avab pordi 8090, mis on vajalik liidese ja veebirakenduse omavaheliseks suhtluseks.

```
        public bool AddFirewallException()
        {
            INetFwMgr icfMgr = null;
            try
            {
                Type TicfMgr = Type.GetTypeFromProgID("HNetCfg.FwMgr");
                icfMgr = (INetFwMgr)Activator.CreateInstance(TicfMgr);
            }
            catch (Exception ex)
            {
                return false;
            }

            try
            {
                INetFwProfile profile;
                INetFwOpenPort portClass;
                Type TportClass = Type.GetTypeFromProgID("HNetCfg.FWOpenPort");
                portClass = (INetFwOpenPort)Activator.CreateInstance(TportClass);
                profile = icfMgr.LocalPolicy.CurrentProfile;
                portClass.Scope = NetFwTypeLib.NET_FW_SCOPE_.NET_FW_SCOPE_ALL;
                portClass.Enabled = true;
                portClass.Protocol =
NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP;
                portClass.Name = "DescriptionOfRule";
                portClass.Port = 8090;
                profile.GloballyOpenPorts.Add(portClass);
                return true;
            }
            catch (Exception ex)
            {
                return false;
            }
        }
```

14) Meetod sulgeb punktis 13 avatud 8090 pordi.

```
        public void RemoveFirewallException()
        {
            INetFwMgr icfMgr = null;
            try
            {
                Type TicfMgr = Type.GetTypeFromProgID("HNetCfg.FwMgr");
                icfMgr = (INetFwMgr)Activator.CreateInstance(TicfMgr);
            }
            catch (Exception ex)
            {
                return;
            }

            try
            {
                INetFwProfile profile;
                INetFwOpenPort portClass;
                Type TportClass = Type.GetTypeFromProgID("HNetCfg.FWOpenPort");
                portClass = (INetFwOpenPort)Activator.CreateInstance(TportClass);

                // Get the current profile
                profile = icfMgr.LocalPolicy.CurrentProfile;

                // Set the port properties
                portClass.Scope = NetFwTypeLib.NET_FW_SCOPE_.NET_FW_SCOPE_ALL;
                portClass.Enabled = true;
                portClass.Protocol =
NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP;
                portClass.Name = "DescriptionOfRule";
                portClass.Port = 8090;

                // Add the port to the ICF Permissions List
                profile.GloballyOpenPorts.Remove(8090,
NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP);
            }
            catch (Exception ex)
            {
            }
        }
```

7 Rakenduse testimine

Lahendus ja selle osasid testiti käsitsi. Läbi testiti järgnevad osad:

a) Sisestades veebirakenduse avalehel vale parooli, kuvatakse sõbralik veateade nii mobiili- kui ka tavavaates
b) Sisestades veebirakenduse avalehel õige parooli suunatakse edasi puldi vaatele
c) Minnes puldi vaatele hetkel, kui Kantaris Mediaplayer'is ei mängi ükski faili, kuvatakse hetkel mängiva faili väljale vastav teade
d) Minnes puldi vaatele hetkel, kui Kantaris Mediaplayer'is mängib fail, kuvatakse puldi vaates hetkel mängiva faili nimi ning Kantaris Mediaplayer'is valitud helitugevus
e) Muutes helitugevust puldi vaatel, muudetakse helitugevust ka Kantaris Mediaplayer'is ning vastavalt kuvatakse muutust ka Kantaris Mediaplayer'i kasutajaliidesel
f) Vajutades puldi vaatel mängi/pausi nuppu, muutub puldi vaatel ka mängi/pausi nupu pilt
g) Vajutades puldi vaatel mängi/pausi nuppu, Kantaris Mediaplayer vastavalt, kas jätkab/paneb pausi hetkel mängiva faili ning muutust kuvatakse ka Kantaris Mediaplayer'i kasutajaliidesel
h) Muutes helitugevust Kantaris Mediaplayer'i kasutajaliidesel, muutub helitugevust indikeeriva nupu väärtus vastavalt
i) Vajutades mängi/pausi nuppu Kantaris Mediaplayer'i liidesel, kuvatakse muutust ka puldi vaatel
j) Kui muutub mängitav fail Kantaris Mediaplayer'is, muutub ka hetkel mängiva faili väljal olev faili nimi
k) Tava puldi vaates kasutades klahve puldi juhtimiseks, muutub puldi ja Kantaris Mediaplayer'i olek vastavalt vajutatud klahvi funktsioonile
l) Vajutades puldi vaatel Off nuppu, sulgetakse Kantaris Mediaplayer ja kasutaja suunatakse puldi vaatest edasi rakenduse pealehele

8 Kokkuvõte

8.1 Kokkuvõte.

Lõputöös tehtud lahendus töötab nii nagu ülesande püstitutes määratud. Veebirakendus koos alla laetava muudetud Kantaris Mediaplayer'iga on saadaval aadressil http://remote.azurewebsites.net. Lahendus sai loodud kasutades Microsofti .NET platvormi, lisaks kasutati vabavaralist Javasript jQuery teeki ning SignalR teeki. Veebirakendus ja Kantaris Mediaplayer'i muudatused tehti kasutades Microsoft Visual Studio IDE't. Puldi vaates on võimalik näha hetkel mängiva faili nime, selle faili olekut mõjutada ning Kantaris Mediaplayer'it sulgeda. Kui muudetakse mängiva faili olekut Kantaris Mediaplayer'i liidesel, siis kajastuvad muudatused ka puldi vaates. Kasutajamugavuse kohapealt jäi suurimaks puuduseks asjaolu, et rakenduse avamisel peab rakendust avama administraatori õigustega.

8.2 Edasiarendamise võimalused

Arenguruumi antud lahendusel on palju, loetlen mõned tähtsamad:

1. Muuta lahendust nii, et funktsioneerimiseks ei peaks mõlemad seadmed olema samas kohalikus võrgus
2. Androidi, iOSi ja Windows Phone rakendused vastavalt platvormide rakenduste poodidesse. Sisuliselt oleks rakendus praeguse veebirakenduse *wrapper*
3. Anda kasutajale võimalus muuta porti, mida rakendus kasutab suhtluseks
4. Nii veebirakendus kui ka liides peaks kontrollima tehtud HTTP päringute vastuseid põhjalikumalt ja vastavalt vastustele käituma
5. Kinnitades meediamängijas soovi kasutada puldi funktsionaalsust, ei tohiks liides graafiliselt kokku joosta ajaks, mil tehakse veebirakendusse päringut

9 Viited

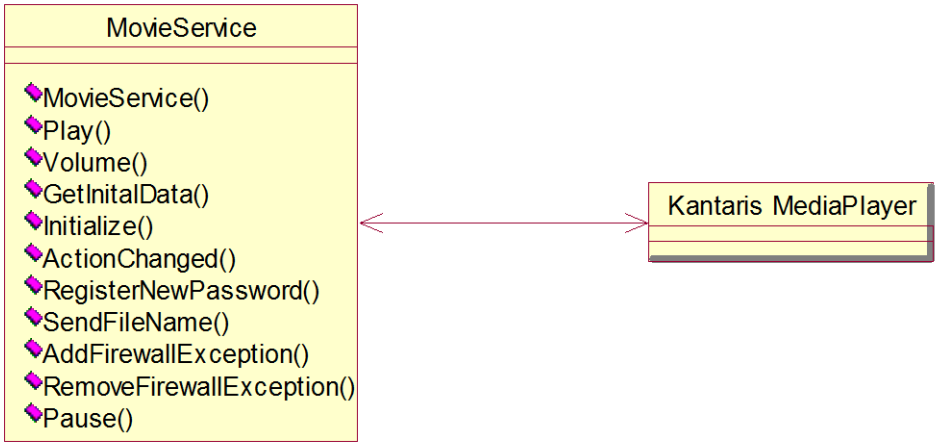
1. Kantaris Mediaplayer. [Võrgumaterjal] [Tsiteeritud: 7. 6 2014. a.] http://kantaris.org/.
2. Remote for VLC. [Võrgumaterjal] [Tsiteeritud: 7. 6 2014. a.] https://play.google.com/store/apps/details?id=org.peterbaldwin.client.android.vlcremote.
3. BSRemote. [Võrgumaterjal] [Tsiteeritud: 7. 6 2014. a.] https://play.google.com/store/apps/details?id=ro.hupca.bsremote.
4. SignalR. [Võrgumaterjal] [Tsiteeritud: 7. 6 2014. a.] http://www.asp.net/signalr.
5. jQuery. [Võrgumaterjal] [Tsiteeritud: 7. 6 2014. a.] http://jquery.com/.

(3)

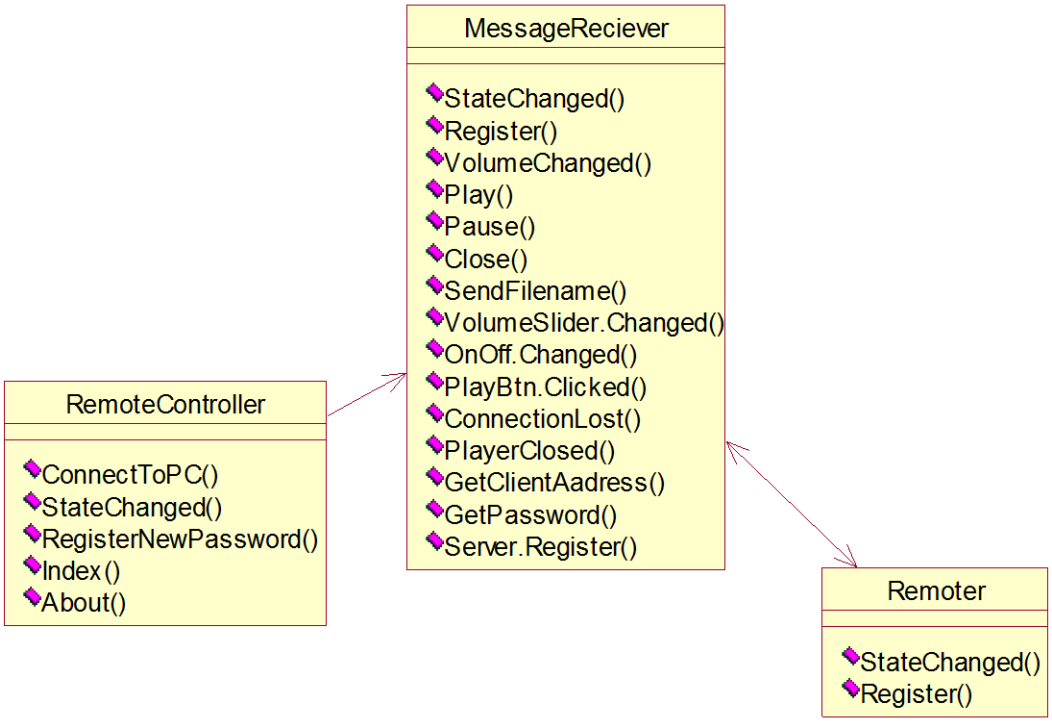
10 Lisad

10.1 Joonised

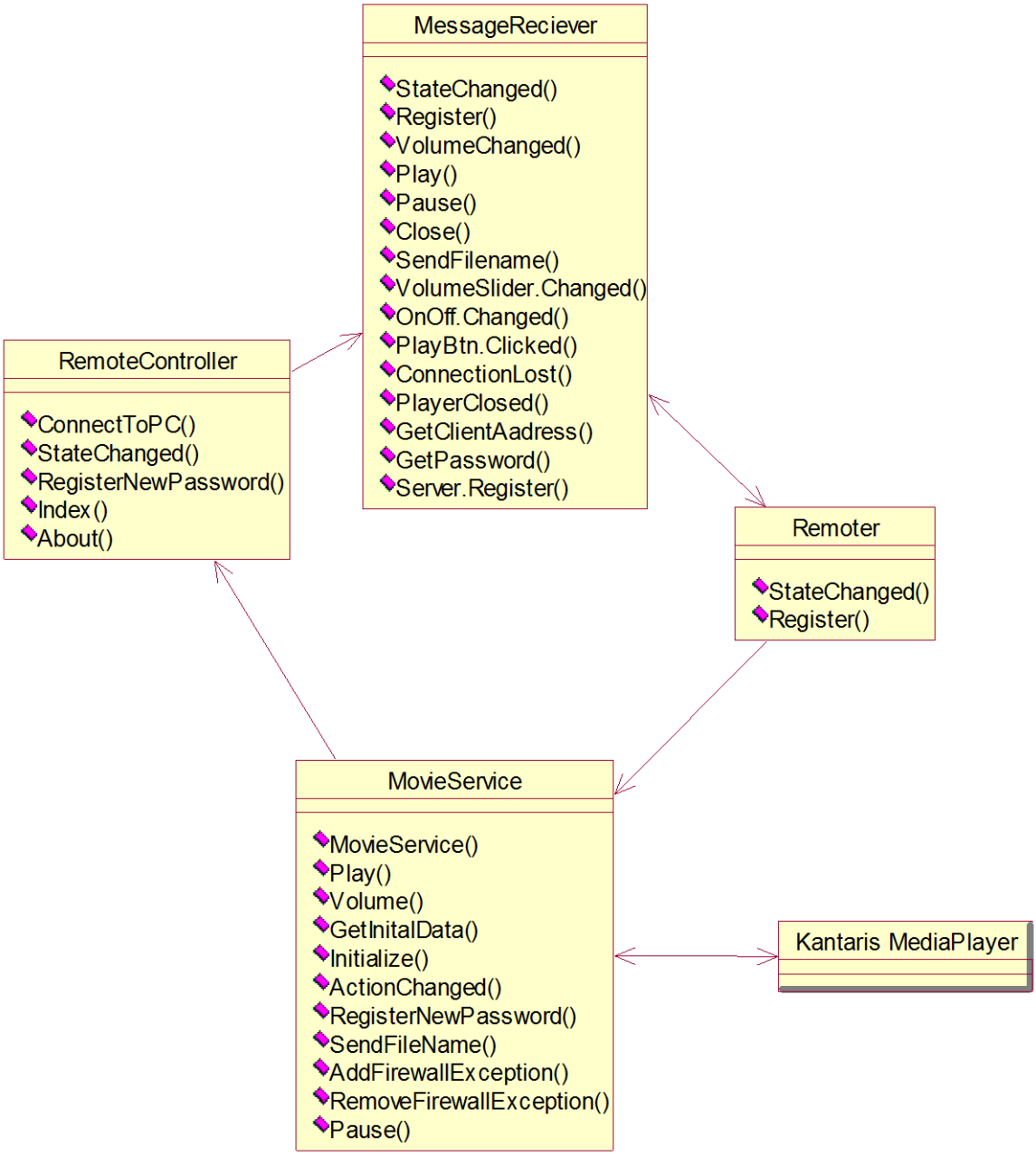
1) Kantaris Mediaplayer'ile lisatud liidestuse klassidiagramm



2) Veebirakenduse ja Javascript faili seoste klassidiagramm



3) Veebirakenduse ja Kantaris Mediaplayer'ile lisatud liidese seoste diagramm



10.2 Kriitiliste failide täielik lähtekood

10.2.1 RemoteController

```
using System;
using System.CodeDom;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading;
using System.Web;
using System.Web.Mvc;
using MediaRemote.Models;
using MediaRemote.Hubs;

namespace MediaRemote.Controllers
{
 public class RemoteController : Controller
 {

  AadressDBContext db = new AadressDBContext();

  public ActionResult Index()
  {
   return View();
  }

  public string RegisterNewPassword(string aadress)
  {
   try
   {
    string password = "";
    while (true)
    {
     int passwN = new Random().Next(10000);
     password = passwN.ToString();
     if (!db.Aadresses.Any(k => k.password == password))
      break;
    }

    Aadress aadressToAdd = new Aadress()
    {
     aadress = aadress,
     password = password
    };
    db.Aadresses.Add(aadressToAdd);
    db.SaveChanges();
    return "aadress registered@" + password;
   }
   catch (Exception e)
   {
    return "aadress not registered "+e.InnerException.ToString()+" "+e.ToString();
   }
  }
  public ActionResult Remoter()
  {
   return View();
  }

  public string StateChanged(string password, string state)
  {

   MessageReciever.StateChanged(password,state);
   return "message sent";
```

```
    }

    public ActionResult About()
    {
      return View("About");
    }
    public ActionResult ConnectToPC(string password)
    {

      Aadress pcAadress = db.Aadresses.Where(t => t.password == password).FirstOrDefault();
      if (pcAadress != null)
      {
        TempData["password"] = pcAadress.password;
        TempData["aadress"] = pcAadress.aadress+":8090/hosting/";
      }
      else
      {
        TempData["error"] = "sorry password not found :(";
        return View("Index");
      }

      return View("Remoter");

    }

  }
}
```

10.2.2 MessageReciever

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Microsoft.AspNet.SignalR;

namespace MediaRemote.Hubs
{
    public class MessageReciever : Hub
    {
        public static void StateChanged(string password, string action)
        {

            var context =
GlobalHost.ConnectionManager.GetHubContext<MessageReciever>();
            if (action.StartsWith("Play"))
                context.Clients.Group(password).Play();
            if (action.StartsWith("Pause"))
                context.Clients.Group(password).Pause();
            if (action.StartsWith("VolumeChanged"))
            {
                string volumeLevel = action.Split(':')[1];
                context.Clients.Group(password).volumeChanged(volumeLevel);

            }
            if (action.StartsWith("Close"))
                context.Clients.Group(password).Close();
            if (action.StartsWith("Filename"))
            {
                string filename = action.Split(':')[1];
                context.Clients.Group(password).SendFilename(filename);
            }
        }


        public void Register(string password)
        {
            Groups.Add(Context.ConnectionId, password);
            return;
        }
    }
}
```

10.2.3 Remoter

```
$(document).ready(function () {
  var isPlaying = true;
  var clientAadress = getClientAadress();
  var sendToPlayer = true;
  var messageReciever = $.connection.messageReciever;
  var volumeLevel = 50;
  var name;
  var reDirectEndpoint = 'http://rmcr.azurewebsites.net';
  $.connection.hub.start().done(function () {

    messageReciever.server.register(getPassword());
  });

  $("#volumeLevel").html('Volumelevel: '+volumeLevel);
  var NameAadress = 'http://' + getClientAadress() + "Data";

  $.ajax({
    type: "GET",
```

```
  url: NameAadress,
  dataType: "jsonp",
  contentType: "application/json;charset=utf-8",
  success: function(data) {
    var bufferData = data.toString();
    var jsonInfo = $.parseJSON(bufferData);
    name = GetNameToShow(jsonInfo.name);
    isPlaying = jsonInfo.playing;
    volumeLevel = jsonInfo.volumeLvl;
    if (!isPlaying) {
      $('#playBtn').attr('src', '/Content/images/PlayBtn.png');

    } else {
      $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
    }

    $('.heading').html(name);
    $('#volumeSlider').val(volumeLevel);
    $('#volumeSlider').slider("refresh");
  },
  error: function(xhr, ajaxOptions, thrownError) {
    ConnectionLost();
  }
});

$("#sliderDiv").change(function () {
  if (sendToPlayer) {
    var aadress = "http://" + getClientAadress() + "Volume/" + $('#volumeSlider').val();
    $.ajax({
      type: "GET",
      url: aadress,
      timeOut: 2000,
      success: function (data) {
      },
      error: function (xhr, ajaxOptions, thrownError) {
      }
    });
  }
});



messageReciever.client.volumeChanged = function (changed) {
  sendToPlayer = false;
  $('#volumeSlider').val(changed);
  $('#volumeSlider').slider("refresh");
  sendToPlayer = true;
};

messageReciever.client.play = function () {
  $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
  isPlaying = true;
};

messageReciever.client.pause = function () {
  isPlaying = false;
  $('#playBtn').attr('src', '/Content/images/PlayBtn.png');
};

messageReciever.client.close = function () {

  $('#onOffFlip').val('off');
```

```
  PlayerClosed();
};

messageReciever.client.sendFilename = function (name) {
  $('.heading').html(GetNameToShow(name));
};

function GetNameToShow(name) {
  if (name.length > 14)
    return name.substring(0, 20) + '...';
  else {
    return name;
  }
}

$(".onoff").change(function () {
  var aadress = "http://" + getClientAadress() + "Close";
  $.ajax({
    type: "GET",
    url: aadress,
    timeOut: 5000,
  });
  PlayerClosed();
});

$('#playBtn').click(function () {
  var aadress;
  if (isPlaying == true) {
    aadress = "http://" + getClientAadress() + "Pause";
    $('#playBtn').attr('src', '/Content/images/PlayBtn.png');

  } else {
    var aadress = "http://" + getClientAadress() + "Play";
    $('#playBtn').attr('src', '/Content/images/PauseBtn.png');
  }
  $.ajax({
    type: "GET",
    url: aadress,
    timeOut: 5000,
    dataType: "jsonp",
    contentType: "application/json;charset=utf-8",
    success: function (data) {
      if (isPlaying)
        isPlaying = false;
      else {
        isPlaying = true;
      }
    },
    error: function (xhr, ajaxOptions, thrownError) {
      ConnectionLost();
    }
  });
});

function ConnectionLost() {
  $.msg({
    bgPath: '/content/images/',
    content: "Connection has been lost...you're being redirected to mainpage"
  });

  messageReciever.server.close(getPassword());
  setTimeout(function () {
```

```
    window.location.href = "http://remote.azurewebsites.net";
   }, 2000);
  }

  function PlayerClosed() {

   $.msg({
    bgPath: '/content/images/',
    content: "Player has been closed...you're being redirected to mainpage"
   });
   messageReciever.server.close(getPassword());
   setTimeout(function () {
    window.location.href = "http://remote.azurewebsites.net";
   }, 2000);
  }

  function getClientAadress() {
   var retval = $('#aadress').val();
   return retval;
  }

  function getPassword() {
   var retval = $('#password').val();
   return retval;
  }
});
```

10.2.4 Global.Asax

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
using System.Web.WebPages;

namespace MediaRemote
{
  // Note: For instructions on enabling IIS6 or IIS7 classic mode,
  // visit http://go.microsoft.com/?LinkId=9394801

  public class MvcApplication : System.Web.HttpApplication
  {
    protected void Application_Start()
    {
      RouteTable.Routes.MapHubs();
      AreaRegistration.RegisterAllAreas();

      WebApiConfig.Register(GlobalConfiguration.Configuration);
      FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
      RouteConfig.RegisterRoutes(RouteTable.Routes);
      BundleConfig.RegisterBundles(BundleTable.Bundles);
      AuthConfig.RegisterAuth();
      DisplayModeProvider.Instance.Modes.Insert(0,
       new DefaultDisplayMode("Mobile")
       {
        ContextCondition = (ctx => (
         (ctx.GetOverriddenUserAgent() != null) && ctx.Request.Browser.IsMobileDevice
```

```
      ))
      });
    }
  }
}
```

10.2.5 MovieService

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.ServiceModel;
using System.ServiceModel.Channels;
using System.ServiceModel.Web;
using KantarisMP;
using System.IO;
using NetFwTypeLib;
using System.Web.Script.Serialization;

namespace KantarisMP
{
  [ServiceContract]
  [ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]
  class MovieService
  {

    private static MovieService instance;
    private MovieService() { }
    public static MovieService Instance
    {
      get
      {
        if (instance == null)
        {
          instance = new MovieService();
        }
        return instance;
      }
    }

    Kantaris mainForm;
    private string password;

    JavaScriptSerializer jsS = new JavaScriptSerializer();
    private string myLocalIp;

    [WebGet(UriTemplate = "/Pause",
    BodyStyle = WebMessageBodyStyle.WrappedRequest,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json)]
    [OperationContract]
    public string Pause()
    {

      mainForm.pauseForService();
      return jsS.Serialize("paused");
    }

    [WebGet(UriTemplate = "/Play",
    BodyStyle = WebMessageBodyStyle.WrappedRequest,
    RequestFormat = WebMessageFormat.Json,
```

```
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string Play()
{

  mainForm.playForService();
  return jsS.Serialize("playing");
}

[WebGet(UriTemplate = "/Volume/{volumeLevel}")]
[OperationContract]
public string Volume(string volumeLevel)
{
  mainForm.volumeChangeForService(volumeLevel);
  return "Playing";
}

[WebGet(UriTemplate = "/Close")]
[OperationContract]
public string Close()
{
  RemoveFirewallException();
  mainForm.Close();
  return "closed";
}

[WebGet(UriTemplate = "/Data",
BodyStyle = WebMessageBodyStyle.WrappedRequest,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json)]
[OperationContract]
public string GetInitalData()
{
  string name="no movie selected";
  int volumeLvl = 50;
  bool playing = false;
  try
  {
    playing = mainForm.isPlaying();
    volumeLvl = mainForm.getVolumeLevel();
    name = mainForm.getPlayList();
  }
  catch (Exception e)
  {

  }
  var kk = new
  {
    name,
    volumeLvl,
    playing

  };
  string json = jsS.Serialize(kk);
  return json;
}
public void Initialize(Kantaris mainForm, string password)
{
  PasswordForm passwordForm = new PasswordForm();
  passwordForm.Show();
  this.mainForm = mainForm;
}

public void ActionChanged(string action)
{
```

```
      if (password != null)
      {
       HttpWebRequest httpWReq = (HttpWebRequest)WebRequest.Create(String.Format(host +
"StateChanged?password={0}&state={1}", password, action));
       Task.Factory.StartNew(() =>
       {
        httpWReq.GetResponse();
       });
      }
    }
    public string RegisterNewPassword(string aadress)
    {
      HttpWebRequest httpWReq = (HttpWebRequest)HttpWebRequest.Create(String.Format(host +
"RegisterNewPassword?aadress={0}",aadress));
      WebResponse response = httpWReq.GetResponse();
      Stream stream = response.GetResponseStream();
      string result = String.Empty;
      using (StreamReader sr = new StreamReader(stream))
      {
       result = sr.ReadToEnd();
      }
      AddFirewallException();
      password = result.Split('@')[1];
      return result;
    }
    public string DeletePassword()
    {
      HttpWebRequest httpWReq =
(HttpWebRequest)WebRequest.Create(String.Format(host+"Remote/DeletePassword?password={0}", password));
      return httpWReq.GetResponse().ToString();
    }
    public void SendFileName()
    {
      HttpWebRequest httpWReq = (HttpWebRequest)WebRequest.Create(String.Format(host +
"Remote/StateChanged?password=test&Filname={1}", password, mainForm.getPlayList()));
      httpWReq.GetResponse();
    }

    public void SavePassword(string password)
    {
      this.password = password;
    }

    public bool AddFirewallException()
    {
      INetFwMgr icfMgr = null;
      try
      {
       Type TicfMgr = Type.GetTypeFromProgID("HNetCfg.FwMgr");
       icfMgr = (INetFwMgr)Activator.CreateInstance(TicfMgr);
      }
      catch (Exception ex)
      {
       return false;
      }

      try
      {
       INetFwProfile profile;
       INetFwOpenPort portClass;
       Type TportClass = Type.GetTypeFromProgID("HNetCfg.FWOpenPort");
       portClass = (INetFwOpenPort)Activator.CreateInstance(TportClass);
       profile = icfMgr.LocalPolicy.CurrentProfile;
       portClass.Scope = NetFwTypeLib.NET_FW_SCOPE_.NET_FW_SCOPE_ALL;
       portClass.Enabled = true;
```

```
        portClass.Protocol = NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP;
        portClass.Name = "DescriptionOfRule";
        portClass.Port = 8090;
        profile.GloballyOpenPorts.Add(portClass);
        return true;
      }
      catch (Exception ex)
      {
        return false;
      }
    }

    public void RemoveFirewallException()
    {
      INetFwMgr icfMgr = null;
      try
      {
        Type TicfMgr = Type.GetTypeFromProgID("HNetCfg.FwMgr");
        icfMgr = (INetFwMgr)Activator.CreateInstance(TicfMgr);
      }
      catch (Exception ex)
      {
        return;
      }

      try
      {
        INetFwProfile profile;
        INetFwOpenPort portClass;
        Type TportClass = Type.GetTypeFromProgID("HNetCfg.FWOpenPort");
        portClass = (INetFwOpenPort)Activator.CreateInstance(TportClass);
        profile = icfMgr.LocalPolicy.CurrentProfile;
        portClass.Scope = NetFwTypeLib.NET_FW_SCOPE_.NET_FW_SCOPE_ALL;
        portClass.Enabled = true;
        portClass.Protocol = NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP;
        portClass.Name = "DescriptionOfRule";
        portClass.Port = 8090;
        profile.GloballyOpenPorts.Remove(8090,
NetFwTypeLib.NET_FW_IP_PROTOCOL_.NET_FW_IP_PROTOCOL_TCP);
      }
      catch (Exception ex)
      {
      }
    }
  }
}
```