TALLINNA POLÜTEHNIKUM

Eric Talviste

**VEEBI-PÕHINE SUHTLUS KESKKOND  
*„CHATCENTER“***

Lõputöö

**Juhendaja:** Dan Piilend  
  
**Koostaja**:

Tallinn 2019

SISUKORD

[SISSEJUHATUS 3](#_Toc535960142)

[EESMÄRGID 4](#_Toc535960143)

[TEOREETILINE OSA 5](#_Toc535960144)

[TERMINID 6](#_Toc535960145)

[RAKENDUSLIK OSA 7](#_Toc535960146)

[KESKKONNA INSTALLEERIMINE 7](#_Toc535960147)

[TÖÖRIISTAD 7](#_Toc535960148)

[INSTALLEERIMINE 7](#_Toc535960149)

[KAUSTA STRUKTUUR 7](#_Toc535960150)

[SERVERI TESTIMINE 8](#_Toc535960151)

[KAHESUUNALISE KOMMUNIKATSIOONI LOOMINE 9](#_Toc535960152)

[KOKKUVÕTE 9](#_Toc535960153)

[KASUTATUD KIRJANDUS 10](#_Toc535960154)

# SISSEJUHATUS

*„ChatCenter“* ehk „*VestlusKeskus“* on veebi-põhine vestlus **keskkond**, kus kõik kasutajad ei pea üksteist tingimata tundma.

Sarnaseid veebi-keskkondi on samuti olemas näiteks *Omegle, Rolechat,* ja kunagi ka *ShamChat*.

*Omegle* oli püstitatud 2009 a. 25. märts ning on selle idee algne algataja,  
*Omegle* alustas tavalise tekstipõhise vestlus-keskkonnaga hiljem lisades ka kaamera funktsiooni ning ka „küsitlus“-funktsioon kus kasutaja saab lisada küsimuse ning lugeda, kuidas 2 võõrast selle üle arutavad.

*Omegle*, kuigi üks vanemaid ja originaalsemaid algatajaid *võõrastega suhtlemises*, on üle aastate omandanud mitmeid **funktsionaalseid/algoritmilisi probleeme** millele pole *Omegle* meeskond tähelepanu pööranud.  
Nimelt omegle võitleb siiamaani ***„reklaami-bottidega“***, ehk kolmanda-isiku poolt ehitatud *automatiseeritud programm*, mis sirvib läbi automaatselt *omegle* vestlused, jättes sinna omalt poolt ka reklaami.

*RoleChat* ja *ShamChat* on praktiliselt samade ideedega, kuid ainult teksti-põhised, nende eripära seisneb selles, et kasutaja saab määrata omale nime, *Omegles* näiteks oled sa alati nimega „*võõras“*.   
*ShamChat* pandi kinni 2018 aasta lõpus omanike ebaaktiivsuse tõttu.  
RoleChat töötab ikka endiselt.

Kõigil kolmel on erinevad probleemid, kuid peamiseks probleemiks ütleksin on *eba-aktiivsus*.  
Arendajad on ehitanud üles hea ideega veebilehed kuhu inimesed *lihtsalt* tulevad ning unustanud kõik muud potentsiaalsed funktsioonid ja probleemide parandamised välja.

Kõik küll valdavad samat põhi ideed, ühildada üks võõras teise võõraga ning jätta see keskkond registreerimis ja tasu vaba.

*„VestlusKeskus“* seisneb veebi põhise kommunikatsiooni avamises ning ka selle edasi arendamises.  
*VestlusKeskuse* eesmärk on vaadata **erinevaid** viise, kuidas me saaksime veebi-põhist kommunikatsiooni kasutada olgu ta meelelahtuslik või praktiline.

# EESMÄRGID

Kuna *„Veebi-põhine vestlus keskkond“* on üsna avar ning laialdane pean sätestama omale paar eesmärki, eesmärgid mis loeksid selle projekti lõpetatuks.

Eesmärk 1:  
Luua kõige algelisem kahe suunaline kommunikatsiooni aplikatsioon.

Eesmärk 2:  
Lisa funktsioon, mis laseb kasutajal otsida teise võõra-kasutajaga **kahe** poolse vestlus ruumi.

Eesmärk 3:  
Funktsioon, mis laseb kasutajal lisada küsimuse mille üle 2-teist võõrast kasutajat saavad sinu küsimuse üle arutleda.

Eesmärk 4:   
4 Liikmeline võõr-kasutajate vestlus ruum.

Eesmärk 5:  
Mikrofoni vestlus

Eesmärk 6:  
Veebi-kaamera vestlus

# TEOREETILINE OSA

# TERMINID

# RAKENDUSLIK OSA

Meie veebi-keskkond on põhiliselt üles ehitatud kahe inimese ühildamisega veebis, seega alustame selle esimese probleemiga:  
  
***Kuidas ühildada 2 või enam kasutajat läbi veebilehe?***

## KESKKONNA INSTALLEERIMINE

## TÖÖRIISTAD

Et veebikeskkonda luua peame algselt paar töörista installeerima, nendeks on:  
 **Ekspress.js; node.js; socket.io;**

Mis asjad need on?  
  
***Ekspress.js*** *–veebi aplikatsioonide raamistik mis töötab node.js peal*

***Node.js*** *-Keskkond, mis võimaldab* ***JavaScriptil*** *töötada* ***browseri*** *välises keskkonnas*

***Socket.io*** *-JavaScripti* ***raamatukogu****, mis võimaldab luua reaalajas, kahesuunalise ühenduse* ***veebi-kliendi*** *ja* ***serverite*** *vahel.*

## INSTALLEERIMINE

Paigaldame kõik vajalikud ressursid kasutades git-bash-i:

Algselt installime ekspressi: npm install --save express

## KAUSTA STRUKTUUR

Enne, kui edasi lähme vaatame üle projekti kausta struktuuri:

ChatCenter

|-- node\_modules

|-- package.json

|-- package-lock.json

`-- server.js

**ChatCenter** – *meie projekti nimi, peamine projekti kaust.*

**Node\_Modules** – *on meie moodulite ehk lisa tööristade kaust, näiteks node.js ja ekspress.js*

***Package&package-lock.json*** *– siin asub meie projekti kirjeldus ja üldine info.*

***Server.js*** *– Siin rakendame kõik* ***serveriga*** *seonduvad koodid.*

## SERVERI TESTIMINE

Et testida kas meie express raamistik töötab lisame selle koodi meie **server.js** faili:

var app = require('express')();

var http = require('http').Server(app);

app.get('/', **function**(req, res){

res.send('<h1>Hello world</h1>');

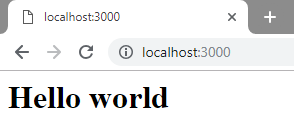
});

http.listen(3000, **function**(){

console.log('listening on \*:3000');

});

Minnes browserisse localhost:3000 lehele näeme seda:



Kuidas see töötab?

var app = require('express')();

var http = require('http').Server(app);

-Siin me toome sisse oma **express raamistiku** ning vormindame selle.

-**HTTP** laseb meil saata andmeid läbi *http protokolli*

-**Server** loob lokaalselt meile arvuti peale serveri.

app.get('/', **function**(req, res){

res.send('<h1>Hello world</h1>');

-**App.Get** on HTTP meetod, mis laseb meil ligipääseda taotlemise ja reageerimise funktsioonidele

-**’/’** määrab marsruudi meie funktsioonide talletamiseks

-**function(req,res)**   
 -req on HTTP taotlemise argument  
 -res on HTTP vastuse tagamise argument

-**res.send()** laseb meil saata antud andmeid

http.listen(3000, **function**(){

console.log('listening on \*:3000');  
-**http.listen** avab meie serveri 3000 pordil  
-**console.log()** annab meile *konsoolis* teada, et server on edukalt käivitatud.

## KAHESUUNALISE KOMMUNIKATSIOONI LOOMINE

Meie arendus keskkond on loodud ning server töötab, nüüd saame alustada oma kommunikatsiooni loomisega.

Algselt loome **html** faili, kus saame kõike veebilehe visuaalseid ehk *frontend* elemente manipuleerida.

# KOKKUVÕTE

# KASUTATUD KIRJANDUS

**REDO THIS A BIT   
ChatCenters main purpose is to implement various  
ways of communication and chatting**

**->>**

Writing a chat application with popular web applications stacks like LAMP (PHP) has traditionally been very hard. It involves polling the server for changes, keeping track of timestamps, and it’s a lot slower than it should be.

Sockets have traditionally been the solution around which most real-time chat systems are architected, providing a bi-directional communication channel between a client and a server.

This means that the server can push messages to clients. Whenever you write a chat message, the idea is that the server will get it and push it to all other connected clients.

[**https://stackoverflow.com/questions/26921117/http-createserverapp-v-http-serverapp**](https://stackoverflow.com/questions/26921117/http-createserverapp-v-http-serverapp)

[**https://www.w3schools.com/nodejs/nodejs\_http.asp**](https://www.w3schools.com/nodejs/nodejs_http.asp)

**Ran into JS External file problem**

**~~Need to implement usernames aside messages~~**

**~~Migration to server~~**

**https://stackoverflow.com/questions/27599614/var-express-requireexpress-var-app-express-what-is-express-is-it**