Manu Kortesmäki

Google Charts & SignalR

Opinnäytetyö
Kevät 2016
SeAMK Tekniikka
Tietotekniikan koulutusohjelma

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Manu Kortesmäki

Työn nimi: Google Charts & SignalR

Ohjaaja: Petteri Mäkelä

Vuosi: 2016 Sivumäärä: 39 Liitteiden lukumäärä: 4

Opinnäytetyön tavoite oli tutkia SignalR ASP.NET -kirjastoa, Google Charts -kaaviopalvelua ja näiden yhteiskäyttöä. Työn alussa tutkittiin SignalR-kirjastoa, Google Chartsia ja näihin liittyvää teknologiaa, kuten vaihtoehtoista tekniikkaa. Työssä toteutettiin palvelinohjelma, joka vastaanottaa dataa toiselta ohjelmalta ja vie vastaanotetun datan Google Charts -kaavioon, jota voi tarkastella web-selaimella. Palvelinohjelmassa on käytetty C#-, HTML- ja JavaScript-koodia.

Avainsanat: SignalR, Google Charts, ASP.NET

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Authors: Manu Kortesmäki

Title of thesis: Google Charts & SignalR

Supervisor: Petteri Mäkelä

Year: 2016 Number of pages: 39 Number of appendices: 4

The purpose of this thesis was to research SignalR, Google Charts and their combined use. At the beginning of the thesis, SignalR and Google Charts were researched along with other technologies connected to them. A client-server program was created during this thesis project. The program receives data from an external source and is visualized in a graph which can viewed using a web-browser. The server program was programmed using C#, HTML and JavaScript languages.

Keywords: SignalR, Google Charts, ASP.NET

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

API	Application Programming Interface eli ohjelmointirajapinta jota ohjelmat käyttävät vaihtaakseen tietoja keskenään.
IETF	Internet Engineering Task Force. Suuri maailmaanlaajuinen avoin kommuuni, joka koostuu mm. verkkosuunnittelijoista, operaattoreista ja tutkijoista, joiden tehtävänä on kehittää internetin arkkitehtuuria ja toimivuutta.
SSL	Secure Sockets Layer. Protokolla, jolla voidaan salata IP-verkkojen yli kulkevaa tietoliikennettä.
W3C	World Wide Web Consortium. Maailmanlaajuinen yhteisö, jossa jäsenorganisaatiot, henkilökunta ja vapaaehtoiset kehittävät verkkostandardeja.
Skripti	Skripti on skriptikielellä kirjoitettu komentosarja (esimerkiksi JavaScript).
Socket	Päätepiste verkkokommunikoinnissa. Kommunikaatioprotokollat lähettävät dataa socketista socketiin ja sovellukset lukevat ja kirjoittavat dataa niihin.
Latenssi	Aika, joka kuluu matkassa paketin lähettäjältä sen vastaanottajalle.

1 Johdanto

1.1 Työn tausta

Opinnäytetyön aiheena oli tutkia datan lähettämistä palvelimen ja selaimen välillä. Opinnäytetyön palvelinohjelmassa käytettiin SignalR-kirjastoa ja sen visualisointiin selaimessa käytettiin Google Charts -kaavioita.

Datan visualisointi on nykyään entistä tärkeämpää datalähteiden lisääntymisen vuoksi. Dataa saadaan esimerkiksi sosiaalisesta mediasta, web-sivuilta ja mittauslaitteista. Yrityksille datan hyödyntäminen on tärkeää, koska sitä voidaan käyttää kilpailukyvyn parantamiseen. Datan suuri määrä voi aiheuttaa sen, että perinteiset taulukot eivät anna riittävän hyvää kuvaa datan sisällöstä. Dataa voidaan esittää selkeämmin visualisoimalla sitä esimerkiksi kaavioilla. (Yuki & Diamond 2014,1.)

Datan määrän kasvaminen lisää myös dataliikenteen määrää. Raskas dataliikenne rasittaa verkkoja, joten liikenteen rasituksen pienentäminen toisi suuria hyötyjä. SignalR-kirjasto ja WebSocket-protokolla pystyvät pienentämään verkkojen kuormitusta ainakin tietynlaisissa sovelluksissa.

1.2 Työn tavoite

Tämän opinnäytetyön tavoitteena on Googlen tarjoaman Google Charts -palvelun sekä Microsoftin kehittämän SignalR ASP.NET -kirjaston tutkiminen ja näiden yhteiskäytön testaaminen ohjelmoinnissa. Opinnäytetyössä toteutetaan palvelinohjelma käyttäen SignalR-kirjastoa, joka lähettää dataa selaimelle. Tätä dataa visualisoidaan käyttäen Google Chartsin kaavioita.

Palvelinohjelma ohjelmoidaan C#-ohjelmointikielellä käyttäen Microsoftin kehittämää Visual Studio-ohjelmankehitysympäristöä. Palvelinohjelmassa käytetään Microsoftin SignalR ASP.NET -kirjastoa, joka erikoistuu reaaliaikaisiin web-sovelluksiin ja pystyy hyödyntämään WebSocket-protokollaa. Valmis palvelinohjelma lähettää dataa, jota visualisoidaan käyttäen kaavioita.

Datan visualisointiin käytetään Googlen kehittämää Google Charts -kaaviopalvelua. Google Charts-kaaviot ovat HTML-pohjaisia, joten visualisointia varten ohjelmoidaan web-sivu, jolla kaaviot näytetään.

Ohjelmoinnin jälkeen palvelinohjelma ja datan visualisointi testataan ja sen perusteella arvioidaan ohjelman toiminta.

1.3 Työn rakenne

Luku 2 käsittelee palvelintekniikoita. Pääaiheina ovat WebSocket ja SignalR, mutta luvussa kerrotaan myös vaihtoehtoisista ja vanhemmista tekniikoista.

Luku 3 käy läpi HTML5-merkintäkieltä ja siihen liittyviä asioita, jotka ovat tämän opinnäytetyön kannalta tärkeitä.

Luku 4 kertoo datan visualisointikeinoista web-sivuilla. Luvun pääosassa on Google Charts -palvelu, mutta siinä käsitellään myös vaihtoehtoisia JavaScript-kirjastoja.

Luku 5 esittelee opinnäytetyötä varten tehtyä ohjelmaa.

Luvussa 6 on opinnäytetyön tulokset ja yhteenveto.

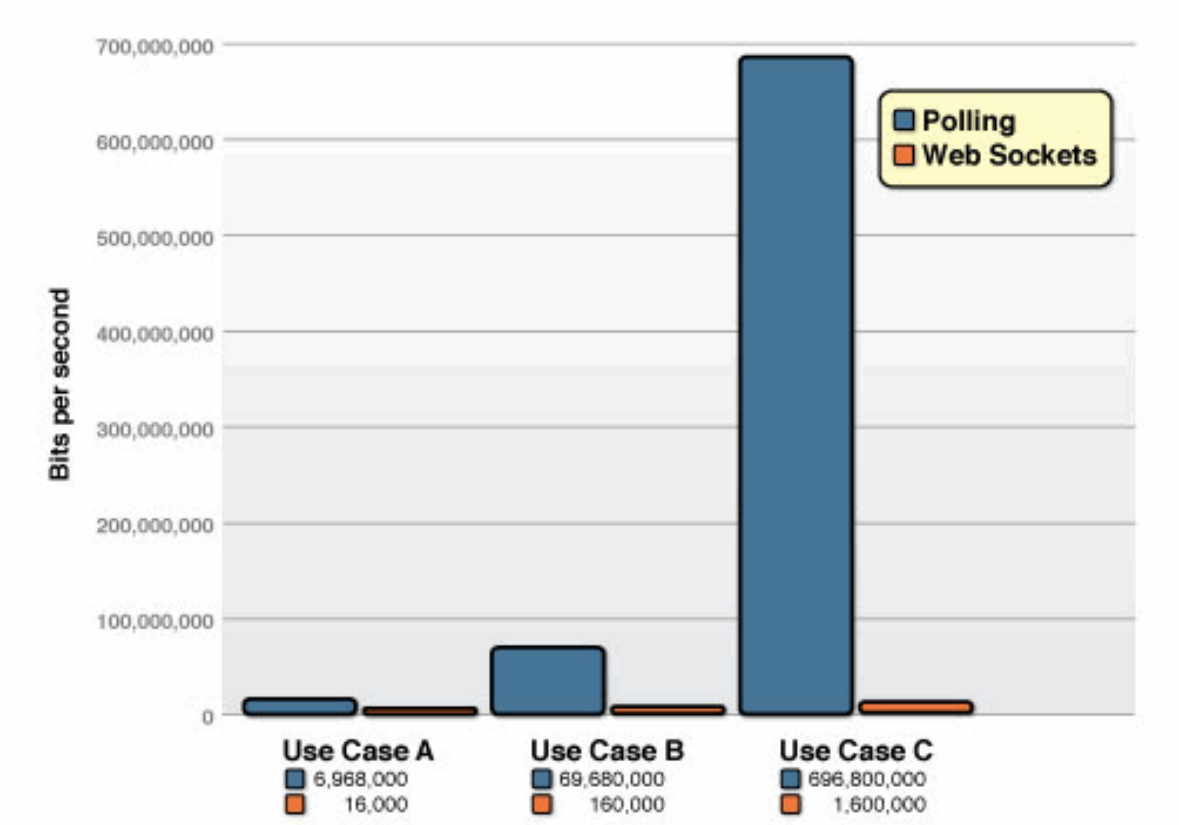
2 WebSocket ja SignalR

2.1 WebSocket

WebSocket on protokolla, joka mahdollistaa pitkäaikaisen kaksisuuntaisen yhteyden web-palvelimen ja selaimen välillä. Näin molemmat osapuolet pystyvät lähettämään dataa toisilleen samanaikaisesti. WebSocketilla, kuten yleisesti käytössä olevalla HTTP-protokollalla, on oma skeema URL-osoitteessaan. HTTP-protokollan URL sisältää skeeman ”http://” ja WebSocketin skeema on ”ws://” (Kuvio 1). Kun käytetään suojattua yhteyttä käyttäen SSL-protokollaa, niin WebSocketin skeemasta tulee ”wss://”. (Pterneas 2013, 7, 11.)

Kuvio 1. WebSocketin URL
(Hansa 2016).

WebSocketin kaksisuuntainen yhteys toimii vain yhden socketin kautta. Vanhemmat tekniikat voivat käyttää kahta socketia saavuttaakseen saman. Näin yhden socketin käyttö pienentää latenssia ja kuormitusta verkossa. Kuviossa 2 kuvataan verkkoliikenteen määrää pollauksella ja WebSocketilla. Testi on toteutettu lähettämällä yksi viesti sekunnissa määrätylle määrälle asiakaskoneita. Use Case A -tapauksessa asiakaskoneita on 1000, tapauksessa B niitä on 10 000 ja tapauksessa C 100 000. Liikenteen määrä on mitattu biteissä sekunnissa. Kuviosta voi havaita WebSocketin käytön pienentävän liikenteen määrää huomattavasti. (Kaazing 2016.)



Kuvio 2. Pollauksen ja WebSocketin vertailu
(Kaazing 2016).

WebSocket-yhteyttä muodostaessa se ei heti alussa ole vielä WebSocket-yhteys, vaan HTTP-yhteys. Yhteyttä muodostaessa asiakaskone lähettää palvelimelle kättelyn, jossa on pyyntö vaihtaa käytössä oleva HTTP-protokolla WebSocket-protokollaksi (kuvio 3). Kun palvelin vastaanottaa ja ymmärtää kättelyn, se lähettää asiakaskoneelle oman kättelynsä, jossa protokollan vaihto on hyväksytty (Kuvio 4). Tämän jälkeen WebSocket-yhteys korvaa käytössä olevan HTTP-yhteyden käyttäen samoja yhteystietoja. (Kaazing 2016.)

```
GET ws://echo.websocket.org/?encoding=text HTTP/1.1
Origin: http://websocket.org
Cookie: __utma=99as
Connection: Upgrade
Host: echo.websocket.org
Sec-WebSocket-Key: uRovscZjNol/umbTt5uKmw==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

Kuvio 3. Pyyntö vaihtaa HTTP-yhteys WebSocket-yhteydeksi
(Kaazing 2016).

```
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Fri, 10 Feb 2012 17:38:18 GMT
Connection: Upgrade
Server: Kaazing Gateway
Upgrade: WebSocket
Access-Control-Allow-Origin: http://websocket.org
Access-Control-Allow-Credentials: true
Sec-WebSocket-Accept: rLHCkw/SKsO9GAH/ZSFhBATDKrU=
Access-Control-Allow-Headers: content-type
```

Kuvio 4. Yhteyden vaihto hyväksytään
(Kaazing 2016).

WebSocket-yhteydellä on mahdollista helpottaa toimintaa palomuurien ja välipalvelinten (proxy) kanssa. WebSocket kykenee havaitsemaan välipalvelimen, jolloin se perustaa tunnelin lähettämällä HTTP CONNECT -pyynnön välipalvelimelle. Tämän jälkeen välipalvelin avaa TCP/IP-yhteyden tietylle palvelimelle ja porttiin. Tunnelin perustaminen mahdollistaa tehokkaan kommunikoinnin välipalvelimen kautta. (Kaazing 2016.)

WebSocket on vielä suhteellisen uusi protokolla, joten kaikki selaimet tai vanhemmat selainversiot eivät välttämättä tue sitä. Taulukossa 1 on listattu selaimet ja selainversiot, jotka tukevat WebSocketia.

Taulukko 1. WebSocketia tukevat selaimet ja selainversiot (Pterneas 2014, 19).

Selain	Versio
Internet Explorer	**10+**
Firefox	**20+**
Chrome	**26+**
Safari	**6+**
Opera	**12.1+**
Safari for iOS	**6+**
Blackberry Browser	**7+**

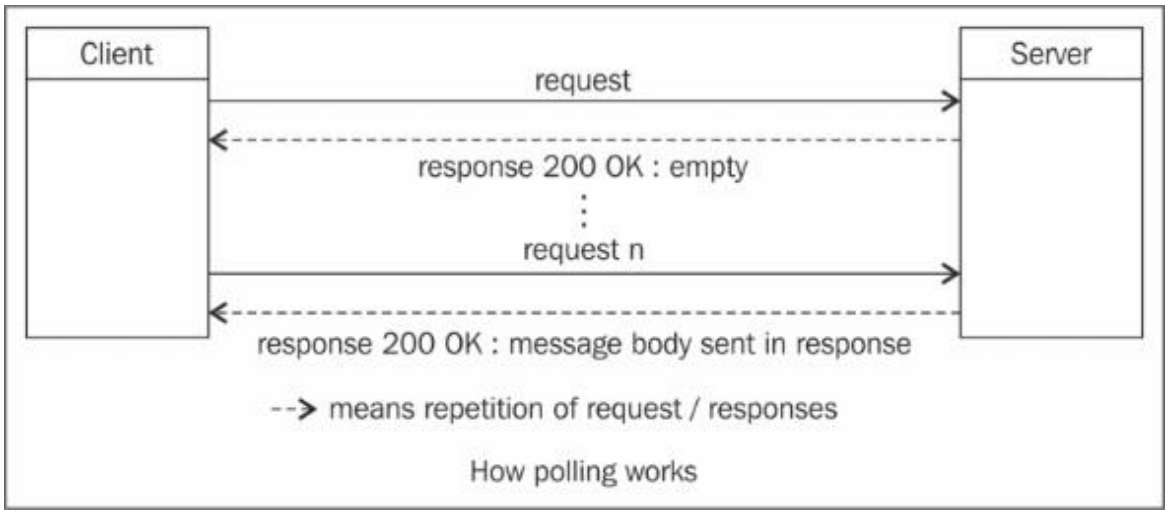
IETF (Internet Engineering Task Force) on standardisoinut WebSocket-protokollan. WebSocket API:n standardisointia hoitaa W3C (World Wide Web Consortium). (Pterneas 2013, 8.)

2.2 WebSocketia edeltävät tekniikat

Seuraavassa luvussa kerrotaan WebSocketia vanhemmista tekniikoista. Luvun tarkoitus on antaa selkeämpi kuva WebSocketin hyödyistä ja tiedonsiirrosta yleisesti.

2.2.1 Pollaus, pitkä pollaus ja suoratoisto

Varhainen tapa hoitaa kaksisuuntaista kommunikointia oli Polling eli pollaus, jossa asiakaskone suorittaa ajoittaisesti pyyntöjä palvelimelle, ja jokaiseen pyyntöön palvelin vastaa saatavilla olevalla datalla tai viestillä (kuvio 5). Pollaus on edelleen toimiva tapa kaksisuuntaiseen viestimiseen, mutta se on turhan raskas verrattuna muihin tapoihin. (Pterneas 2013, 8.)



Kuvio 5. Viestit pollauksessa
(Simpson 2016).

Long Polling eli pitkä pollaus on askel eteenpäin tavallisesta pollauksesta. Siinä asiakaskone avaa yhteyden palvelimelle, ja palvelin pitää yhteyden auki niin kauan, että dataa on lähetetty tai aikakatkaisuun asti. (Pterneas 2013, 8.)

Streaming eli suoratoisto on taas askel eteenpäin. Siinä asiakaskoneen yhteyspyynnön jälkeen palvelin pitää yhteyden auki jatkuvasti ja hakee uutta dataa. Suoratoisto on nykyäänkin hyvä reaaliaikaiseen datan lähetykseen, mutta on olemassa myös kevyempiä ja nopeampia tapoja. (Pterneas 2013, 8.)

2.2.2 HTTP

HTTP eli Hypertext Transfer Protocol on OSI-mallin sovelluskerroksen tiedonsiirtoon tarkoitettu protokolla, jota on käytetty tiedonsiirtoon jo 1990-vuosikymmenestä lähtien. Ensimmäinen käyttöönotettu HTTP-versio on HTTP 0.9. Asiakaskone ottaa isäntäkoneeseen TCP–IP yhteyden käyttämällä tämän domain-nimeä tai IP-osoitetta ja porttia 80. HTTP:n kaksi yleisintä metodia pyyntöihin ja vastauksiin ovat GET, joka pyytää dataa määritetystä resurssista ja POST, joka lähettää dataa resurssiin. (W3Schools 2016 ; Berners-Lee 1999.)

IETF sai vuonna 2015 valmiiksi Hypertext Transfer Protocollan version 2 (HTTP/2) (Nottingham 2015). Sen on tarkoitettu toimivan yhdessä vanhempaa HTTP/1-protokollaa käyttävien kirjastojen kanssa. HTTP/2 ei sisällä uusia metodeja tai

protokollaa liikaa muuttavia muutoksia, mutta sen uudet ominaisuudet voivat vaatia hienosäätöä toimiakseen oikein. (Nottingham 2014.)

HTTP/2 pyrkii pienentämään pyyntöjen aiheuttamaa rasitetta multipleksauksella, joka helpottaa viestiliikennettä. Se myös kykenee pakkaamaan viestien otsikoinnit, mikä vähentää huomattavasti kaistan käyttöä. HTTP/2 voi hoitaa yhdellä asiakaskoneen ja palvelimen välisellä TCP-yhteydellä saman mihin HTTP/1 saattoi tarvita useampaa yhteyttä. Näiden lisäksi HTTP/2 parantaa käyttökokemusta nopeuttamalla sitä muillakin tavoin, kuten paremmalla kryptauksella ja poistamalla turhia ja viallisia asioita. (Nottingham 2014.)

2.3 Socket.IO

Socket.IO on JavaScript-kirjasto Node.js-kehitysympäristölle, jonka on kehittänyt Guillermo Rauch. Se mahdollistaa samanaikaisen tiedonsiirron palvelimen ja asiakaskoneen välillä eli reaaliaikaisen kommunikoinnin. Socket.IO-kirjaston 1.0-versio julkaistiin 28.5.2014. Kehittämisen taustalla oli tekijän tarve sovelluskehykselle (framework), joka mahdollistaisi datan työntämisen palvelimelta asiakaskoneelle helposti. (Rauch 2014.)

Node.js on asynkroninen JavaScript-kehitysympäristö, joka perustuu Googlen V8 JavaScript -moottoriin. Se on tarkoitettu skaalautuvien verkko-sovellusten luomiseen. Ensimmäinen Node.js-versio julkaistiin vuonna 2009, ja kirjoitushetkellä uusin versio on julkaistu huhtikuussa 2016. (Node.js Foundation 2016.)

Socket.IO on tarkoitettu käyttämään pääasiassa WebSocket-protokollaa. Normaalissa yhteydenmuodostamisessa yhteys muodostetaan aluksi käyttäen XHR- (XMLHttpRequest) tai JSONP (JSON with Padding) -tekniikoita. Tämän jälkeen yhteys yritetään päivittää WebSocket-yhteydeksi. Socket.IO tukee selaimia, jotka tukevat WebSocket-protokollaa (taulukko 1). (Rauch 2014.)

Socket.IO-kirjaston mainittavia ominaisuuksia ovat

- käyttö palvelinkoneen lisäksi myös asiakaskoneessa
- binääridatan lähettäminen
- skaalautuvuus

- helppo sopeuttaminen. (Rauch 2014.)

Socket.IO-kehittäjät ovat luoneet Engine.IO-moduulin, joka auttaa Socket.IO-kirjaston yhteensopivuusongelmissa, jotka liittyvät selaimiin ja datan kuljetukseen. Engine.IO yksinkertaistaa palvelimen ohjelmointia ja lisää joustavuutta ja toimintavarmuutta. (Rauch 2014.)

Socket.IO-kirjastoa kehitetään jatkuvasti. Tämän kirjoitushetkellä uusin Socket.IO-versio on 1.4.5, joka on julkaistu tammikuussa 2016. Kehittäjäyhteisö myös lataa tekemiään Socket.IO-sovelluksia GitHub-pakettivarastoon, josta kuka tahansa voi ladata niitä omaan käyttöönsä. (Rauch 2014.)

2.4 SignalR

ASP.NET SignalR on avoimen lähdekoodin ASP.NET-kirjasto, jonka tarkoitus on yksinkertaistaa reaaliaikaisten verkkotoimintojen lisäämisen sovelluksiin. Esimerkkinä tällaisesta verkkotoiminnosta on erilaiset chatit ja valvontasovellukset, joiden toimintaperiaate perustuu reaaliaikaiseen päivitykseen. SignalR tarjoaa myös API:n, jolla palvelin voi lähettää etäismenettelyn kutsuja asiakaskoneen alustan (esim. web-selain) JavaScript-toimintoihin. (Fletcher 2014.)

SignalR käyttää ensisijaisesti WebSocket-protokollaa, mutta jos sen käyttö ei ole mahdollista, se käyttää vanhempia menetelmiä. SignalR-kirjaston käyttö helpottaa myös WebSocketin käyttöä. Jos WebSocketia päivitetään, niin myös SignalR päivitetään tukemaan mahdollisia muutoksia WebSocketissa. SignalR:n ja WebSocketin yhteiskäyttö vaatii, että palvelin käyttää ohjelmistoa, joka tukee näitä teknologioita. Jos palvelin ei täytä vaatimuksia, niin SignalR pyrkii käyttämään muita siirtotekniikoita. (Fletcher 2014.)

SignalR-palvelin tarvitsee käyttöjärjestelmäkseen jonkin seuraavista

- Windows Server 2012
- Windows Server 2008 r2
- Windows 8
- Windows 7. (Fletcher 2014.)

SignalR toimii myös Microsoft Azuressa. Palvelimen käyttöjärjestelmän on oltava Windows Server 2012 tai uudempi, jos haluaa käyttää SignalR-kirjastoa WebSocketin kanssa. Windows Azure tukee myös WebSocketia, jos .NET 4.5 on käytössä. (Fletcher 2014.)

Asiakaskone tarvitsee SignalR-sovelluksille sitä tukevan selaimen.

Seuraavat selaimet tukevat SignalR-kirjastoa.

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Safari
- Opera
- Android. (Fletcher 2014.)

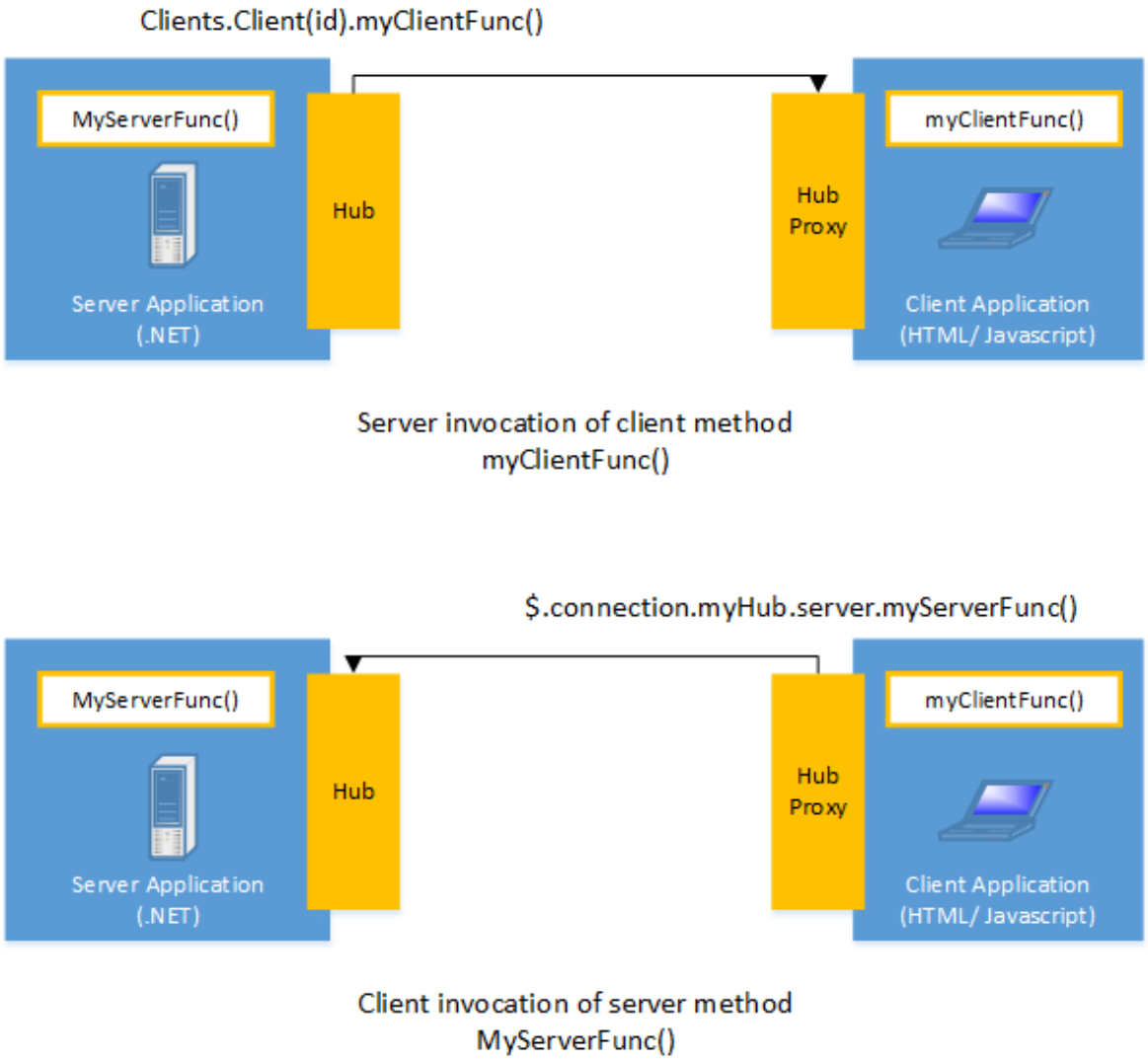
Taulukko 2. Selainten versiovaatimukset siirtotavan mukaan (Fletcher 2014).

	Internet Explorer	**Chrome**	**Firefox**	**Safari**	**Android**
WebSocket	10+	34.0+	30.0+	9.1+	Ei tukea
Server-Sent Events	Ei tukea	34.0+	30.0+	9.1+	Ei tukea
ForeverFrame	8+	Ei tukea	Ei tukea	Ei tukea	4.1
Long Polling	8+	34.0+	30.0+	9.1+	4.1

Taulukossa 2 on kuvattu eri selaimien tukemat siirtotavat ja vaadittu selaimen versio niille. On suositeltavaa käyttää vähintään taulukossa mainittu versiota. SignalR voi toimia muissakin selaimissa, mutta käyttö voi olla epävakaata, eikä mahdollisia vikoja korjata. (Fletcher 2014.)

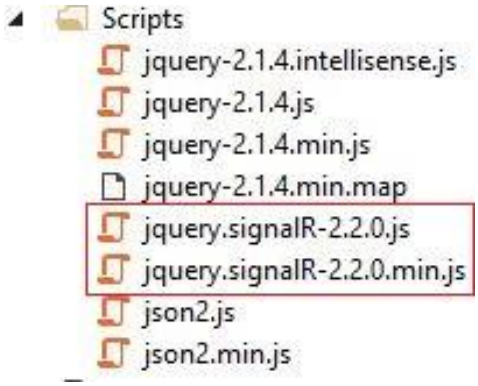
SignalR:n API voi käyttää kahta eri mallia asiakaskoneen ja palvelimen väliseen viestimiseen: ne ovat Persistent Connections ja Hubs. Persistent Connections toimii yhtenä päätepisteenä viestien lähettämiseen ja sen API antaa SignalR:n kanssa

kehittäjälle käyttöoikeuden matalatasoiseen kommunikaatioprotokollaan. Hubs on korkeamman abstraktiotason järjestelmä, joka mahdollistaa asiakkaan ja palvelimen suoran kommunikoinnin kutsumalla toistensa toimintoja (kuvio 6). (Fletcher 2014.)



Kuvio 6. Palvelin ja asiakas kutsuvat toistensa toimintoja (Fletcher 2014).

SignalR-toiminnot voi lisätä esimerkiksi Visual Studiossa käyttämällä paketinhallintakonsolia. Lisäys tapahtuu kirjoittamalla paketinhallintakonsoliin komennon ”install-package Microsoft.AspNet.SignalR” ja suorittamalla sen. (Fletcher 2014.) SignalR-kirjaston asentumisen voi tarkistaa ohjelman ”Scripts”-kansiosta (kuvio 7).



Kuvio 7. SignalR-kirjasto asennettuna

2.4.1 ASP.NET

ASP.NET on web-sovelluskehys, jonka on kehittänyt Microsoft. Ensimmäinen versio julkaistiin vuonna 2002. ASP.NET tukee Visual Basic-, C#- ja J#-ohjelmointikieliä. ASP.NET itse mahdollistaa kolmen erityyppisen kehyksen käytön, joilla kaikilla on omat käyttötarkoituksensa. (Microsoft 2016.)

Kehykset ovat

- ASP.NET Web Forms
- ASP.NET MVC
- ASP.NET Web Pages. (Microsoft 2016.)

Yhden tavan valitseminen ei kuitenkaan sulje pois toisten käyttöä, sillä kaikissa on samat ydintoiminnot, kirjautumissuojat ja samat käytännöt pyyntöjen ja sessioiden hallintaan. Tämä mahdollistaa useamman kehyksen käytön samassa sovelluksessa, jossa esimerkiksi iso osa sivusto käyttää MVC-arkkitehtuuria, mutta jokin pienempi osa siitä käyttää Web Formsia. (Microsoft 2016.)

Kehittäjät päivittävät ASP.NET-kehystä edelleen. Opinnäytetyön kirjoitushetkellä uusin ASP.NET-versio on ASP.NET 4.6. (Fritz 2015.)

3 HTML 5

HTML (Hypertext Markup Language) on web-sivujen merkintäkieli, joka alun perin kehitettiin vuonna 1991. HTML5 on uusin HTML-versio, se otettiin virallisesti käyttöön vuonna 2014. (W3Schools 2016a.)

HTML5-kielen parhaiten tunnettu käyttökohde on web-sivut, mutta se ei rajoitu pelkästään niihin. HTML5-kieltä voi käyttää myös yleisesti web-sovelluksiin, sähköpostissa ja vaikka e-kirjoihin. (Korpela 2014, 26.)

HTML-kielellä määritellään web-sivuille elementtejä, joihin kuuluu kolme osaa. Elementin osat ovat: alkutagi, sisältö ja lopputagi (Korpela 2014, 46). Jos esimerkiksi haluaa web-sivulle otsikon ja tekstiä, niin ne voi määritellä <h1>- ja <p>-elementeillä (kuvio 8).

Kuvio 8. HTML-kielen tekstielementtejä

Elementeillä voi myös olla määritteitä (attributes), joilla voi määritellä elementin ominaisuuksia tarkemmin. Määritteillä voi esimerkiksi määrittää tekstille fonttikokoa, tai haluamalleen elementille luokan. (Korpela 2014, 104.)

Kaikki yleisessä käytössä olevat web-selaimet tukevat HTML5-kieltä. Kaikki selaimet tosin eivät oletuksena tunnista kaikkia HTML5-elementtejä. On kuitenkin mahdollista ”opettaa” selaimelle elementtejä, käyttäen document.createElement()-metodia. (W3Schools 2016a.)

3.1 XML

XML (Extensible Markup Language) on merkintäkieli, joka on tarkoitettu tiedon varastointiin ja kuljetukseen. XML ei sisällä tietoa datan esittämisestä, joten sillä ei

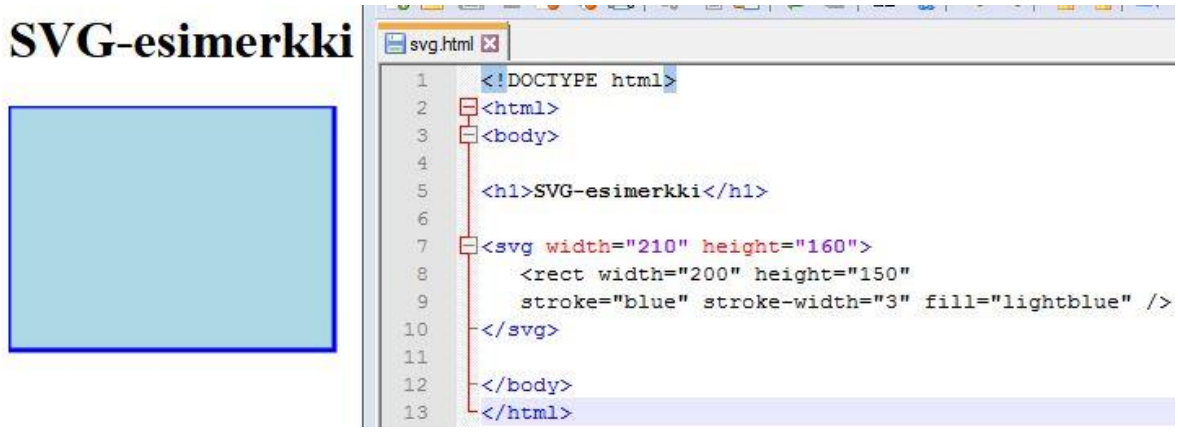
voi täysin korvata HTML-kieltä. Näitä tosin voi käyttää yhdessä: XML-kieltä datan varastointiin ja kuljettamiseen ja HTML-kieltä saman datan muotoiluun ja näyttämiseen. (W3Schools 2016b.)

XML-kielen virallinen määrittäminen tapahtui vuonna 1998. W3C on kehittänyt HTML-kieltä enemmän XML-pohjaiseksi. Näin kehitettiin XHTML (eXtensible Hypertext Markup Language). (Korpela 2014, 28–29.)

XHTML-kielen ensimmäinen versio julkaistiin vuonna 2000. HTML-kielen muokkaaminen XML-pohjaiseksi teki siitä rajoitetumpaa ja pidensi merkintöjä. XHTML-kielellä voidaan korvata HTML-kieli, mutta sen edut tulevat esiin järjestelmissä, joissa XML-yhteensopivuus on hyödyllistä. HTML5 on kuitenkin suositumpi kieli web-sivujen merkintään kuin XHTML. (Korpela 2014, 28–29.)

3.2 SVG

SVG (Scalable Vector Graphics) on XML-pohjainen kuvan tallennusmuoto. SVG-kuvat ovat XML-muodossa, johon kuvien tiedot on kirjoitettu SVG-kielellä. SVG-tallennusmuodon etu on, että kuvat ovat aina juuri niin tarkkoja kuin näyttölaitteen ominaisuudet sallivat. SVG on käytännöllinen kuvioille, kaavioille ja muille yksinkertaisille kuville, mutta tekstipohjaisuus tekee siitä kömpelön kuvamuodon valokuville ja muille monimutkaisille kuville. Kuviossa 9 on kuvattu, miten SVG-kielellä voi ohjelmoida yksinkertaisen kuvan. (Korpela 2014, 804–805.)

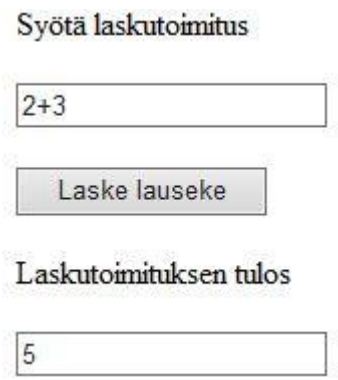


Kuvio 9. SVG-kielellä ohjelmoitu kuva

SVG on ollut olemassa jo 2000-luvun alusta asti, mutta sitä ei käytetä kovin laajasti vielä. Yksi syy on SVG:n vähäinen tuki. Internet Explorer alkoi tukemaan SVG-muotoa vasta versiossa 9 ja Android-versiossa 3. (Korpela 2014, 804.)

3.3 JavaScript

JavaScript on ohjelmointikieli, jolla selaimen voi saada suorittamaan scriptejä. Scriptit ovat skriptikielellä kirjoitettuja ohjelmia, jotka suoritetaan HTML-sivun näyttämisen yhteydessä. Sillä voi muokata HTML-sivuja monella tavalla, kuten lisätä kuvia, taulukoita ja toimintoja (kuvio 10). Jos selaimella on puutteellinen HTML5-tuki, JavaScriptillä voi paikata tilannetta (document.createElement()). JavaScriptiä voi käyttää selainohjelmoinnin lisäksi myös palvelinohjelmointiin. (Korpela 2014, 55.)

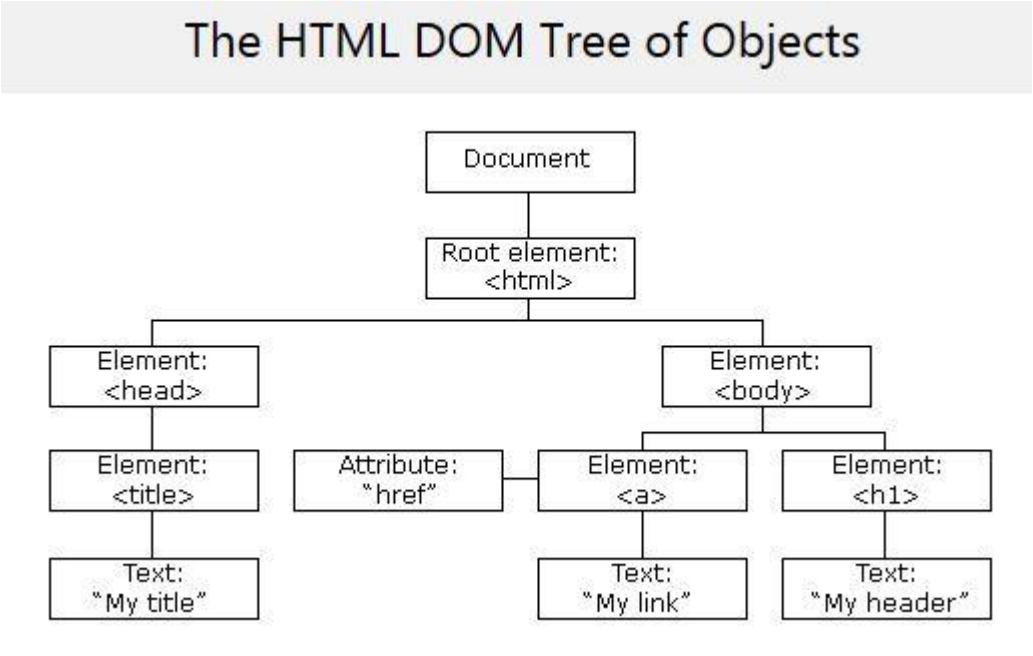


Kuvio 10. JavaScriptillä ohjelmoitu laskin

Esimerkkejä skripteistä ovat esimerkiksi:

- script-elementti
- ”javascript”-alkuinen URL-määrite
- tapahtumat, kuten onclick
- SVG-kuvat, joissa kuva piirretään sivulle. (Korpela 2014, 600.)

DOM (Document Object Model) eli dokumenttioliomalli on tärkeä, kun JavaScriptiä käytetään HTML-sivun käsittelyyn. DOM kuvaa, mitkä elementit ja selaimen osat ovat JavaScriptin käsiteltävinä (kuvio 11). (Korpela 56–57.)



Kuvio 11. HTML DOM -puukaavio
(W3Schools 2016c).

JavaScriptin lisäksi on olemassa muitakin selainohjelmointikieliä, mutta JavaScript on näistä ylivoimaisesti käytetyin. Tätä auttaa sen laaja tuki selaimissa (Korpela 2014, 56). Ecma International, joka on tieto- ja viestintäteknologioiden standardisoitiin omistautunut yhdistys, on standardisoinut JavaScriptin. ECMA-standardisoidun kielen nimi on ECMAScript. (Ecma International 2015.)

JavaScript-ohjelmoinnissa on mahdollista käyttää myös JavaScript-kirjastoja. Yksi näistä kirjastoista on esimerkiksi tässä opinnäytetyössä käytössä ollut jQuery-kirjasto. jQueryn tarkoitus on yksinkertaistaa asiakaspuolen toimintojen ohjelmointia HTML-sivuilla. (The jQuery Foundation 2016.)

4 Datan visualisointi

Datan visualisointi on tiedon esittämistä muilla tavoin kuin perinteisessä tekstimuodossa, kuten esimerkiksi kaavioilla ja kuvilla. Yleinen tapa visualisoida tietoa on tehdä infograafeja jotka koostuvat useista asioista. Nämä infograafit voivat pitää sisällään vaikka kuvia, kaavioita ja tekstiä. (Yuki & Diamond 2014, 7-8.)

Hyvässä datan visualisoinnissa on usein huomioitu kolme asiaa:

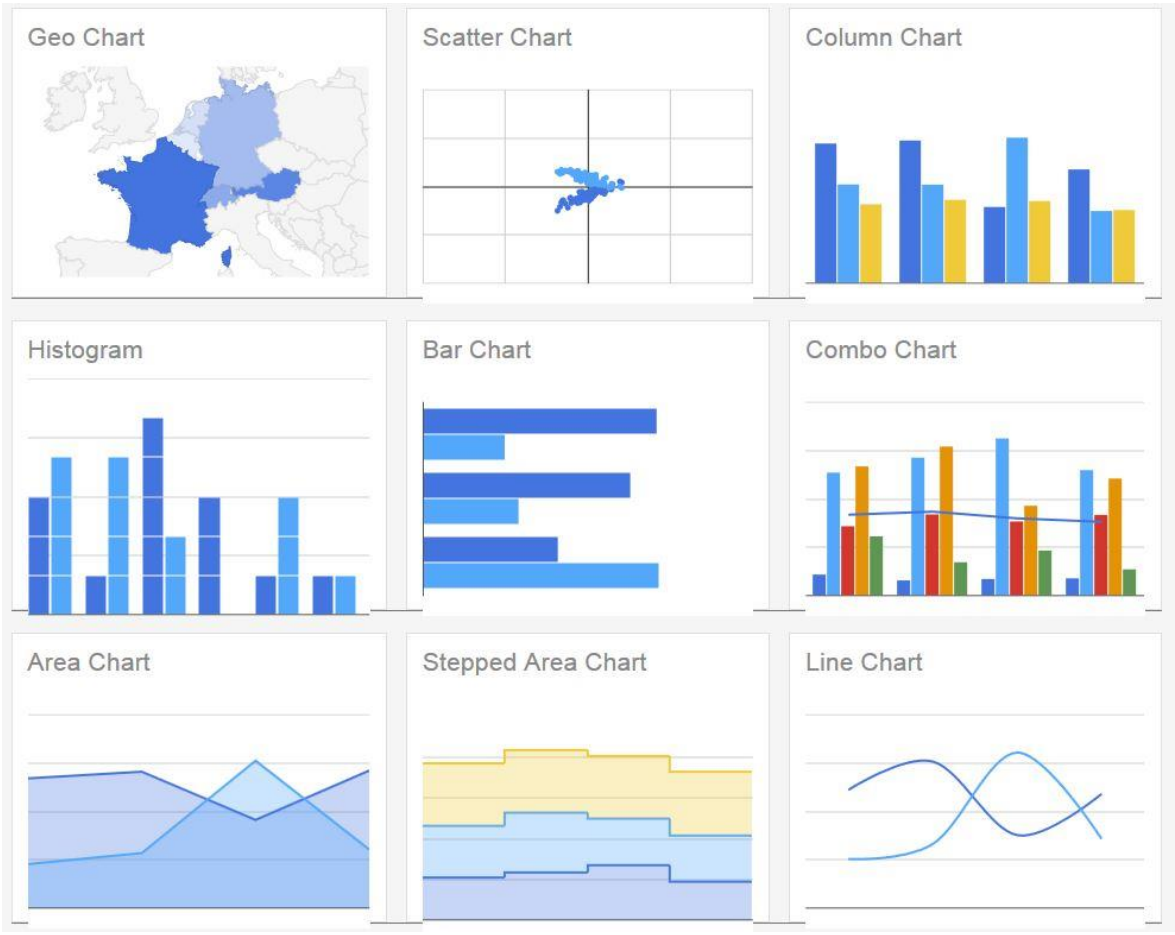
- tarpeellisuus, tietoa tarvitaan usein ja sitä käytetään tärkeisiin päätöksiin
- mukavuus, visualisoitua tietoa on helppo ja mukava hyödyntää
- käytettävyys, tiedon käyttäjät saavuttavat sen avulla tavoitteensa helposti. (Yuki & Diamond 2014, 9.)

4.1 Google Charts

Google Charts on opinnäytetyössä käytetty Googlen kaaviopalvelu, jonka avulla voi helposti lisätä omalle web-sivulleen erilaisia kaavioita datan visualisointiin. Google Charts tarjoaa kymmeniä erilaisia kaavioita, joita voi käyttää oletusmuodossaan tai myös muokata tarvittaessa mieleisekseen, jos tyyli ei sovi web-sivulle. Kaavioita voi jopa yhdistellä isommiksi kokonaisuuksiksi erilaisilla toiminnoilla. Google Chartsin kaavioita on havainnollistettu kuviossa 12. (Google 2015.)

Google Charts tarjoaa esimerkiksi näitä kaavioita:

- Pylväskaavio
- Palkkikaavio
- Gantt-kaavio
- Piirakkakaavio
- Aikajana
- Taulukkokaavio
- Maakaavio
- Kartta
- Janakaavio. (Google 2015.)



Kuvio 12. Joitain Google Chartsin kaavioita (Google 2015).

Palvelun kaaviot esitellään JavaScript-luokkina ja ne renderoidaan HTML5- ja SVG-teknologioilla. Vanhemmat Internet Explorer -versiot käyttävät VML-tallennusmuotoa. Nämä mahdollistavat kaavioiden yhteensopivuuden useimmilla web-selaimilla, jopa mobiililaitteilla kuten iPhone, iPad ja Android. Google Charts toimii ilman selainten ladattavia lisäominaisuuksia tai muita ylimääräisiä ohjelmia. Google Chartsin kaaviot voi saada toimimaan Flashillakin, jos esimerkiksi lataa tämän mahdollistavan kirjaston. (Google 2015.)

Kaikki taulukot käyttävät tietolähteenään DataTable-luokkaa, tämä helpottaa eri kaavioiden välillä vaihtelun oikeanlaista kaaviota etsiessä. DataTable-objekti on kaksiulotteinen taulukko, jonka jokaisella pystyrivillä on määritetty ainakin sen pystyrivin ID, nimiö ja datatyyppi. DataTableen voi lisätä dataa esimerkiksi web-sivuilta, tietokannoista tai käyttäen Chart Tools Datasource -protokollaa. Kuviossa 13 havainnollistetaan DataTablen luomista. Siinä tehdään taulukko kahdella sarakkeella ja viidellä pystyrivillä. (Google 2015.)

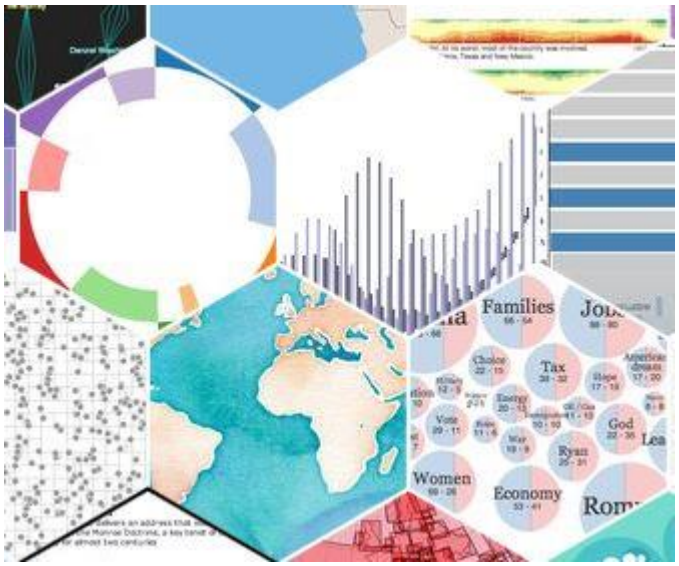
```
var dt = new google.visualization.DataTable({
    cols: [{id: 'task', label: 'Task', type: 'string'},
           {id: 'hours', label: 'Hours per Day', type: 'number'}],
    rows: [{c:[{v: 'Work'}, {v: 11}]},
           {c:[{v: 'Eat'}, {v: 2}]},
           {c:[{v: 'Commute'}, {v: 2}]},
           {c:[{v: 'Watch TV'}, {v:2}]},
           {c:[{v: 'Sleep'}, {v:7, f:'7.000'}]}]
    }, 0.6);
```

Kuvio 13. DataTable-taulukon luominen
(Google 2015).

Google Chartsia kehitetään edelleen aktiivisesti ja siitä julkaistaan uusia versioita ajoittain. Uusin virallisessa käytössä oleva versio on julkaistu lokakuussa 2015. Uusin versio, joka kuitenkaan ei ole virallisessa käytössä, on julkaistu 23.2.2016. (Google 2015.)

4.2 D3.js

D3.js (Data-Driven Documents) on avoimen lähdekoodin JavaScript-kirjasto. D3.js-kirjaston vahvuuksia ovat sen muokattavuus ja ominaisuudet, kuten Enter ja Exit. Se on myös tarkoitettu hyödyntämään tehokkaasti HTML-, SVG- ja CSS-teknologiaa. D3.js-kirjaston aiheuttama rasite tietokoneelle on minimoitu, tällä saavutetaan sen nopea toiminta ja tuki erilaisille toiminnoille ja animaatioille. D3.js ei itsessään sisällä valmiita kaavioita, toisin kuin esimerkiksi Google Charts. D3.js tarvitsee vähintään Internet Explorer 8 -version toimiakseen. Kuviossa 14 on kuvattu kaavioita, jotka on tehty käyttäen D3.js-kirjastoa. (Bostock 2015.)



Kuvio 14. D3.js-kaavioita
(Bostock 2015).

4.3 Chart.js

Chart.js on avoimen lähdekoodin JavaScript-kirjasto. Se on pienempi kuin Google Charts tai D3.js, sisältäen vain murto-osan edellisten erilaisista kaaviota. Chart.js on HTML5-pohjainen ja se hyödyntää sen Canvas-elementtiä kaavioihinsa. Chart.js vaatii toimiakseen selaimen, joka tukee HTML Canvas -elementtiä ja vähintään Internet Explorer -selaimen version 7. (Downie 2015.)

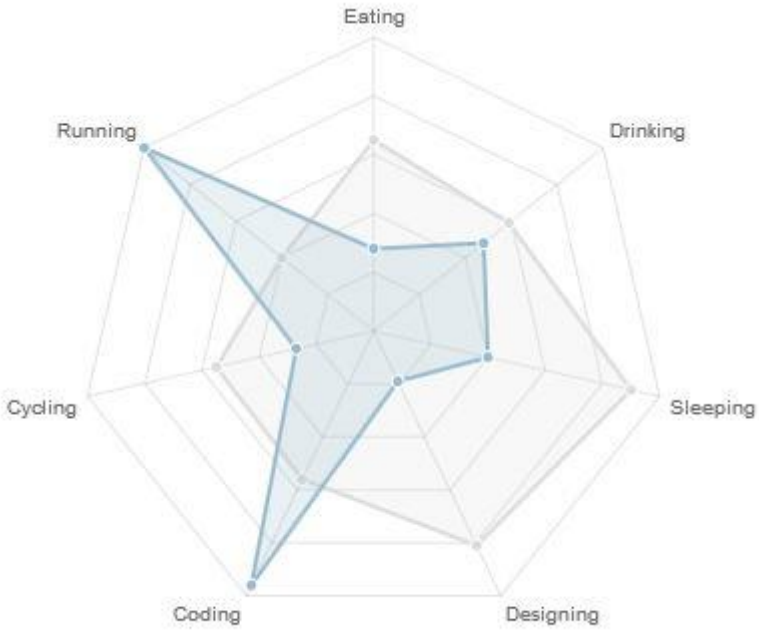
Charts.js vahvuudet ovat:

- yksinkertaisuus ja joustavuus
- responsiivisuus
- modulaarisuus
- vuorovaikutteisuus. (Downie 2015.)

Charts.js kaaviot ovat:

- Janakaavio
- Palkkikaavio
- Tutkakaavio
- Piiraskaavio. (Downie 2015.)

On mahdollista ohjelmoida myös uudenlaisia kaavioita Chart.js-kirjastoon, tai kehittää olemassa olevia pidemmälle. Kuviossa 15 on kuvattu yksinkertainen tutkakaavio, joka on tehty Chart.js-kirjastolla. (Downie 2015.)



Kuvio 15. Chart.js-tutkakaavio
(Downie 2015).

5 Ohjelma

Opinnäytetyötä varten kirjoitettu ohjelma on luotu käyttäen Visual Studio 2013 -ohjelmankehitysympäristöä. Ohjelma ottaa vastaan dataa, jota erillinen ohjelma generoi. Dataa generoiva ohjelma luo sekunnin välein kolme eri ”mittausarvoa”, jotka ovat numeroita väliltä 1 ja 20. Data lähetetään määrättyyn porttiin, josta palvelinohjelma voi hyödyntää sitä.

Palvelimen ohjelmointiin on käytetty C#-, HTML- ja JavaScript-kieliä.

5.1 Palvelin

Tässä luvussa keskitytään palvelinohjelman SignalR-toimintaan.

Työn palvelin käyttää toimintaansa monia luokkia, joita ovat esimerkiksi:

- Startup.cs
- MeasurementController.cs
- MeasurementHub.cs.

Palvelimella on muitakin luokkia, mutta edellä mainitut luokat liittyvät SignalR-toimintaan.

Startup.cs-luokka on kooltaan pieni, mutta se sisältää palvelimelle tärkeän toiminnon. Luokka sisältää OWIN-käyttöliittymän toiminnon, joka mahdollistaa verkkopalvelinten ja verkkosovellusten välistä toimintaa. Tämä tapahtuu työn palvelimella komennolla ”app.MapSignalR();” (kuvio 16).

```
using System;
using System.Threading.Tasks;
using Microsoft.Owin;
using Owin;

[assembly: OwinStartup(typeof(WebApiServerSR.Startup))]

namespace WebApiServerSR
{
    1 reference
    public class Startup
    {
        0 references
        public void Configuration(IAppBuilder app)
        {
            app.MapSignalR();
        }
    }
}
```

Kuvio 16. Startup.cs-luokka

MeasurementController.cs-luokka sisältää funktioita, joilla palvelin kommunikoi dataa generoivan ohjelman kanssa ja vie dataa kaavioita operoivaan skriptiin (kuvio 17). Luokan koodissa määritellään SignalR Hub -API, joka mahdollistaa aliohjelmien kutsumisen palvelimelta asiakkaille ja toisin päin. HttpResponseMessage-metodi kommunikoi dataa generoivan ohjelman kanssa HTTP POST -metodilla. Metodi lähettää myös kaikille asiakaskoneille generoidun datan, joka sisältää tunnistenumeron ja kolme mittausarvoa. Tätä lähetyskomentoa tullaan kutsumaan datan visualisoinnin aikana.

```
public MeasurementController()
{
    _uptimeHub = GlobalHost.ConnectionManager.GetHubContext<MeasurementHub>();
}

0 references
public HttpResponseMessage PostMeasurement(Models.Measurement item)
{
    // Google Chartin päivitys
    _uptimeHub.Clients.All.broadcastMeasToChart(item.Id, item.Meas1,
                item.Meas2, item.Meas3);

    var response = Request.CreateResponse<Models.Measurement>(HttpStatusCode.Created, item);

    string uri = Url.Link("DefaultApi", new { id = item.Id });
    response.Headers.Location = new Uri(uri);
    return response;
}
```

Kuvio 17. MeasurementController-luokan metodit

MeasurementHub.cs-luokan tarkoitus on toimia enimmäkseen välikätenä palvelimen toiminnassa. Se yksinkertaistaa palvelimen toimintaa ja helpottaa metodien paikallista toimintaa. Luokkaan ei ole tämän työn kannalta tarpeellista kirjoittaa omaa koodia (kuvio 18).

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

namespace WebApiServerSR
{
    1 reference
    public class MeasurementHub : Hub
    {

    }
}
```

Kuvio 18. MeasurementHub-luokka

5.2 Datan visualisointi

Sen jälkeen kun data on saatu palvelimelle, on vuorossa sen visualisointi kaaviolla. Kaavio on nähtävissä selaimella luettavalla HTML-sivulla. Sivun koodin ohjelmointiin on käytetty kahta kieltä: HTML-merkintäkieltä ja JavaScript-ohjelmointikieltä tarvittavilla kirjastoilla.

Kun kyseessä oleva HTML-sivu avataan, niin asiakaskoneen selain suorittaa sivulle kirjoitettua koodia. Koodissa viitataan useisiin kirjastoihin, joita tarvitaan datan visualisointiin. Viittaukset tehdään Google Charts-, jQuery-, SignalR- ja SignalR hub -kirjastoihin (kuvio 19).

```
<!--Scriptiviittaukset. -->
<!--Google Charts-viittaus-->
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<!--Viitataan jQuery-kirjastoon.-->
<script src="Scripts/jquery-2.1.4.min.js"></script>
<!--Viitataan SignalR-kirjastoon.-->
<script src="/Scripts/jquery.signalR-2.2.0.js"></script>
<!--Viittaus automaattisesti generoituun SignalR hub-scriptiin. -->
<script src="/signalr/hubs"></script>
```

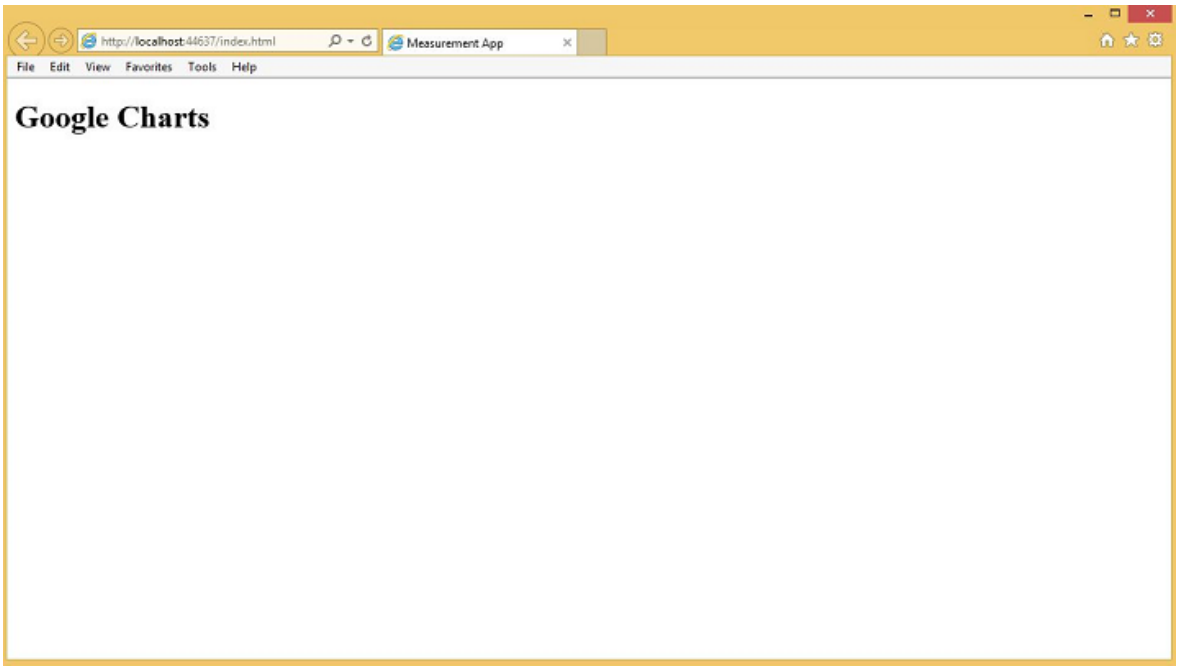
Kuvio 19. HTML-sivun viittaukset

Google Charts -viittauksella saadaan käyttöön kyseisen palvelun tarvittavat resurssit. jQuery-kirjasto mahdollistaa yksinkertaisen skriptikoodin, jonka avulla on helpompi saada sivulle toimintoja. SignalR- ja SignalR hub -viittaukset ovat selaimen ja SignalR-palvelimen yhteistoimintaa varten.

Ennen kuin ohjelma alkaa suorittaa skriptiä, joka luo kaavion sivulle, se luo ensin HTML-elementit. Näistä elementeistä näkyvimmät ovat sivun otsikko, datan tekstimuodossa näyttävä kenttä ja alue, jolle dataa näyttävä kaavio piirretään (kuvio 20). Tässä vaiheessa ainoastaan sivun otsikko on nähtävissä silmin (kuvio 21).

```
<!--Tilat tekstille ja taulukolle-->
<div><h1>Google Charts</h1></div>
<div> <p id="textp"></p> </div>
<div id="chart"></div>
```

Kuvio 20. HTML-elementit



Kuvio 21. Tilanne ennen kaaviota

Seuraavaksi suoritetaan sivulle kirjoitettua skriptiä. Datan visualisointiin tarkoitettu kaavio alustetaan ja sille määritellään asetukset ja taulukko dataa varten. Asetukset määritellään ”options”-muuttujalla, jossa määritetään kahden akselin otsikot (kuvio 22). Taulukossa on pystyrivit tarvittavaa dataa varten ja 20 vaakariviä. Vaakarivien määrä tarkoittaa sitä, että siinä voi kerralla olla maksimissaan 20 mittauksen arvot. Taulukossa siis voi olla näin monta mittausta yhtäaikaisesti, mutta uusille mittauksille täytyy tehdä tilaa poistamalla vanhat mittaukset (kuvio 23).

```
//Google Charts
google.load('visualization', '1', { packages: ['corechart', 'line'] });
google.setOnLoadCallback(initChart);

var data;
var chart;
var options = {
    hAxis: {
        title: 't'
    },
    vAxis: {
        title: 'Arvo',
    },
    curveType: 'none'
};
```

Kuvio 22. Kutsutaan taulukon alustus ja määritetään muuttujia

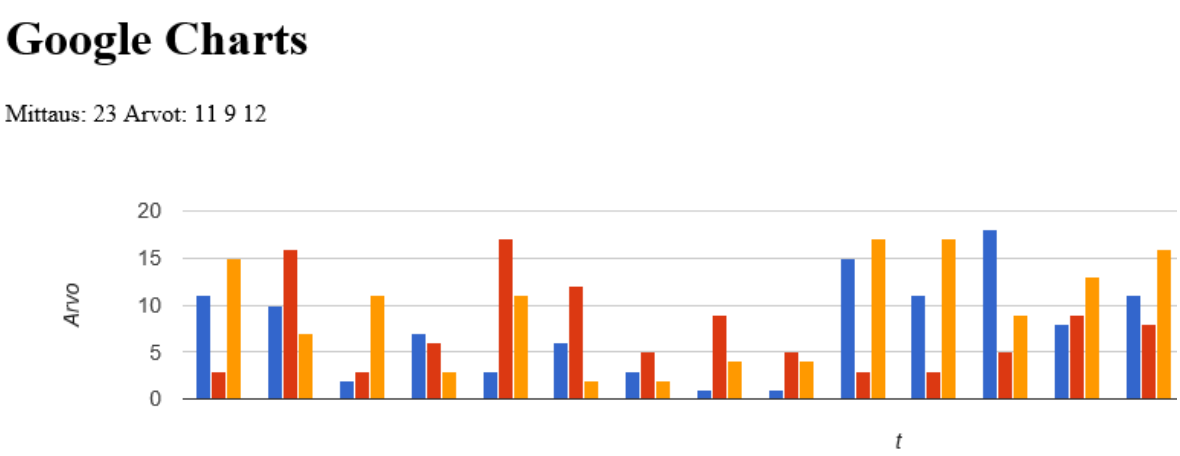
```
function initChart() {
    //Valmistellaan taulukko
    data = new google.visualization.DataTable();

    data.addColumn('string', 'X');
    data.addColumn('number', 'A');
    data.addColumn('number', 'B');
    data.addColumn('number', 'C');

    var da = [];

    for (var i = 0; i < 20; i++) {
        da.push(['', 0, 0, 0]);
    }

    data.addRows(da);

    //Kaavio piirretään "chart"-elementtiin
    chart = new google.visualization.ColumnChart(document.getElementById('chart'));
}
```

Kuvio 23. Funktio taulukon alustukseen

Kaavion valmistelun jälkeen voidaan aloittaa sivun päätoiminnon suorittaminen. Selain avaa yhteyden palvelimelle, jonka jälkeen palvelin alkaa lähettämään dataa. Lähetetystä datasta saadaan kolme mittaustulosta ja mittauksen tunnistenumero (id). Tässä vaiheessa kaavion datataulukosta poistetaan vanhin rivi ja tilalle lisätään uusi rivi, jossa on uusimmat mittaukset. Tämän jälkeen päivitetty kaavio piirretään sivulle ja kaavion yllä olevaan tekstikenttään kirjoitetaan myös uusimmat mittausarvot. Funktion koodi on nähtävissä kuviossa 24. Tämä skripti toistaa itseään niin kauan kuin kyseinen selaimen ikkuna on auki. Näin saadaan visualisoitua jatkuvasti uusia mittausarvoja ilman, että asiakkaan tarvitsisi itse päivittää sivua. Kuviossa 25 näytetään selaimen näkymä, kun dataa on kuvattu pylväskaaviossa.

```
$(function () {

    var hub = $.connection.measurementHub;

    //Funktio joka päivittää kaaviota
    hub.client.broadcastMeasToChart = function
        (id, meas1, meas2, meas3) {
        data.removeRow(0);
        data.addRow(['', meas1, meas2, meas3]);
        chart.draw(data, options);
        //Kirjoitetaan mitat tekstinä
        $("#textp").text
            ('Mittaus: ' + id + ' Arvot: ' + meas1 + ' ' + meas2 + ' ' + meas3);

    };

  $.connection.hub.start();
});
```

Kuvio 24. Funktio, jolla päivitetään sivua



Kuvio 25. Datan visualisointi pylväskaaviona

6 Tulokset ja yhteenveto

6.1 Tulokset

Työssä tutkittiin SignalR ASP.NET -kirjastoa, Google Charts -visualisointia ja näiden yhteiskäyttöä. Työssä toteutettiin myös palvelinohjelma, joka vastaanottaa erillisestä dataa generoivasta ohjelmasta tulevaa dataa ja visualisoi datan kaavioon. Palvelin käyttää SignalR-kirjastoa, joka on tarkoitettu reaaliaikaisten sovellusten luontiin. Data visualisoidaan HTML-sivulle käyttäen Google Charts -kaaviota.

Ohjelman toimintaa on onnistuneesti testattu Internet Explorer-, Mozilla Firefox- ja Google Chrome -selaimilla.

Ohjelmaa voisi vielä kehittää pidemmälle lisäämällä siihen WebSocket-toimintaa, mahdollisuuden käyttää internetin välityksellä ja toimintoja Google Charts -kaavioihin.

6.2 Yhteenveto

Tämän opinnäytetyön tuloksista voi todeta näissä tekniikoissa olevan potentiaalia. SignalR ja Google Charts ovat jo itsessään toimivia tekniikoita, mutta yhdessä ne ovat erittäin käytännöllisiä. Niiden käyttäminen mahdollistaa hyödyllisten ja tehokkaiden seurantasovellusten luomisen yksinkertaisesti. Muita vaihtoehtoja ei työn aikana testattu, mutta kappaleessa 4 mainitut D3.js ja Chart.js voivat hyvinkin olla samaa tasoa Google Chartsin kanssa.

Työn tekeminen oli hyödyllinen kokemus. Sen aikana pystyi kertaamaan teoriaa monista asioista ja syventämään tietämystään monessa ohjelmointikielessä. Työn aikana tuli esille myös täysin uusia asioita, joita täytyi opiskella työtä varten.

LÄHTEET

Berners-Lee, T. 1999. The HTTP Protocol As Implemented In W3. [www-dokumentti]. W3C. [Viitattu 4.4.2016]. Saatavissa: https://www.w3.org/Protocols/HTTP/AsImplemented.html

Bostock, M. 2015. Data-Driven Documents. [www-dokumentti]. D3 Data-Driven Documents. [Viitattu 14.4.2016]. Saatavissa: https://d3js.org/

Downie, N. 2015. Chart.js. [www-dokumentti]. Chart.js. [Viitattu 14.4.2016]. Saatavissa: http://www.chartjs.org/

Ecma International. 2015. What is Ecma International. [www-dokumentti]. Ecma International. [Viitattu 12.4.2016]. Saatavissa: http://www.ecma-international.org/memento/index.html

Fletcher, P. 2014. SignalR. [www-dokumentti]. Microsoft Corporation. [Viitattu 6.4.2016]. Saatavissa: http://www.asp.net/signalr

Fritz, J. 2015. Announcing ASP.NET 4.6 and ASP.NET 5 beta 5 in Visual Studio 2015 Release. [www-dokumentti]. Microsoft Corporation. [Viitattu 20.4.2016]. Saatavissa: https://blogs.msdn.microsoft.com/webdev/2015/07/20/announcing-asp-net-4-6-and-asp-net-5-beta-5-in-visual-studio-2015-release/

Google. 2015. Charts. [www-dokumentti]. Google Inc. [Viitattu 12.4.2016]. Saatavissa: https://developers.google.com/chart/

Hansa, U. 2016. Start Using HTML5 WebSockets Today. [www-dokumentti]. Envato Pty Ltd. [Viitattu 27.5.2016]. Saatavissa: http://code.tutsplus.com/tutorials/start-using-html5-websockets-today--net-13270

The jQuery Foundation. 2016. jQuery. [www-dokumentti]. The jQuery Foundation. [Viitattu 17.5.2016]. Saatavissa: https://jquery.com/

Kaazing. 2016. About HTML5 Websocket. [www-dokumentti]. Kaazing Corporation. [Viitattu 31.3.2016]. Saatavissa: http://www.websocket.org/aboutwebsocket.html

Korpela, J. 2014. HTML5-käsikirja. Jyväskylä: Docendo Oy.

Microsoft. 2016. ASP.NET Overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 6.4.2016]. Saatavissa: https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

Node.js Foundation. 2016. About Node.js®. [www.dokumentti]. Node.js Foundation. [Viitattu 26.4.2016]. Saatavissa: https://nodejs.org/en/about/

Nottingham, M. 2014. mnot's blog: Nine Things to Expect from HTTP/2. [www-dokumentti]. IETF. [Viitattu 4.4.2016]. Saatavissa: https://www.mnot.net/blog/2014/01/30/http2_expectations

Nottingham, M. 2015. mnot's blog: HTTP/2 is Done. [www-dokumentti]. IETF. [Viitattu 4.4.2016]. Saatavissa: https://www.mnot.net/blog/2015/02/18/http2

Pterneas, V. 2013. Getting Started With HTML5 Websocket Programming. Birmingham: Packt Publishing Ltd.

Rauch, G. 2014. Introducing Socket.IO 1.0. [www-dokumentti]. [Viitattu 25.4.2016]. Saatavissa: http://socket.io/blog/introducing-socket-io-1-0/#

Simpson, R. 2016. A Well Designed API Approach. [www-dokumentti]. [Viitattu 27.5.2016]. Saatavissa https://www.airpair.com/rest/posts/a-well-designed-api-approach

W3Schools. 2016. HTTP Methods GET vs POST. [www-dokumentti]. W3Schools. [Viitattu 4.4.2016]. Saatavissa: http://www.w3schools.com/tags/ref_httpmethods.asp

W3Schools. 2016a. HTML Introduction. [www-dokumentti]. W3Schools. [Viitattu 8.4.2016]. Saatavissa: http://www.w3schools.com/html/html_intro.asp

W3Schools. 2016b. Introduction to XML. [www-dokumentti]. W3Schools. [Viitattu 20.4.2016]. Saatavissa: http://www.w3schools.com/xml/xml_whatis.asp

W3Schools. 2016c. JavaScript Introduction. [www-dokumentti]. W3Schools. [Viitattu 20.4.2016]. Saatavissa: http://www.w3schools.com/js/js_intro.asp

Yuk, M. & Diamond, S. 2014. Data Visualization For Dummies. For Dummies/John Wiley & Sons Inc.

LIITTEET

Liite 1. Startup-luokka

Liite 2. MeasurementController-luokka

Liite 3. MeasurementHub-luokka

Liite 4. HTML-sivu

LIITE 1. Startup-luokka

```
using System;
using System.Threading.Tasks;
using Microsoft.Owin;
using Owin;

[assembly: OwinStartup(typeof(WebApiServerSR.Startup))]

namespace WebApiServerSR
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.MapSignalR();
        }
    }
}
```

LIITE 2. MeasurementController-luokka

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using WebApiServerSR.Models;
using Microsoft.AspNet.SignalR;

namespace WebApiServerSR.Controllers
{
    public class MeasurementController : ApiController
    {
        private static readonly IMeasurementRepository repository = new Measuremen-
tRepository();
        private readonly IHubContext _uptimeHub;

        public MeasurementController()
        {
            _uptimeHub = GlobalHost.ConnectionManager.GetHubContext<Meas-
urementHub>();
        }

        public HttpResponseMessage PostMeasurement(Models.Measurement item)
        {
            // Google Chartin päivitys
            _uptimeHub.Clients.All.broadcastMeasToChart(item.Id, item.Meas1,
                        item.Meas2, item.Meas3);

            var response = Request.CreateResponse<Models.Measurement>(HttpStatus-
Code.Created, item);

            string uri = Url.Link("DefaultApi", new { id = item.Id });
            response.Headers.Location = new Uri(uri);
            return response;
        }
    }
}
```

LIITE 3. MeasurementHub-luokka

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

namespace WebApiServerSR
{
    public class MeasurementHub : Hub
    {

    }
}
```

LIITE 4. HTML-sivu

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Measurement App</title>
</head>
<body>
    <!--Scriptiviittaukset. -->
    <!--Google Charts-viittaus-->
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <!--Viitataan jQuery-kirjastoon.-->
    <script src="Scripts/jquery-2.1.4.min.js"></script>
    <!--Viitataan SignalR-kirjastoon.-->
    <script src="/Scripts/jquery.signalR-2.2.0.js"></script>
    <!--Viittaus automaattisesti generoituun SignalR hub-scriptiin. -->
    <script src="/signalr/hubs"></script>

    <!--Scripti sivun päivittämiseen.-->
    <script type="text/javascript">

        //Google Charts
        google.load('visualization', '1', { packages: ['corechart', 'line'] });
        google.setOnLoadCallback(initChart);

        var data;
        var chart;
        var options = {
            hAxis: {
                title: 't'
            },
            vAxis: {
                title: 'Arvo',
            },
            curveType: 'none'
        };


        $(function () {

            var hub = $.connection.measurementHub;

            //Funktio joka päivittää kaaviota
            hub.client.broadcastMeasToChart = function (id, meas1, meas2, meas3) {
                data.removeRow(0);
                data.addRow(['', meas1, meas2, meas3]);
                chart.draw(data, options);
                //Kirjoitetaan mitat tekstinä
                $("#textp").text('Mittaus: ' + id + ' Arvot: ' + meas1 + ' ' + meas2
+ ' ' + meas3);


            };

          $.connection.hub.start();
        });

        function initChart() {
            //Valmistellaan taulukko
            data = new google.visualization.DataTable();

            data.addColumn('string', 'X');
```

```
            data.addColumn('number', 'A');
            data.addColumn('number', 'B');
            data.addColumn('number', 'C');

            var da = [];

            for (var i = 0; i < 20; i++) {
                da.push(['', 0, 0, 0]);
            }

            data.addRows(da);

            //Kaavio piirretään "chart"-elementtiin
            chart = new google.visualization.ColumnChart(document.getEle-
mentById('chart'));
        }

    </script>

    <!--Tilat tekstille ja taulukolle-->
    <div><h1>Google Charts</h1></div>
    <div> <p id="textp"></p> </div>
    <div id="chart"></div>

</body>
</html>
```