Ville Soikkola

MICROSOFT AZURE, HTML5 JA JAVASCRIPT

Opinnäytetyö
Tietojenkäsittely

Kesäkuu 2012

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** 6. kesäkuuta 2012
Tekijä(t) Ville Soikkola	**Koulutusohjelma ja suuntautuminen** **Tietojenkäsittely**
Nimeke Microsoft Azure, HTML5 ja JavaScript	

Tiivistelmä

Työn tavoitteena oli selvittää, millaiset verkkotekniikat ovat tällä hetkellä internetohjelmoinnissa käytössä sekä millaiset tekniikat lähivuosien aikana todennäköisesti yleistyvät. Koska toimenkuva on melko laaja, rajattiin työtä erityisesti käsittelemään HTML5:n mahdollistamia teknisiä toteutuksia yhdessä JavaScriptin kanssa sekä Windows Azuren pilvipalveluita.

Tutkimusongelma ratkaistiin ensin tutkimalla annettuja osa-alueita erikseen teoreettisella tasolla sekä tekemällä useita verkkotekniikoita yhdistävä HTML5-sivusto, jonka avulla pystyy soveltamaan teoriaa käytäntöön. Tutkimus osoitti, että uudet verkkotekniikat tuovat huomattavan määrän uusia mahdollisuuksia tehdä nettisivustoista entistäkin näyttävämpiä, interaktiivisempia sekä käyttäjäystävällisempiä.

Pilvipalveluiden osalta tutkimus osoittaa, että vaikka pilvipalvelut ovat eräänlainen trendi-ilmiö, ne soveltuvat hyvin vain suurille ja globaaleille toimijoille. Pienempien yritysten tarpeisiin, ainakin vielä toistaiseksi sopivat paremmin perinteiset palvelinratkaisut. Pilvipalveluiden ongelmiksi muodostuvat erityisesti niiden teknillisen toteutuksen myötä syntyvät kustannukset, tietoturvaongelmat sekä hallinnallisuuden rajoittuneisuus.

Asiasanat (avainsanat)
HTML5, JavaScript, Ajax, Azure, Pilvipalvelut

Sivumäärä	**Kieli**	**URN**
35 sivua	Suomi	URN:NBN:fi:amk-2012061512783

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Jukka Selin	Mikkelin ammattikorkeakoulu

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

Date of the bachelor's thesis

6 June 2012

Author
Ville Soikkola

Degree programme and option
Business Information Technology

Name of the bachelor's thesis

Microsoft Azure, HTML5 and JavaScript

Abstract

The goal of my thesis was to investigate which web techniques would be relevant in internet programming today and what kind of techniques would become generally used in near future. Due to the topic being so broad, the thesis was focused specifically on HTML5, JavaScript, and Windows Azure cloud services. The research problem was solved by studying the theory of the selected parts individually. These parts were then used to make a program that mixed multiple different web techniques and was based on a HTML5 website. The program's aim was only to demonstrate the theory in use.

The study showed that the new web techniques allowed the websites to be even more attractive, interactive and user friendly. Even though cloud services are a kind of trend, the study indicates that they suit best for the needs of big global companies. For the needs of smaller companies the traditional servers offer more efficient solutions. The problem of cloud services turn out to be especially in the expenses in the technical execution of the services, data security and in the limited management of the data used.

Subject headings, (keywords)
HTML5, JavaScript, Ajax, Azure, Cloud services

Pages	Language	URN
35 pages	Finnish	URN:NBN:fi:amk-2012061512783

Remarks, notes on appendices

Tutor
Jukka Selin

Bachelor's thesis assigned by
Mikkeli University of Applied Sciences

SISÄLTÖ

1 JOHDANTO

Opinnäytetyöni tilaajana toimi Mikkelin ammattikorkeakoulu ja työni tutkimusongelmana oli tutkia millaiseen suuntaan internet on kehittymässä sekä mitkä voisivat olla tulevaisuuden kannalta olennaisia tekniikoita internetohjelmoinnissa. Koska käytännössä ei ollut mahdollista edes koettaa käsitellä kaikkia mahdollisia tekniikoita, joita internetohjelmoinnissa käytetään, työni edetessä valitsin keskittyä erityisesti HTML5:n yleistymisen myötä aukeaviin mahdollisuuksiin.

HTML5:n uusista elementeistä mielestäni mielenkiintoisin oli canvas-elementti, sekä sen tuomat mahdollisuudet nettiohjelmoinnissa. Elementin sisältöä pystytään päivittämään esimerkiksi JavaScriptin avulla. Myös JavaScriptin käsitteleminen työssäni oli loogista, sillä se on muodostunut standardiksi verkko-ohjelmointi kieleksi vuosien varrella ja sen mahdollistamat Ajax-tekniikat ovat oleellinen osa nykyaikaisia verkkosivustoja.

Tutkimusongelman teoreettisen selvityksen lisäksi tein ohjelman, jossa testataan useiden eri verkkotekniikoiden yhdistämistä HTML5 -pohjaisessa verkkosivustossa. Ohjelmalla luodaan käyttäjän selaimeen silmukka, jonka avulla käyttäjän ruudulle tuodaan jatkuvasti päivittyvää materiaalia. Ohjelman oli tarkoitus toimia myös interaktiivisesti, jolloin sen käyttöliittymää voisi käyttää useampi käyttäjä samaan aikaan. Tämä interaktiivisuus mahdollistettiin nimenomaan JavaScriptin Ajax-käskyillä.

Nykypäivänä it-alalla puhutaan lisäksi paljon pilvipalveluista. Siksi opinnäytetyössäni keskityn myös selvittämään, mitä oikeastaan pilvipalvelut pohjimmiltaan ovat ja avartaa ajatusta, millaisissa tilanteissa pilvipalveluita yritysmaailmassa voisi hyödyntää. Pyrin myös osoittamaan mahdollisia ongelmakohtia, joita pilvipalveluiden käyttöönottoon voi liittyä. Koska erot pilvipalveluissa eri palveluntarjoajien välillä ovat suuria, opinnäytetyön tilaajan toiveiden mukaisesti keskityn opinnäytetyössäni erityisesti Microsoft Azure -pilvipalveluihin.

2 PILVIPALVELUT

Nykyaikana yritykset tarvitsevat joustavuutta toiminnassaan ja niiden on kyettävä reagoimaan nopeasti markkinoilla muuttuviin tilanteisiin. Pilvipalvelut ovatkin vastaus yritysmaailman kasvaneisiin it-tarpeisiin. Verkkopalveluissa pilvipalvelut eivät välttämättä aina ole paras mahdollinen toimintatapa, mutta varsinkin suuryrityksissä saadut edut pilvipalveluista voivat olla huomattavat. (Hurwitz ym. 2009, 7.) Joustavuutta tarjoaa pilvipalveluille ominainen skaalautuvuus, se tarkoittaa että palveluun voidaan ohjata tarvittaessa enemmän resursseja joko nopeasti, tai jopa täysin automaattisesti. Tämä on tärkeää esimerkiksi jos palveluntarjoaja ei voi ennustaa palvelun suosiota ennalta. (Hurwitz ym. 2009, 10.)

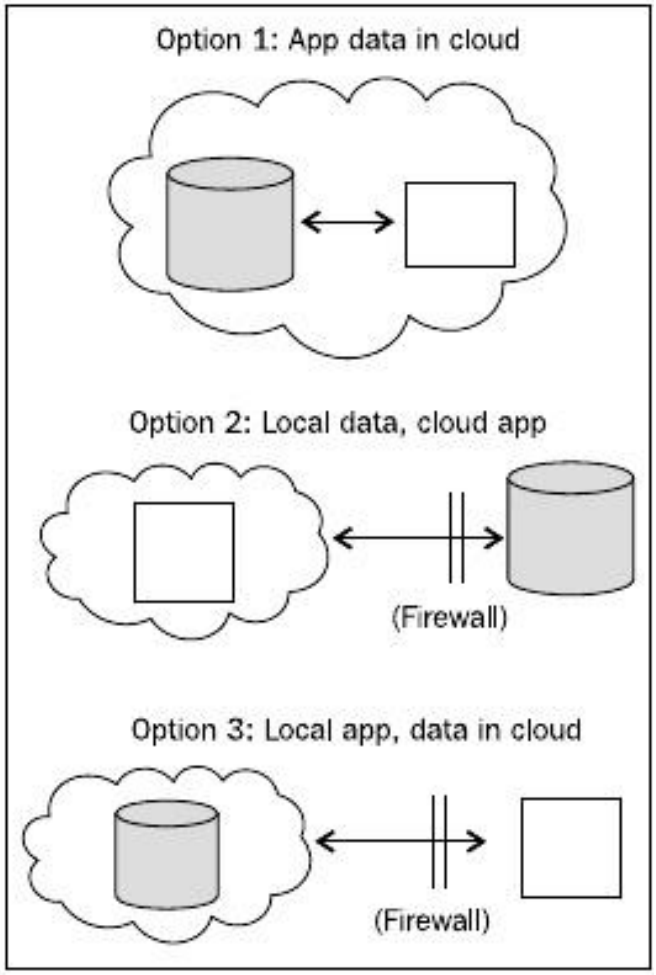
Pilvipalvelut ovat vastaus markkinoilla kasvaneeseen kysyntään it-palveluista ja niitä tarkastellessa täytyy pitää mielessä useat toimijat, joiden tarpeista palvelut ovat kehittyneet. Näitä toimijoita ovat esimerkiksi suuret it-kauppiaat, heidän yhteistyökumppanit sekä tämänhetkiset it-markkinajohtajat, jotka käyttävät useita eri pilvipalveluita. Pilvipalvelut voivat myös olla yksityisesti firman sisäisessä käytössä, julkisia tai yhdistää sekä julkista että yksityistä palvelua. Pilvipalvelut eivät siis kaikissa tapauksissa ole asiakkaille suoraan suunnattuja palveluita. (Hurwitz ym. 2009, 8.)

Pilvi itsessään on suuri kokonaisuus laitteistoja, verkkoja, tallennustilaa, ohjelmistopalveluita ja rajapintoja, joita yritykset voivat vuokrata käyttöönsä. Palvelun loppukäyttäjän ei tarvitse tietää mitään palveluiden alla olevasta tekniikasta tai laitteistoista. Pilvipalvelut ovat siis vuokrapalveluita, joissa vuokraavan yrityksen johto hallinnoi tietoa jota pilvessä liikkuu ja palvelun tarjoaja pitää taas huolen it-laitteiston toiminnasta ja ylläpidosta. (Hurwitz ym. 2009, 9.)

2.1 Ohjelmistot pilvipalveluissa

Pilvipalvelu yksinkertaisimmillaan tarkoittaa sisäisessä verkossa toimivan ohjelman siirtämistä osin tai kokonaan ulkoiseen verkkoon. Pilvipalvelun vuokraaja vuokraa käyttöönsä laitteistoa ja ohjelmistoa ja palvelun tarjoaja huolehtii siitä, että vuokraaja saa halutessaan palvelulleen lisää laskentatehoa, laitteisto palautuu virhetilanteista automaattisesti, datan synkronoinnista ym. Vuokraaja siis ulkoistaa koko ylläpidon, näin usein saadaan kustannustehokkaasti parempaa tietoturvaa sekä säästetään kustan-

nuksissa, kun omaa laitteistoa ei tarvitse huoltaa ja ylläpitää. Kuten kuvassa 1 nähdään palvelun vuokraaja voi itse päättää, haluaako ohjelmistostaan vain osan toimivaksi pilvipalveluiksi vai kokonaisen ohjelmiston pilveen. (Dudley 2010, 8 - 10.)



KUVA 1. Ohjelmisto voi toimia joko kokonaan, tai vain osittain pilvipalveluna (Dudley 2010, 8.)

2.2 Paketoidut pilvipalveluiden mallit

Pilvipalvelut markkinoidaan palvelupaketeissa, jotka voidaan jaotella infrastruktuuri-, alusta- ja ohjelmisto -palveluiksi. Yksinkertaistettuna infrastruktuuri palvelutaso tarjoaa palvelintilaa sekä laskentaresursseja, joita esimerkiksi it-ohjelmatalot voivat käyttää omissa ohjelmissaan. Alusta palvelutasolla on valmiina kehitysympäristö, jolle pilvipalvelu rakennetaan. Ohjelmisto tasolla yritykselle taas tarjotaan suoraan jokin valmis palvelu pilvestä. (Hurwitz ym. 2009, 18.)

Infrastructure as a Service (IaaS)

Infrastruktuuri palveluna tarkoittaa, että palveluntarjoajalta vuokrataan käyttöön serverit, verkkotekniikka, tallennustila ja muu laitteisto palveluna. IaaS -palvelutasoon kuuluu usein myös käyttöjärjestelmä joilla palvelua voidaan hallinnoida. Palvelun vuokraajan ei siis tarvitse virittää omia servereitä, palomuureja, reitittimiä yms. eikä ylläpitää niitä. Myös tämän tason palvelut ovat tyypillisesti hyvin skaalautuvia, asiakas siis saa käyttöönsä tarvittaessa enemmän laitteistoa, kussakin tilanteessa vallitsevien tarpeiden mukaan. (Hurwitz ym. 2009, 19.)

Platform as a Service (PaaS)

Alusta palveluna on paljon enemmän kuin pelkkä infrastruktuuri. Tällä tasolla palveluntarjoaja vuokraa koko pinon tarvittavaa laitteistoa ja ohjelmistoa, mitä tarvitaan tuottamaan kokonainen ohjelmisto, testausympäristöineen sekä toimintaympäristöineen. Tämän tason palveluntarjontaa katsotaan internetohjelmoinnin evoluution seuraavana askeleena, tulevaisuudessa ohjelmiston koko elinkaari suunnittelusta asti elää pilvessä. (Hurwitz ym. 2009, 20.)

Ohjelmiston tekeminen vuokratun alustan päälle kuitenkin asettaa myös hankalia ehtoja. Vuokratun alustan päälle rakennettu ohjelmisto ei välttämättä toimi toisen palveluntarjoajan vuokraamalla alustalla, kalliin palveluntarjoajan vaihtaminen edullisempaan voi siis koitua hyvinkin ongelmalliseksi. Tätä varten pilvipalvelimiin on kehitetty myös erityisiä avoimia alustoja. (Hurwitz ym. 2009, 20.)

Software as a Service (SaaS)

Yksi ensimmäisiä pilvipalveluiden malleja oli ohjelmistot palveluna. Ohjelmiston vuokraaja vuokraa siis sekä ohjelmiston, että kaiken sen pyörittämiseen vaadittavan laitteiston. Valmiiden ohjelmistojen vuokraamisessa on selkeä etu: Mikäli samanlaisen ohjelman käyttäjiä on jo paljon, ei ohjelmiston vuokraaminen maksa paljoa. Ohjelmiston vuokraamisen etuna on mm. se, että palvelun vuokraaja voi etsiä kannattavampia ratkaisuja toiminnan edetessä. Ohjelmistoa voidaan vaihtaa ilman suurempia kuluja. (Hurwitz ym. 2009, 23.)

Hyvä esimerkki ohjelmistosta palveluna ovat sähköpostipalvelut: Jokaisen yritysten ei ole järkevää rakentaa omaa sähköpostiohjelmistoa yrityksen käyttöön alusta asti. Oman sähköpostiohjelmiston tekeminen, siihen vaadittavan laitteiston hankkiminen ja sen ylläpitäminen on kallista ja lopputulos tuskin on yhtä hyvä kuin suurten palvelun-tarjoajien valmiit ohjelmistot. Esimerkiksi Yahoo Maililla on yli 260 miljoonaa käyttäjää, kun kaikkien käyttäjien lähettämät palvelinpyynnöt ovat teknisesti katsoen samanlaisia, voidaan kokonainen palvelinkeskus optimoida käsittelemään juuri näitä palvelinpyyntöjä erittäin tehokkaasti. (Hurwitz ym. 2009, 23.)

2.3 Palvelinkeskukset

Perinteiset palvelinkeskukset tarjoajat yrityksille serveritilaa, joilla pyöritetään esimerkiksi asiakkuuksien hallintaan ja liiketoiminnan vuorovaikutteisuuteen tarvittavia ohjelmistoja. Näistä palveluita on voinut tulla yrityksen toiminnalle vuosien saatossa elintärkeitä. Palvelinkeskuksissa olevat serverit pyörittävät aina sellaisia palveluita, kun niillä halutaan pyöritettävän. Pilvipalveluiden palvelinkeskusten luonne on lähtökohtaisesti erilainen ja tekee omalla tasollaan selvän eron siihen, mikä on pilvipalvelu ja mikä ei ole. Ohjelmistoa suunniteltaessa pilvipalveluksi, on pidettävä mielessä, että todennäköisesti pilvipalveluiden palvelinkeskus ei yksin kykene tarjoamaan kaikkea mitä asiakas palvelimeltaan vaatii. (Hurwitz ym. 2009, 49-51.)

Kun perinteisessä palvelinkeskuksessa suoritetaan tuhansia erilaisia tehtäviä ja pyöritetään tuhansia erilaisia palveluita, pilvipalveluja tarjoavat palvelinkeskukset pyrkivät käsittelemään mahdollisimman vähän erilaisia laskentataakkoja. Parhaimmillaan pilvipalvelinkeskukset voivatkin käsitellä vain yhdenlaisia tehtäviä. Tästä johtuen laitteisto pilvipalvelinkeskuksessa on täysin homogeenistä, sen vuoksi myös hallinnointityökalut ovat yhdenmukaiset. Homogeeninen ympäristö yksinkertaistaa organisointia ja toimintaa, sekä mahdollistaa sen skaalautuvuuden. (Hurwitz ym. 2009, 51-56.)

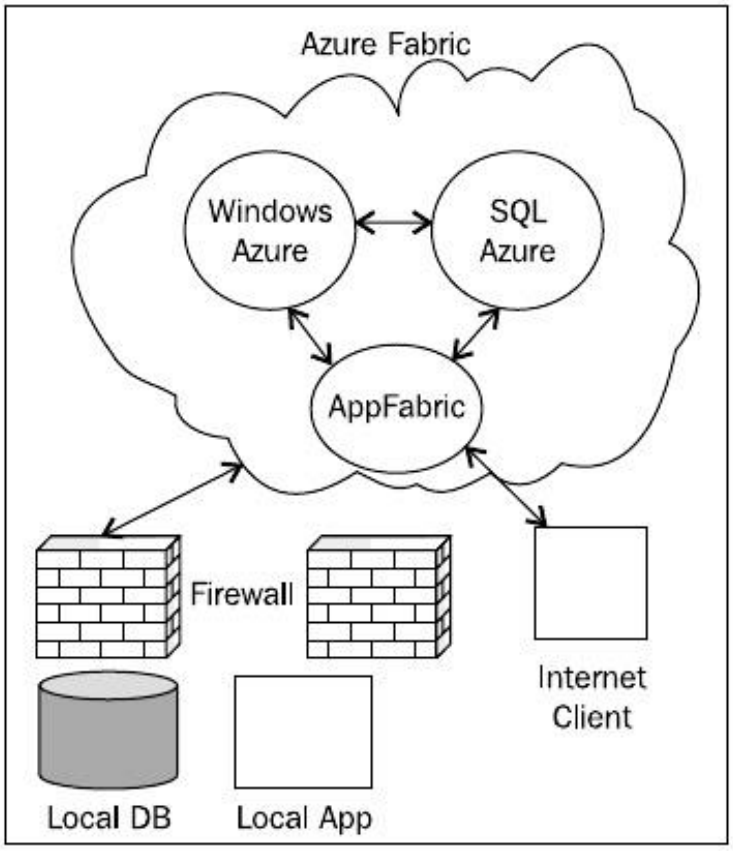
TAULUKKO 1. Vertailutaulukko perinteisestä palvelinkeskuksesta ja pilvipalveluiden palvelinkeskuksesta (Hurwitz ym. 2009, 55)

Perinteinen palvelinkeskus	**Pilvipalveluiden palvelinkeskus**
Tuhansia erilaisia ohjelmistoja	Vähän ohjelmistoja (joskus jopa vain 1)
Erilaisia laitteistoja ja ympäristöjä	Homogeenistä laitteistoa ja yhtenäinen ympäristö
Paljon erilaisia hallinnointityökaluja	Standardisoidut hallinnointityökalut
Jatkuvaa ohjelmistojen paikkailua ja päivittämistä.	Minimoitu päivitysten tarve.
Useita monenlaisia laskentakuormia	Yksinkertaiset laskentakuormat
Useita ohjelmisto arkkitehtuureita	Yksi standardisoitu ohjelmistoympäristö

3 MICROSOFT AZURE

Vuonna 2010 suurimmat pilvipalveluiden tarjoajat, olivat Google, Amazon ja Microsoft. Kaikkien näiden kolmen suurimmat tuotteet tarjotaan SaaS tason palveluna. Palvelut vuokrataan kuitenkin käytännössä aina tarpeiden mukaisesti pakettina johon sisältyy useita eri osia, riippuen minkälaista ohjelmistoa tarvitsee ja rupeaa rakentamaan. Vaihtoehtoja tutkiessa kannattaa pitää mielessä, että palveluntarjoajan vaihtaminen voi olla hyvinkin ongelmallista, koska käytetyt standardit eivät ole tuettuja kilpailijan palvelinkeskuksessa.

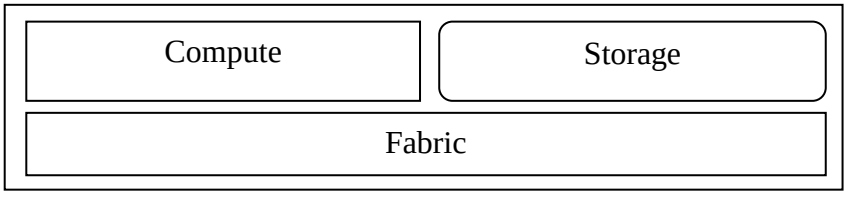
Microsoft Azure on Microsoftin tarjoama pilvipalveluiden kokonaisuus. Kuten kuvassa 2 on osoitettu, se rakentuu kolmesta osasta: Windows Azure, SQL Azure ja AppFabric. Näiden kolmen osan kokonaisuutta kuvataan nimellä Azure kehikko. (Dudley 2010, 12 - 13.)



KUVA 2. Micorsoft Azure -kehikko (Dudley 2010, 16)

3.1 Windows Azure

Kuten Microsoft Azure .-kehikko kaaviossa, Windows Azure voidaan jaotella vielä kuvan 3 mukaisesti kolmeen osaan: Laskenta-osaan jossa ajetaan ohjelmia, varasto-osaan, jonne tieto varastoidaan sekä Windows Azuren omaan kehikko-osaan, jonka varaan edellä mainitut kaksi on rakennettu. (Chapell, 2010, 3.) Nimensä mukaisesti Windows Azure osa on varsinaisesti itse Azuren käyttöjärjestelmä osio (Dudley 2010, 17).



KUVA 3. Windows Azuren osat (Chapell, 2010, 3)

Windows Azuren laskennalliset ominaisuudet

Kun ohjelmoija tekee ohjelman Microsoft Azure pilvipalveluun, ohjelmaan tehdään yhdistelmä web-rooleja ja worker-rooleja. Luoduille rooleille luodaan pilvipalvelussa aina omat virtuaalikone-ilmentymät. Ilmentymiä ei tarvitse ohjelmoijan itse varsinaisesti tehdä, vaan kun ohjelma siirretään pilveen, samalla sille siirretään XML -tiedosto, jossa palvelimelle kerrotaan tarvittavat virtuaalikoneiden ilmentymät. Näin tehdään kolmesta syystä: Varmistetaan että käsiteltävä data on aina eristyksissä yhdessä paikassa ja siirretty tieto pysyy aina turvassa. Virtuaalisten koneiden avulla pystytään varmistamaan, ettei haluttu data ole koskaan saamattomissa, mikäli jokin virhe kaataa virtuaalisen ilmentymän, automaattinen turvajärjestelmä käynnistää toisen instanssin kaatuneen tilalle. Lisäksi virtuaalikoneiden ilmentymien avulla pystytään seuraamaan tarkasti syntynyttä laskentakuormaa ja tietoliikennettä. (Chapell, 2010, 4.)

Web-roolit nimensä mukaisesti käsittelee verkon yli tapahtuvia pyyntöjä, jossa Windows Azuren sisäänrakennettu laitteisto käsittelee ne. Worker-rooli on taas tarkoitettu funktioille, joiden ei tarvitse vastata suoraan verkkopyyntöihin. On myös mahdollista että web-rooli vastaanottaa verkkopyynnön ja lähettää sen worker-roolille käsiteltäväksi. (Chapell, 2010, 4.)

Microsoft tarjoaa Visual Studioon Azuren ohjelmistoille oman kehitys-ympäristön nimeltään Developer fabric, jossa koko käyttöjärjestelmää voi testata paikallisesti ennen varsinaista käyttöönottoa. (Chapell, 2010, 4.)

Varastointi

Loogista tiedostojen varastointia varten Windows Azuressa on oma varasto-osio. Varastointiosioon voidaan tallentaa tietoa kolmessa eri muodossa: Blob, taulukko ja jono. Jokaisella eri muodolla on palvelussa oma tarkoituksensa. (Dudley 2010, 18.) Myös web-roolin ja worker-roolien pysyvät tiedostot tallennetaan varasto-osioon (Chapell 2010, 5).

Kaikkiin muotoihin tietoa tallennetaan standardeilla http:n get, put ja delete -komennoilla. Jokaiseen erimuotoiseen dataan pääsee myös käsiksi Windows Azure:n ulkopuolelta, esimerkiksi toimistokäytössä olevan ohjelmiston käyttäjä voisi valita

tallentavansa suuria videotiedostoja suoraan Windows Azureen, näitä osiota kutsutaan myös nimellä Azure asemat. (Chapell 2010, 5.)

Blob

Blob on lyhenne sanoista ”binary large object”, eli niillä tarkoitetaan suuria yksittäisiä tiedostomuotoja. Blobit tallennetaan säiliöihin, joiden koko voi olla enintään 50gb. Yhdessä säiliössä voi olla useita blob-tiedostoja. Mikäli Azure-pohjaisessa verkko-ohjelmassa näkyy esimerkiksi jokin logo, tulee logo blobi-varastosta. Tiedostoja kutsutaan standardeilla REST-kutsuilla tai .NET kirjastoiden avulla. (Dudley 2010, 18-19.)

Taulukko

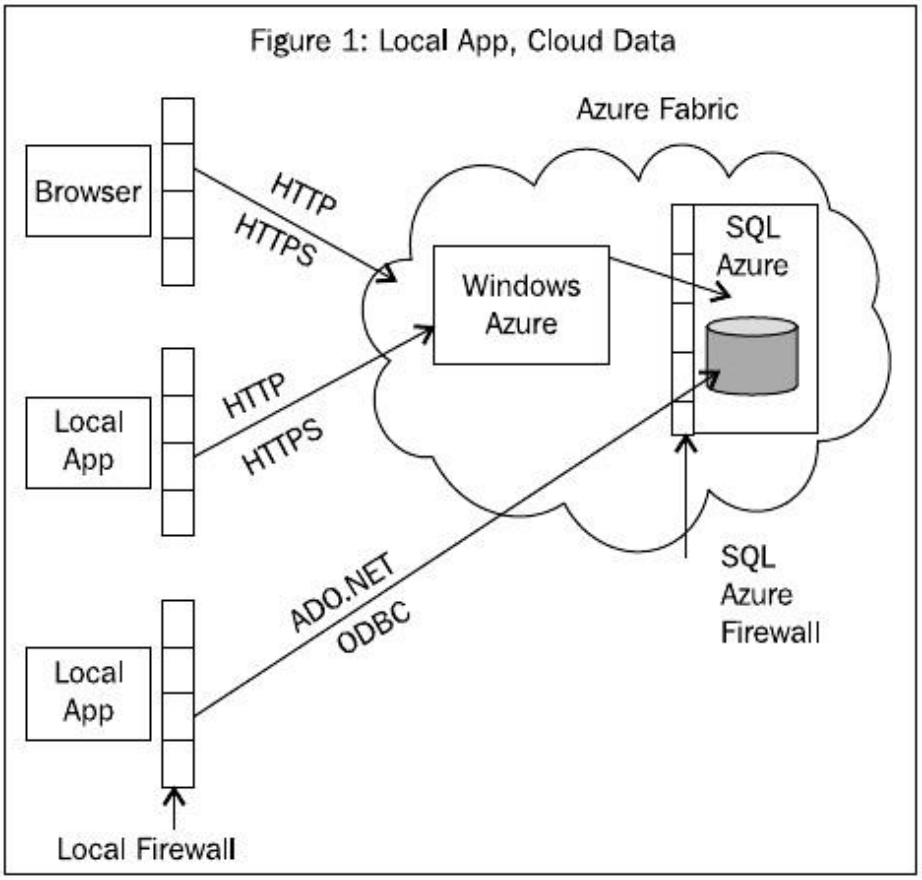
Tärkeää on huomata, että taulukkomuotoinen data ei tarkoita SQL -taulukoita eikä relaatiotaulukoita. Taulukkomuotoisena voidaan tallentaa tietoa itsenäisinä kokonaisuuksina. Taulukoihin voi tallentaa tarvittaessa vaikka terabittien kokoisia tiedostoja ja taulukoiden ominaisuuksia voidaan säätää laajasti. Taulukot luodaan ja niitä hallitaan ohjelman sisäisellä koodilla. Vaikka taulukkorakenne voi aluksi kuulostaa omituiselle, se on useassa tilanteessa todella tehokas tapa hallita suuria määriä dataa. (Dudley 2010, 19.)

Jono

Windows Azuressa jono on paikka, jonne tallennetaan pyynnöt, jotka ovat menossa käsiteltäväksi worker-roolille, joten se poikkeaa edellä esitellyistä tallennusmuodoista. Jokaisella eri Windows Azure tilillä voi olla useita jonoja, ja niihin voidaan viitata standardeilla REST-kutsuilla tai .NET kirjastoiden avulla. Myös muut käyttäjät voivat kutsua jonon dataa, mikäli käyttöoikeudet sen sallivat. (Dudley 2010, 19.)

3.2 SQL Azure

Micorosft Azure kehikon toisena osana on SQL Azure, se on varsinainen relaatiotietokanta, joka perustuu SQL Server 2008:aan. Kuten kuva 4 näyttää SQL Azurea voi käyttää muiden pilvipalveluiden tapaan joko suoraan paikallisen ohjelman kautta, taikka Azure kehikon läpi erillisellä ohjelmalla. Yleisin tapa kuitenkin on ottaa yhteys tietokantaan Windows Azuren web-roolin tai worker-roolin kautta. (Dudley 2010, 41-42.)



KUVA 4. SQL Azureen yhdistäminen (Dudley 2010, 42)

Microsoft luettelee Azure SQL:n avainvahvuuksien olevan hallittavuudessa, tiedon korkeassa saatavuudessa, palvelun skaalautuvuudessa, relaationaalisessa tietomuodossa sekä ohjelmoijille tutussa toimintaympäristössä (Dudley 2010, 42). Eroavaisuutena perinteisiin tietokantapalveluihin Azure SQL vaatii huomattavasti vähemmän huomiota hallinnointiin ja ylläpitoon, toisaalta taas ylläpidon yksinkertaistaminen ottaa pois työkaluja ja toimintatapoja, jotka ovat ohjelmoijille entuudestaan tuttuja. Azure SQL:n tietokannoille ei voi enää määrittää esimerkiksi tiedoston tallennuspaikkaa tai tiedos-

toryhmiä. Myös datan keräämiseen on totuttu tekemään asetukset palvelintasolle, tämäkään ei ole enää mahdollista, sillä erillisiä palvelimia ei enää ole. (Dudley 2010, 50.)

Kun Azure SQL otetaan käyttöön, käyttäjälle luodaan tili, jonne tehdään SQL Azure ilmentymä. Erona perinteisiin taulukoihin Azure SQL:ään pitää tehdä myös hakemistoklusteri, ennen kuin tietoa voidaan taulukoihin tallentaa. (Dudley 2010, 50 - 51.) Hakemisto klusteria tarvitaan kertomaan järjestelmän automaatiolle tarvittavat tiedot jotta käsitelty tieto voidaan tallentaa, varmuuskopioida ja tarvittaessa myös palauttaa (Microsoft SQL Azure Blog, 2010).

Hallittavuus

Suurimman edun tietokannan hallittavuudesta asiakas saa, kun asiakkaan ei itse enää tarvitse ostaa tarvittavaa laitteistoa, eikä kuluttaa resurssejaan etsimään ammattilaisia suorittamaan asennus ja huoltotöitä. Azure SQL:n hallinnointi tapahtuu kokonaan ohjelmallisesti ja pilvipalveluun tallennettua dataa pääsee hallinnoimaan käytännössä mistä tahansa päin maailmaa, vaikka yhteydet yksittäisiin palvelinkeskuksiin eivät olisi käytettävissä. Tietoturvan tietokannoille ja hallinnointipalveluille tarjoaa Azure-kehikon kolmannen osan App-kehikosta löytyvä, käyttöoikeuksien hallinnointiin tarkoitettu Access control. Kuten pilvipalveluille on ominaista, myös Azure SQL skaalautuu tarvittaessa. (Dudley 2010, 43.)

Kuitenkin SQL Azuren rajoitetussa hallittavuudessa on myös ongelmia: Ohjelmoijille tuttuja tietokantojen hallinnointiin tarkoitettuja työkaluja ei pysty käyttämään Azure SQL:n ympäristössä. Useat ohjelmistojen virheiden korjaamiseen, sekä suorituskyvyn optimointiin tarkoitetut apuohjelmat eivät toimi enää, kun tietokannat siirretään Azure SQL palveluun. (Dudley 2010, 43.)

Suurena miinuksena voidaan myös pitää CLR:n puuttumista SQL Azuresta (Dudley 2010, 43). Common Language Runtime on Microsoftin kehittämä virtuaalinen komponentti, jonka avulla .NET ohjelmia ajetaan. CLR:n avulla ohjelmoijat ovat voineet mm. käyttää eri ohjelmointikielillä toteutettuja komponentteja yhdessä (MSDN, 2012).

Korkea saatavuus

Azure SQL pystyy myös varmistamaan perinteisiä tietokantoja paremman saatavuuden tallennetulle tiedolle. Mikäli yhteys perinteiseen palvelinkeskukseen katkeaa, tieto ei ole enää saatavilla. Azure SQL:n tietokannat tallennetaan automaattisesti kolminkertaisena kolmelle eri palvelimelle. Vaikka yksittäinen palvelinyhteys katkeaisi, ei käyttäjä huomaa katkosta, vaan tiedonsiirto reititetään automaattisesti kahden muun palvelinkeskuksen kautta. Azure SQL tarjoaa myös sisäänrakennetun tietosuojan tallennetuille tiedoille. Palvelun vuokraajan ei tarvitse itse miettiä esimerkiksi varmuuskopiointia tai ongelmatilanteessa datan palauttamista. (Dudley 2010, 45 - 46.)

Vaikka data pyritään tarjoamaan käyttäjälle 100 % ajasta, mihin tahansa päin maailmaa, on kuitenkin huomioitava, ettei ohjelmoija voi olettaa, että koodissa pyyntöihin vastataan aina. Syitä datan katkasuun voi olla esim. liiallinen resurssien käyttöaste, liian pitkät tiedustelut tai transaktiot, yhteyksien avoimiksi jättäminen, tai serverillä tapahtuvat virheet. Kuten muussakin ohjelmoinnissa, koodatessa ohjelmaa joka käyttää Azure SQL:ää, täytyy aina kompensoida virheiden varalta. (Dudley 2010, 45 - 46.)

Skaalautuvuus

SQL Azure portal -työkalulla voidaan ohjata lisää resursseja Azure SQL:n käyttöön, ilman että resurssien lisäämisestä koituu käyttökatkoksia. Esimerkiksi yrityksen tuotetietokanta voi olla aluksi 1gb kokoinen, mutta viiden vuoden päästä se voi olla kasvanut 5gb kokoiseksi. Tietokannan kokoa voidaan muokata yhdellä ”ALTER DATABASE” -komennolla. Azure kehikko pitää huolen automaattisesti, että tietokanta pysyy pystyssä ja asiakasohjelman pyynnöt pysyvät järjestyksessä. Koska Windows Azuren taulukot toimivat suuriakin tiedostoja käsittelevissä tietomuodoissa hyvin , sitä voidaan sanoa todelliseksi relaationaalisen tietokannan hallintajärjestelmäksi. (Dudley 2010, 46.)

Tuttu kehitysympäristö

Microsoftin tuoteperheessä Azure SQL:ää edeltävänä tuotteena voidaan pitää Microsoft SQL Server 2008:aa. Vahvasti edeltäjäänsä pohjautuvana vaihtaessa tietokantaa SQL Serveri 2008:sta SQL Azure:n ei käyttäjän parhaimmillaan tarvitse kuin luoda pilvipalveluun uusi tietokanta ja muuttaa koodista merkkijonoa joka muodostaa yhteyden tietokantaan. Kuitenkin SQL Server 2008:n ja SQL Azure:n väleillä on eroja, joten täytyy puhua tutusta kehitysympäristöstä, ei identtisestä. Taulukoiden ominaisuudet ovat useilta osin samat: Tutut INSERT, SELECT yms. lausekkeet toimivat molemmissa tietokannoissa samalla tavalla. (Dudley 2010, 46 - 47.) Kuitenkin esimerkiksi SQL Server 2008:n järjestelmä taulukot puuttuu Azure SQL:stä kokonaan (Dudley 2010, 45).

3.3 AppFabric

Aikaisemmin .NET palveluina tunnettu Azure AppFabric on Microsoft Azuren kolmas osakokonaisuus. Nimestään huolimatta se ei ole Windows Server AppFabric:in seuraaja. Azure AppFabric tarjoaa laitteiston palveluväylän, sekä pilvipalvelun käyttöoikeuksia käsittelevät toiminnot Windows Azure ympäristössä. Käyttöoikeuksia käsitteleviä toimintoja voidaan käyttää myös erikseen jokaisen eri Microsoft Azuren osan kanssa. (Dudley 2010, 177.)

Käyttöoikeudet

Nykyään suosittuja sosiaalisia palveluita joihin käyttäjä voi kirjautua on paljon: Esimerkiksi Google, Twitter, Facebook ja WindowsLive käyttävät kaikki erilaisia tekniikoita todentaakseen käyttäjän. Microsoft Azuren liittoutettu todentaminen on uusi ja ainutlaatuinen konsepti. Käyttäjien ei tarvitse enää luoda uutta erillistä tiliä kirjautuakseen palveluun sisälle. Käyttäjät voivat tunnistautua palveluun useista jo olemassa olevasta suosituista palveluista, vaikka ne ovat erillään toimivia yrityksiä. (Dudley 2010, 178 - 179.)

AppFabric mahdollistaa, että mihin tahansa tuettuun palveluun kirjautuvalle henkilölle luodaan yksilöllinen poletti. Poletin avulla ohjelmoijan ei siis tarvitse huolehtia eri palveluista tulevien datavirtojen käsittelyä, vaan voi käsitellä näitä kaikkia kerralla

poletin avulla. Varsinkin suurille henkilömäärille julkiseen käyttöön tulevissa ohjelmissa polettien käyttö on varmasti houkutteleva vaihtoehto, mutta kuitenkin ohjelmistoa suunniteltaessa täytyy miettiä, onko palvelulle järkevämpää tehdä esimerkiksi perinteisempi ASP.NET kirjautumiseen pohjautuva ratkaisu. (Dudley 2010, 178 - 179.)

Palveluväylä

Kun käyttöön halutaan kolmannen osapuolen ohjelmistoja, niihin otetaan yhteyttä palveluväylän kautta. Väylän läpi saadaan myös automaattisesti käyttöön Microsoft Azuren tarjoama palomuuri. Väylän avulla voidaan joko välittää yksittäisiä viestejä ohjelmien välillä, taikka avata yhteys ohjelmistojen väliin. Tässä prosessissa palveluväylältä pyydetään yhteyttä, ja väylä huolehtii että yhteyksien väliin löydetään yhteinen protokolla jolla kommunikointi hoidetaan. (Dudley 2010, 196-197.)

3.4 Haasteet ja kustannukset

Pilvipalvelut tarjoavat houkuttelevia mahdollisuuksia, sekä mahdollistavat skaalautuvuutensa avulla nopean reagoinnin muuttuviin tilanteisiin. Kuitenkin Pilvipalveluiden käyttöönotossa on haasteita, sekä kustannuksia jotka täytyy ottaa huomioon, ennen kuin minkään yrityksen toimintaa ryhdytään siirtämään pilvipalveluksi. Näistä kysymyksistä suurimmat liittynee palveluntarjoajan luotettavuuteen: Voiko palveluntarjoajaan luottaa esimerkiksi arkaluontoisen tiedon tallentamisessa pilvipalvelimelle? (Chapell 2010, 9-11.)

Pilvipalveluiden kehitystä myös rajoittavat erilaiset määräykset ja säännöt, valtion omistuksessa olevat yritykset eivät voi käyttää pilvipalveluita joissa tietoja mahdollisesti tallennetaan muissa maissa sijaitseviin palvelinkeskuksiin. Lisäksi useita yrityksille tärkeitä ohjelmistoja on teknisesti mahdotonta toteuttaa pilvipalveluina. (Chapell 2010, 9-11.)

Rajoitettu pilvipalveluiden hallinnointi myös poistaa yrityksen käytöstä paljon työkaluja ja toiminnan mittareita. Kuitenkin on huomattavaa että pilvipalvelut ovat nuori ilmiö ja ne varmasti kehittyvät jatkossa palvelemaan entistäkin suurempaa osaa yrityksistä. (Chapell 2010, 9-11.)

Pilvipalveluiden vuokraamisesta syntyville kustannuksille on ominaista, että ne muodostuvat palveluiden käytön mukaan. Vuonna 2010 Microsoft veloitti Azure:n pilvipalveluista seuraavasti

- Laskennalliset ominaisuudet: 0,12 – 0,96 $ / tunti jokaiselta Windows Azure ilmentymältä, riippuen sen laajuudesta.
- Tiedon varastointi: 0,5 $ jokaista gigatavua kohden, kuukausittain ja 0,01 $ lisää jokaista transaktiota kohden (GET, PUT, DELETE).
- SQL Azure: 9,99 $ jokaista gigatavua kohden, kuukausittain.
- AppFabric: Käyttöoikeuksien hallintapalvelut 1,99 $ / 100 000 transaktiota ja palveluväylä 3,99 $ kuukausittain jokaista yhteyttä kohden.
- Lisäksi asiakasta laskutetaan kaistasta, joka määritellään datavirtana palvelinkeskuksesta sisään, sekä ulos. Eu:n ja Amerikan alueella hinta gigatavua kohden oli 0,10 $ sisään ja 0,15 $ ulospäin. Aasian ja Tyynenmeren alueella 0,30 $ sisään ja 0,45 $ ulos

Näiden hintojen ohella Microsoft myöntää erilaisia alennuksia tuotepaketeissaan. (Chapell 2010, 7.)

4 KEHITTYVÄT VERKKOTEKNIIKAT

Internetin sivustot perustuvat W3C-yritysten ja yhdistysten yhteenliittymän luomien standardeihin. Tunnetuin ja yleisin käytössä oleva näistä on merkkauskieli HTML. Merkkauskeli on elänyt Internetin mukana ja se on nähnyt jo suuriakin muutoksia sen alkuajoista. W3C jatkaa HTML:n kehitystä edelleen maailman muuttuvien tarpeiden mukaan. (Gosney 2003, 13.)

4.1 HTML5

Aikaisempi versio HTML:stä, HTML 4.01 ilmestyi jo vuonna 1999. Internet on muuttunut paljon vuosien varrella ja HTML5 on vielä keskeneräinen, mutta selkeästi seuraava askel internetin evoluutiossa. Vaikka HTML5 ei ole vielä valmis, useimmat uusimmat selaimet, sekä älypuhelimet tukevat jo sen uusia ominaisuuksia ja elementtejä. HTML:n osalta W3C on jo päättänyt että HTML5 tulee perustumaan HTML-,

CSS-, DOM- ja JavaScript -tekniikoihin. Näiden tekniikoiden ominaisuuksilla se pyrkii merkittävästi vähentämään nettiselaimien liitännäisohjelmien tarvetta. (w3schools.com, 2012.)

HTML5 tulee tarjoamaan internetohjelmoijan käyttöön nipullisen uusia elementtejä. Uusilla elementeillä tulee olemaan helpompaa tuoda nettisivuille esimerkiksi videoita ja ääntä. Uudet video- ja audio-elementit ovat erityisesti laiteriippumattomuuden näkökulmasta tervetulleita. Videon ja äänen tuomiseen nettisivuille aikaisemmin on vaadittu liitännäisiä, jotka eivät välttämättä toimi kaikissa selaimissa, tai esimerkiksi älypuhelimissa. Tulevaisuudessa nettisivuilla voidaan kuitenkin käyttää yksinkertaisia video- ja audio-elementtejä ja laitteen oma selain huolehtii tekniikasta, jolla video tai ääni saadaan käyttäjälle esitettyä. (W3C, 2012).

Uusista elementeistä mielenkiintoisimpana itse kuitenkin pidän canvas-elementtiä. Canvas-elementti on määrätyn kokoinen alue, johon voidaan piirtää grafiikkaa, tai muuta visuaalista sisältöä suoraan lennosta (W3C, 2012). Canvas-elementin myötä nettisivustoille on mahdollista tehdä sisältöä joka aikaisemmin oli tehtävissä vain liitännäisten avulla. Koska canvas-elementin sisältöä voidaan reaaliaikaisesti ohjelmakoodissa myös tulkita, mahdollistetaan sen avulla esimerkiksi nettisivuston käyttöliittymän suunnittelu graafisesti.

Video-, audio- ja canvas-elementtien lisäksi HTML:n seuraavassa versiossa on keskitytty tarjoamaan ohjelmoijille lukuisia uusia elementtejä joiden tarkoitus on mahdollistaa hakukoneiden tarkempi toiminta. Näissä elementeissä voidaan eritellä esimerkiksi sivustojen sisällöstä tekstissä esiintyvät kappaleet toisistaan. Lisäksi käyttöön tulee merkintätapoja joiden avulla pystytään korjaamaan esimerkiksi eri kirjoitusjärjestelmien merkistöjen välillä tapahtuvia virheitä. (W3C, 2012.)

4.2 JavaScript

JavaScript on ohjelmointikielenä nopea tapa lisätä interaktiivisuutta nettisivuille. Se on sisäänrakennettuna suoraan nettiselaimeen ja se on ainoa kieli, joka toimii suoraan miltei kaikkien selaimien kanssa. Kielillä kuten Java, Perl, PHP ja C ei pysty suoraan vaikuttamaan kuviin, kaavioihin ja ikkunoihin joita nettisivuilla on. (Thau 2006, 2.)

JavaScriptin avulla voidaan esimerkiksi vaihtaa sivuilla esiintyviä kuvia, lisätä satunnaisia tapahtumaketjuja, vaikuttaa sivuun kuluvan ajan perusteella ja dynaamisesti muuttaa mitä sivustolla tapahtuu kun vierailija tekee jotain. JavaScriptin toiminnasta Thau käyttää esimerkkiä, jossa vierailija täyttää rekisteröitymis-kaavakkeen ja JavaScriptillä tarkastetaan onko kaavake täytetty oikein. Jos kaavake on täytetty väärin, käytäjää pyydetään korjaamaan virheet ja vain oikein täytetyt kaavakkeet lähetetään selaimesta käsiteltäväksi palvelimelle. (Thau 2006, 4.)

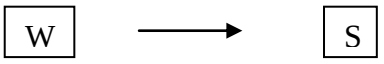
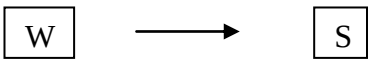
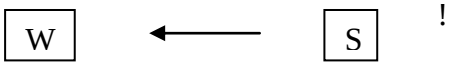
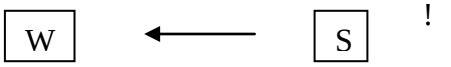
JavaScriptissä on myös rajoituksia, se ei kykene kommunikoimaan muiden laitteiden kanssa, kuin selaimen. Hyvänä puolena tässä rajoituksessa on, että JavaScript on huomattavasti nopeampi, kun kommunikointi tapahtuu vain selaimen sisällä. Mutta esimerkiksi tietokantahakuja JavaScriptillä ei voi tehdä. Mutta kuten edellisessä esimerkissä, JavaScript pystyy lähettämään käskyjä käynnistää jokin toiminto selaimen läpi. Nämä komennot käynnistetään suoraan web-serverillä ja mahdollistavat monimutkaisetkin toiminnot. Heikkoutena voidaan myös pitää sitä, että JavaScript tulkataan suoraan selaimessa, se voi siis myös käyttäytyä eri selaimissa hieman eri tavoin. Tämä on tuttu ongelma web-ohjelmoinnissa. (Thau 2006, 7-8.)

Ajax

Ajax on lyhenne sanoista Asynchronous JavaScript and XML. Sen avulla on mahdollista tehdä nettisivustoja jotka toimivat enemmän kuten stand-alone -ohjelmat. Tunnetuimpana Ajax-ohjelmana Thau (2006, 262) pitää Google mapsia. Suuri karttapalvelu toimii netin yli käyttäjän ruudulla, ilman että käyttäjän tarvitsee välillä päivittää sivua. (Thau 2006, 262). Nykyään Ajax -lyhenne kuitenkin viittaa ohjelmointimalleissa tekniikkaan, joissa JavaScript lähettää ja vastaanottaa tietoa web-palvelimelta ilman, että koko sivua ladataan uudestaan (Wilton, McPeak 2010, 491).

Ajaxin toiminta voidaan jakaa kolmeen osaan. Ensimmäisenä käyttäjä aktivoi tapahtuman nettisivulla, joka lähettää yhden tai useamman samanaikaisen pyynnön serverille. Sillä välin kun serveri suorittaa pyyntöä, JavaScriptin ohjelma jatkaa toimintaa tavalliseen tapaansa, ja käyttäjä voi edelleen käyttää ohjelmaa. Ja lopuksi kun kaikki pyynnöt ovat suoritettu serverillä, voidaan takaisin palautettu data käyttää päivittämään sivusto. (Thau 2006, 263.)

TAULUKKO 2. Vertailutaulukko Ajaxilla ja perinteisellä tavalla toteutetusta kommunikoinnista serverin kanssa (Thau 2006, 264)

Perinteinen	Ajax
 1. Käyttäjä painaa ”lähetä” ja lähettää kaavion serverille.	 1. Tekstikentästä poistuminen lähettää serverille käsittelypyynnön.
 2. Käyttäjä odottaa kun serveri käsittelee dataa.	 2. Serveri käsittelee dataa ja käyttäjä jatkaa nettisivun käyttämistä.
 3. Serveri on valmis ja se lähettää vastauksen takaisin.	 3. Serveri on valmis ja se lähettää vastauksen takaisin, käyttäjää ei vieläkään keskeytetä.
 4. Sivusto latautuu uudestaan ja käyttäjä voi jatkaa.	 4. Sivusto päivittyy automaattisesti, ilman että käyttäjää on keskeytetty kertaakaan.

Ajaxilla toteutetulla kommunikoinnilla selain tekee pyynnön serverille, ilman että käyttäjä huomaa pyyntöä. Selaimen ikkunan kulmassa oleva ”lataa” ikoni ei pyöri, ja käyttäjä pystyy käyttämään selainta normaalisti. Kun serveri on käsitellyt informaation, se palauttaa halutun datan ilman uudelleenlatausta, jolloin koko prosessi voidaan suorittaa ilman että käyttäjää keskeytetään käyttämästä sivua hetkeksikään. (Thau 2006, 264.)

JavaScript frameworks

JavaScriptillä ohjelmoitaessa huomaa usein törmäävänsä useissa eri projekteissa samoihin ongelmiin, esimerkiksi selainten yhteensopivuuden kanssa. Näihin ongelmiin voidaan tehdä ratkaisuja varsinaisen ohjelmakoodin ulkopuolelle, jolloin samaa ratkaisua voidaan käyttää useammassa projektissa. Ammattilaiset ovat julkaisseet näitä kehysratkaisuja ilmaiseksi julkiseen käyttöön ja näistä osa on saanut melkoisen mää-

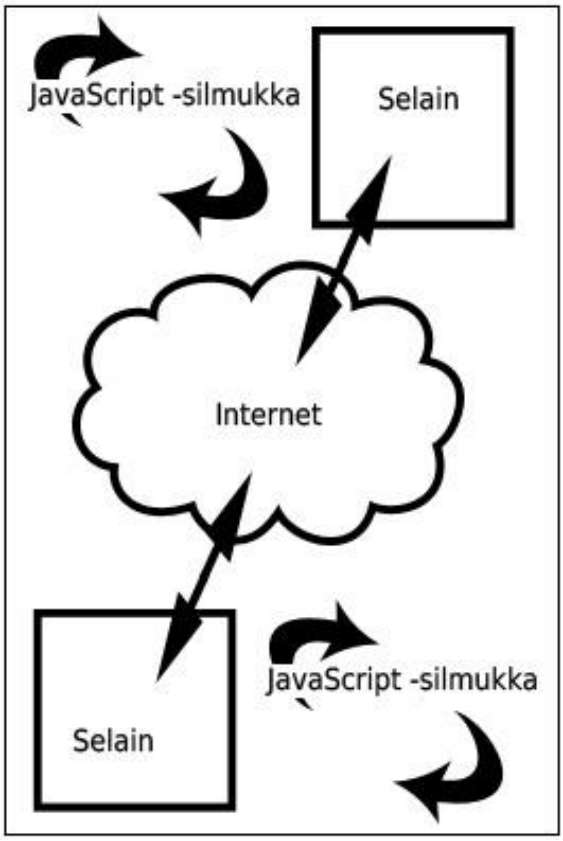
rän käyttäjiä ympäri maailman, näistä tunnetuimpia ovat esimerkiksi jQuery, Prototype ja MooTools. (Wilton, McPeak 2010, 527.) Esimerkiksi Ajax -pyynnöt eivät varsinaisesti ole kovin vaikeita, mutta jQueryn avulla nekin voidaan hoitaa vielä nopeammin ja vähemmillä riveillä koodia (Wilton, McPeak 2010, 551).

5 VAATIMUSMÄÄRITTELYT JA SUUNNITTELU

Tutkimusongelmana oli luoda ohjelma joka osoittaa HTML5:n uusien elementtien avulla, kuinka internetin lukuisien sivustojen toimintatapa voisi tulla muuttumaan muutaman vuoden kuluessa. Lähtökohtana oli että ohjelman pohjana toimi HTML5-elementein rakennettu nettisivu, joka sisältää canvas-elementin. Canvas-elementin tuli demonstroida kuinka HTML5 mahdollistaa jatkuvasti päivittyvän sisällön tuomisen nettisivuille.

Koska JavaScriptin ohjelmalogiikka toimii selaimessa, se mahdollistaa canvas-elementin sisällön päivittyvän nopeastikin. Lisäksi JavaScript on yleisesti tuettu miltei kaikissa selaimissa. HTML5 pyrkii laiteriippumattomuuteen, joten ohjelman pitäisi toimia kaikilla HTML5:ttä ja JavaScriptiä tukevilla laitteilla. Opinnäytetyössäni kuitenkin keskityn vain tutkimaan ohjelman toimintaa Internet explorer-, Google chrome- ja Firefox -selaimin, sekä Nokia Lumia 800 -puhelimen selaimella.

Päätin työssäni tehdä JavaScriptillä silmukan, joka päivittyy haluamani päivitysnopeuden mukaan. Silmukan sisältö päivitetään canvas-elementin sisälle. Sisältö täytyi olla samanaikaisesti useamman käyttäjän hallittavissa, mahdollisimman reaaliaikaisesti. Käyttöliittymään tarvittiin siis myös kuvan 5 mukainen logiikka, jossa useampi samanaikainen käyttäjä voi käyttää sovellusta yhdessä. Tämä käytännössä tarkoitti, että nettisivuston täytyi kyetä kertomaan kaikille sivustoa näyttäville selaimille dataa omasta selaimesta, sekä lukemaan muiden selaimien lähettämä tieto, sekä tulkitsemaan se.



KUVA 5. Ohjelman suunnitelmanmukainen toiminta

6 TOTEUTUS

Tutkimusongelmaa ratkaistaessa tein ensin yksinkertaisen HTML sivun, jonne lisäsin canvas-elementin. Canvas-elementti luodaan yksinkertaisella komentorivillä:

<canvas id='c'></canvas>.

Canvas-elementin luomisen jälkeen, sivusto ottaa ajoon JavaScrtiptillä tehdyn osion joka pyörittää selaimessa jatkuvasti toistuvaa silmukkaa. Silmukan sisältö päivitetään käyttäjälle reaaliaikaisesti canvas-elementtiin, silmukassa tapahtuvan ohjelmalogiikan avulla.

6.1 Ohjelman JavaScript -osion toteutus

Canvaksen luomisen jälkeen HTML -tiedostossa kerrotaan selaimelle, että tässä kohdassa halutaan ajaa JavaScript koodia. JavaScript otetaan käyttöön script -tagien avulla:

```
<script src="game.js"></script>.
```

JavaScript game.js -tiedostossa täytyi ensimmäiseksi määritellä canvas-elementille korkeus ja leveys, sekä muuttuja canvas-elementin sisällölle.

```
var  c = document.getElementById('c');
var  width = 500, height = 500;
var ctx = c.getContext('2d');
c.width = width;  c.height = height;
```

Kun canvas-elementti on saatu tehtyä, täytyi seuraavana luoda itse silmukka, josta muuttuvaa sisältöä käyttäjälle näytetään. Silmukan tekemiseen tarvitaan myös apumuuttuja jolla voidaan määritellä silmukan pyörimisen nopeus. Esimerkissä päivitetään silmukan sisältö 30 kertaa sekunnissa.

```
var gLoop;
var GameLoop = function(){
  gLoop = setTimeout(GameLoop, 1000 / 30);
}
GameLoop();
```

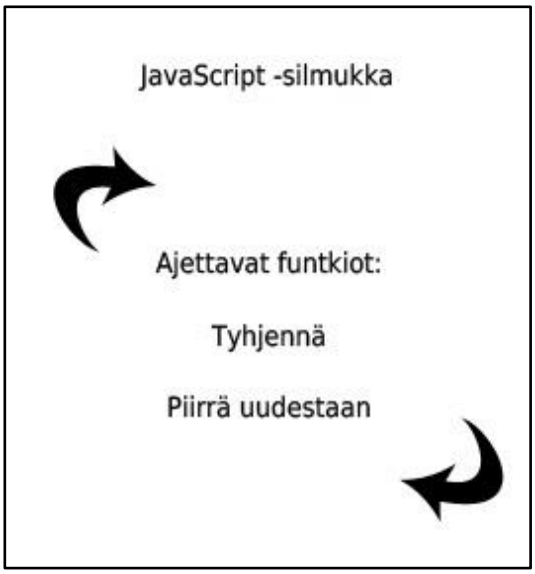
Tässä vaiheessa on tärkeää oivaltaa, että canvas-elementtiin jo piirrettyjä objekteja ei voi enää muokata tai liikuttaa. Joten vanha sisältö pitää aina ensin pyyhkiä, jonka jälkeen päivitetty sisältö piirretään elementille uudestaan. Siksi silmukan toimiminen vaatii, että se tyhjennetään jokaisella kierroksella. Kuvan 6 mukaisesti tyhjentämisen jälkeen päivitetty sisältö piirretään silmukasta uudestaan canvas-elementille. Tyhjentäminen käytännössä tapahtuu niin, että koko elementti peitetään yhdellä värillä. (Budzynski 2010.)

```
var Clear = function(){
 ctx.fillStyle = '#aabfe6';
 ctx.beginPath();
 ctx.rect(0, 0, width, height);
 ctx.closePath();
 ctx.fill();
```

```
}
```

Seuraavana silmukassa täytyi piirtää käyttäjälle pieni laatikko, käyttäjän laatikon piirtämiseen määriteltiin vielä muuttujat jotka määrittävät käyttäjän laatikon paikan. Näitä muuttujia muuttamalla, käyttäjän laatikkoa voidaan myöhemmin liikuttaa.

```
var DrawPlayer1 = function () {
  ctx.fillStyle = '#00FF12';
  ctx.beginPath();
  ctx.rect(paikkaX, paikkaY, 20, 20);
  ctx.closePath();
  ctx.fill();
} // DrawPlayer1
```



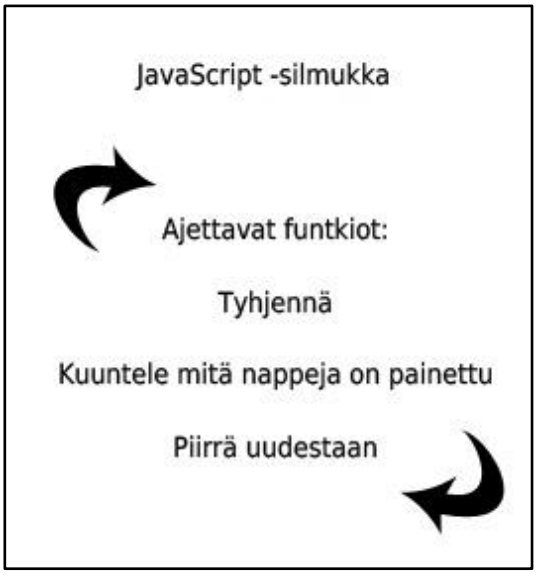
KUVA 6. JavaScript silmukan alkufunktiot

Kun silmukka ja piirtäminen canvas-elementtiin JavaScriptillä toimivat, tein ohjelmaan selaimelle tapahtumakuuntelijat näppäimistöön, jonka avulla käyttäjä voi liikuttaa laatikkoansa. Ohjelman toiminnan kannalta on tärkeää että kuuntelijoiden toiminta koodiin tehdään erillisinä funktioina. Funktioiden avulla voidaan tutkia onko käyttäjä esimerkiksi painanut jotain nappia, samalla kun canvas-elementtiin piirretään uutta sisältöä (Moore 2011).

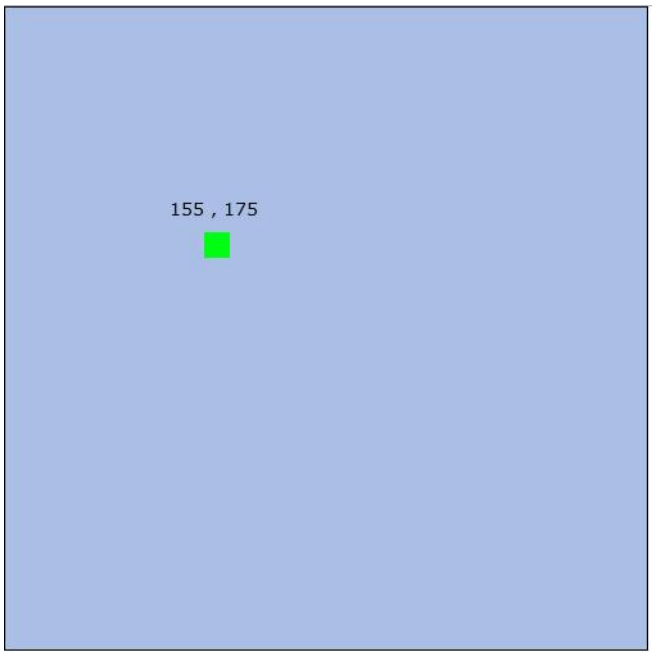
Kuuntelijoiden funktiot ja niiden muuttujat tein näin:

```
var  rightKey = false, leftKey = false, upKey = false, downKey = false;
document.addEventListener('keydown', keyDown, false);
document.addEventListener('keyup', keyUp, false);

function keyDown(e) {
  if (e.keyCode == 39) rightKey = true;
  else if (e.keyCode == 37) leftKey = true;
  if (e.keyCode == 38) upKey = true;
  else if (e.keyCode == 40) downKey = true;
}

function keyUp(e) {
  if (e.keyCode == 39) rightKey = false;
  else if (e.keyCode == 37) leftKey = false;
  if (e.keyCode == 38) upKey = false;
  else if (e.keyCode == 40) downKey = false;
}
```

Kuten mainitsin, ohjelman toiminnan kannalta on tärkeää, että kaikki toiminnallisuudet ovat tehty funktioihin, sillä ohjelma logiikka vaatii kuvan 7 mukaisesti jokaista funktiota kutsuttavan halutussa järjestyksessä JavaScriptin silmukassa.



KUVA 7. JavaScript silmukan funktiot ohjelman toisessa vaiheessa

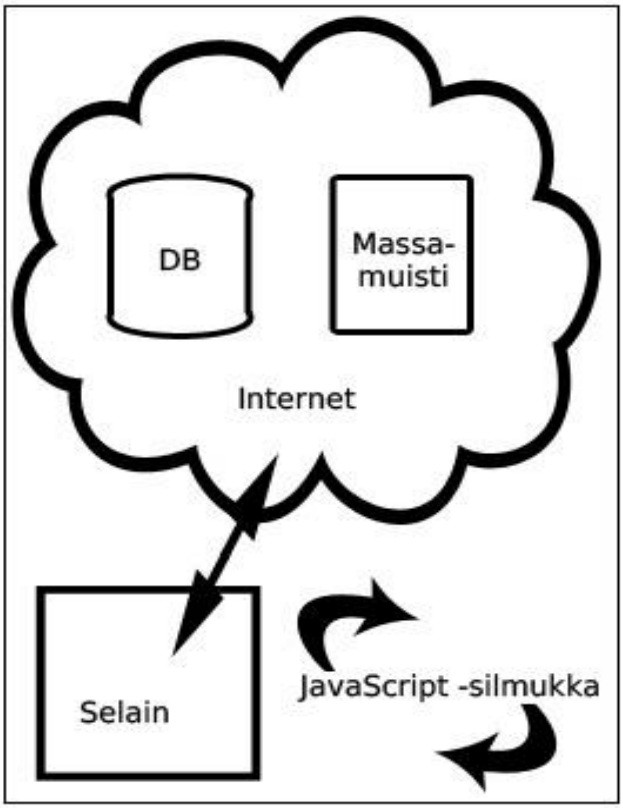


KUVA 8. Kuvakaappaus ohjelmasta

6.2 Kommunikointi muihin selaimiin Ajaxin avulla

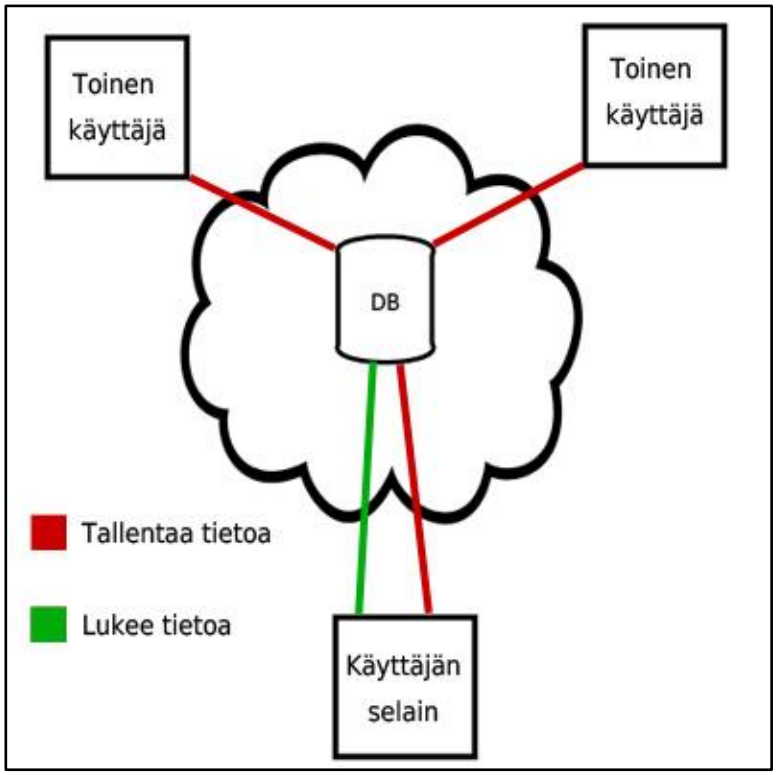
Seuraavana ongelmana oli selvittää, kuinka selain voisi kommunikoida toisten selaimien kanssa. Ongelmaa pohtiessani tulin lopputulokseen, että JavaScriptissä ajettavassa silmukassa täytyy tehdä Ajax -pyyntö selaimelle, joka puolestaan pyytää kuvan 9 mukaisesti palvelinta tallentamaan halutun datan. Kaikkien selainten serverille tallentama tieto luetaan uudestaan ja tulkitaan selaimessa, jolloin muiden käyttäjien tiedot saadaan tuotua myös omaan selaimeen.

JavaScriptin avulla muodostetaan tehtyyn silmukkaan funktioita, jotka käskevät selainta antamaan käskyjä serverille, esimerkiksi tietokantaan tai massamuistiin. Ratkaisin ongelman selaintenvälisestä kommunikoinnista tekemällä erilliset funktiot, jotka pyytävät selaimen välityksellä serveriä tallentamaan käyttäjän oman laatikon paikan MySql tietokantaan tai serverin tarjoamaan massamuistiin.



KUVA 9. Tiedon tallentaminen selaimesta palvelimelle

Ensimmäisenä ongelmana oli selvittää, kuinka selain saadaan pyytämään tietoa palvelimelta sekä kuinka tuo tieto saadaan takaisin käyttöön. Päädyin ratkaisemaan ongelman tekemällä oman SQL tietokannan jonne kaikkien käyttäjien laatikon paikan tiedot tallennetaan. Kuvan 10 mukaisesti tietokannasta tiedot voidaan Ajax-käskyin tallentaa ja hakea selaimelle. SQL tietokantaan tallennettua tietoa ei kuitenkaan voi tallentaa ja lukea samanaikaisesti yhdellä pyynnöllä, siksi täytyi tehdä kaksi funktiota: Ensimmäinen tiedon tallentamiseen, sekä toinen tiedon lukemiseen. Jotta tietoa pystyi tallentamaan käyttäjäkohtaisesti, tein myös JavaScriptillä muuttujan joka arpoo numeron käyttäjälle välillä 1 - 9999. Arvottu numero toimii käyttäjän tunnisteena SQL tietokannassa.



KUVA 10. Tietokulku tietokantaan käyttäjän selaimen näkökulmasta

SQL -tietoa tallentavan ja lukevan funktion toteutin erillisillä .php -tiedostoilla. Tietokannasta lukevaan funktioon tein myös pienen toiminnallisuuden joka tarkastaa onko omalla id -numerolla jo tallennettu tietokantaan tietoja. Mikäli on, ajaa funktio update -lauseen, insert -lauseen sijaan.

```
<?php
$id = $_GET["id"]; $x = $_GET["x"]; $y = $_GET["y"];

$laskuri = 0;
require_once "globals.php";

// Haetaan kannasta tunnuslukua vastaava rivi
$sql="SELECT * FROM pelipaikat WHERE id = '".$id."'";
$result=mysql_query($sql);
while ($row=mysql_fetch_array($result,MYSQL_ASSOC))
   //Jos tulosjoukossa on jotain (samalla id:llä) ajetaan update.
        if (mysql_num_rows($result)){
        mysql_query("UPDATE pelipaikat SET paikkaX='".$x."', paikkaY='".$y."' WHERE id = '".$id."'");
```

```
$laskuri++;
        echo "omat paikkatiedot päivitetty";
        } // if

        if ($laskuri == 0) {
        mysql_query("INSERT INTO pelipaikat (Id, paikkaX, paikkaY)
VALUES ($id, $x, $y)");
        echo "omat paikkatiedot lisätty";
        } // jos tietokannassa ei ollut jo tietoja.

?>
```

Tietokannan lukevan .php tiedoston koodi:

```
<?php
// Ajaxilta tuleva id Get - tarvitaan koska haetaan muiden palikat (ei
omaa).
$id = $_GET["id"];
require_once "globals.php";
// valitaan kaikki missä id ei oo meidän id.
$sql="SELECT * FROM pelipaikat WHERE id != '".$id."'";
$result = mysql_query($sql);
while($row = mysql_fetch_array($result))
  {
    // ja palautetaan yksinkertaisena CSV:nä.
    echo $row['paikkaX'];
    echo ", ";
    echo $row['paikkaY'];
    echo "; ";
  }
mysql_close($con);
?>
```

JavaScriptissä .php tiedostoa kutsutaan Ajax -komennoilla. Tiedot palauttava funktio myös tulkitsee vastauksen jaettuVastaus-nimiseen taulukko muuttujaan, joka mahdollistaa palautetun datan käyttämisen koodissa myöhemmin.

```
var ReturnValues = function () {

// tökätään php:tä josta saadaan muiden palikoiden paikat.
   xmlhttp.open("GET","mySqlResult.php?id="+id,true);
   xmlhttp.send();

// kun readystate muuttuu valmiiksi (arvo 4)
 xmlhttp.onreadystatechange=function()
 {
 if (xmlhttp.readyState==4 && xmlhttp.status==200)
   {
 //  document.getElementById("myDiv").innerHTML="Kavereiden palikat haettu.";
         jaettuVastaus = xmlhttp.responseText.split(";");
         // tässä vaiheessa siellä on jaettuVastaus[indexi] jossa on aina 124, 198 muotoinen palikka.

   } // if (tässä toiminta)
 } // onreadystatechange
} // ReturnValues
```

JavaScriptissä Ajaxilla SQL:lle tietojen lähettämisen funktio:

```
var SendValues = function () {

xmlhttp2.open("GET","mySqlInsert.php?id="+id+"+&x="+paikkaX+"&y="+paikkaY,true);
   xmlhttp2.send();
   // kun readystate muuttuu valmiiksi (4)
   xmlhttp2.onreadystatechange=function()
   {
```

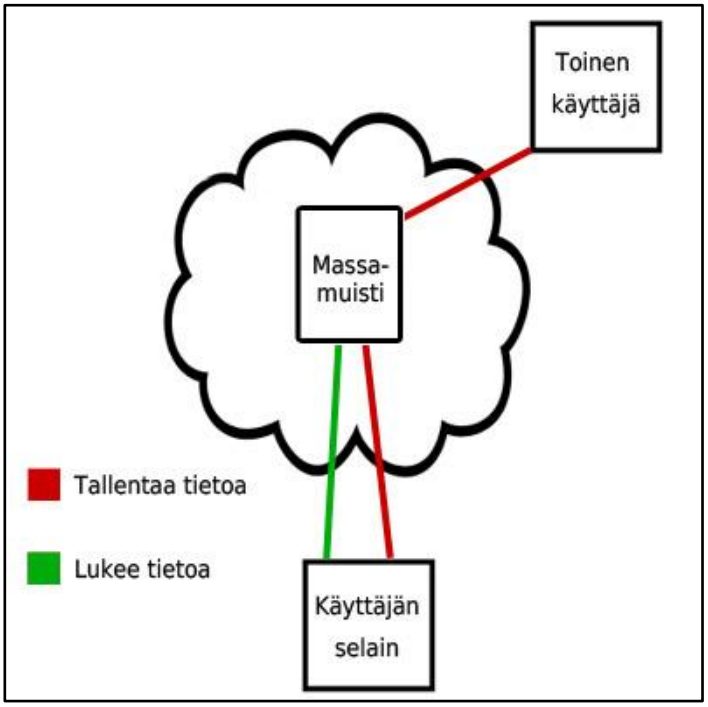
```
    if (xmlhttp2.readyState==4 && xmlhttp2.status==200)
      {
      } // if (tässä toiminta)
  } // onreadystatechange
} // SendValues
```

30 kertaa sekunnissa kysytyt Ajax käskyt eivät kerkeä ikinä muuttumaan readyState -arvoltaan ”valmiiksi”. Siksi kuvan 11 mukaisesti Ajax-käskyjen funktioille oli tehtävä ajastimet, jotka pyytävät tallentamaan ja lukemaan datan vain hetkittäin. Tämän ongelman ratkaisin lisäämällä silmukkaan funktioihin laskurin, jonka avulla Ajax -käskyt lähetettiin viiveellä.

```
  // laskurin ollessa 30 lähetetään oman paikan tiedot
  if (laskuri == 30) {
  SendValues();
  } // jos laskuri on 30

  // lskurin ollessa 60 kysytään muiden tiedot
  if (laskuri == 60) {
  ReturnValues();
  laskuri = 0;
} // jos laskuri on 60
```

KUVA 11. JavaScript silmukan funktiot 3

Ohjelmakoodia testatessa havainnoin, että Ajax käskyjä voi lähettää ja lukea tietokannasta noin 0,7 sekunnin välein ilman ongelmia. Tein ohjelmasta myös kuvan 12 mukaisen version, joka tallentaa halutun datan suoraan serverin massamuistiin.



KUVA 12, CSV -tekniikan tiedonsiirto käyttäjän selaimen näkökulmasta.

Pyrin parantamaan koodin toimintanopeutta tallentamalla käyttäjän laatikon paikan suoraan serverillä olevaan massamuistiin. CSV -tekniikkaan perustuvaan koodiin tein muutoksen jossa käyttäjä valitsee sivulle liittyessään id -numerokseen joko arvon 1 tai 2. Tuon arvon mukaan valitaan tiedosto jonne serveri tallentaa tiedon. Vastaavasti kun dataa luetaan, luetaan vain toisen käyttäjän id -arvon mukainen tieto.

Lisäksi Ajax käskyillä kutsuttavat .php -tiedostojen koodit oli kirjoitettava uudelleen. CSV -muotoisen tiedon tallentamiseen käytetty koodi:

```
<?php
$id = $_GET["id"]; $x = $_GET["x"]; $y = $_GET["y"];
if ($id == 2) {
//$file = "http://127.0.0.1/home/oppariDemoCsv/2.csv";
$file = $_SERVER['DOCUMENT_ROOT'] . '/misc/oppariCsv/2.csv' ;
}
else {
// $file = "http://127.0.0.1/home/oppariDemoCsv/1.csv";
$file = $_SERVER['DOCUMENT_ROOT'] . '/misc/oppariCsv/1.csv' ;
} // else
// $file = $_SERVER['DOCUMENT_ROOT'] . '1.csv' ;

$fh = fopen($file, 'w') or die("can't open file");
$data = $x.", ".$y.";";
fwrite($fh, $data);
fclose($fh);
?>
```

Ja CSV -muotoisen tiedon lukemisen .php tiedoston koodi:

```
<?php
$id = $_GET["id"];
if ($id == 1) {
$file = $_SERVER['DOCUMENT_ROOT'] . '/misc/oppariCsv/2.csv' ;
}
else {
```

```
$file = $_SERVER['DOCUMENT_ROOT'] . '/misc/oppariCsv/1.csv' ;
} // else
// $file = $_SERVER['DOCUMENT_ROOT'] . 'data.csv' ;
$data = file_get_contents($file);
echo $data;
?>
```

Datan tallentaminen suoraan palvelimen massamuistiin, tietokannan sijaan ei kuitenkaan osoittautunut merkittävästi paremmaksi, tai nopeammaksi tekniikaksi. Ajax-käskyt pystyin uusimaan nyt noin 0,6 sekunnin välein. Päinvastoin tietokantaan tallennettu tieto on paremmin turvassa, sekä mahdollistaa suoraan käsitellä kaikkia käyttäjiä joita sivustolla on.

6.3 Ongelmakohdat

Tutkimusongelmaa ratkaistaessa tuli vastaan useita ongelmakohtia. Ongelmia kohtasin erityisesti liittyen internetohjelmoinnissa yleisesti tiedostettavaan ongelmaan virheilmoituksista: Mikäli koodissa oli yksikin virhe, JavaScriptin silmukka ei lähtenyt käyntiin. Ilman vikailmoitusta ongelmaa oli usein hidas paikallistaa ja korjata.

Iso ongelmakohta tuli esiin JavaScriptin myös silmukassa toteutettujen Ajax -pyyntöjen kanssa. Koska Ajax -pyynnöt vaativat readyState -arvon palautuvan serveriltä palautuvien tietojen mukana, dataa ei voi pyytää lisää ja käyttää sitä mukaa kun serveri sitä kerkeää palauttaa, vaan tietoa täytyy käyttää aina yksittäisissä sykäyksissä. Ajax -käskyjen lähettäminen 30 kertaa sekunnissa onnistuu kyllä, mutta Ajax käskyt kuljettavat mukanaan readyState -arvoa, eikä Ajaxin muuttujissa kuljettamaa tietoja voinut käyttää ennen kuin tieto on palautunut palvelimelta takaisin. Jatkuvasti lähetetty uusi pyyntö tiedon hakemisesta ei tuo tietoa koskaan käytettäväksi. Ongelmaa kuitenkin pystyi kiertämään tekemällä useampia pyyntöjä eri muuttujien avulla. Myöskään tiedon tallentaminen ja lukeminen samalla funktiolla ei ollut mahdollista, vaan tietoa täytyy tallentaa ja lukea vuorotellen, eri funktioilla.

Myös käyttöliittymää tekiessäni kohtasin vaikeuksia. JavaSctiptin tukemat tapahtumakuuntelijat eivät ymmärtäneet älypuhelimella annettuja käskyjä. Tätä ongelmaa vartein tein canvas-elementin päälle haamunapit joilla käyttäjä voi liikuttaa omaa laatik-

koaan myös ilman näppäimistöä, mutta haamunappien toiminta ei ole kovin käyttäjäystävällinen. Lisäksi koska kaikki canvas-elementin grafiikkakuorma piirretään prosessorilla (Grossbart 2011), käyttöliittymä puhelimella on huomattavasti hitaampi kuin pöytäkoneella testattuna.

Ongelmakohtana voidaan pitää myös materiaalin tuomista canvas-elementille. Yksinkertaisimmillaan esimerkiksi tekstin tuominen näkyviin oli ongelmallista. Canvas-elementille materiaalin tulostamista varten kuitenkin on tehty kattava määrä erilaisina frameworkseja jotka ratkaisevat ongelman. Tekstin tuomiseen ruudulle käytin CanvasText -nimistä frameworksia.

7 TESTAUS JA KEHITYSKOHTEET

Ohjelmistoa testasin Firefox, Internet explorer sekä Google Chrome selaimella. Ohjelman molemmat versiot testasin sekä paikallisesti, että sijoitettuna SunComet -webhotellin verkkopalvelimelle. Tietokoneelle tarkoitettujen selaimien lisäksi testasin ohjelmiston toimintaa myös Nokia Lumia 800 -puhelimen selaimella.

Ohjelmisto toimi kaikilla testatuilla alustoilla, eli yhteensopivuusongelmiin en törmännyt ja näyttäisi tosiaankin siltä, että HTML5 todella tulee olemaan edeltäjiään huomattavasti riippumattomampi laitteistosta ja lisä liitännäisistä. Selaimien välillä käytössä ei ollut eroa.

Testatessa kävi ilmi että puhelimella nettisivua näyttäessä, Ajax -kutsut eivät kaikissa tilanteissa kerkeä palauttamaan haluttua dataa, ennen uutta käskyä. Ohjelmaan täytyisi siis kehittää toiminnallisuus joka tarkkailisi syntyneitä latensseja, ja kontrolloisi niiden mukaan Ajax -käskyjen ajamista. Puolestaan serverille massamuistiin tallentava versio ei aiheuttanut samanlaisia ongelmia puhelimen selaimessa.

Koska halusin pitää ohjelman yksinkertaisena ja keskittyvän enemmän uusiin mahdollisuuksiin internetohjelmoinnissa, joita canvas-elementti tarjoaa, oikaisin ohjelmoinnissa useassa kohdassa. Mikäli sivustoa haluaisi käyttää esimerkiksi kaupallisissa tarkoituksissa, tulisi kaikki otetut oikotiet korjata. Yksi näistä kohdista on käyttäjälle luotu id, sekä käyttäjien liittyminen sivustolle. Ohjelma ei itse asiassa tarkasta milloin

joku käyttäjä liittyy sivustolle, ja milloin joku lähtee pois, vaan tyhjentää koko tietokannan aina kun uusi käyttäjä tulee sivustolle. Lisäksi tällä hetkellä jokaiselle käyttäjälle luotu id arvotaan, ohjelma ei tarkasta, onko arvottu id jo arvottu jollekin toiselle käyttäjälle. Näin ollen sama id-numero, jonka kuuluisi olla yksilöllinen, voi sattua useammalle käyttäjälle ja aiheuttaa toivomattomia ongelmia sivuston toiminnassa.

Toisena kehityskohteena pitäisin frameworksien käyttöä ohjelmakoodissa, esimerkiksi jQuery -kirjastosta löytyy paljon ominaisuuksia, joilla esimerkiksi kuvien tuominen canvas-elementtiin onnistuu helposti. Näitä ominaisuuksia olisi ollut syytä hyödyntää ohjelmakoodissa alusta saakka. Ohjelmakoodissa voisi myös käyttää järkevämpiä muuttujien ja funktioiden nimiä, pyrin nimeämään kaikki funktiot ja muuttujat englanniksi mahdollisimman kuvaavasti, mutta useassa tapauksessa valmiit funktiot eivät enää toiminnallisesti vastanneetkaan annettuja nimiä. Lisäksi nyt muuttujissa on sekaisin suomenkielisiä, sekä englanninkielisiä muuttujia. Kehityskohteena toki voisi myös pitää sitä, ettei sivuston ohjelmakoodi itse asiassa tee vielä mitään järkevää.

8 PÄÄTÄNTÖ

Varsinaisena tutkimusongelmana oli tutkia, millaiseen suuntaan internet on kehittymässä sekä millaiset tekniikat todennäköisesti tulevat parin vuoden sisällä olemaan yleisiä ja olennaisia. Uudenlaiset pilvipalvelimet mahdollistavat jo nyt jopa useiden miljoonien ihmisten käyttävän internetissä tarjottuja palveluita samanaikaisesti. Kun palvelun vuokraaja maksaa vain todellisuudessa käytetystä tiedonsiirto kaistasta, pienennetään kulurakennetta silloin, kun palvelulla ei vielä ole paljon käyttäjiä. Kuitenkin mielestäni Suomen kokoisessa maassa kokonaisten pilvipalveluohjelmistojen tarjoamat mahdollisuudet eivät vielä ole useassakaan tilanteessa kovin houkuttelevia. Suurten pilvipalveluiden suomat edut ovatkin hyödyllisempiä globaaleille yrityksille, jotka pystyvät hyödyntämään pilvipalveluille ominaisen skaalautuvuuden ja vaativat ohjelmistolleen turvallista ja nopeaa toimintaa ympäri maailman. SaaS -malliset pilvipalvelut puolestaan palvelevat parhaillaan pieniäkin yrityksiä erinomaisesti.

Pilvipalveluiden hyödyntämisestä parhaimmillaan saavutetut edut ja hyödyt ovat usein niin houkuttelevia, että niiden käyttöön otossa vastaan tulevat riskit unohdetaan. Tietoviikko.fi (2012) uutisoi vastikään että Uudenmaan luterilaiselle seurakunnalle oli

koitunut puolen miljoonan euron tappiot epäonnistuneesta pilvipalvelu-projektista. Pilvipalveluiden käyttöönotto ei ollut loppujenlopuksi onnistunut johtuen esimerkiksi väestökirjanpitoon liittyvistä lakisäädöksistä (Tietoviikko.fi 2012). Pilvipalveluiden yleistyessä, niiden käyttöönotossa tehtyjen lapsusten määrä tulee varmasti vähenemään, samalla kun pilvipalveluiden hallintaan tarkoitetut työkalut paranevat.

Perinteiseen HTML -ympäristöön tulevat W3C -standardien mukaiset uudistukset tulevat kuitenkin muokkaamaan arvioni mukaan koko internetin ulkonäköä tulevaisuudessa. Uudistus ei tule olemaan nopea, eikä tapahtumaan yhden yön aikana. Ilmiö voisi olla aikataulultaan samanlainen kun aikaisemmatkin HTML -versio uudistukset. Esimerkiksi 15 vuotta sitten tehdyt nettisivut näyttävät Matti Meikäläisen silmään vanhoilta sivuilta. Syy miksi sivustot vuosia aikaisemmin on näyttänyt erilaiselta, ei johdu siitä, että graafikot olisivat suunnitelleet näyttävämpiä nettisivustoja vasta 2000-luvulla. Syy on yksinkertaisempi: Tekniikka ei ole mahdollistanut aikaisemmin näyttävämpiä sivustoja.

HTML5:n tuoma canvas-elementti, sekä JavaScriptin Ajax -ominaisuudet tulevat varmaankin yleistymään nettisivustoilla, ja tätä kautta nettisivustoista tulee entistä käyttäjäystävällisempiä ja interaktiivisempia. HTML5:n tarkoituksena on myös poistaa yhteensopivuusongelmia eri laitteiden välillä, sekä vähentää liitännäisten tarvetta. Eri liitännäisien avulla on voitu tehdä näyttäviä sivustoja jo pitkän aikaa, mutta mikäli sivusto on vaatinut asennettavaksi jonkin liitännäisen, on ollut miltei varmaa, ettei sivuston sisältö tavoita haluttua asiakaskuntaa kovinkaan tehokkaasti. Siksi liitännäisiä on pyritty kaupallisissa sivustoissa välttämään.

Uskon, että internetin evoluutio jatkuu kuin se on tähänkin asti jatkunut. Uudet tekniikat mahdollistavat uusia innovatiivisia ideoita, mutta internetin kehitys jatkossakin etenee yritysmaailman ja W3C:n luomien standardien ehdoilla. Yritysmaailmassa erityisesti suunnittelemisen, testauksen ja dokumentoinnin merkitys tulee kasvamaan entisestään. Käyttäjille jatkossa suunnitellaan entistä käyttäjäystävällisempiä ja näyttävämpiä sovelluksia, jotka toimivat samalla tavalla riippumatta käytössä olevasta laitteistosta tai ohjelmistosta.

LÄHTEET

Budzynski, Michal 2010. Tutorial: Simple game with HTML5 Canvas. Verkkojulkaisu.
http://michalbe.blogspot.com/2010/09/simple-game-with-html5-canvas-part-1.html
Kirjoitettu 10.9.2010, Luettu 17.5.2012

Chapell, David. 2010. The Windows Azure platform and ISVs. Verkkojulkaisu.
http://www.davidchappell.com/writing/white_papers/Windows_Azure_platform_and_ISVs,_v2.0--Chappell.pdf
Luettu 28.4.2012.

Dudley, Richard J. 2010. Microsoft Azure: Enterprise Application Development. Olton Birmingham, Britannia: Packt Publishing Ltd.

Gosney, John 2003. HTML Professional Projects. Independence, KY, USA: Premier Press, Incorporated.

Grossbart, Zack 2011. How To Create Web Animations With Paper.js. Verkkojulkaisu.
http://coding.smashingmagazine.com/2011/11/21/create-web-animations-with-paperjs/
Kirjoitettu 21.12.2011, Luettu 29.5.2012.

Hurwitz, Judith, Bloor, Robin, Kaufman, Marcia 2009. Cloud computing for dummies. Hoboken, NJ, USA: For dummies.

McPeak, Jeremy Wilton, Paul 2010. Beginning JavaScript (4nd edition). Hoboken, NJ, USA: Wrox.

Microsoft SQL Azure Team Blog. 2010. Why Do I Need a Clustered Index?, Verkkojulkaisu.
http://blogs.msdn.com/b/sqlazure/archive/2010/05/12/10011257.aspx
Kirjoittanut nimim. David_R1. 12.3.2010. Luettu 5.5.2012.

Moore, Daniel X. 2011. No tears guide to HTML5 games. Verkkojulkaisu.
http://www.html5rocks.com/en/tutorials/canvas/notearsgame/
Kirjoitettu 1.1.2011, Luettu 17.5.2012.

MSDN. 2012. Common Language Runtime Overview. Verkkojulkaisu.
http://msdn.microsoft.com/en-us/library/ddk909ch(v=vs.71).aspx
Luettu 5.5.2012.

Thau, Dave 2006. Book of JavaScript: A Practical Guide to Interactive Web Pages (2nd edition). San Francisco, CA, USA: No Starch Press, Incorporated.

Tietoviikko.fi 2012. Kirkko mokasi Citrix-hankkeen - puolen miljoonan euron tappiot. Verkkojulkaisu.
http://www.tietoviikko.fi/kaikki_uutiset/kirkko+mokasi+citrixhankkeen++puolen+miljoonan+euron+tappiot/a795089#
Kirjoitettu 27.5.2012, Luettu 30.5.2012.

W3C 2012. HTML5 differences from HTML4. Verkkojulkaisu.
http://www.w3.org/TR/html5-diff/
Kirjoitettu 29.3.2012, Luettu 11.5.2012.

W3C 2012. HTML5 A vocabulary and associated APIs for HTML and XHTML. Verkkojulkaisu.
http://www.w3.org/TR/html5/the-canvas-element.html#the-canvas-element
Kirjoitettu 29.3.2012, Luettu 29.5.2012.

w3schools.com 2012. HTML5 Introducition. Verkkojulkaisu.
http://www.w3schools.com/html5/html5_intro.asp
Luettu 6.5.2012.