Niko Leppänoro

QUICKBLOXIN KÄYTTÖ WEB-SOVELLUSKEHITYKSESSÄ

QUICKBLOXIN KÄYTTÖ WEB-SOVELLUSKEHITYKSESSÄ

Niko Leppänoro
Opinnäytetyö
Kevät 2016
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehityksen suuntautumisvaihtoehto

---

Tekijä(t): Niko Leppänoro
Opinnäytetyön nimi: QuickBloxin käyttö web-sovelluskehityksessä
Työn ohjaaja(t): Juha Alakärppä, Hanna-Leena Huttunen
Työn valmistumislukukausi ja -vuosi: Kevät 2016 Sivumäärä: 35

Työn aiheena oli kehittää Vitilumi Oy:lle web-sovellus, jonka käyttäjät koostuvat henkilökohtaisista valmentajista ja heidän asiakkaistaan sekä yksittäisistä kuntoilijoista. Tavoitteena oli saada valmiiksi web-sovelluksesta julkaisukelpoinen versio. Työssä tutkittiin myös QuickBloxin käyttöä web-sovelluskehityksen työkaluna.

Web-sovelluksen kehityksessä käytettiin HTML-, CSS-, JavaScript- ja PHP-kieliä, QuickBloxin kirjastoja, tietokantoja ja palvelimia sekä WordPressiä ulkoasun ja web-sovelluksen rungon luomiseen. Web-sovelluksen kehitys aloitettiin vuoden 2015 kesän alussa kahden opiskelijan kehitysryhmässä harjoitteluprojektina, jota jatkettiin osana opinnäytetyötä.

Web-sovellukseen ehdittiin toteuttaa käyttäjän kirjautumis- ja rekisteröimissivut sekä lähes kaikki vaaditut toiminnot sisältävä pikaviestintäpalvelu. Web-sovelluksen kehitystä jatketaan myöhemmällä ajankohdalla. Jatkokehitykseen jäivät nykyisten toimintojen valmiiksi saaminen, puuttuvien ominaisuuksien toteutus ja web-sovelluksen laajempi testaus. QuickBlox on hyvä työkalu käyttäjien välisen kommunikaation toteuttamista varten, sillä se on helposti käyttöön otettavissa ja se sisältää työkalut käyttäjäprofiilien luomista ja käyttäjien välistä pikaviestintää varten.

Asiasanat: web-sovellus, QuickBlox, WordPress, HTML, CSS, JavaScript, PHP

ALKUSANAT

Haluan kiittää koko Vitilumi Oy:n henkilökuntaa ja erityisesti Hanna-Leena Huttusta opinnäytetyömahdollisuudesta ja kehitystyön aikaisesta tuesta. Kiitokset myös kaikille työtovereille, kun olitte mukana kehittämässä web-sovellusta.

23.5.2016

Niko Leppänoro

SISÄLLYS

SANASTO

asiakasohjelma	Sovellus, joka on verkon yli yhteydessä palvelimeen ja jolla voidaan käyttää palvelimella olevia palveluja
back end	Web-sovelluksen osa, joka vastaa asiakasohjelman ja palvelimen välisestä kommunikaatiosta sekä tiedon tallentamisesta ja välittämisestä
CSS	Cascading Style Sheets, web-sivuille kehitetty tyylioh-jeiden laji, jonka avulla voidaan muokata web-sivujen esitystapaa
front end	Web-sovelluksen osa, jonka kanssa käyttäjä voi olla vuorovaikutuksessa
HTML	HyperText Markup Language, web-sivujen luomiseen käytetty avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä
JavaScript	Ohjelmointikieli, joka mahdollistaa dynaamisten toiminnallisuuksien lisäämisen web-sivuille
palvelin	Tietokoneessa suoritettava palvelinohjelmisto, joka tarjoaa palveluja muille ohjelmille joko paikallisesti tai verkon välityksellä
PHP	PHP: Hypertext Preprocessor, ohjelmointikieli, jota käytetään dynaamisten web-sivujen luonnissa
Scrum	Projektinhallinnan viitekehys, joka koostuu tietyn ajanjakson suuruisista kehitysjaksoista eli sprinteistä
WWW	World Wide Web, internet-verkossa toimiva hajautettu hypertekstijärjestelmä

1 JOHDANTO

Työn tilaajana toimi Vitilumi Oy, joka on vuonna 2015 perustettu hyvinvointialan ohjelmistotalo. Vitilumi Oy:n tavoitteena on tarjota laadukasta ohjelmistokehitystä ja kehittää omia hyvinvointialan sovelluksia.

Työn aiheena oli kehittää henkilökohtaisten valmentajien ja heidän asiakkaidensa sekä yksittäisten kuntoilijoiden käyttöön tarkoitettua web-sovellusta käyttäen hyväksi QuickBloxin tarjoamaa, pilvipalveluna toimivaa back endiä eli asiakasohjelman ja palvelimen välisestä kommunikoinnista vastaavaa kokonaisuutta. QuickBloxia haluttiin käyttää, sillä sen uskottiin tehostavan web-sovelluksen kehitystä. Valintaan vaikutti myös se, että QuickBloxin käyttö on ohjelmistokehittäjälle ilmaista ja vasta web-sovelluksen käyttäjämäärän ja verkkoliikenteen kasvaessa tarpeeksi QuickBlox muuttuu maksulliseksi.

Web-sovellus sisältää muun muassa käyttäjän rekisteröimisen ja kirjautumisen sekä käyttäjien välisen pikaviestintäpalvelun. Tavoitteena oli toteuttaa web-sovelluksesta julkaisukelpoinen versio, joka sisältäisi käyttötarkoitukseen soveltuvat perustoiminnallisuudet, kuten valmentajan ja asiakkaan välisen pikaviestinnän.

Web-sovelluksen kehitys aloitettiin vuoden 2015 kesän alussa kahden opiskelijan kehitysryhmässä osana harjoitteluprojektia. Projektin läpiviennissä sovellettiin Scrum-viitekehystä: viikoittain pidetyissä palavereissa käytiin läpi edellisen viikon tavoitteita ja niiden toteutumista sekä sovittiin seuraavan viikon tavoitteet.

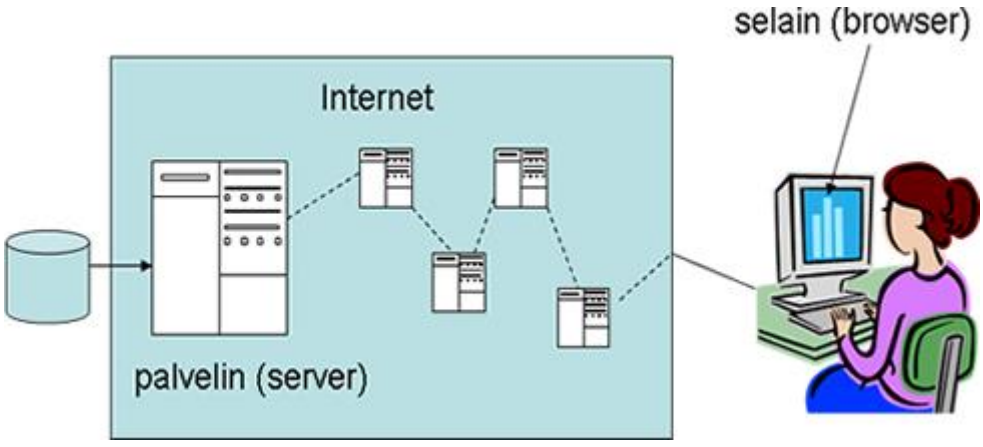
2 WEB-SOVELLUSKEHITYS

Työn aiheena olleen web-sovelluksen kehityksessä käytettiin QuickBloxin tarjoamaa, pilvipalveluna toimivaa back endiä, joka sisältää tarvittavat tietokannat ja palvelimet. Web-sovelluksen käyttöliittymän ulkoasun suunnittelussa käytettiin WordPress-nimistä julkaisualustaa. Kehitystyökaluina työssä käytettiin HTML-, CSS-, JavaScript- ja PHP-kieliä.

2.1 Web-sovellus

Web-sovelluksella tarkoitetaan internetin kautta jaettavaa ohjelmistoa, jota ajetaan verkkoselaimen kautta. Web-sovellukset ovat suosittuja monista syistä, joista päällimmäisinä ovat verkkoselainten yleisyys, verkkoselaimen käyttö asiakasohjelmana web-sovellusten ylläpitämiseen ilman tarvetta asentaa erillistä ohjelmaa tietokoneelle sekä luontainen yhteensopivuus eri järjestelmien välillä. Tavallisimpia web-sovelluksia ovat esimerkiksi sähköposti, verkkokaupat ja pikaviestintäpalvelut. (1.)

Web-sovelluksen käyttöliittymänä toimivat verkkoselaimessa näytettävä sivut, jotka perustuvat yhdeltä tai useammalta palvelimelta saatavaan tietoon. Tietokoneen verkkoselain toimii web-sovelluksen asiakasohjelmana (client) ja on yhteydessä palvelimeen (server) verkon välityksellä kuvan 1 mukaisesti. (2.)



KUVA 1. Web-sovelluksen toimintaperiaate (2)

2.2 Front end ja back end

Web-sovelluskehityksestä puhuttaessa käytetään monesti termejä front end ja back end. Front endillä tarkoitetaan web-sovelluksen osaa, jonka käyttäjä pystyy näkemään ja jonka kanssa hän voi olla vuorovaikutuksessa. Front end -kehitykseen liittyviä ohjelmointikieliä ovat muun muassa HTML, CSS ja JavaScript. (3; 4.)

Web-sovelluksen taustalla toimiva back end koostuu kolmesta osasta: palvelimesta, sovelluksesta ja tietokannasta. Back end keskittyy asiakasohjelman ja palvelimen väliseen kommunikointiin sekä tiedon tallettamiseen ja välittämiseen. Back end-kehityksessä käytettyjä ohjelmointikieliä ovat esimerkiksi PHP, Ruby ja Python. (3; 4.)

2.3 QuickBlox

QuickBlox on pilvipalveluna toimiva back end, joka mahdollistaa muun muassa pikaviestinnän ja videopuhelut mobiili- ja web-sovelluksissa. QuickBloxia käyttävät sadat sosiaalisten verkkojen sovellukset ja yritykset. (5.)

QuickBlox sisältää ohjelmointirajapinnat, ohjelmistokehityspaketit ja dokumentaatiot eri käyttöjärjestelmiä varten sekä hallintapaneelin QuickBlox-sovellusten ylläpitäjille, jonka kautta he voivat hallita omien sovellusten sisältöä. QuickBlox on jaettu kuuteen eri moduuliin, joita voidaan yhdistellä tarpeen mukaan. (6.)

QuickBloxin yhteydet on suojattu QuickBloxin omalla yhteysprotokollalla, joka on mukautettu versio OAuth-viitekehyksestä. Järjestelmän suoritus tapahtuu joko QuickBloxin tai yritysten omalla AWS (Amazon Web Services) -palvelimella. (6.)

2.4 WordPress

WordPress on alun perin blogien luomiseen ja ylläpitämiseen tarkoitettu ilmainen sisällönhallintaohjelmisto, joka sai alkunsa vuonna 2003. WordPress perustuu avoimeen lähdekoodiin, on kirjoitettu PHP-kielellä ja käyttää MySQL-tietokantaa tietojen tallentamiseen. WordPressille on myös kehitetty lukuisia vapaasti käytettäviä teemoja ja pienoisohjelmia. (7; 8.)

Q-Successin ylläpitämän W3Techs-sivuston mukaan 24,9 % kaikista maailman verkkosivustoista käyttää WordPressiä. WordPress on samalla myös maailman suosituin sisällönhallintaohjelmisto. (9.)

2.5 HTML

HTML eli HyperText Markup Language on WWW-sivujen rakenteen kuvaamiseen käytetty merkintäkieli. HTML mahdollistaa muun muassa tekstiä, kuvia ja ääni- ja videoleikkeitä sisältävien verkkodokumenttien julkaisemisen, verkkotiedonhaun hyperlinkkien avulla sekä web-sovellusten liittämisen dokumentteihin. (10.)

HTML-koodi koostuu sisäkkäisistä ja peräkkäisistä elementeistä, joita edustavat kulmasulkeilla merkityt tunnisteet eli tägit. HTML-elementillä on yleensä aloitustunniste ja vinoviivalla merkitty lopetustunniste, joiden väliin jää elementin sisältö. On kuitenkin olemassa myös HTML-elementtejä, joilla ei ole mitään sisältöä, kuten esimerkiksi
-tunniste, jolla kuvataan pakollista rivinvaihtoa. Tällaiset elementit eivät tarvitse lopetustunnistetta, ellei kyseessä ole XHTML-koodi, jolloin lopetustunnisteen vinoviiva merkitään heti aloitustunnisteen loppuun. (11.)

HTML-elementeillä kuvataan dokumentin sisältö, kuten otsikot, taulukot ja tekstikappaleet kuvan 2 esimerkkidokumentin mukaisesti.

KUVA 2. Esimerkki HTML-dokumentista

HTML sai lukuisia uudistuksia ja laajennuksia vuosien 1990 ja 1995 välisenä aikana, jolloin sitä isännöivät aluksi CERN ja myöhemmin IETF (Internet Engineering Task Force). Kun W3C (World Wide Web Consortium) sai alkunsa, HTML-

kielen kehitys sai uuden käänteen ja vuonna 1995 kielestä julkaistiin uusi versio HTML 3.0. Vuonna 1997 julkaistiin seuraava versio HTML 3.2, joka oli hieman käytännöllisempi edeltäjäänsä nähden. Samana vuotena julkaistiin myös HTML4. (12.)

Vuonna 1998 HTML-kielen kehitys pysähtyi ja sen sijaan W3C alkoi kehittämään tälle XML-pohjaista vastinetta nimeltä XHTML. Tuloksena syntyi XHTML 1.0 vuonna 2000, joka ei sisältänyt uusia ominaisuuksia HTML4:ään nähden. Tämän jälkeen W3C keskittyi tekemään XHTML:stä helpommin laajennettavan sekä alkoi kehittämään uutta aiempiin kieliin yhteensopimatonta kieltä nimeltä XHTML2. (12.)

Vuonna 2003 julkaistu XForms herätti uudelleen kiinnostuksen jatkaa HTML4:n kehitystä. Mozilla ja Opera esittivät W3C:lle varhaisluonnoksen HTML5:n ominaisuuksista vuonna 2004, mutta se peruttiin sillä perusteella, että se ei sopinut yhteen aiemmin sovitun webin kehittämissuunnan kanssa. Apple, Mozilla ja Opera ryhtyivät yhteistoimin jatkokehittämään tätä ajatusta yhteisnimikkeellä WHATWG (Web Hypertext Application Technology Working Group), jonka perusperiaatteina olivat takautuva yhteensopivuus, määritelmien ja implementaatioiden yhteensopivuus sekä määritelmien riittävä tarkkuus implementaatioiden yhteentoimivuuden varmistamiseksi. HTML5:n määrityksen oli siis sisällytettävä aikaisempien dokumenttien, HTML4:n, XHTML1:n ja DOM2 HTML:n, määritykset. (12.)

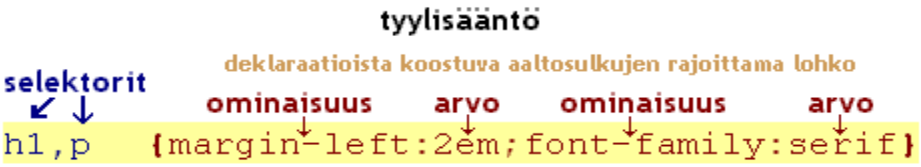
Vuonna 2006 W3C muutti mieltään ja halusi mukaan HTML5:n kehittämiseen, ja muodosti työryhmän kehittämään HTML5:n määritystä WHATWG:n kanssa vuonna 2007. Ryhmät tekivät yhteistyötä Ian Hicksonin johtamana useita vuosia, kunnes vuonna 2011 ryhmät tulivat siihen johtopäätökseen, että ryhmien päätavoitteet olivat erilaiset: W3C halusi määritellä HTML5:n suositusta, kun taas WHATWG halusi jatkaa niin sanotun elävän standardin kehittämistä. Vuonna 2012 W3C valitsi uuden työryhmän luomaan HTML 5:n suositusta, joka julkaistiin vuoden 2014 lokakuussa, sekä valmistelemaan luonnosta HTML:n seuraavaa versiota varten. (12.)

2.6 CSS

CSS eli Cascading Style Sheets on tyylikieli, joka mahdollistaa tyyliehdotuksien yhdistämisen HTML- ja XML-dokumentteihin. Tyyliehdotuksilla voidaan muokata haluttujen elementtien esitystapaa, kuten väriä ja sijoittelua. CSS mahdollistaa dokumenttien käyttöalueiden laajentamisen, helpottaa dokumenttien luomista ja ylläpitoa sekä pienentää tiedostokokoja. (13;14.)

CSS-kielen tärkein ominaisuus on niin sanottu Cascade-prosessi. Kun tyylimäärittelyt määritellään yhdessä paikassa, selain asettaa tyylimäärittelyt kaikkiin dokumentin elementteihin, jotka vastaavat tiettyjä sääntöjä. (14.)

Tyyliehdotukset (style sheets) koostuvat vähintään yhdestä tyylisäännöstä (style rule), jotka puolestaan koostuvat vähintään yhdestä valitsimesta (selector) sekä deklaraatiolohkosta, jossa on vähintään yksi deklaraatio eli ominaisuus-arvo-pari. Tyylisäännössä useammat valitsimet erotetaan toisistaan pilkulla, ominaisuus-arvo-parit kaksoispisteellä ja deklaraatiot puolipisteellä kuvan 3 mukaisesti. (13.)



KUVA 3. Tyylisäännön rakenne (13)

2.6.1 Valitsimet

Valitsimien avulla deklaraatiot yhdistetään HTML-dokumentin eri kohteisiin. Eniten käytetyimpiä valitsimia ovat

- tyyppivalitsimet, joissa elementtityyppeihin viitataan niiden nimellä
- luokkavalitsimet, joissa elementteihin viitataan elementille annetun CLASS-attribuutin perusteella
- tunnistevalitsimet, joissa elementteihin viitataan käyttämällä yksilöllistä ID-attribuuttia. (14.)

Näiden lisäksi on olemassa myös valitsimia, jotka perustuvat tarkempiin kriteereihin. Näitä ovat

- universaalit valitsimet (universal selectors)
- attribuuttivalitsimet (attribute selectors)
- lapsivalitsimet (child selectors)
- jälkeläisvalitsimet (descendant selectors)
- sisarusvalitsimet (adjacent sibling selectors)
- pseudoluokat (pseudo-classes)
- pseudoelementit (pseudo elements). (14.)

2.6.2 Tyyliehdotusten liittäminen HTML-sivulle

Tyyliehdotukset voidaan liittää HTML-dokumenttiin neljällä eri tavalla. Paras vaihtoehto on ulkoinen tyylimäärittely, jossa tyyliehdotukset kootaan yhteen CSS-tiedostoon, joka liitetään HTML-dokumenttiin käyttäen <link>-elementtiä <head>-elementin sisällä. (Kuva 4.)

```
<link rel="stylesheet" href="styles.css" type="text/css" media="screen">
```

KUVA 4. Ulkoinen tyylimäärittely (14)

Rivinsisäisessä tyylimäärittelyssä käytetään HTML-elementin style-attribuuttia. Tällä tavalla määriteltynä selain pakotetaan käyttämään valittuja tyyliehdotuksia, mutta ongelmaksi voi tulla ylläpidon vaikeus. Tässä ei myöskään käytetä hyväksi Cascade-prosessia. (Kuva 5.)

```
<p style="background:blue; color:white; padding:5px;">Paragraph</p>
```

KUVA 5. Rivinsisäinen tyylimäärittely (14)

Upotetussa tyylimäärittelyssä tyyliehdotukset kootaan <style>-elementin sisään, joka sijoitetaan HTML-dokumentin <head>-elementtiin. (Kuva 6.)

```
<style type="text/css" media="screen">
    p {
        color:white;
        background:blue;
        padding:5px;
    }
</style>
```

KUVA 6. Upotettu tyylimäärittely (14)

Tuodussa tyylimäärittelyssä tyyliehdotus käytetään @import-sääntöä, kuten kuvassa 7.

```
<style type="text/css" media="screen">
    @import url(styles.css);
    /* Other import statements or CSS styles could go here */
</style>
```

KUVA 7. Tuotu tyylimäärittely (14)

2.7 JavaScript

JavaScript on järjestelmäriippumaton, olioperustainen ohjelmointikieli. Se sisältää peruskirjaston objekteja sekä kielielementtejä, jota voidaan laajentaa asiakasohjelmapuolen ja palvelinpuolen JavaScriptillä. (15.)

JavaScriptiä ei tule sekoittaa Java-ohjelmointikielen kanssa, sillä ne eivät tarkoita samaa kieltä, vaikka niillä on jonkin verran yhtäläisyyksiä. JavaScriptin ja Javan eroja vertaillaan taulukossa 1.

TAULUKKO 1. JavaScript- ja Java-kielten vertailua (15)

JavaScript	**Java**
Olioperustainen ohjelmointikieli	Luokkiin perustuva ohjelmointikieli
Muuttujia ei tarvitse esitellä etukäteen (dynaaminen tyypitys)	Muuttujat täytyy esitellä etukäteen (staattinen tyypitys)
Ei voi kirjoittaa kovalevylle automaattisesti	Voi kirjoittaa kovalevylle automaattisesti

Entinen Netscapen ja nykyään Mozillan työntekijä Brendan Eich kehitti JavaScriptin vuoden 1995 toukokuussa. Tuolloin sen nimi ei vielä ollut JavaScript, vaan siitä käytettiin Netscapen perustajan Marc Andreessenin keksimää nimitystä Mocha. Vuoden 1995 syyskuussa nimi muutettiin LiveScriptiksi, ja kun LiveScript sai lisenssin Sun Microsystemsiltä, kieli sai lopullisen nimityksensä. (16.)

Vuosien 1996 ja 1997 välisenä aikana ECMA (European Computer Manufacturers Association) standardisoi JavaScriptin, jotta sitä voitaisiin käyttää muissakin selaimissa. Lopputuloksena syntyi ECMAScript-standardi, josta tunnetuimpia implementaatioita olivat JavaScript ja ActionScript 3. Seuraava standardi ECMAScript 2 julkaistiin vuonna 1998 ja ECMAScript 3 vuonna 1999. (16.)

Vuonna 2000 entinen NetScapen työntekijä Waldemar Horwat ryhtyi johtamaan JavaScript 2:n eli ECMAScript 4:n kehitystä, jossa Microsoft oli myös alustavasti mukana. Myöhemmin kuitenkin selvisi, että Microsoftilla ei ollut aikomustakaan tehdä yhteistyötä tai implementoida JavaScriptiä Internet Exploreriin, joten vuonna 2003 JS2:n kehitystä lykättiin. (16.)

Vuonna 2005 Brendan Eich ja Mozilla ryhtyivät yhdessä kehittämään E4X:ää (ECMAScript for XML) Macromedian kanssa. ECMAScript 4:n kehitys sai siten uuden alun, jonka tavoitteena oli standardisoida ActionScript 3 ja implementoida se Mozillan JavaScript-moottoriin nimeltä SpiderMonkey.

Vastaavasti Adobe julkaisi avoimen lähdekoodin projektin AVM2:n (ActionScript Virtual Machine 2), josta käytettiin myös koodinimikettä Tamarin. ActionScript 3 ja Tamarin olivat kuitenkin liian erilaisia yhdistettäväksi. (16.)

Samana vuonna Jesse James Garrett julkaisi artikkelin, jossa hän loi termin ”Ajax” sekä kuvasi joukon teknologioita web-sovellusten luomiseen, joissa dataa voidaan ladata taustalla luoden siten dynaamisempia sovelluksia. Näiden teknologioiden perustana käytettiin JavaScriptiä. Tämä johti lukuisten avoimen lähdekoodin kirjastojen, kuten jQueryn, syntymiseen. (16.)

Vuoden 2009 alkupuolella ECMAScript 3.1 nimettiin uudelleen ECMAScript 5:ksi. Standardin jatkokehitys tunnetaan nykyään nimellä Harmony. (16.)

2.8 PHP

PHP eli PHP: Hypertext Preprocessor on web-sovelluskehityksessä käytettävä, avoimeen lähdekoodiin perustuva komentosarjakieli, joka mahdollistaa muun muassa dynaamisen verkkosivujen sisällön luomisen. PHP:n merkittävimpiin ominaisuuksiin kuuluvat tuet Linux-, Windows- ja Mac-käyttöjärjestelmille sekä monille nykyisille verkkopalvelimille ja tietokannoille. (17; 18.)

PHP:n käyttökohteet voidaan jakaa kolmeen eri pääalueeseen:

- palvelinpuolen ohjelmointi, joka vaatii PHP-jäsentäjän, verkkopalvelimen sekä WWW-selaimen toimiakseen
- komentoriviohjelmointi, johon tarvitaan vain PHP-jäsentäjä
- työpöytäsovellusten kehittäminen. (18.)

Palvelimella suoritettavan PHP-koodin yleisin käyttötapa on tulostaa HTML-koodia, joka puolestaan lähetetään asiakasohjelmalle. PHP:llä voidaan tulostaa myös esimerkiksi kuvia, PDF-tiedostoja, Flash-elokuvia sekä mitä tahansa muuta tekstiä. (17; 18.)

PHP-koodia voidaan sisällyttää HTML-dokumentteihin käyttämällä aloitustunnistetta ”<?php” ja lopetustunnistetta ”?>”, joiden avulla PHP-jäsentäjä erottaa PHP-koodin muusta sisällöstä. (Kuva 8.)

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

KUVA 8. PHP-esimerkki (17)

Rasmus Lerdorfin vuonna 1994 luoma ensimmäinen versio PHP:stä oli joukko C-kielellä kirjoitettuja CGI-binäärejä, joita hän käytti sivustojen kävijämäärien seuraamiseen. Lerdorf käytti niistä silloin nimitystä PHP Tools (Personal Home Page Tools). Lisäominaisuuksille oli kuitenkin tarvetta, joten hän kirjoitti PHP Toolsin uudelleen, kunnes vuonna 1995 Lerdorf julkaisi PHP Toolsin lähdekoodin muidenkin käytettäväksi. (19.)

Saman vuoden syyskuussa Lerdorf jatkoi PHP:n laajentamista ja alkoi käyttää siitä nimitystä FI (Forms Interpreter). Lokakuussa Lerdorf julkaisi jälleen uudelleen kirjoitetun version, joka tunnettiin hetkellisesti nimellä Personal Home Page Construction Kit. Kieli muistutti rakenteeltaan C-kieltä ja oli siten helposti ohjelmistokehittäjien omaksuttavissa. (19.)

Huhtikuussa vuonna 1996 Lerdorfin julkaisema uusi versio PHP/FI oli kehittynyt kokonaan omaksi ohjelmointikielekseen ja kesäkuussa kielestä julkaistiin 2.0-versio. Toukokuussa vuonna 1998 tehdyn Netcraft-tutkimuksen mukaan PHP oli asennettu lähes 60000 verkkotunnukselle ja käsitti siten noin prosentin kaikista tuolloin olemassa olevista Internet-verkkotunnuksista. (19.)

Tähän asti PHP:n kehitys oli tapahtunut pääasiallisesti pelkästään Lerdorfin toimesta. Vuonna 1997 Andi Gutmans ja Zeev Suraski halusivat parantaa PHP/FI 2.0:n toiminnallisuutta ja alkoivat kehittämään yhdessä Lerdorfin kanssa uutta ohjelmointikieltä, jonka nimi yksinkertaistettiin PHP 3.0:ksi. Tämän jälkeen kehitysryhmään liittyi muitakin kehittäjiä ympäri maailmaa. Lopulta vuoden 1998 kesäkuussa kieli nimettiin viralliseksi PHP/FI 2.0:n jälkeläiseksi. Kielen helppo laajennettavuus saavutti suuren suosion: jo ennen virallista julkaisua PHP 3.0 oli asennettu yli 70000 verkkotunnukselle ja parhaimmillaan noin 10 %:lle kaikista Internet-palvelimista. (19.)

Pian PHP 3.0:n julkaisun jälkeen Gutmans ja Suraski aloittivat PHP 4.0:n kehityksen pyrkimyksenä parantaa PHP-sovellusten suorituskykyä ja PHP:n lähdekoodin modulaarisuutta. Toukokuussa vuonna 2000 julkaistussa PHP 4.0:ssa oli uusi Zend Engine -nimellä tunnettu ydin sekä uusia ominaisuuksia. (19.)

Vuoden 2004 heinäkuussa julkaistiin PHP 5.0, joka sisälsi uuden Zend Engine 2.0 -ytimen sekä lukuisia uusia ominaisuuksia. Aiempien vuosien tilastojen perusteella PHP:n arvioitiin olleen asennettuna jopa sadoille miljoonille verkkotunnuksille ympäri maailmaa. (19.)

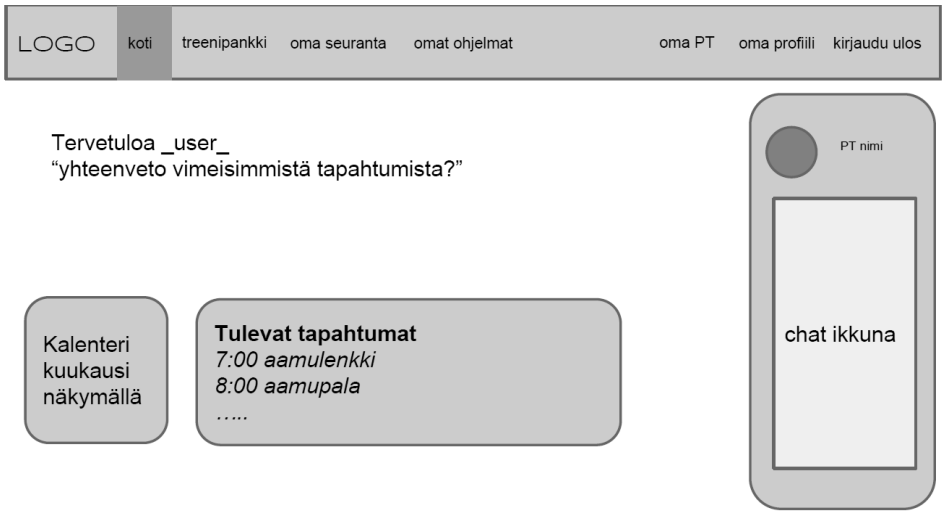
3 WEB-SOVELLUKSEN TOTEUTUS

Ennen varsinaista kehitystyön aloittamista oli aluksi suunniteltava sivuston perusrakenne, pystytettävä kehitysympäristö, valittava käytettävät työkalut ja otettava selvää QuickBloxin JavaScript-kirjaston rakenteesta ja sen käytöstä. Kehityksen alkuvaiheessa web-sovellus rakennettiin HTML-sivuista, mutta myöhemmässä vaiheessa siirryttiin WordPressin käyttöön. Tästä syystä oli myös selvitettävä, miten aiemmat tuotokset saataisiin siirrettyä PHP-sivuista koostuvalla WordPress-sivustolla toimiviksi. Web-sovelluksen eri komponentteihin liittyviä vaatimuksia ja käyttöliittymän ulkoasua käsiteltiin viikkopalavereissa.

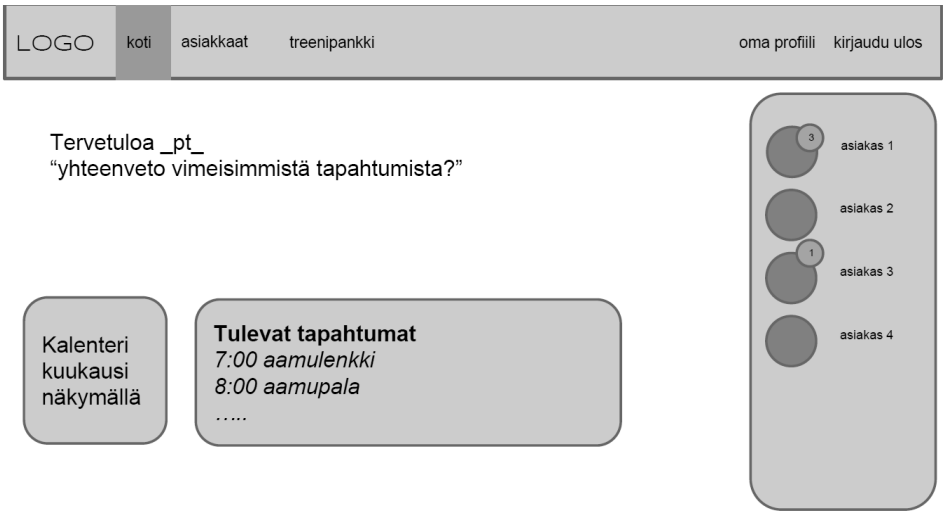
Koska web-sovelluksen lähdekoodeja ei voitu tuoda julki salassapitosyistä johtuen, toimintoja on havainnollistettu käyttäen QuickBloxin ja WordPressin esimerkkejä.

3.1 Web-sovelluksen rakenne

Web-sovellus koostuu aloitussivusta, jonka kautta käyttäjä luo itselleen profiilin ja kirjautuu sisään web-sovellukseen. Tämän jälkeen käyttäjä siirretään pääsivulle, josta käyttäjä pääsee käyttämään web-sovelluksen toimintoja. Käyttäjälle näkyvä sisältö riippuu siitä, onko kyseessä harjoittelija (kuva 9) vai valmentaja (kuva 10).



KUVA 9. Alustava luonnos harjoittelijan pääsivusta



KUVA 10. Alustava luonnos valmentajan pääsivusta

3.2 Kehitysympäristö

Web-sovelluksen HTML-, JavaScript- ja PHP-koodien kirjoittamisessa käytettiin Brackets-nimistä ilmaista tekstieditoria, jonka yhtenä hyväksi todettuna etuna muihin tekstieditoreihin oli koodiin tehtyjen muutosten reaaliaikainen esikatselu selaimen kautta.

Versionhallinnassa käytettiin SourceTree-nimistä Git-asiakasohjelmaa, jolla ohjelmakoodit sai helposti siirrettyä BitBucketiin talteen. Versionhallintaa käytettiin myös siitä syystä, että web-sovellusta oli kehittämässä useampia henkilöitä.

WordPressin paikallinen asennus tehtiin WampServer-kehitysympäristöön, joka sisältää Apache2-verkkopalvelimen, MySQL-tietokannan ja PHP:n asennukset.

3.3 QuickBloxin liittäminen web-sovellukseen

Jotta QuickBloxia voitaisiin käyttää web-sovelluksessa, on HTML-sivun koodiin lisättävä HTML-tunniste <script>, jolle annetaan lähteeksi QuickBloxin JavaScript-kirjaston sijainti kuvan 11 osoittamalla tavalla. Toimiakseen kirjasto tarvitsee myös joko jQuery- tai Zepto-kirjaston, joista toinen on myös lisättävä ennen QuickBlox-kirjastoa. (20.)

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/quickblox/2.1.1/quickblox.min.js"></script>
```

KUVA 11. JQuery- ja QuickBlox-kirjaston lisääminen HTML-sivulle (20)

Tämän jälkeen aina, kun QuickBloxiin halutaan olla jollain tavalla yhteydessä, on koodissa ensin alustettava viitekehys, luotava QuickBlox-sessio ja kirjauduttava QuickBlox-sovellukseen jo olemassa olevalla käyttäjällä tai rekisteröimällä uusi käyttäjä (20).

3.3.1 Viitekehyksen alustaminen

Ennen viitekehyksen alustamista on rekisteröidyttävä QuickBloxiin luomalla tunnukset ja sen jälkeen luomalla uusi QuickBlox-sovellus. Jokainen QuickBlox-sovellus saa kolme automaattisesti luotua tunnistearvoa (appId, authKey ja authSecret), joita tarvitaan viitekehyksen alustamisessa.

Viitekehys alustetaan kutsumalla QuickBlox-kirjaston init()-funktiota QB-nimisen olion avulla. init()-funktion parametreiksi annetaan käytettävän QuickBlox-sovelluksen tunnistearvot kuvan 12 osoittamalla tavalla.

```
var CREDENTIALS = {
    appId: 28287,
    authKey: 'XydaWcf8OO9xhGT',
    authSecret: 'JZfqTspCvELAmnW'
};

QB.init(CREDENTIALS.appId, CREDENTIALS.authKey, CREDENTIALS.authSecret);
```

KUVA 12. Viitekehyksen alustaminen (20)

3.3.2 QuickBlox-session luominen

Kun viitekehys on alustettu, voidaan luoda QuickBlox-sessio, joita on kahta eri tyyppiä: sovellussessio ja käyttäjäsessio. Sovellussessio antaa pelkästään datan lukuoikeudet (kuva 13), kun taas käyttäjäsessio antaa datan luku-, kirjoitus-, päivitys- ja poistamisoikeudet (kuva 14).

```
QB.createSession(function(err, result) {
  // callback function
});
```

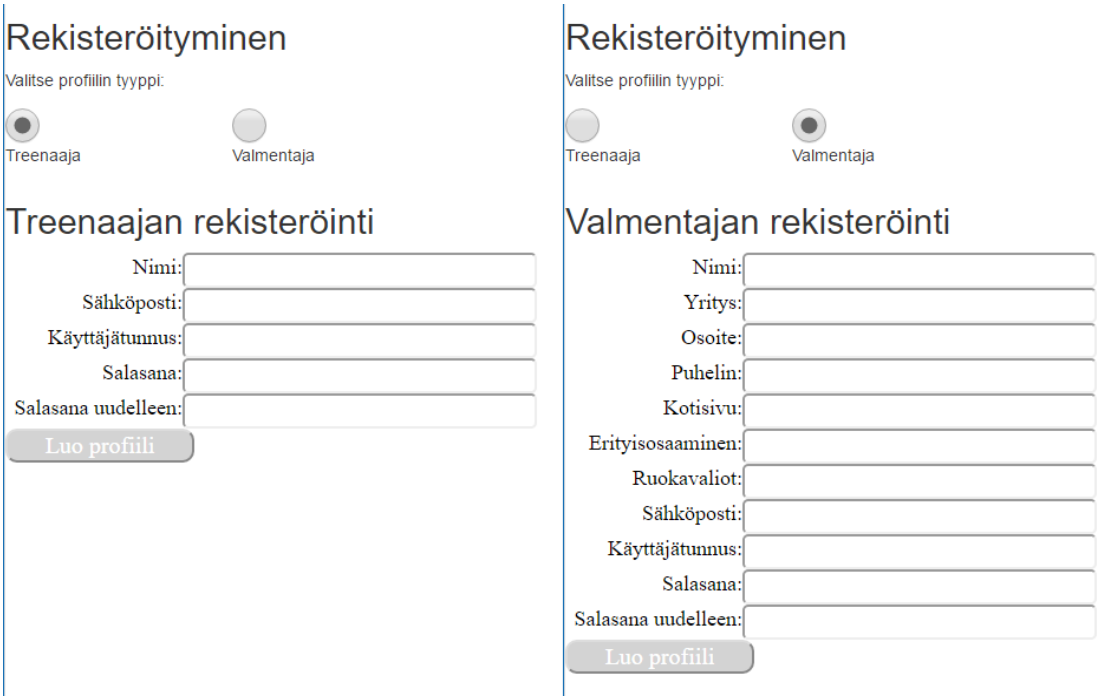
KUVA 13. Sovellussession luominen (20)

```
var params = {login: 'garry', password: 'garry5santos'};

// or through email
// var params = {email: 'garry@gmail.com', password: 'garry5santos'};

// or through social networks (Facebook / Twitter)
// var params = {provider: 'facebook', keys: {token: 'AM46dxjhisdffgry26282352fdusdfusdfgsdf'}};

QB.createSession(params, function(err, result) {
  // callback function
});
```

KUVA 14. Käyttäjäsession luominen (20)

3.3.3 Käyttäjäprofiilin luominen

Rekisteröityessään käyttäjän on valittava aluksi profiilin tyypiksi treenaaja tai valmentaja, jota varten rekisteröitymislomakkeessa on valintapainikeet sekä tekstikentät käyttäjätiedoille. Rekisteröitymiseen tarvittavat käyttäjätiedot vaihtelevat profiilityypin mukaan. (Kuva 15.)



KUVA 15. Rekisteröitymislomakkeet

Jotta vääränlaisilta arvoilta vältyttäisiin, tiettyihin tekstikenttiin sisällytettiin JavaScriptillä ohjelmoituja ennakkotarkistuksia:

- sähköpostiosoitteessa on oltava @-merkki
- kahta samannimistä käyttäjätunnusta ei voi luoda
- salasanan on oltava vähintään kahdeksan merkkiä pitkä.

Kun käyttäjä on syöttänyt arvon tekstikenttään, tekstikentän viereen ilmestyy kuvake, joka kertoo käyttäjälle oliko syötetty arvo kelvollinen. Kun käyttäjä on täyttänyt jokaisen tekstikentän hyväksytysti, käyttäjä voi luoda profiilin.

Käyttäjä luodaan kutsumalla QuickBloxin Users-moduulin create()-funktiota, joka luo käyttäjäprofiilin annettujen tietojen perusteella. Valmentajan rekisteröinnissä luodaan tavallisen QuickBlox-käyttäjän lisäksi Custom-objekti lisätietojen tallentamista varten. (Kuva 16.)

```
var params = {
    'login': "emporio",
    'password': "somepass"
};

QB.users.create(params, function (err, user) {
    if (user) {
        // success
    } else {
        // error
    }
});
```

KUVA 16. Käyttäjän rekisteröinti (21)

3.3.4 Sisäänkirjautuminen web-sovellukseen

Sisäänkirjautumislomake sisältää tekstikentät käyttäjätunnusta ja salasanaa varten, sisäänkirjautumispainikkeen sekä linkin edellä mainittujen tietojen palautussivulle (kuva 17).

KUVA 17. Sisäänkirjautumislomake

Sisäänkirjautumisfunktiossa kutsutaan QuickBloxin login()-funktiota, jolle annetaan parametrina käyttäjän syöttämät tiedot. Mikäli sisäänkirjautuminen epäonnistuu, käyttäjälle annetaan tästä ilmoitus. (Kuva 18).

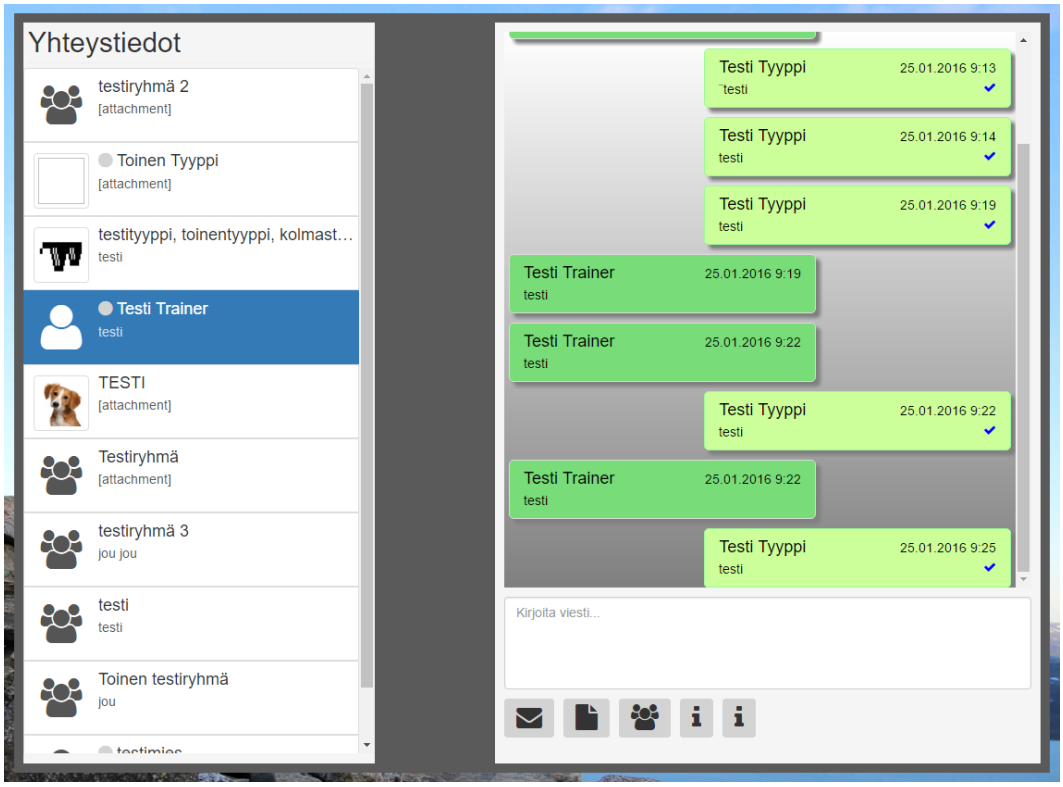
```
var params = {
    'login': "emporia",
    'password': "somepass"
};

QB.login(params, function (err, user) {
    if (user) {
        // success
    } else {
        // error
    }
});
```

KUVA 18. Käyttäjän sisäänkirjautuminen (21)

3.3.5 Pikaviestintäpalvelu

Pikaviestintäpalvelun kautta valmentajat voivat keskustella reaaliaikaisesti kaikkien asiakkaidensa kanssa sekä luoda keskusteluryhmiä. Harjoittelijat voivat puolestaan keskustella vain omien henkilökohtaisten valmentajiensa kanssa. Käyttäjät voivat lähettää toisilleen viestien lisäksi kuva-, ääni- ja videotiedostoja sekä nähdä yhteyshenkilöiden online/offline-tilat. Käyttäjä näkee lähettämistään viesteistään, onko vastaanottaja lukenut viestin.

Pikaviestintäpalvelun käyttöliittymässä on lista yhteystiedoista, viesti-ikkuna, jonka viestit on värikoodattu lähettäjän perusteella, sekä tekstikenttä viestien kirjoittamista varten. Tekstikentän alapuolella olevista painikkeista käyttäjä voi lähettää kirjoitetun viestin, lähettää tiedostoja tietokoneelta, luoda uusia keskusteluryhmiä sekä tarkastella valitun yhteyshenkilön tai ryhmän tietoja. Yhteyshenkilön nimen vieressä sijaitseva kuvake kuvaa yhteyshenkilön online/offline-tilaa. Lähetetyn viestin alakulmassa sijaitseva kuvake kertoo käyttäjälle, kun viesti on lähetetty, vastaanotettu tai luettu. (Kuva 19.)



KUVA 19. Pikaviestintäpalvelun käyttöliittymä

Kun käyttäjä kirjautuu tunnuksillaan pikaviestintäpalveluun, JavaScript-koodilla luodaan QuickBlox-sessio, haetaan käyttäjän tiedot ja luodaan yhteys QuickBloxin chat-palvelimeen. Tämän jälkeen koodilla haetaan keskustelujen dialogit ja käynnistetään joukko QuickBloxin tapahtumakuuntelijoita (event listener), jotka vastaavat muun muassa saapuvien viestien päivittämisestä viesti-ikkunaan. Lopuksi aktivoidaan yhteystietolistan ensimmäinen keskustelu ja päivitetään viesti-ikkuna.

Viestin lähetys tehdään kutsumalla QuickBloxin Chat-moduulin send()-funktiota, jonka parametreiksi annetaan vastaanottajan tunnusta kuvaava opponentId-muuttuja sekä msg-muuttuja, joka sisältää viestin tyypin ja lähetettävän viestin sisällön. (Kuva 20.)

```
var msg = {
    type: 'chat',
    body: "How are you today?",
    extension: {
        save_to_history: 1,
    }
};

var opponentId = 78;
QB.chat.send(opponentId, msg);

...

QB.chat.onMessageListener = onMessage;

function onMessage(userId, msg) {

}
```

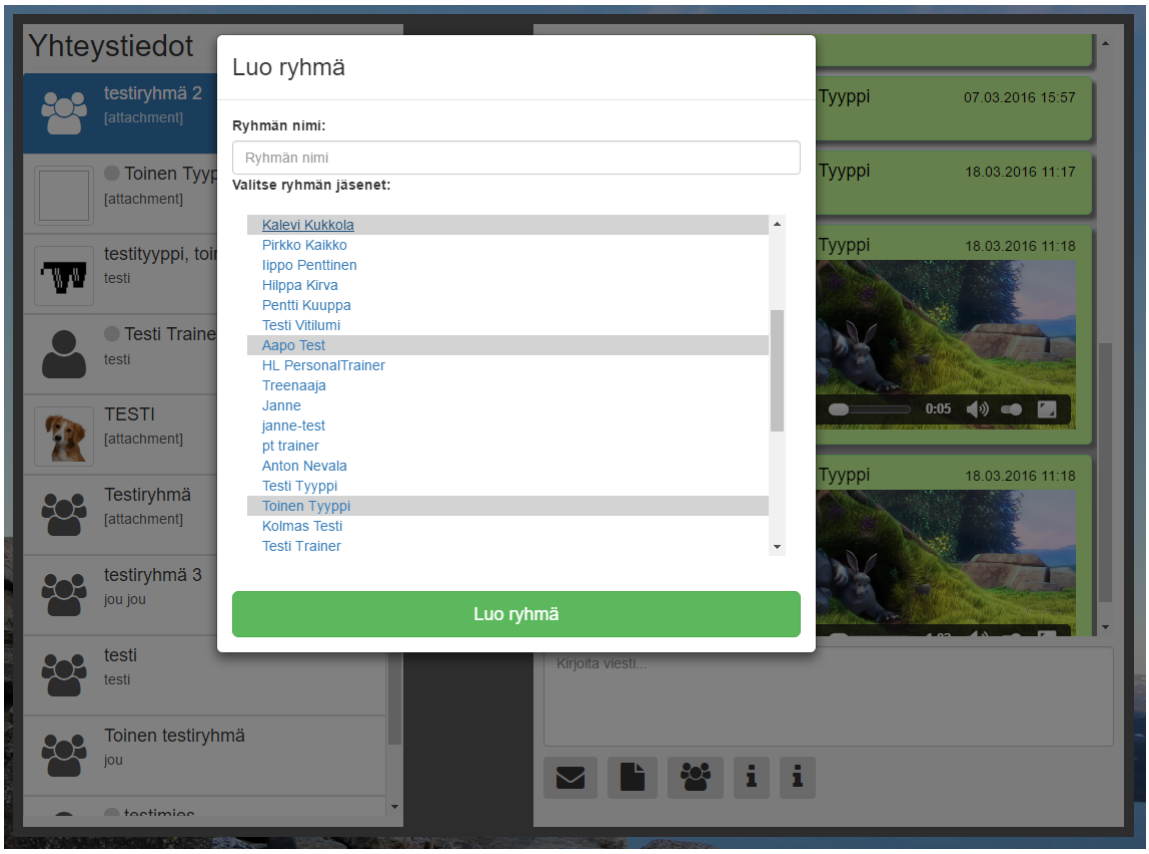
KUVA 20. Viestin lähettäminen (22)

Tiedostojen lähettämisessä käytettiin HTML-tunnistetta <input>, jonka type-attribuutin arvoksi annettiin ”file” ja joka piilotettiin käyttäen CSS:ää. Kun käyttäjä painaa Lisää tiedosto -painiketta, käyttäjälle avautuu ikkuna, josta lähetettävän tiedoston voi valita. Tiedosto lähetetään kutsumalla QuickBloxin Content-moduulin createAndUpload()-funktiota, joka tallentaa tiedoston QuickBlox-sovelluksen Content-luokkaan. Tämän jälkeen lähetetty tiedosto näytetään viesti-ikkunassa. (Kuva 21.)

```
// Upload a file to the Content module
var inputFile = $("input[type=file]")[0].files[0];

var params = {
    name: inputFile.name,
    file: inputFile,
    type: inputFile.type,
    size: inputFile.size,
    'public': false
};
QB.content.createAndUpload(params, function (err, response) {
    if (err) {
        console.log(err);
    } else {
        var uploadedFileId = response.id;

        // prepare a message
        //
        var msg = {
            type: currentDialog.type == 3 ? 'chat' : 'groupchat',
            body: "attachment",
            extension: {
                save_to_history: 1,
            }
        };
        msg["extension"]["attachments"] = [{
            id: uploadedFileId,
            type: 'photo'
        }];

        // send a message
        // ...
    }
});
```

KUVA 21. Tiedoston lähettäminen (22)

Uuden ryhmäkeskustelun luomista varten on tehty oma osionsa, joka on luotu käyttämällä JavaScriptin Bootstrap Modal -liitännäistä. Bootstrap Modal on ponnahdusikkuna, joka esitetään nykyisen sivun päällä. Käyttäjä voi asettaa ryhmän nimen ja valita ryhmän jäsenet käyttäjälistasta, jonka jälkeen keskusteluryhmä luodaan Luo ryhmä -painikkeella. (Kuva 22.)



KUVA 22. Uuden ryhmäkeskustelun luonti-ikkuna

Uusi ryhmäkeskustelu luodaan type-, occupant_ids- ja name-muuttujien avulla. Type-muuttujan arvo määrittelee keskustelun tyypin: kahden käyttäjän välistä keskustelua merkitään arvolla 1, ryhmäkeskustelua arvolla 2 ja avointa ryhmä-keskustelua, johon kuka tahansa voi liittyä, arvolla 3. Muuttujat kootaan yhteen params-muuttujaan, joka annetaan QuickBloxin Chat-moduulin dialogs.create()-funktion parametriksi. Lopuksi keskustelun jäsenille lähetetään ilmoitus ryhmän luomisesta ja keskustelu lisätään käyttäjän yhteystietolistaan. (Kuva 23).

```
var params = {
    type: 2,
    occupants_ids: [45, 78],
    name: "The A team"
};

QB.chat.dialog.create(params, function (err, createdDialog) {
    if (err) {
        console.log(err);
    } else {}
});
```

KUVA 23. Ryhmäkeskustelun luominen (22)

3.3.6 Lapsiteeman luominen WordPressissä

WordPress-sovelluksia varten on luotu monia erilaisia teemoja. Yksi niistä on Customizr, joka on monipuolinen, helposti muokattavissa ja laajennettavissa, sopeutuu tarvittaessa erilaisille laitteille ja tukee kaikkia nykyaikaisia selaimia. Teemasta on myös olemassa maksullinen versio Customizr Pro. (23.)

Koska WordPress-teemojen päivitykset kumoaisivat alkuperäisen teeman tiedostoihin tehdyt muutokset, oli teemasta luotava niin sanottu lapsiteema (child theme). Lapsiteemalla on oma kotikansionsa, joka sisältää vähintään seuraavat tiedostot: style.css ja functions.php.

Lapsiteeman luonti aloitettiin Customizr-teeman lisäämisellä WordPress-asennukseen. Teeman sisältävä zip-tiedosto purettiin WordPressin kotikansiossa sijaitsevaan themes-kansioon. Tämän jälkeen luotiin tyhjä kansio lapsiteemaa varten, johon sisällytetään kaikki alkuperäiseen teemaan tehtävät muutokset sekä muut omat tiedostot.

Seuraavaksi lapsiteeman kansioon luotiin style.css-tiedosto, jonka alussa on oltava kuvan 24 kaltainen kommenttilohko eli tyyliehdotusten otsikko, josta ilmenevät muun muassa teeman nimi, tekijä ja kuvaus.

```
/*
 Theme Name:   Twenty Fifteen Child
 Theme URI:    http://example.com/twenty-fifteen-child/
 Description:  Twenty Fifteen Child Theme
 Author:       John Doe
 Author URI:   http://example.com
 Template:     twentyfifteen
 Version:      1.0.0
 License:      GNU General Public License v2 or later
 License URI:  http://www.gnu.org/licenses/gpl-2.0.html
 Tags:         light, dark, two-columns, right-sidebar, responsive-layout, accessibility-ready
 Text Domain:  twenty-fifteen-child
*/
```

KUVA 24. Tyyliehdotusten otsikko (24)

Lopuksi lapsiteemaan oli luotava functions.php-tiedosto, jossa pääteeman ja lapsiteeman tyyliehdotukset sekä kaikki web-sovelluksen käyttämät CSS- ja JavaScript-tiedostot asetetaan jonoon. Tiedostoon luotiin funktio, jossa käytettiin

WordPressin wp_enqueue_style()- ja wp_enqueue_script()-funktioita tyyliehdotusten ja JavaScript-koodien jonoon asettamiseen. Funktioon lisätyt tyyliehdotukset ja ohjelmakoodit asetetaan jonoon wp_enqueue_scripts-toimintolla, joka lisätään kutsumalla WordPressin add_action()-funktiota, jonka parametreiksi annetaan halutun toiminnon nimi ja funktio. (Kuva 25.)

```php
<?php
function theme_enqueue_styles() {

    $parent_style = 'parent-style';

    wp_enqueue_style( $parent_style, get_template_directory_uri() . '/style.css' );
    wp_enqueue_style( 'child-style',
        get_stylesheet_directory_uri() . '/style.css',
        array( $parent_style )
    );
}
add_action( 'wp_enqueue_scripts', 'theme_enqueue_styles' );
?>
```

KUVA 25. Functions.php (24)

4 YHTEENVETO

Työn aiheena oli kehittää web-sovellusta Vitilumi Oy:lle sekä tutkia QuickBloxin käyttöä web-sovelluskehityksen työkaluna. Tavoitteena oli saada web-sovelluksen päärakenne ja perustoiminnallisuudet valmiiksi web-sovelluksen julkaisua varten. Annettuihin työtehtäviini kuuluivat käyttäjän sisäänkirjautumisen ja rekisteröinnin sekä pikaviestintäpalvelun toteutus QuickBloxin kirjastoja ja back endiä käyttäen. Aikarajojen puitteissa ehdittiin toteuttaa HTML-sivut käyttäjän rekisteröintiä ja sisäänkirjautumista varten sekä pikaviestintäpalvelu, joka sisälsi lähes kaikki vaaditut toiminnot. Pikaviestintäpalvelun toimivuutta ehdittiin kokeilla WordPress-sivustossa, mutta käytön aikana ilmeni ongelmia, joita ei kaikkia saatu korjattua. Varsinaista testausvaihetta ei web-sovelluksen kehityksessä ollut, vaan web-sovelluskehittäjien oli itse varmistettava omien tuotosten toimivuus sitä mukaa, kun ne valmistuivat.

Web-sovelluskehitys eteni suhteellisen hitaasti. Viikkopalavereissa sovitut tavoitteet eivät aina tulleet toteutetuiksi satunnaisista ongelmatilanteista johtuen, jolloin tavoitteita siirrettiin seuraavalle viikolle. Yhtenä syynä ongelmatilanteiden syntymiselle saattoi olla vähäinen kokemus web-sovelluskehittämisestä ja kehitysympäristöstä. QuickBloxin tullessa tutuksi kehittäminen sujui tehokkaammin.

QuickBloxin back end on helppo ottaa käyttöön omia web-sovelluksia varten lyhyen aiheeseen perehtymisen jälkeen. QuickBloxin valmiita esimerkkejä seuraamalla saa myös hyvän käsityksen QuickBlox-funktioiden käytöstä sekä back endin toimivuudesta. QuickBlox soveltuu erittäin hyvin käyttäjien välisen kommunikoinnin toteuttamiseen, sillä se sisältää tarvittavat työkalut muun muassa käyttäjäprofiilien luomista ja pikaviestintää varten.

Web-sovellusta kehittäessä oppi paljon uutta erityisesti HTML-, CSS- ja JavaScript-kielten käytöstä. Lisäksi työtä tehdessä oppi käyttämään muutamia uusia työkaluja, kuten WordPressiä.

Web-sovelluksen jatkokehitykselle jäi paljon tilaa. Ensisijaista olisi saada tähän mennessä toteutetut toiminnallisuudet toimimaan oikein, jonka jälkeen voitaisiin

alkaa kehittämään muita web-sovelluksesta puuttuvia toimintoja. Web-sovelluksen käyttöliittymää ja toimintoja voidaan hioa tehokkaammin toimiviksi. Kattavammalla testauksella voidaan varmistaa web-sovelluksen toimivuus ja löytää ohjelmointivirheitä korjattavaksi.

LÄHTEET

1. Web application. 2015. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Web_application. Hakupäivä 7.11.2015.

2. Laine, Harri 2006. Web-sovelluksen rakenteesta. Saatavissa: https://www.cs.helsinki.fi/u/laine/tikas/material/web_sovellus.html. Hakupäivä 7.11.2015.

3. Long, Josh 2012. I Don't Speak Your Language: Frontend vs. Backend. TreeHouse Blog. Saatavissa: http://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend. Hakupäivä 7.11.2015.

4. Girdley, Michael 2014. Front-end vs Back-end. CodeUp. Saatavissa: http://codeup.com/different-types-of-programmers-front-end-vs-back-end/. Hakupäivä 7.11.2015.

5. QuickBlox company details. CrunchBase. Saatavissa: https://www.crunchbase.com/organization/quickblox. Hakupäivä 7.11.2015.

6. FAQ. QuickBlox. Saatavissa: https://quickblox.com/faq/. Hakupäivä 19.5.2016.

7. About Wordpress. WordPress. Saatavissa: https://wordpress.org/about/. Hakupäivä 7.11.2015.

8. Wordpress. 2015. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/WordPress. Hakupäivä 7.11.2015.

9. Usage of content management systems for websites. 2015. W3Techs Web Technology Surveys. Saatavissa: http://w3techs.com/technologies/overview/content_management/all/. Hakupäivä 7.11.2015.

10. HTML & CSS. W3C. Saatavissa: https://www.w3.org/standards/webdesign/htmlcss. Hakupäivä 22.3.2016.

11. HTML. 2016. Wikipedia. Saatavissa: https://fi.wikipedia.org/wiki/HTML. Hakupäivä 22.3.2016.

12. HTML5. W3C. Saatavissa: https://www.w3.org/TR/html/. Hakupäivä 22.3.2016.

13. Saarikumpu, Osmo 2016. Johdanto CSS-tyyliehdotuksien käyttöön Web-sivuilla. Saatavissa: http://weppipakki.com/css/tekstit/cssintro.htm. Hakupäivä 22.3.2016.

14. Heilmann, Christian 2008. CSS Basics. Saatavissa: https://dev.opera.com/articles/css-basics/. Hakupäivä 13.4.2016.

15. Introduction. 2016. Mozilla Developer Network. Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction. Hakupäivä 29.3.2016.

16. A Short History of JavaScript. 2012. W3C. Saatavissa: https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript. Hakupäivä 22.3.2016.

17. What is PHP? 2016. The PHP Group. Saatavissa: http://php.net/manual/en/intro-whatis.php. Hakupäivä 20.4.2016.

18. What can PHP do? 2016. The PHP Group. Saatavissa: http://php.net/manual/en/intro-whatcando.php. Hakupäivä 20.4.2016

19. History of PHP. 2016. The PHP Group. Saatavissa: http://php.net/manual/en/history.php.php. Hakupäivä 19.4.2016.

20. JavaScript. QuickBlox Docs. Saatavissa: http://quickblox.com/developers/Javascript. Hakupäivä 14.11.2015.

21. Sample-users-javascript. QuickBlox Docs. Saatavissa: http://quickblox.com/developers/Sample-users-javascript. Hakupäivä 4.5.2016.

22. Web XMPP Chat Sample. QuickBlox Docs. Saatavissa: http://quickblox.com/developers/Web_XMPP_Chat_Sample. Hakupäivä 4.5.2016.

23. Customizr WordPress Theme. 2016. Press Customizr. Saatavissa: https://wordpress.org/themes/customizr/. Hakupäivä 25.4.2016

24. Child Themes. WordPress. Saatavissa: https://codex.wordpress.org/Child_Themes. Hakupäivä 4.5.2016