TAMPEREEN
AMMATTIKORKEAKOULU

Web-kehitys ASP.NET MVC:llä

Joonas Eloranta

Sami Korhonen

Opinnäytetyö
Joulukuu 2016
Tietotekniikka
Ohjelmistotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

ELORANTA JOONAS & KORHONEN SAMI
Web-kehitys ASP.NET MVC:llä

Opinnäytetyö 32 sivua, joista liitteitä 17 sivua
Joulukuu 2016

Työn tarkoituksena oli opiskella web-kehitystä, ja samalla toteuttaa asiakkaalle web-sovellus ASP.NET MVC:llä. Sovelluksen tavoitteena oli korvata asiakkaan olemassa oleva bisnesprosessi suoraviivaisemmalla ja modernimmalla tavalla, joka tehostaa ja helpottaa loppukäyttäjien työntekoa.

Työssä käydään läpi sovelluksen kehitys kokonaisuudessaan aina määrittely- ja suunnitteluvaiheesta sovelluksen toteutukseen. Lisäksi työssä kerrotaan käytetyistä teknologioista ja esimerkkisovelluksen tavoitteista, rakenteesta ja lopputuloksesta.

Asiasanat: ASP.NET, MVC, IIS, C#, .NET, JavaScript, Web-kehitys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Information Technology
Software Engineering

ELORANTA JOONAS & KORHONEN SAMI
Web development with ASP.NET MVC

Bachelor's thesis 32 pages, appendices 17 pages
December 2016

The purpose of the thesis was to study web development and to create a web application using ASP.NET MVC for client. The goal for application was to replace existing business process with more straightforward and modern way. Application aims to improve the work efficiency of end users.

The thesis describes the application development process as a whole, starting from the design and specification phases moving on to development of the application. In addition the thesis goes through technologies, aims, structure and result of the example application.

Key words: ASP.NET, MVC, IIS, C#, .NET, JavaScript, Web development

SISÄLLYS

LYHENTEET JA TERMIT

.NET Framework	Microsoftin ohjelmistokirjasto, joka koostuu CLR –ympäristöstä ja luokkakirjastoista (FCL). Perustana kaikille .NET –pohjaisille kielille.
C#	.NET –perheeseen kuuluva ohjelmointikieli. Kehitettiin C:n ja C++:n pohjalta.
CLR	Common Language Runtime –virtuaaliympäristö toimii välikielen ja konekielen välisenä tulkkina .NET-frameworkissa.
Entity Framework	.NET –kirjasto mahdollistamaan Microsoftin SQL Server –tietokantojen käytön sovelluksissa.
FCL	.NET Framework Class Library, kirjasto joka tarjoaa .NET –kielille pääsyn järjestelmän toiminnallisuuteen.
Hallittu koodi	Sovelluskoodia, joka voidaan ajaa vain CLR –virtuaaliympäristössä. Yleisesti .NET –kielet kirjoitetaan tässä muodossa.
Konekieli	Konekieli on tietokoneen suoraan ymmärtämään muotoon käännettyä tai kirjoitettua kieltä.
MVC	Model View Controller –arkkitehtuuri internet-sivujen rakentamista varten. Model on tietomalli, joka rakennetaan kontrollerilla ja viedään näytettäväksi viewille.
SQL	Structured Query Language on kieli tietokantojen hallinnointiin. Käytetään yleisesti myös viittaamaan koko tietokantaan.
Välikieli	.NET –pohjaiset kielet käännetään välikieleksi, josta ajettaessa CLR tulkkaa kielen konekieleksi.

1 JOHDANTO

Tässä opinnäytetyössä tutustutaan web-sovelluksen kehittämiseen ASP.NET MVC:llä esimerkkiprojektin kautta. Web-sovelluksilta vaaditaan nykypäivänä yhtä vaativia ja monipuolisia toiminnallisuuksia kuin työpöytäohjelmilta, jonka lisäksi web-sovellusten tulee olla usein selain- ja laiteriippumattomia. Nämä tekijät aiheuttavat omat haasteensa web-kehitykseen.

Työssä esimerkkinä käytetty projekti toteutettiin käyttämällä ASP.NET MVC-kehystä. Sovelluksen ohjelmointikielenä käytettiin C#:a, ja käyttöliittymä toteutettiin HTML5- ja JavaScript -kielillä. Sovelluksen tietomalli toteutettiin Entity Frameworkin koodi-ensin -työkalulla, jolla luotiin SQL –tietokanta.

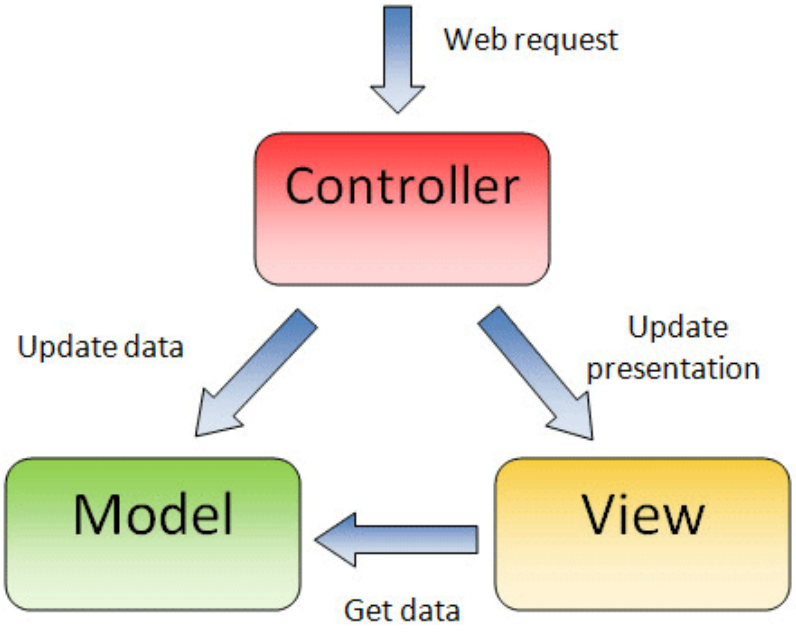
Työn kakkosluvussa esitellään ASP.NET –kehitykseen käytettäviä, tai siihen suoraan liittyviä teknologioita. Kolmosluku käsittelee web-sovelluksen suunnittelu- ja kehitysprosessia ottaen kantaa sekä asiakkaan että sovelluksen tekijän rooleihin. Neloskappaleessa on pohdintaa työn tuloksista.

2 TEKNOLOGIAT

2.1 ASP.NET MVC

ASP.NET MVC on Microsoftin kehittämä web-sovelluskehys, joka toteuttaa MVC -arkkitehtuuria. ASP.NET MVC:n ensimmäinen versio julkaistiin maaliskuussa 2009. Tämä pohjautui olemassa olevaan ASP.NET –sovelluskehykseen. Alkuperäisen ASP.NET:in tavoitteena oli mahdollistaa dynaamisten nettisivujen kehitys .NET teknologiaa käyttäen. Microsoft kuitenkin halusi tarjota kehittäjille mahdollisuuden käyttää MVC-arkkitehtuuria web-kehityksessä, joka mahdollistaa esimerkiksi haasteiden eriyttämisen, sekä purkaa nettisivun ja palvelimen välistä suhdetta erittelemällä koodin eri tasoihin. Näin syntyi ASP.NET MVC. (Ghani, 2013)

ASP.NET MVC:llä kehittäessä palvelinpään koodin voi toteuttaa joko C#:lla tai Visual Basic.NET:llä. Käyttöliittymän kehityksessä puolestaan käytetään HTML:ää yhdistettynä jompaankumpaan edellä mainituista kielistä, sekä JavaScriptiä ja CSS:ää. Esimerkkisovelluksessa käytettiin C#:a.



KUVA 1. MVC –arkkitehtuuri (Gupta, 2015)

2.2 C#

C# on Microsoftin kehittämä oliopohjainen ohjelmointikieli. Microsoft julkisti C#:n 26. kesäkuuta vuonna 2000, ja kielen versio 1.0 ilmestyi 13. helmikuuta vuonna 2002. C# on kehittynyt tasaisesti siitä asti, kuten taulukosta 1 näkee. Nykyinen versio on 6.0, joka julkaistiin 20. heinäkuuta vuonna 2015. Kieli kehitettiin C:n ja C++:n pohjalta, mutta C# on ottanut myös vahvoja vaikutteita muista kielistä, kuten Delphistä ja Javasta. Kielen sanotaankin muistuttavan syntaksiltaan paljon sekä C++:aa sekä Javaa. (CSharp-Station.com, 2016)

TAULUKKO 1. C#:n kehitys

C#:n versio	Julkaisu	Ensimmäisenä tuettu	.NET versio
1.0	Helmikuu 2002	Visual Studio .NET 2002	.NET Framework 1.0
1.2	Huhtikuu 2003	Visual Studio .NET 2003	.NET Framework 1.1
2.0	Marraskuu 2005	Visual Studio 2005	.NET Framework 2.0
3.0	Marraskuu 2007	Visual Studio 2008	.NET Framework 3.5
4.0	Huhtikuu 2010	Visual Studio 2010	.NET Framework 4
5.0	Elokuu 2012	Visual Studio 2012	.NET Framework 4.5
6.0	Heinäkuu 2015	Visual Studio 2015	.NET Framework 4.6

C# on niin kutsuttu hallitun koodin kieli, eli vaatii toimiakseen CLR-virtuaaliympäristön. Yleensä tässä käytetään .NET Framework – kirjastoa, mutta myös muita vaihtoehtoja on olemassa, kuten avoimen lähdekoodin Mono, joka mahdollistaa C# -ohjelmien ajamisen esimerkiksi Unix –ympäristössä. Myös Microsoft itse on julkaissut tuoreen .NET Core-frameworkin joka on avoin alustariippumaton .NET –toteutus. Koska C# on hallitun koodin kieli, ei tätä käännetä suoraan konekieleksi, vaan kääntäjä tuottaa välikieltä, jota CLR – ympäristö tulkitsee konekieleksi lennosta. Tämän vuoksi C# on riippuvainen omista CLR –ympäristöistään, sillä välikieli ei sellaisenaan ole tietokoneen tulkittavissa. (Mono Project, 2016)

C# ei itsessään sisällä monia yleisiä ohjelmointikielille ominaisia toimintoja, kuten esimerkiksi tiedostonkäsittelyä, vaan nämä kielelle tarjoaa .NET FCL -kirjasto. Tästä tulee se etu, että koska kaikki .NET –pohjaiset ohjelmointikielet tuottavat samaa välikieltä, voidaan samaa FCL –kirjastoa käyttää kaikissa näistä kielistä, sekä myös kaikki kielet

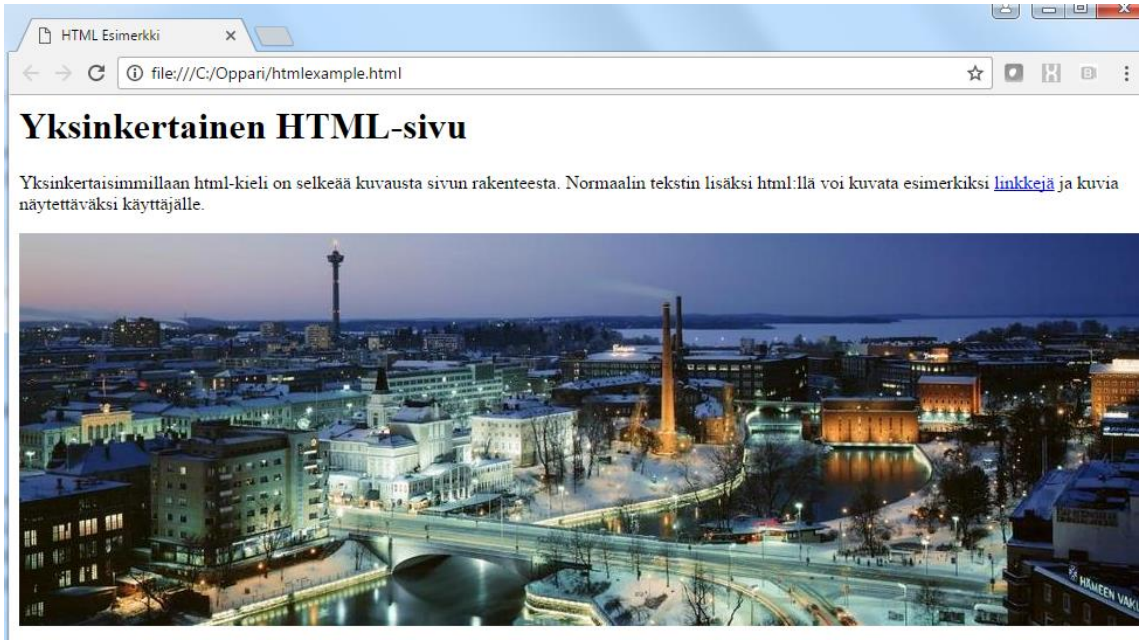
voivat käyttää kaikilla muilla .NET –kielillä kirjoitettuja apukirjastoja hyödykseen. (CSharp-Station.com, 2016) (Lander, 2015)

2.3 HTML

HTML on standardoitu merkintäkieli, jota käytetään verkkosivujen rakenteen ja sisällön kuvaamiseen. Kaikkien verkkosivujen sisältö on kuvattuna HTML:llä. Kuvaus kertoo selaimelle miten selaimen tulee näyttää verkkosivun teksti ja kuvat käyttäjälle. HTML-kuvaus koostuu elementeistä, joita voi olla sisäkkäin ja peräkkäin. Elementti koostuu alkutunnisteesta, sisällöstä ja lopputunnisteesta. Elementin avainsana ja attribuutit määrittelevät, miten sen sisältö näytetään ruudulla. Kuvassa 2 on yksinkertaista HTML-koodia, jonka tuloksena selain näyttää käyttäjälle kuvan 3 näkymän.

```
<html>
<head>
<title>
HTML Esimerkki
</title>
</head>
<body>
<h1>
Yksinkertainen HTML-sivu
</h1>
<p>
Yksinkertaisimmillaan html-kieli on selkeää kuvausta sivun rakenteesta. Normaalin tekstin lisäksi html:llä voi kuvata esimerkiksi
<a href="https://lmgtfy.com/?q=html+linkki">linkkejä</a> ja kuvia näytettäväksi käyttäjälle.</p>
<img src="http://www.ria.fi/files/1297/960/Tampere2etus.jpg"/>
</body>
</html>
```

KUVA 2. Esimerkkisivun html-koodi



KUVA 3. Esimerkkisivun näkymä web-selaimessa

HTML-standardia on ylläpitänyt vuodesta 1994 standardointia varten perustettu kansainvälinen W3C-konsortio. Vuosien varrella HTML:stä on julkaistu useita eri versioita, joista viimeisin on HTML 5. Virallisesti se julkaistiin 28.10.2014. HTML 5 on ensimmäinen versio, jossa HTML-kieleen on tuotu mukaan mahdollisuudet toistaa videota ja ääntä, kun aiemmin on tarvinnut turvautua kolmannen osapuolen liitännäisiin. Lisäksi HTML 5 mahdollistaa vektorigrafiikan käyttämisen osana HTML:ää. Selainten tuki HTML:n eri versioille on ollut kirjavaa, eikä kaikki selaimet ole osanneet näyttää sivua standardin mukaisesti. Nykyään kaikki modernit selaimet tukevat varsin hyvin HTML 5-standardia. (Rouse, 2005)

2.4 JavaScript

JavaScript on alustariippumaton oliopohjainen tulkittava ohjelmointikieli. JavaScriptin historia alkoi vuonna 1995, kun internetin yleistyessä Netscape halusi selaimelleen helpon ja kevyen tulkittavan ohjelmointikielen. HTML ei riittänyt kehitysympäristönä vastaamaan uusiin tarpeisiin tuottaa entistä monipuolisempaa sisältöä internetiin, joten sen rinnalle tarvittiin uusi kieli. JavaScriptin syntaksiin otettiin paljon vaikutteita C++ -kielestä, mutta siihen haluttiin yksinkertaisempi syntaksi ja dynaamisen muistin hallinta. Yleisestä harhaluulosta poiketen JavaScriptillä ei ole muuta tekemistä Javan kanssa kuin nimi. Ensimmäinen versio julkaistiin nimellä LiveScript, mutta Netscapen ja Sunin yhdistyessä nimeksi muutettiin JavaScript markkinoinnin edistämiseksi. Pian JavaScriptin julkaisun jälkeen Microsoft julkaisi oman vastaavan ohjelmointikielen nimellä JScript. Vuonna 1997 näitä kieliä yhdistämään perustettiin ECMAScript-standardi, joka määrittelee miten sitä käyttävien kielten tulee toimia. (Severance, 2012)

JavaScript poikkeaa perinteisistä luokkapohjaisista kielistä huomattavasti. JavaScriptissä ei ole luokkia, vaan kaikki muuttujat ovat yksinkertaisesti objekteja, joiden ominaisuuksia voidaan muokata milloin vain objektin elinkaaren aikana. JavaScriptissä voidaan myös käyttää mitä vain funktiota objektin alustajana, ja kaikki objektit voivat periä toisilta objekteilta ominaisuuksia. Dynaamisuuden vuoksi JavaScriptissä objekteja ei myöskään tarvitse tyypittää, vaan objektin tyyppi asetetaan automaattisesti ohjelman ajossa objektin arvosta riippuen. Nämä kaikki ominaisuudet luovat JavaScriptistä todella monipuolisen ja joustavan kielen, jolla voi tehdä käytännössä mitä vain selaimeen halutaan. (Mozilla, 2016)

2.5 IIS

IIS, eli Internet Information Services, on Microsofin kehittämä Web –serveri. IIS tunnettiin aiemmin nimellä Internet Information Server, ja sen ensimmäinen versio julkaistiin ilmaisena lisäosana Windows NT 3.51:lle osana tämän Service Pack 3 -päivitystä. Nykyinen versio on 10, ja se tulee ilmaisena Windows 10:n ja Windows Server 2016:n osana. IIS on tällä hetkellä maailman käytetyin internetin palvelinohjelmisto. (TheFreeDictionary, 2015) (NetCraft, 2016)

IIS tukee ASP-, ASP.NET- ja PHP –pohjaisia sivustoja. Näiden lisäksi IIS:istä löytyy myös esimerkiksi FTP- sekä SMTP -palvelintuki. IIS tukee useita eri autentikointimekanismeja, ja IIS:n toimintaa on mahdollista lisätä moduulien avulla. Moduuleita on mahdollista tehdä itse, tai hakea esimerkiksi Microsoftin Web Platform Installer –sovelluksen avulla. (TheFreeDictionary, 2015)

3 SUUNNITTELU JA TOTEUTUS

Internet-sovelluskehityksessä, kuten sovelluskehityksessä yleensäkin, on tarve ensin suunnitella mitä lähdetään tekemään. Asiakas määrittää mitä sovelluksen pitäisi pääpiirteissään tehdä, ja tämän jälkeen sitä lähdetään suunnittelemaan. Kun suunnitelmat ovat kunnossa, aloitetaan itse sovelluksen tekeminen. Tekeminen on harvoin kuitenkaan yksinkertaista, vaan suunnitelmat saattavat muuttua ja tarkentua myös tekemisvaiheessa. Sovelluskehitys onkin nykyään yleensä molemminpuolista vuoropuhelua, jossa asiakas ja toteuttaja keskustelevat ja pohtivat sovellusta sen eri vaiheissa.

3.1 Suunnittelu

Suunnitellessa on tärkeää, että roolijako on asiakkaan ja toteuttajan välillä selvä. Asiakkaalta valitaan henkilö toimimaan projektin omistajan roolissa, jolla on vastuullaan keskittyä laajempaan kokonaiskuvaan. Tämä vastaa siitä, että sovellus toteuttaa sen tehtävän jota varten se on suunniteltu, ja tarvittaessa auttaa tekijöitä ymmärtämään asiakkaan bisnesmallia, tavoitteita ja muita mahdollisia laajemman kokonaiskuvan vaatimia ongelmia. Sanotaan, että projektin omistajan tulisi keskittyä tarjoamaan ongelmia eikä ratkaisuja. Ongelma on esimerkiksi ”miten saada vieraat ostamaan enemmän?”, kun taas ratkaisu on ”yksinkertaistakaa ostoprosessia”. Jos projektin omistaja alkaa tarjota ratkaisuja, hämärtää tämä suunnittelijan ja omistajan välisiä rooleja, eikä lopputulos välttämättä ole silloin paras mahdollinen.

Suunnittelija taas on sovellustoimittajan henkilö, ja tämän vastuulla on tarjota ratkaisuja projektin omistajan ongelmiin. Suunnittelija päättää miltä sivun tulisi näyttää ja miten eri asiat tulevat toimimaan sivulla. Suunnittelija alkaa koostaa suunnitteludokumenttia projektin sisällöstä, ja pikkuhiljaa dokumentti kehittyy kun suunnittelija ja asiakas projektin omistajan johdolla käyvät läpi mitä ja miten sovellus tulee tekemään, on kasassa dokumentaatio jonka perusteella sovellusta voidaan lähteä tekemään. (Elvio web design, 2016) (Reimer, 2011)

3.2 Toteutus

Sovelluksen toteutus alkaa, kun ensimmäiset suunnitelmat ovat valmiita. Toteutusvaiheen alussa on tärkeää rakentaa toimiva perusta sovellukselle jonka päälle lähdetään lisäämään ominaisuuksia. Asiakkaan kannalta on hyvä, että sovellusta lähdetään rakentamaan pienissä palasissa, ja tarjoten asiakkaalle mahdollisimman aikaisin testiympäristö, jossa he voivat tarkastella ja testata sovelluksen toimivuutta, myöskin keskittyen aina pieneen palaan kerrallaan. Toteutuksen aikana suunnitelmat yleensä myös jalostuvat saadun palautteen pohjalta, joskus enemmän, joskus vähemmän. Testauksen aloittaminen aikaisin ja sen tekeminen jatkuvana prosessina on tärkeää, sillä mitä aikaisemmin virheitä huomataan, sitä helpompi niitä on korjata ja suunnitelmaa muuttaa. (StrateComm, 2016) (Reimer, 2011)

4 POHDINTA

Sovelluksen kehittäminen alusta loppuun oli todella opettava kokemus. Varsinkin määrittelyvaihe oli opettavaista, koska siitä ei juuri kokemusta etukäteen ollut. Emme saaneet asiakkaalta mitään valmiita speksejä, vaan asiakkaan tarpeet ja tavoitteet tuli saada selville palaveraamalla ja kyselemällä oikeita kysymyksiä. Vaikka päätavoitteena olikin korvata asiakkaan vanha järjestelmä, oli tämä mahdollisuus samalla parantaa loppukäyttäjien käyttökokemusta ja tehdä heidän työnteosta helpompaa. Mielestämme tässä onnistuttiin varsin hyvin. Loppukäyttäjät olivat tyytyväisiä sovellukseen, ja se on aktiivisessa tuotantokäytössä.

Ongelmilta projektissa ei kuitenkaan vältytty. Vaikka lopputulokseen voikin olla tyytyväinen, työmääräarviot eivät pitäneet vaan toteutukseen kului ennakoitua enemmän aikaa. Tämä on kuitenkin ymmärrettävää, koska kyseessä oli kehittäjien ensimmäinen suurempi projekti, ja paljon haasteita voi laittaa oppimisen piikkiin. Onneksemme projektilla ei ollut varsinainen kiire valmistua, joten aikataulua voitiin muokata projektin edetessä. Suurimpana haasteena oli kokonaiskuvan saaminen: pieniin osiin pilkottuina yksittäiset osat valmistuivat aikataulussa, mutta niiden yhteen liittämiselle ei oltu varattu riittävästi aikaa. Lisäksi omat haasteensa käyttöliittymän toteutukseen toi selainyhteensopivuus, sillä kaikista lupauksista huolimatta Internet Explorer ei edelleenkään toteuta kaikkea JavaScriptiä ja tyylejä samalla tavalla kuin muut yleisimmät selaimet.

Kokonaisuutena projekti antoi hyvän kuvan siitä, mitä sovelluksen kehitys sisältää ja mitä se vaatii kehittäjiltä. Olemme aiemminkin olleet mukana projekteissa, mutta tämä oli ensimmäinen kerta kun olimme mukana kaikissa projektin vaiheissa, ja olimme vastuussa suurimmasta osasta kehitystyötä. Kaiken kaikkiaan projekti antoi hyvät eväät jatkoa ajatellen ohjelmistokehityksen parissa, sekä opetti paljon työskentelystä asiakkaan kanssa.

LÄHTEET

CSharp-Station.com 2016. C# Station. Luettu 5.12.2016. http://www.csharp-station.com/

Ghani 2013. Difference betweeen ASP.NET WebForms and ASP.NET MVC. Luettu 2.12.2015. https://www.codeproject.com/articles/668182/difference-betweeen-asp-net-webforms-and-asp-net-m

Gupta 2015. Spring MVC Hello World Example. Luettu 10.12.2016. http://howtodoinjava.com/spring/spring-mvc/spring-mvc-hello-world-example/

Elvio web design 2016. Role Of Client And Designer. Luettu 11.12.2016. http://www.elviowebdesign.com/role-of-client-and-designer

Lander 2015. Announcing .NET Framework 4.6. Luettu 4.12.2016. http://blogs.msdn.com/b/dotnet/archive/2015/07/20/announcing-net-framework-4-6.aspx

Mono Project 2016. Cross platform, open source .NET framework. Luettu 5.12.2016. http://www.mono-project.com/

Mozilla 2016. Details of the object model. Luettu 20.6.2016. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model

NetCraft 2016. November 2016 Web Server Survey. Luettu 5.12.2016. https://news.netcraft.com/archives/2016/11/22/november-2016-web-server-survey.html

Reimer 2011. Following A Web Design Process. Luettu 11.12.2016. https://www.smashingmagazine.com/2011/06/following-a-web-design-process/

Rouse 2005. HTML (Hypertext Markup Language). Luettu 2.12.2016. http://searchsoa.techtarget.com/definition/HTML

Severance 2012. JavaScript: Designing a Language in 10 Days. Luettu 20.6.2016. http://www.computer.org/csdl/mags/co/2012/02/mco2012020007.pdf

StrateComm 2016. How involved should the client be in Web Design. Luettu 11.12.2016. http://www.stratecomm.com/faqs/howinvolved/

TheFreeDictionary 2015. Internet Information Server. Luettu 2.12.2015. http://encyclopedia2.thefreedictionary.com/Internet+Information+Server

LIITTEET

Liite 1. Esimerkkisovelluksen kehitys, LUOTTAMUKSELLINEN.