OULUN SEUDUN
AMMATTIKORKEAKOULU

Jarmo Tolonen

WEB-SOVELLUKSEN KIRJAUTUMISEN MÄÄRITTELY JA TOTEUTUS

WEB-SOVELLUKSEN KIRJAUTUMISEN MÄÄRITTELY JA TOTEUTUS

Jarmo Tolonen
Opinnäytetyö
Kevät 2013
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä(t): Jarmo Tolonen
Opinnäytetyön nimi: Web-sovelluksen kirjautumisen määrittely ja toteutus
Työn ohjaaja(t): Liisa Auer
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 43 + 2

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena oli määritellä, suunnitella sekä toteuttaa web-sovellukselle sovellukseen kirjautumisen mahdollistava komponentti. Komponentti pitää sisällään kirjautumiseen vaadittavan käyttöliittymän joka toimii samalla uuden kehitteillä olevan sovelluksen etusivuna. Komponentti sisältää lisäksi tekniset toiminnallisuudet käyttäjien autentikointia ja salasanojen käsittelyä varten. Työssä perehdytään lisäksi automaattisen rajapintadokumentaation luontiin Adoben ColdFusion web-teknologiaan perustuvissa järjestelmissä. Toimeksiantaja toimi Suomen harjoitusyritystoiminnan keskus - Finnish Practice Enterprises Centre (FINPEC).

Työssä kuvataan kirjautumisen ja käyttäjäautentikaation suunnittelu ja toteutus Adobe Coldfusion web-teknologialla toteutettaviin sovelluksiin. Annettua kehitystehtävää tarkastellaan teknisestä näkökulmasta, selvittäen Adobe ColdFusion web-sovelluspalvelimen toimintaa sekä samalla tutkien vaadittavia teknisiä ratkaisuja käyttäjäautentikaation rakentamiseen. Työssä huomioidaan nykyaikaiselle web-sovellukselle vaadittavat tietoturvaominaisuudet ja -käytänteet. Lisäksi työssä kuvataan automaattisen rajapintadokumentaation luomiseen vaadittavia käytänteitä ja toimintaa.

Opinnäytetyön tietosisällössä keskitytään tarkastelemaan toteutettavan komponentin vaatimuksille asetettuihin tärkeisiin elementteihin liittyviä asiakokonaisuuksia alkaen web-sovellusten rakenteesta siirtyen käyttäjien autentikaatiossa tarvittaviin teknisiin ratkaisuihin. Tietosisällössä tarkastellaan lisäksi käytettyjä web-teknologioita asiakaspuolen sekä palvelin-puolen osalta.

Työn toiminnallisen vaiheen tuloksena syntyi kehitteillä olevalle sovellukselle kirjautumissivu web-sovellukseen, kirjautumisen ja käyttäjien autentikoinnin mahdollistava ohjelmistokomponentti sovellukselle. Työssä toteutetun web-sovelluksen sovelluslogiikka toteutettiin Adobe ColdFusion web-ohjelmointitekniikkaa käyttäen. Sovelluksen käyttöliittymä toteutettiin web-ohjelmoinnissa perinteisesti käytettävillä HTML (HyperText Markup Language) -, CSS (Cascading Style Sheets) - ja JavaScript-tekniikoilla. Web-sovelluksen tietokantaratkaisuna toimii MySQL-tietokanta. Automaattinen rajapintadokumentaatio luotiin ColdFusion Component Doc -rajapintaa apuna käyttäen.

Asiasanat: web-sovellus, kirjautuminen, autentikointi, salaus, suojaus, Cold Fusion

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author(s): Jarmo Tolonen
Title of thesis: Web application for login specification and implementation
Supervisor(s): Liisa Auer
Term and year when the thesis was submitted: Spring 2013 Number of pages: 43 + 2

ABSTRACT

The aim of this bachelor thesis was to define, design and implement a web application login component. The component includes the required login form and view of the user interface that serves as an application home page. The component also includes the technical functionalities of user authentication and password processing. The automatic documentation created by Adobe ColdFusion web-technology-based systems was also studied. The commissioner of the thesis was Finnish Practice Enterprises Centre (FINPEC).

The thesis describes the login and user authentication design and implementation using Adobe ColdFusion web technologies. The technical aspects of the development task have been studied from the operational side by using Adobe ColdFusion web application server. The Thesis also focused on the technical solutions required for the construction of user authentication. The work takes into account the required security features and practices of a modern web application. In addition, the study describes required practices and performances in the automatic creation of the documentation.

The content of the thesis focuses on the implementation of component requirements: the web application structure and the technical solutions for user authentication. Both client-side and server-side web technologies are taken in to account in this study.

The result of this thesis was the login page of the web application, login and user authentication component for the application. The work was implemented using Adobe ColdFusion web programming technique. The user interface of application was carried out using HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript technologies. Web application uses MySQL database. The documentation was created with the help of the ColdFusion Component Doc.

Keywords: web application, login, authentication, encryption, protection, Cold Fusion

SISÄLLYS

1 JOHDANTO 6

2 KIRJAUTUMISKOMPONENTIN PERUSTEET 8

2.1 Web-sovellus 8

2.2 Suojattu yhteys 10

2.4 Autentikointi 11

2.4 Salasanojen suojaus 13

2.5 Ohjelmakoodin dokumentointi 15

3 KÄYTETYT ASIAKASTEKNIIKAT 16

3.1 JavaScript 16

3.2 jQuery 20

4 ADOBE COLDFUSION 22

4.1 Adobe ColdFusion 10 -sovelluspalvelin 22

4.2 ColdFusion-merkkikieli 23

4.3 ColdFusion-scriptikieli 25

4.4 ColdFusion-komponentti 26

4.5 ColdFusion-komponenttidokumentoija 29

5 WEB-SOVELLUKSEN KIRJAUTUMISKOMPONETTI 32

5.1 Vaatimukset 32

5.2 Suunnittelu 33

5.3 Toteutus 33

6 POHDINTA 40

LÄHTEET 42

LIITTEET 44

1 JOHDANTO

Tässä opinnäytetyössä käsitellään erään kehitteillä olevan web-sovelluksen kirjautumiskomponentin määrittelyä, suunnittelua sekä toteutusta. Lisäksi työssä perehdytään tuotettavan ohjelmakoodin dokumentointiin automaattista rajapintadokumentaatiota hyväksi käyttäen.

Opinnäytetyön toimeksiantajana toimii Suomen harjoitusyritystoiminnan keskus - Finnish Practice Enterprises Centre (FINPEC). FINPEC tukee ja avustaa harjoitusyritysten perustamisessa ja niiden päivittäisten asioiden hoidossa. FINPEC tarjoaa harjoitusyritystoimintaan tällä hetkellä VEPEN (Virtual Environment for Practice Enterprises Network) web-sovellusta, jonka avulla yritystoimintaa voidaan harjoittaa virtuaalisessa oppimisympäristössä. Suomessa sovellusta käyttää 30 oppilaitosta ja noin 110 harjoitusyritystä. Vuosittain toiminnassa on mukana yhteensä noin 1500 toisen asteen, ammattikorkeakoulun ja aikuiskoulutuksen opiskelijaa. FINPEC kuuluu kansainväliseen EUROPEN harjoitusyritysjärjestöön ja toimii Suomen edustajana oppimismuodon kansainvälisissä kehittämisryhmissä. FINPEC on osa Oulun seudun koulutuskuntayhtymää ja toimii Oulun seudun ammattiopiston alaisuudessa.

Opinnäytetyön toimeksiantaja on kehittämässä harjoitusyritystoiminnan jatkoksi oman elämän hallintasovellusta, käytettäväksi ensisijaisesti yläasteen ja toisen asteen opiskelijoille. Sovelluksen tarkoituksena on antaa opiskelijoille oppimisympäristö rahan käyttöä sekä sopimusten solmimista ja etuuksien hakemista varten. Lisäksi kehitettävän sovelluksen tarkoitus on laajentaa harjoitusyritystoiminnan oppimiskokemusta myös harjoitusyritysten työntekijöihin, joille maksetaan harjoitusyrityksestä palkkaa. Näille työntekijöille luodaan henkilökohtaiset tilit palkanmaksua varten elämän hallintasovellukseen, mikä mahdollistaa oppimiskokemuksen laajentumisen samalla oman elämän hallinnan osalta.

Toteutettavan web-sovelluksen kehitystyö aloitettiin opinnäytetyötä aloittaessani täysin alusta. Katsoimme toimeksiantajan kanssa parhaaksi aloittaa sovelluksen rakentaminen kirjautumiskomponentista, jonka avulla luodaan toimintaperusta kehitettävälle sovellukselle. Web-sovelluksen kehitystyön lähtökohtana oli toimeksiantajan laatima sovelluksen vaatimusmäärittely, jonka pohjalta lähdin kehittämään sovellukseen vaadittavaa komponenttia. Toimeksiantajan puolelta kirjautumiskomponentille ei ollut asetettu varsinaista määrittelyä lukuun ottamatta

tietokantataulujen rakennetta ja sovellukseen liitettävien osien tiedonvälitystä. Sain näin ollen hyvin vapaat kädet suunnitella ja toteuttaa web-sovellukseen vaadittavan osan.

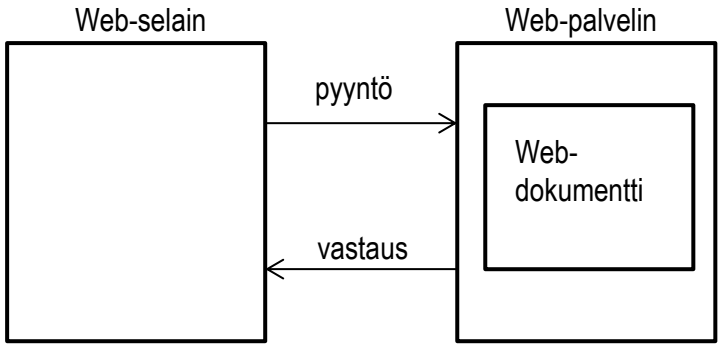
Opinnäytetyössä toteutettavan kirjautumiskomponentin tekniseen toteutukseen käytetään Adobe ColdFusion -sovelluspalvelin alustaa. Toteutettava kirjautumiskomponentti pitää sisällään HTML-kielellä (HyperText Markup Language) toteutettavan kirjautumissivun, johon tuodaan lisäominaisuuksia JavaScript ja CSS (Cascading Style Sheets) -selaintekniikoiden avulla. Toteutettava kirjautumissivu toimii samalla kehitettävän web-sovelluksen etusivuna. Komponentin muihin ominaisuuksiin kuuluvat kirjautumisen ja autentikonnin mahdollistava sovelluslogiikka, jonka toteutuksessa käytetään CFML (ColdFusion Markup Language) ja CFScript (ColdFusion Script) -tekniikoita. Toteutettavan web-sovelluksen tietokantaratkaisuna toimii avoimen lähdekoodin MySQL-tietokanta, joka on liitetty Adobe ColdFusion -sovelluspalvelimeen. Kirjautumiskomponentin ulkopuolisena toimintona opinnäytetyössä tutkitaan automaattisen rajapintadokumentaation käyttöä Adobe ColdFusion web-teknologiaan perustuvan ohjelmistokehityksen yhteydessä. Automaattisen rajapintadokumentaation toteuttamiseen käytetään ColdFusion Component Doc -työkalua.

2 KIRJAUTUMISKOMPONENTIN PERUSTEET

Kirjautumiskomponentin perusteet luvussa käydään läpi keskeisimpiä käsiteteitä ja käytänteitä, jotka on huomioitava toteutettavan komponentin suunnittelussa ja toteutuksessa. Luvun tarkoituksena on selvittää web-sovelluksen toimintaa, yhteyskäytänteitä asiakkaan (selaimen) ja palvelimen välillä sekä tähän liittyviä riskejä. Luvussa käsitellään autentikonnin perusteita ja toimintaperiaatetta yleisellä tasolla. Salasanoja muodostamisen kannalta sekä vertaillaan erityyppisiä salausmenetelmiä sekä niiden ominaisuuksia toisiinsa.

2.1 Web-sovellus

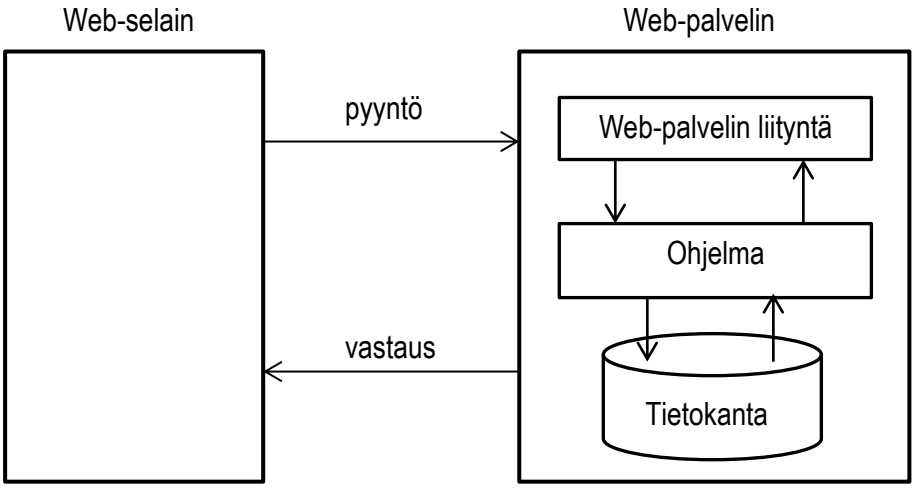
Yleisesti web-julkaiseminen perustuu asiakas-palvelinmalliin, jossa web-dokumentteja ladataan web-selaimen pyynnöstä web-palvelimelta (kuvio 1.). Web-julkaisemisessa web-dokumentit voidaan tehdä staattisiksi eli muuttumattomiksi. Web-palvelimelta noudettavat dokumentit pysyvät muuttumattomina siihen asti kunnes tekijä tekee muutoksia dokumentin sisältöön. Staattisissa web-julkaisuissa käyttäjän mahdollisuus vaikuttaa web-dokumentin sisältöön on hyvin rajallinen. (Rantala 2005, 3.)



KUVIO 1. Staattinen web-dokumentti (Rantala 2005, 3)

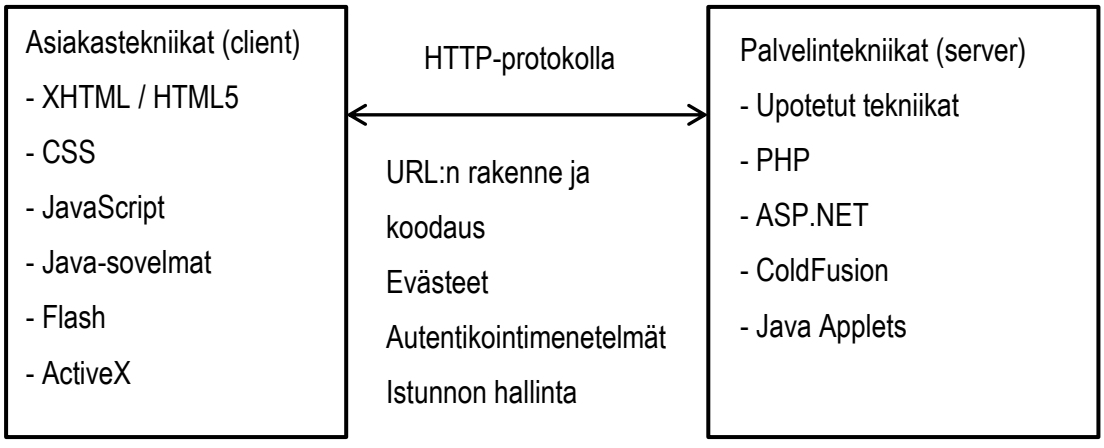
Web-sovellus puolestaan on käyttäjän kannalta dynaaminen ja interaktiivinen web-dokumentti, joka antaa mahdollisuuden käyttäjälle vaikuttaa dokumenttien sisältöön. Web-sovelluksessa toiminta perustuu tekniikoihin, joiden avulla web-dokumentti luodaan osittain tai kokonaan dynaamiseksi. Dynaaminen web-sivu voi sisällöltään muodostua esimerkiksi tietokannasta tulevasta sisällöstä tai käyttäjän tekemien toimintojen tai valintojen perusteella (kuvio 2.). Web-

dokumentit luodaan automaattisesti kulloisenkin lähtötilanteen pohjalta. Dynaamisten dokumenttien toteutus vaatii lähes poikkeuksetta toimintojen ohjelmointia, joka mahdollistaa vuorovaikutteisen web-sovelluksen toteutuksen. (Rantala 2005, 4.)



KUVIO 2. Dynaaminen web-dokumentti (Rantala 2005, 4).

Web-sovellusten toteutukseen käytettävät tekniikat jaetaan yleisesti kolmeen pääluokkaan (kuvio 3.). Web-selaimessa suoritettaviin asiakastekniikoihin (client), web-palvelimessa suoritettaviin palvelintekniikoihin (server) sekä yhteyskäytäntöön liittyviin tekniikoihin. Kokonaisuutena web-sovellus rakentuu käyttöliittymästä sekä sovelluslogiikasta. Lisäksi toimivan web-sovelluksen tarvitaan näiden kahden välistä viestintää. (Rantala.2005, 5-6)



KUVIO 3. Asiakas-palvelinmalli (Rantala 2005, 2).

Web-selaimessa suoritettavia asiakastekniikoita voidaan pitää web-sovelluksen käyttöliittymänä. Nykyisten web-sovellusten käyttöliittymät rakentuvat vähintään HTML-elementeistä ja web-

selainten omista ominaisuuksista. Lisäksi käyttöliittymissä voidaan käyttää elävöittämistekniikoita kuten CSS (Cascading Style Sheets), JavaScript, Java-sovelmat (applets) tai ActiveX - ja Flash -komponentteja. Asiakastekniikoita käyttämällä web-sovelluksen käyttöliittymään saadaan tiettyä dynaamisuutta, mutta pääsääntöisesti dynaamisuus rajoittuu kuitenkin vain käyttöliittymän hallintaan. (Rantala 2005, 6-7)

Web-sovelluksissa varsinainen sovelluslogiikka suoritetaan web-palvelimessa, palvelintekniikkaa hyväksi käyttäen. Web-palvelimen sovelluslogiikat ovat upotettuja tekniikoita, jotka tulkitsevat html-dokumenttien sisään upotetun ohjelmakoodin web-palvelimella. Web-sovelluksissa käytetään tyypillisesti paljon relaatiotietokantoja, joita hallitaan sovelluslogiikan avulla (Rantala 2005, 7-8.)

2.2 Suojattu yhteys

Web-selaimen ja web-palvelimen välinen viestintä toteutetaan HTTP-protokollaa (HyperText Transfer Protocol) hyväksi käyttäen. HTTP on yhteyskäytäntö, jonka avulla käyttöliittymä ja palvelimen sovelluslogiikka viestivät keskenään. HTTP-protokollaa hyödynnetään datan välityksen lisäksi evästeissä, autentikointimenetelmissä ja istunnon hallinnassa. (Rantala 2005, 99)

Viestinnässä käytettävä HTTP-protokolla yhteys on täysin suojaamaton, eli kaikki data selaimen ja palvelimen välillä välitetään selväkielisenä. Salattu selaimen ja palvelimen välinen viestintä toteutetaan HTTPS-protokollaa (Hypertext Transfer Protocol Secure) hyväksi käyttäen. HTTPS-protokollaa käytettäessä välitettävä tieto muutetaan lähettäessä salatuksi ja puretaan alkuperäiseen muotoonsa vasta vastaanottavassa web-palvelimessa. HTTPS-protokollaa käyttävästä sivustosta käytetään termiä suojattu sivusto. Protokollaa käyttävät verkko-osoitteet alkavat etuliitteellä https://. Protokolla perustuu SSL (Secure Sockets Layer) ja TLS (Transport Layer Security) -salausprotokolliin. (if.fi 2013, hakupäivä 23.3.2013)

Uskottavan suojatun sivuston ylläpitäminen vaatii myös digitaalisen sertifikaatin eli varmenteen käytön. Sertifikaatin avulla selain pystyy tunnistamaan SSL-suojauksella suojatun sivuston. SSL-varmenne takaa sekä palvelimen aitouden, että liikenteen luottamuksellisuuden. Varmenne on sähköinen todistus, jonka aitous voidaan tarkistaa ohjelmallisesti. Varmennetekniikka perustuu

matemaattiseen laskutoimitukseen. Julkisen avaimen järjestelmässä luodaan kaksi avainta, joiden välillä on matemaattinen riippuvuus. (Järvinen 2003, 41)

Sertifikaatin sivuston aitoudesta, sivustolle antaa varmentaja (Certificate Authority, CA). Varmentaja on luotettu taho, joka digitaalisella allekirjoituksellaan vahvistaa varmenteen tietojen oikeellisuuden ja varmenteen aitouden. Web-palvelun ylläpitäjä ostaa varmenteen kaupalliselta varmennepalvelun tarjoajalta. Varmentaja tarkastaa varmenteen hankkijan tiedot, ennen varmenteen luomista. Varmenteet ovat voimassa yhdestä kahteen vuoteen, jonka jälkeen varmenne on uusittava. (Järvinen 2003, 165)

Kun käyttäjä tulee varmenteella suojattuun palveluun, selain tarkistaa varmenteen tiedot automaattisesti. Tarkastusta varten selaimella on oltava kyseisen varmentajan juurivarmenne. Selaimen mukana toimitetaan joukko tunnettujen, luotettavien yritysten varmenteita. Tarkistuksen jälkeen selain luo kertakäyttöisen salausavaimen, jolla yhteyden web-liikenne salataan. Samalla HTTP-protokolla vaihtuu HTTPS-protokollaksi. (Järvinen 2003, 168-169)

2.4 Autentikointi

Käyttäjän tunnistaminen on nykyisten web-sovellusten perusominaisuus. Käyttäjän tunnistamisen eli autentikoinnin avulla pystytään määrittelemään kyseisen käyttäjän identiteetti. Autentikoinnin tarkoituksena ei ole pelkästään lisätä web-sovellusten tietoturvaa, vaan myös lisätä kyseiselle käyttäjälle räätälöityjen ominaisuuksien tarjontaa sekä käyttöoikeuksien antamista tiettyihin resursseihin. Tätä kutsutaan pääsynvalvonnaksi, jossa käyttäjäntunnuksen tehtävänä on identifioida käyttäjä ja salasanan tehtävänä autentikoida. (Gilmore 2005, 281)

Web-sovelluksessa käyttäjäautentikointi voidaan toteuttaa kolmea erilaista autentikointitekniikkaa hyväksi käyttäen. Nämä autentikointitekniikat voidaan jakaa periaatteeltaan yksinkertaisiin tekniikoihin, integroitaviin tekniikoihin sekä ulkopuolisiin tunnistautumisjärjestelmätekniikoihin. (Gilmore 2005, 281)

Yksinkertaisissa tekniikoissa autentikointitekniikka perustuu HTTP-perusautentikointiin (HTTP Basic Authentication). HTTP-perusautentikoititekniikassa asiakas pyytää palvelimelta resurssia, jonka käyttöoikeus on asetettu rajoitetuksi. Palvelin vastaa pyyntöön antamalla vastauksen, jonka asiakaan käyttämä selain tunnistaa. Vastauksen perusteella selain avaa autentikoitikehotteen,

johon käyttäjä syöttää tunnistetiedot, käyttäjätunnuksen ja salasanan. Tunnistetiedot palautetaan palvelimelle, joiden perusteella asiakkaan pääsy pyydettyyn resurssiin joko sallitaan tai estetään. Jos tunnistetiedot vahvistetaan, selain tallentaa autentikointitiedot välimuistiin. Autentikointitiedot säilyvät selaimen välimuistissa kunnes välimuisti tyhjennetään tai tilalle tallennetaan toisen palvelun autentikointitiedot. (Gilmore 2005, 282)

Integroitavat tekniikat mahdollistavat käyttäjäautentikonnin toteuttamisen suoraan web-sovelluksen logiikassa. Käyttäjien tunnistetiedot käsitellään ohjelmakoodista käsin. Integroidut tekniikat mahdollistavat lisäksi integrointiin muiden sovelluksen komponenttien kanssa, sisällön räätälöintiin ja käyttäjän oikeuksien määrittämiseen. Tunnistetiedot voidaan tallentaa suoraan koodiin, kovakoodattu autentikointi yksinkertaisin tapa ja nopea toteuttaa integroitu käyttäjäautentikaatio. Tunnistetietojen tallentaminen suoraan ohjelmakoodiin rajoittaa huomattavasti tunnisteiden määrä ja tekee yksilöinnistä hankalaa. Tiedostopohjaisessa autentikoinnissa käyttäjien tunnistetiedot tallennetaan erilliseen tiedostoon palvelimella. Tunnistetiedot haetaan kirjautumisprosessin yhteydessä tiedostosta. Tiedostopohjainen autentikointi soveltuu vain pienelle käyttäjä joukolle. Tiedostopohjainen autentikointi hankaloittaa tunnusten hallinnointia ja hidastaa autentikointitapahtumaa tunnistetietojen määrien ollessa suuret. Tietokantapohjaista autentikointia voidaan pitää tehokkaimpana tapana hoitaa integroitavilla tekniikoilla toteutettu käyttäjäautentikaatio. Tietokantapohjaisessa autentikaatiossa pystytään käyttämään jo olemassa olevaa autentikointitietokantarakennetta. Käyttäjäautentikoinnin toteuttaminen tietokantapohjaista autentikaatiota hyväksi käyttäen on nopein tapa toteuttaa autentikaatio ja hallinnoida suuria tunnistetieto määriä. (Gilmore 2005, 283 - 289)

Käyttäjäautentikointi voidaan toteuttaa myös ulkopuolista tunnistautumisjärjestelmää hyväksi käyttäen. Käyttäjäautentikoinnin perusperiaate on sama kuin integroitavissa tekniikoissa. Käyttäjien tiedot on tallennettu järjestelmään ohjelmankoodiin, tiedostoon tai tietokantaan. Erona on, että varsinainen tunnistautuminen tapahtuu muualla toisessa palvelussa tai järjestelmässä. Tunnistautumisessa käytettävä palvelu palauttaa vaadittavat tunnistautumistiedot kohdepalvelulle, joiden perusteella käyttäjä tunnistetaan. (OpenID Fundation. 2013, hakupäivä 30.3.2013)

Toisin kuin yksinkertaisissa tekniikoissa, ei integroitavissa tekniikoissa ja ulkopuolista tunnistautumisjärjestelmää käytettäessä tallenneta selaimen välimuistiin lainkaan käyttäjän

tunnistetietoja. Näillä tekniikoilla sovellukselle luodaan tunnistautumisen yhteydessä istuntokohtainen tunniste, jota kuljetetaan vain siirtotapahtumien yhteydessä. Varsinainen istuntoon kuuluva data säilytetään palvelimella istunnon ajan. Istuntokohtaista tunnistetta käytettäessä mahdolliset autentikointitiedot (käyttäjätunnus, salasana) välitetään vain kerran eikä niitä tallenneta asiakaskoneelle. Istunnon tiedot tallennetaan istunnonhallintaan, jossa ne ovat voimassa jonkin määritellyn ajan. Istuntoajan loppuessa tai käyttäjän toimesta lopetettaessa nämä tiedot tuhotaan. (OWASP. 2013, hakupäivä 30.3.2013)

2.4 Salasanojen suojaus

Web-sovellusten autentikoinnissa käytettävät tunnistetiedot, esimerkiksi tunnistepari käyttäjätunnus ja salasana, on tallennettu joko tietokantaan, erilliseen autentikoititiedostoon tai joissain tapauksissa suoraan ohjelmakoodiin. Tallennustavasta riippumatta autentikonnin yhteydessä vertaillaan käyttäjän antamia syötteitä tallennettuihin tietoihin.

Web-palveluihin kohdistuvien tietomurtojen yhteydessä haetaan yleensä reittiä juurin näihin autentikointitietoihin. Anastettujen autentikointitietojen avulla voidaan kirjautua palveluun ja esiintyä jonkin tietyn käyttäjän tiedoilla sekä profiililla. Lisäksi ylläpitäjien profiililla voidaan aiheuttaa haittaa myös itse palvelulle. Mikäli autentikointitiedot tallennetaan selväkielisinä sovellukseen, voidaan tunnistetietoja käyttää sellaisenaan palveluun kirjautumiseen.

Salasanojen suojaamisella pyritään estämään kirjautumistietojen joutuessa vääriin käsiin niiden käyttö ja tekemään se mahdollisimman vaikeaksi. Yleisesti salasanojen suojauksen tarkoituksena toteuttaa prosessi, jossa käyttäjän tiedossa oleva selväkielinen salasana, tallennetaan järjestelmään salattuna. Tällä prosessilla pyritään estämään todellisen tiedon näkyminen.

Salasanojen suojaamiseen on tarjolla useita eri menetelmiä. Yleisimmin salasanojen suojaukseen käytetään kahta toisistaan poikkeavaa menetelmää, symmetrisiä salausmenetelmiä tai yksisuuntaisia tiivisteitä. Symmetrisissä menetelmissä toiminta perustuu salausavaimeen, jonka avulla salattava tieto salataan ja puretaan matemaattisen laskutoimituksen avulla. Tunnettuja ja yleisesti käytössä olevia symmetrisiä tekniikoita ovat DES (Data Encryption Standard), AES (Advanced Encryption Standard) sekä Blowfish. Yksisuuntaisissa tiivisteissä toiminta perustuu merkkijonojen matemaattiseen tulkintaan ja laskutoimitukseen, joiden pohjalta tuotetaan tiiviste alkuperäisestä tiedosta. Tiiviste on muodoltaan heksakoodattu merkkijono, joka

on pituudeltaan 32 – 128 merkkiä pitkä, riippuen käytetystä tiivistealgorytmistä. Tunnettuja ja yleisti käytössä olevia yksisuuntaisia tiivisteitä ovat MD5 (Messade digest) sekä SHA (Secure Hash Algorithm) -tiivistealgorytmit. (Järvinen 2003, 81-97)

SHA-tiivistealgorytmien ensimmäinen versio julkaistiin vuonna 1993 NIST:n (National Institute of Standards and Technology) ja FIPS:n (U.S. Federal Information Processing Standard) toimesta nimellä SHA-0. Kehitystyön pohjana on toiminut MD5-tiivistealgoritmi, jonka matemaattista laskutoimitusta on kehitetty eteenpäin ja tiivistekokoa kasvatettu. SHA-tiivistealgoritmiestä on SHA-0:n lisäksi käytössä tällä hetkellä versiot SHA-1 ja SHA-2. SHA-2 algoritmit on jaettu neljään eri tasoon SHA-224, SHA-256, SHA-348, ja SHA-512, joista käytetään yleisnimitystä SHA-2 tiivistealgoritmifunktiot. SHA-2 tiivistealgoritmifunktioiden nimen luvulla kuvataan kyseisen funktion bittimäärä. SHA-512-funktio tuottaa 512-bittisen tiivisteen, joka muunnetaan 128 merkkiä pitkäksi heksakoodatuksi merkkijonoksi. Jokainen SHA-tiivistealgoritmillä käsitelty merkkijono muodostuu yksittäisten merkkien perusteella yksilölliseksi (Taulukko 1). (Järvinen 2003, 122-129)

TAULUKKO 1. Tiiviste merkkijonosta SHA-512-algoritmillä

Merkkijono	Tiiviste
password12!	09caf6983248ddbf17a2d76f8f48ee825b93be87946c039e6e556ee4f6fbae67a6 631a60dd569d9ddb48c0485b63122111edc0cc14164110bb773a2f108ee532
Password12!	b667d9ce75359514d54e5d3a4a884fa0ac6323c4341aad3b8490ff09d50bb8312 e2822fc813f2d2ef7fb59ed0f2ebcf69cd8afe92037e6ddce1f64efc66fef3d

Tiivistefunktioiden avulla suojattuja salasanoja voidaan selvittää ns. rainbow-taulukoiden avulla. Rainbow-taulukot ovat kokoelmia valmiiksi lasketuista tiivisteitä merkkijonoista. Tiivistettä verrataan valmiiksi laskettuun tiivisteeseen, jonka avulla saadaan selvittyä alkuperäinen merkkijono. (365 Computer Security Training 2013, hakupäivä 9.5.2013)

Salasanojen suojausta voidaan tehostaa lisäämällä merkkijonoon lisämerkkejä, joko alkuperäisen merkkijonon alkuun, loppuun tai molempiin yhtä aikaa. Lisämerkkejä merkkijonossa kutsutaan suolaksi ja merkkien lisäämistä suolaamiseksi. Suolaamisen tarkoituksena on muuttaa käyttäjän tunnistautumisessa käyttämä salasana pidemmäksi ja vaikeammin generoitavaksi. Tärkeimpänä tarkoituksena on muuttaa salauksen tulosta. Käytettäessä suolausta, salasanan salaustavasta riippumatta, salaus suoritetaan suolatulle salasanalle, ei alkuperäiselle. (Järvinen 2003, 246-247)

2.5 Ohjelmakoodin dokumentointi

Osana ohjelmistoprojektia syntyy usein paljon dokumentaatiota. Syntyneen dokumentaation tulisi olla ajantasaista eikä heijastella projektin kehitysvaiheita silloin, kun dokumentaatio on kirjoitettu. Näin dokumentaatiosta saadaan mahdollisimman paljon hyötyä tulevaisuudessa. Kaikkea ohjelmistoprojektin dokumentaatiota ei luultavasti kannata ylläpitää, sillä osa siitä on ollut tarpeen vain osana kehittämisprosessia. (Haikala & Mikkonen 2011, 194)

Yleensä dokumentaation tuottaminen ja ylläpito vaativat suuren työpanoksen, mitään turhaa ei näin ollen kannata siis dokumentoida. Dokumentaatiossa kuvauksen tulisi olla siis hyvin tiivis ja kuvaava kuin on mahdollista. Tällöin myös vältetään ristiriitaisuudet, joita ylläpidon myötä dokumentaatioon voisi muuten jäädä. (Haikala & Mikkonen 2011, 194)

Ohjelmistoprojektissa toteutusvaihe keskittyy yleensä ohjelmakoodin tuottamiseen. Tällöin myös dokumentaation paikka on luonnollisesti olla siellä. Mahdollisuuksien mukaan dokumenttimuotoinen kuvaus ohjelmistossa kannattaa tuottaa automaattisesti ohjelmakoodista, sillä mikäli kuvausten päivittäminen on ylimääräinen tehtävä, se jää helposti ohjelmoijalta tekemättä. Automaattinen rajapintadokumentaatio voidaan hoitaa sopivia työkaluja hyväksi käyttäen. Dokumentit kannattaa päivittää viimeistään projektin päättyessä. Muutoin niistä ei ole juurikaan hyötyä jatkossa. (Haikala & Mikkonen 2011, 195)

Vaikka itsensä dokumentoivasta koodista usein innostutaankin, tämä harvoin toteutuu oikeasti. Siksi ohjelmakoodiin kannattaa upottaa sen toimintalogiikkaan liittyvä dokumentaatio yleisellä tasolla. Kommentoinnin pitää tällöin kuvata tarkoitus ja ratkaisu yleisellä tasolla, eikä missään tapauksessa saa kopioida ohjelmointikielen semantiikkaa kommentteihin. Ohjelmakoodin kommenttien kirjoittamiseen pätevät samat periaatteet kuin muuhunkin määrittelyyn ja dokumentointiin, eli asiat tulee kommentoinnissa ilmaista lyhyesti, täsmällisesti ja tarkoituksen mukaisesti. (Haikala & Mikkonen 2011, 195)

3 KÄYTETYT ASIAKASTEKNIIKAT

Käytetyt asiakastekniikat luvussa käydään läpi vaadittavat perusasiat opinnäytetyössä toteutettavan kirjautumiskomponentin toteutuksessa vaatimista, selaimen käyttöön tarvitsemista asiakastekniikoista. Kirjautumiskomponentin pääasiakastekniikkana käytetään JavaScript-skriptikieltä. Luvun tarkoituksena on kuvata asiakastekniikoiden käyttöä ja perusominaisuuksia. Luvussa käsiteltävä jQuery ei varsinaisesti ole tekniikka, vaan JavaScript-kirjasto, jonka toiminta perustuu JavaScript-tekniikkaan.

3.1 JavaScript

JavaScript luokitellaan skriptikieliin, joita käytetään erilaisten sovellusten toimintojen laajentamiseen. JavaScript on tulkattava kieli muiden skriptikielten tapaan. JavaScript-tulkki on upotettu isäntäsovellukseen, kuten web-selaimeen tai web-palvelimeen. Tulkin avulla JavaScript-ohjelmaa tulkataan rivi riviltä sitä mukaan kun ohjelmaa suoritetaan. (Peltomäki & Nykänen 2006, 96)

JavaScript on Netscapen vuonna 1995 Navigator 2 -selaimen mukana julkaisema JavaScript 1.0 -kieli. Alkuperäiseltä nimeltään kieli oli LiveScript, mutta markkinoinnin vuoksi nimi muutettiin. Nimen valintaan vaikutti Netscapen yhteistyö Java-kielen kehittäjän Sun Microsystemsin kanssa. JavaScriptin pohjalta määriteltiin ECMAScript-standardikieli. JavaScript on Netscapen tavaramerkki. Microsoftin vastine JavaScriptille on JScript joka noudattaa ECMAScript-standardia. (Peltomäki & Nykänen 2006, 90.)

HTML-dokumentteihin liitettyjen skriptien tarkoituksena on tuoda dokumentteihin aitoa dynaamista vuorovaikutteista toimintaa. Yksi keskeinen syy JavaScriptin käyttöön on HTML-kielen yksinkertaisuus. JavaScriptillä pystytään antamaan käyttäjälle välitöntä palautetta, johon HTML-kieli ei pysty. JavaScriptin tehokkuus web-dokumenttien muokkauksessa perustuu dokumenttioliomallin DOM (Document Object Model) -ohjelmointirajapinnan hyväksikäyttöön. (Peltomäki & Nykänen 2006, 94 -95.)

Nykymuotoinen JavaScript on dynaamisesti tyypitetty ja oliopohjainen kieli, joka muistuttaa löyhästi syntaksiltaan C-ohjelmointikieltä. Viimeisin versio Javascript-kielestä on 1.8.5, jonka pohjana toimii EcmaScript-standardi ECMA-262 Edition 3. (Mozilla Developer Network. 2013, hakupäivä 30.3.2013.)

JavaScriptiä voidaan liittää usealla eri tavalla HTML-dokumenttiin. JavaScript-koodia on mahdollista upottaa <script>-merkkejä käyttäen suoraan HTML-dokumentin body-elementtiin, scriptin suorituskohtaan (kuvio 4.). Toinen tapa käyttää JavaScript-koodia on asettaa kaikki suoritettava JavaScript-koodi <script>-merkkejä käyttäen HTML-dokumentin head-elementin sisään (kuvio 5.). (w3schools 2013, hakupäivä 30.3.2013.)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4
5  <head>
6      <title>Sivun otsikko</title>
7      <meta http-equiv="content-type" content="text/html;charset=utf-8" />
8  </head>
9
10 <body>
11     <script type="text/javascript">
12         document.write('Tulostettava teksti');
13     </script>
14 </body>
15 </html>
```

KUVIO 4. JavaScript upotettuna body-elementin sisään

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4
5  <head>
6      <title>Sivun otsikko</title>
7      <meta http-equiv="content-type" content="text/html;charset=utf-8" />
8
9      <script type="text/javascript">
10         document.write('Tulostettava teksti');
11     </script>
12 </head>
13 <body>
14
15 </body>
16 </html>
```

KUVIO 5. JavaScript upotettuna head-elementin sisään

JavaScript-koodia voidaan suorittaa erillisestä tiedostosta käsin. Suoritettava JavaScript-koodi tallennetaan .js-päätteiseen tiedostoon. JavaScript-tiedostojen liittäminen HTML-dokumenttiin tapahtuu head-elementin sisällä <script> -merkkejä käyttäen. JavaScript-tiedoston sijaintiin viitataan script-elementin src-parametrilla (kuvio 6.). Liitettävän tiedoston sisälle on kirjoitettu mahdolliset funktiot ja muu toiminnallisuus, joita halutaan dokumentissa käyttää. (Peltomäki & Nykänen 2006, 103-104)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4
5  <head>
6      <title>Sivun otsikko</title>
7      <meta http-equiv="content-type" content="text/html;charset=utf-8" />
8      <script type="text/javascript" src="script.js"></script>
9  </head>
10 <body>
11     <h1>Otsikko</h1>
12 </body>
13 </html>
```

KUVIO 6. JavaScript -tiedoston lataus HTML-dokumenttiin

Yksi JavaScript-kielen keskeisimmistä toiminnoista ovat funktiot. Funktiot ovat aliohjelmia, joiden tarkoituksena on suorittaa jokin tehtävä ohjelmakoodissa. Nämä tehtävät voivat olla joukko erilaisia perättäisiä toimintoja. Funktio suorittaminen vaatii, että sitä kutsutaan jonkin toimesta. Kutsuja voi olla toinen funktio tai tapahtumakäsittelijä. Yksinkertaisin funktiotyyppi on parametritonfunktio (Kuvio 7.). (Peltomäki & Nykänen 2006, 129-130)

```
1  // Funktion kutsu
2  funktion_nimi();
3
4  // Funktion rakenne
5  function funktion_nimi() {
6      // suoritettava koodi;
7  }
8
9  // Esimerkkifunktio
10 function suoritettavaFunktio() {
11     document.write("Tulosteva teksti");
12 }
```

KUVIO 7. Parametritonfunktion kutsumien, rakenne ja esimerkkifunktio

Toinen mahdollinen funktiotyyppi on parametrillinenfunktio. (Kuvio 8.). Funktiota kutsuttaessa, funktio kutsuun annetaan funktiolle välitettävien parametrien arvot. Funktiolle tuotujen

parametrien arvoja ei pystytä muuttamaan suoraan kutsuttavassa funktiossa. (Peltomäki & Nykänen 2006, 130)

```
1  // Funktion kutsu
2  funktion_nimi(arvo_1, arvo_2,...,arvo_N);
3
4  // Funktion rakenne
5  function funktion_nimi(param_1, param_2,...,param_N) {
6      // suoritettava koodi;
7  }
8
9  // Esimerkkifunktio
10 function suoritettavaFunktio( teksti ) {
11     document.write( teksti );
12 }
```

KUVIO 8. Parametrillisenfunktion kutsumien, rakenne ja esimerkkifunktio

Molemmat funktiotyypit, parametriton- sekä parametrillinenfunktio eivät palauta kutsujalleen minkäänlaista arvoa. Funktiot ainoastaan suoritetaan rivi riviltä, viimeisen rivin jälkeen funktion suoritus lopetetaan. Molemmista funktiotyypeistä voidaan tehdä palauttavia return sanaa käyttäen (Kuvio 9.). (Peltomäki & Nykänen 2006, 130-131)

```
1  // Parametritonfunktio
2  function laske_yhteen() {
3      var luku_1 = 1;
4      var luku_2 = 2;
5      var tulos = luku_1 + luku_2;
6      return tulos;
7  }
8
9  // Funktion kutsu
10 var tulos = laske_yhteen();
11
12 //Parametrillinenfunktio
13 function laske_yhteen( luku_1, luku_2 ) {
14     var tulos = luku_1 + luku_2;
15     return tulos;
16 }
17
18 // Funktion kutsu
19 var tulos = laske_yhteen( 1, 2);
20
```

KUVIO 9. Esimerkki palauttavista funktioista

Web-dokumentissa käyttöliittymästä generoituu aina tapahtumia. Tapahtumat ovat tyypillisesti painikkeen painamisia, tekstin kirjoitusta tekstikenttään tai valintojen tekemistä hiiren osoittimella. Ohjelmakoodissa on pystyttävä tarvittaessa reagoimaan web-dokumentin käyttäjän toimiin.

JavaScript-kieli on tapahtumaohjautuva ohjelmointikieli. JavaScript-funktiota voidaan kutsua HTML-dokumentissa HTML-elementtien kautta. HTML-elementteihin voidaan lisätä tapahtumankäsittelijöitä. Tapahtumankäsittelijät kutsuvat tapahtuman seurauksena tapahtumankäsittelijäfunktiota. Web-dokumenteissa yleisiä tapahtumankäsittelijöitä ovat esimerkiksi hiiren napin painallus (onClick) tai web-dokumentin lataamisesta tuleva tapahtuma (onLoad). (kuvio 10.). (Peltomäki & Nykänen 2006, 110)

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4
5  <head>
6      <title>Sivun otsikko</title>
7      <meta http-equiv="content-type" content="text/html;charset=utf-8" />
8      <script type="text/javascript" src="script.js"></script>
9  </head>
10 <body onload="suoritettavaFunktio();">
11     <h1>Otsikko</h1>
12 </body>
13 </html>
```

KUVIO 10. onload -tapahtumankäsittelijä kutsuu funktiota HTML body -elementissä

3.2 jQuery

Tammikuussa 2006 John Resig julkaisi jQuery-nimisen JavaScript-kirjaston helpottamaan HTML-dokumenttien muokkaamista ja dynamiikan kehittämistä. jQuery on selainriippumaton avoimen lähdekoodin JavaScript-kirjasto. jQuery on suunniteltu helpottamaan JavaScriptin käyttöä helpomman ja selkeämmän ohjelmakoodin avulla. jQuery mahdollistaa muun muassa selainten DOM-elementtien (Document Object Model) ohjelmointirajapinnan käsittelyn, animaatioiden tekemisen ja Ajax-sovellusten toteutuksen. jQuery JavaScript-kirjasto on kaikkien ladattavissa projektin kotisivulta. jQuery-kirjaston käyttöönotto vaatii kirjaston lisäämisen html-dokumenttiin (Kuvio 11). (jQuery Board 2012, hakupäivä 5.3.2012.)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <title>Sivun otsikko</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
</head>
<body>
    <h1>Otsikko</h1>
</body>
</html>
```

KUVIO 11. jQuery version 1.9.1 JavaScript-kirjaston lisääminen HTML-dokumenttiin Googlen CDN (Content Delivery Network) -palvelusta.

jQuery JavaScript-kirjaston funktiota kutsutaan jQuery-objektin avulla. jQuery-funktiota kutsuttaessa paluuarvona saadaan jQuery-objekti. Käytön helpottamiseksi jQuery-kirjastoon on tehty $-merkillä nimetty muuttuja, joka pitää sisällään jQuery-objektin. Sulkujen sisälle annetaan HTML-dokumentissa oleva elementti, johon toiminta kohdistuu tai josta saadaan jokin tapahtuma. Elementille lisätään haluttu jQuery-toimintafunktio. (MacLess 2012, 17-18)

Kuviossa 12 on esimerkki jQuery-kirjaston käytöstä JavaScript-koodissa. Rivillä 1 jQuery-funktioita kutsutaan $-merkillä, joka pitää sisällään jQuery-objektin. Sulkeiden sisällä document-määre viittaa HTML-dokumentiin. Pisteellä erotettu ready on funktio, joka odottaa, kunnes HTML-dokumentti on latautunut valmiiksi. function() on ns. nimetön funktion, joka tehtävä on niputtaa sisälleen suoritettava ohjelmakoodi. Rivillä 2 jQuery-objektin avulla muokataan HTML-dokumentissa olevan div-elementin toimintaa, jonka id-arvo on example. Toiminnan muokkaaminen tapahtuu fadeIn()-funktion avulla, jonka toiminta tuo div-elementin näkyviin hitaasti, HTML-dokumentin latauduttua kokonaan. Rivillä 3 nimetön funktio suljetaan kaarisulkeella ja ready-funktio suljetaan sulkumerkkiä sekä puolipistettä käyttäen.

```
$(document).ready(function() {
    $("div#example").fadeIn("slow");
});

```

KUVIO 12. Esimerkki jQuery:n käytöstä.

4 ADOBE COLDFUSION

Adobe ColdFusion luvussa käsitellään opinnäytetyössä toteutettavan kirjautumiskomponentin toteutusteknologiaa. Toteutusteknologiasta tarkastellaan tarkemmin sovelluspalvelinta, ohjelmointikieliä sekä järjestelmän dokumentointirajapintaa. Adobe ColdFusion web-teknologia on Suomessa vähemmän tunnettu web-teknologia. Muualla maailmassa ColdFusion on suosittu web-teknologia ja sillä on laaja käyttäjäkunta. Adobe ColdFusion on J2EE (Java Platform Enterprise Edition) -pohjainen järjestelmä, joka toimii kaikilla pääpalvelinalustoilla (Linux, Unix, OS X Server ja Windows Server). (Adobe 2013, hakupäivä 23.3.2013)

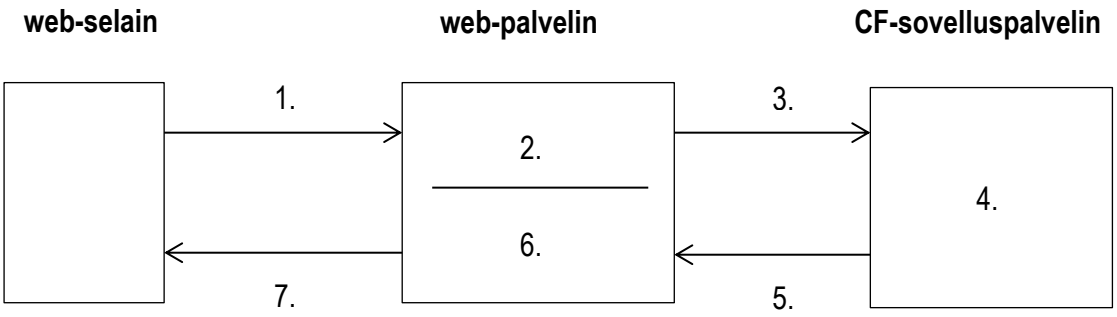
Ensimmäinen versio ColdFusion-sovelluspalvelimesta julkaistiin vuonna 1995. Alussa sovelluspalvelimen kehityksestä vastasivat veljekset Joseph J. ja Jeremy Allaire. Sovelluspalvelimen ensimmäisten versioiden julkaisusta vastasi veljesten omistama Allaire yhtiö. Sovelluspalvelimen kehityksen yhteydessä, kehitettiin myös sovelluspalvelimelle oma skriptikieli, ColdFusion Markup Language (CFML). Myöhemmässä vaiheessa ColdFusioniin kehitettiin oma server-side ActionScript -kieli, joka muistuttaa syntaksiltaan JavaScriptiä ja C-kieltä. Kieli on nimetty ColdFusion Script (CFScript) nimellä. Sovelluspalvelimen alkuperäisenä tarkoituksena oli mahdollistaa tiedon tuominen tietokannoista HTML-dokumentteihin, mahdollisimman helposti. Teknologia tunnettiin 2000 vuoteen asti Cold Fusion -nimellä. Macromedian ostaessa Allaire yhtiön vuonna 2001, teknologian nimi muuttui Macromedia ColdFusion MX -teknologiaksi. Vuonna 2007 Adoben ostaessa Macromedian web-teknologin nimi muuttui Adobe ColdFusioniksi. Adobe on julkaissut tähän mennessä ColdFusion web-teknologiasta kolme versiota. Adobe on ilmoittanut kehittävänsä tulevaisuudessa teknologiaa edelleen eteenpäin. Viimeisin versio sovelluspalvelimesta on versio 10. ColdFusion web-teknologiaa kehitetään Yhdysvalloissa ja Intiassa. (Wikipedia 2013, hakupäivä 23.3.2013)

4.1 Adobe ColdFusion 10 -sovelluspalvelin

Adobe ColdFusion -sovelluspalvelin käsittelee tiedostot, joiden pääte on .cfm, .cfml tai .cfc. ColdFusion-sovelluspalvelimen toimintaperiaate on samantyyppinen kuin muillakin tekniikoilla toteutettujen web-sovelluspalvelinten (Kuvio 13.). Sovelluspalvelimen tehtävänä on suorittaa kirjoitettu ohjelmakoodi web-selaimen ymmärtämään muotoon. Sovelluspalvelimen tehtäviin

kuuluu tarjota turvallinen alusta web-sovellukselle, suorittaa annetut tietokantakyselyt sovelluspalvelimeen kytkettyyn tietokantaan, käsitellä dynaamisia lomake-elementtejä, sekä mahdollistaa integraatiot muihin järjestelmiin tavallisimpia yhteysprotokollia (HTTP, FTP, LDAP, POP ja SMTP) hyväksi käyttäen sekä mahdollistaa vuorovaikutteisten web-sovellusten tekeminen. (Farrar 2010, 7-12)

Kuviossa 13 selvitetään web-sovelluksen toimintaa, jossa sovelluspalvelimena toimii Adobe ColdFusion -sovelluspalvelin. 1. Käyttäjä pyytää web-selaimen välityksellä jotakin sivua (resurssia) web-palvelimelta. 2. Web-palvelin tunnistaa tiedostopäätteestä, vaatiko käyttäjän pyytämä resurssi sovelluspalvelimen prosessointia. 3. Web-palvelin ohjaa pyynnön sovelluspalvelimelle. 4. Sovelluspalvelin käsittelee ohjelmakoodin. Sovelluspalvelin suorittaa tarvittavat yhteydet tietokantaan ja tarvittaessa muihin järjestelmiin. 5. Sovelluspalvelin palauttaa web-palvelimelle staattisessa muodossa olevan web-dokumentin. 6. Web-palvelin ottaa vastaan luodun dokumentin sovelluspalvelimelta ja julkaisee sen 7. Lähetetään valmis web-sivu web-selaimelle käyttäjän käytettäväksi. (Farrar 2010, 9)



KUVIO 13. Adobe ColdFusion 10-sovelluspalvelimen toimintaperiaate (Farrar 2010, 9).

4.2 ColdFusion-merkkikieli

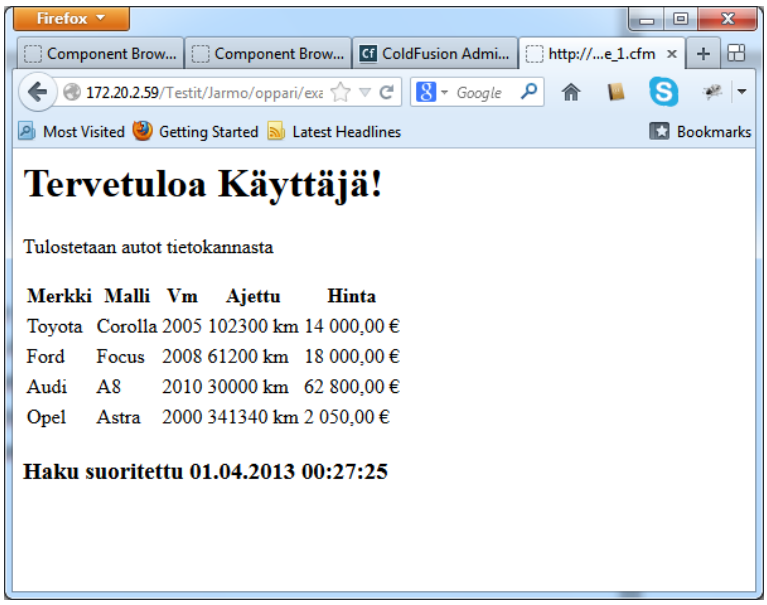
Ohjelmointikielenä CFML (ColdFusion Markup Language) on verrattavissa PHP (PHP, Hypertext Preprocessor) -ohjelmointikieleen. CFML-kielen käyttötarkoitus ja ominaisuudet ovat lähes samat. Molempia edellä mainituista kielistä käytetään web-palvelinten sovelluslogiikassa upotustekniikalla, toisin sanoen CFML-koodi voidaan kirjoittaa dokumentissa suorituskohtaan. CFML-kieli muistuttaa HTML-kieltä, koska kaikki käskyt on ympäröity kulmasulkeilla, esimerkiksi

<cfoutput></cfoutput>. Suoritettavan CFML-merkkikielellä toteutetun tiedoston pääte on cfm. (Farrar 2010, 11)

```
1  <cfset muuttuja = "Käyttäjä">
2  <!--- Kommentti --->
3  <cfoutput><h1>Tervetuloa #muuttuja#!</h1></cfoutput>
4
5  <p>Tulostetaan autot tietokannasta</p>
6
7  <cfquery name="getData" datasource="test">
8      SELECT merkki, malli, vuosimalli, ajettu, hinta
9      FROM test.example
10 </cfquery>
11
12 <table>
13     <tr>
14         <th>Merkki</th><th>Malli</th><th>Vm</th><th>Ajettu</th><th>Hinta</th>
15     </tr>
16     <cfloop query="getData">
17         <tr>
18             <cfoutput>
19                 <td>#merkki#</td><td>#malli#</td><td>#vuosimalli#</td>
20                 <td>#ajettu# km</td><td>#DecimalFormat(hinta)# &euro;</td>
21             </cfoutput>
22         </tr>
23     </cfloop>
24 </table>
25
26 <h3>Haku suoritettu <cfoutput>#dateformat(now(), "dd.mm.yyyy")# #TimeFormat(now(), "HH:mm:ss")#</cfoutput></h3>
```

KUVIO 14. Esimerkki CFML-kielen käytöstä.

Kuviossa 14 on esimerkki CFML-tiedostosta. Rivillä 1 alustetaan muuttuja. Rivillä 3 cfoutput-tagin sisällä tulostetaan muuttujan sisältö. Riviltä 7 alkaen suoritetaan kysely tietokannasta. Riviltä 16 alkaen suoritetaan kyselyn tulostus rivi riviltä. Rivillä 26 tulostetaan dateformat-funktion avulla, tulostushetken päivämäärä ja timeformat-funktion avulla tulostushetken kellonaika. Sovelluspalvelimella tehdyn kääntämisen jälkeen tulokseksi saadaan staattinen HTML-merkkikielellä oleva web-dokumentti (Kuvio 15.).



KUVIO 15. Esimerkki CFML-tiedostosta generoidusta html-sivusta.

4.3 ColdFusion-scriptikieli

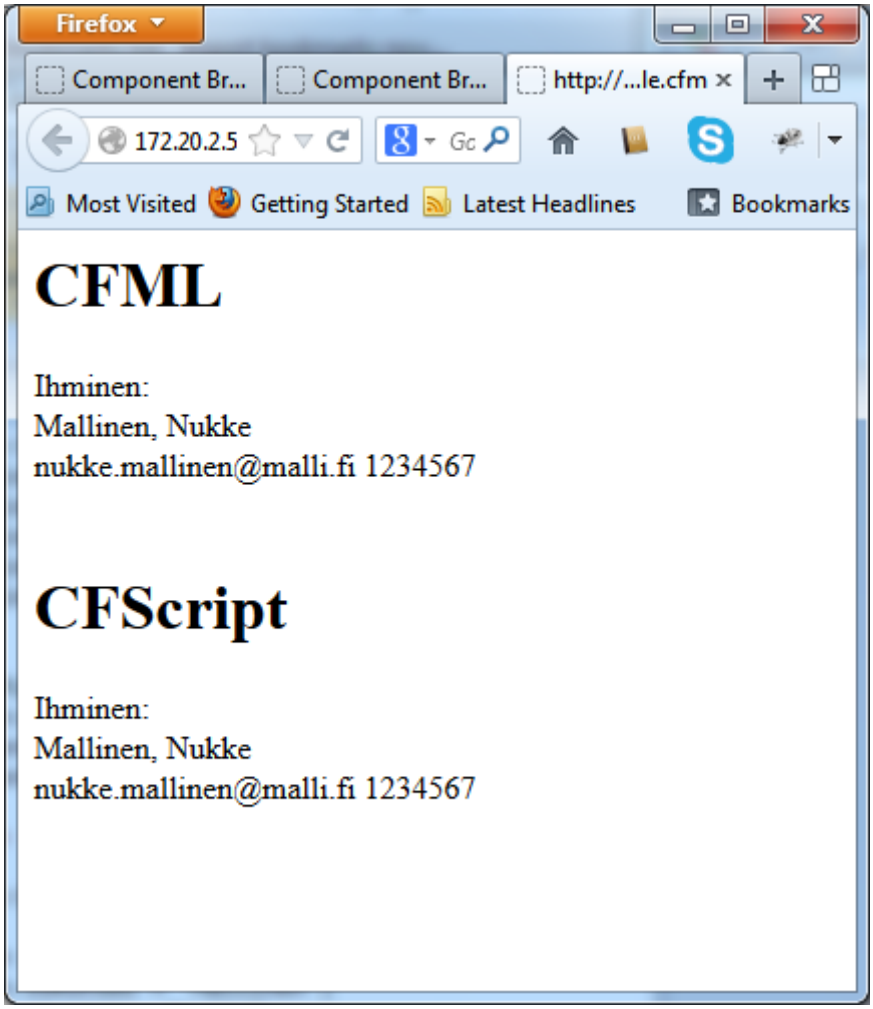
Adobe ColdFusion -sovelluspalvelimessa on mahdollista käyttää ohjelmointikielenä CFML-kielen lisäksi ColdFusion Script (CFScript) -kieltä. CFScript-kielellä on mahdollista toteuttaa samat toiminnallisuudet kun CFML-kielellä, missä ColdFusion ilmauksen käyttö on sallittua. Periaatteessa voidaan puhua ohjelmointikielestä kielen sisällä. Syntaksiltaan CFScript-kieli on samanlainen kieli kun JavaScript. Yleisesti CFScript-kieltä käytetään koodin yksinkertaistamiseen ja nopeuttamiseen. CFScript-kielen suorituslohko on ympäröity <cfscript></cfscript> -tageilla. (Farrar 2010, 147)

```
1  <!--- Käytetään CFML-kieltä --->
2  <cfset ihminen = structNew()>
3  <cfset ihminen.sukunimi = "Mallinen">
4  <cfset ihminen.etunimi = "Nukke">
5  <cfset ihminen.email = "nukke.mallinen@malli.fi">
6  <cfset ihminen.puhelinnro = "1234567">
7
8  <cfoutput>
9      <h1>CFML</h1>
10     Ihminen:<br />
11     #ihminen.sukunimi#, #ihminen.etunimi#<br />
12     #ihminen.email# #ihminen.puhelinnro#<br />
13     <br />
14 </cfoutput>
15
16 <!--- Käytetään CFScriptiä --->
17 <cfscript>
18     ihminen = structNew();
19     ihminen.sukunimi = "Mallinen";
20     ihminen.etunimi = "Nukke";
21     ihminen.email = "nukke.mallinen@malli.fi";
22     ihminen.puhelinnro = "1234567";
23
24     WriteOutput("
25         <h1>CFScript</h1>
26         Ihminen:<br />
27         #ihminen.sukunimi#, #ihminen.etunimi#<br />
28         #ihminen.email# #ihminen.puhelinnro#<br />
29         <br />
30     ");
31 </cfscript>
```

KUVIO 16. CFML-merkkikielen ja CFScript-scriptikielen eroavaisuusesimerkki.

Kuviossa 16 on esimerkki CFML-merkkikielen ja CFScritp-scriptikielellä toteutetun ohjelmakoodin eroavaisuuksista. Riveillä 1–14, ohjelmakoodin toteutuksen on käytetty CFML-merkkikieltä. Riveillä 16–31 ohjelmakoodi on toteutettu CFScript-scriptkielellä. Molemmilla kielillä määritellään aluksi ihminen niminen tietue, jolle annetaan sisällöksi muuttujat sukunimi, etunimi, email ja puhelinnumero. Seuraavaksi tietueen sisältö tulostetaan. Sovelluspalvelimella tehdyn

kääntämisen jälkeen tulokseksi saadaan staattinen HTML-merkkikiellä oleva web-dokumentti (Kuvio 17.).



KUVIO 17. CFML-merkkikielen ja CFScript-scriptikielen eroavaisuusesimerkki web-selaimessa.

4.4 ColdFusion-komponentti

Kaikissa nykyisin käytössä olevista web-sovelluspalvelin ympäristöissä on mahdollista käyttää ohjelmointitapana olio-ohjelmointia (OOP, Object-Oriented Programming). Adobe ColdFusion -ympäristössä oliopohjaisuus on toteutettu ColdFusion-komponenttien avulla. ColdFusion-komponentti vastaa olio-ohjelmoinnin luokka-käsitettä. Komponentti pitää luokan tavoin sisällään funktiota ja muuttujia, jotka voivat olla joko julkisia (public) tai yksityisiä (private). ColdFusion-komponentteja käytetään luokkien tapaan luomalla ohjelmakoodissa komponentista olio. ColdFusion-komponentin määrittämiseen voidaan käyttää CFML-merkkikieltä tai CFScript-scriptkieltä. ColdFusion-komponentin tiedostopääte on .cfc. (Gifford 2010, 7-9)

ColdFusion-komponentin määrittely tapahtuu luomalla .cfc-päätteinen tiedosto. Komponentin määrittely tapahtuu <cfcomponent></cfcomponent>-tagien sisään. Komponentin funktiot muodostetaan käyttäen cffunction-tagia, jolle annetaan lisäattribuutteja, pakollinen attribuutti on name, joka nimeää funktion. Funktioista voidaan tehdä palauttavia cfreturn-tagin avulla. Funktioille voidaan määrätä parametrejä cfargument-tagin avulla, pakollinen attribuutti on name, joka nimeää parametrin. (Kuvio 18.)

```
<cfcomponent>
    <!--- Constructor--->
    <cffunction name="init">
        <cfscript>
            variables.ominaisuus = structNew();
            variables.ominaisuus.nimi = "";
            variables.ominaisuus.pituus = 0;
            variables.ominaisuus.paino = 0;
        </cfscript>

        <cfreturn this>

    </cffunction>
    <!--- get/set nimi --->
    <cffunction name="getNimi">
        <cfreturn variables.ominaisuus.nimi>
    </cffunction>

    <cffunction name="setNimi">
        <cfargument name="nimi">
        <cfset variables.ominaisuus.nimi = arguments.nimi>
    </cffunction>
    <!--- get/set pituus--->
    <cffunction name="getPituus">
        <cfreturn variables.ominaisuus.pituus>
    </cffunction>

    <cffunction name="setPituus">
        <cfargument name="pituus">
        <cfset variables.ominaisuus.pituus = arguments.pituus>
    </cffunction>
    <!--- get/set paino--->
    <cffunction name="getPaino">
        <cfreturn variables.ominaisuus.paino>
    </cffunction>

    <cffunction name="setPaino">
        <cfargument name="paino" required="true" type="numeric">
        <cfset variables.ominaisuus.paino = arguments.paino>
    </cffunction>
</cfcomponent>
```

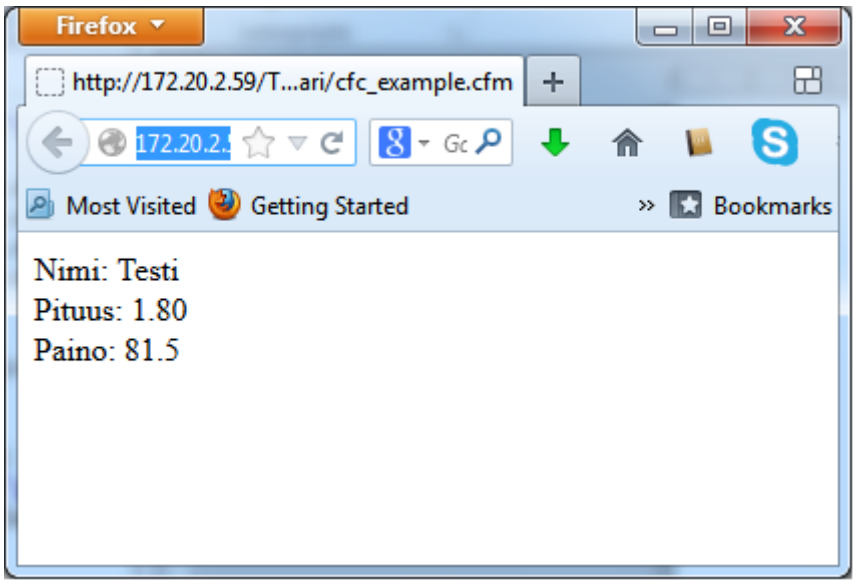
KUVIO 18. Esimerkki ColdFusion-komponentin määrittelystä.

Komponenttien käyttö tapahtuu periaatteeltaan samalla tavalla kuin muissakin olio-ohjelmointikielissä. Luokan funktioiden käyttöönotto vaatii olion luonnin luokasta. Luodun olion funktiota kutsutaan tarvittaessa. (Kuvio 19.)

```
1  <cfscript>
2
3      objHenkilo = createObject("component", "Henkilo").init();
4
5      objHenkilo.setNimi("Testi");
6      objHenkilo.setPituus(1.80);
7      objHenkilo.setPaino(81.5);
8
9  </cfscript>
10
11 <cfoutput>
12     Nimi: #objHenkilo.getNimi()#<br />
13     Pituus: #objHenkilo.getPituus()#<br />
14     Paino: #objHenkilo.getPaino()#<br />
15 </cfoutput>
```

KUVIO 19. Esimerkki ColdFusion-komponentin käytöstä.

Kuviossa 19 riveillä 5-9 komponentin funktioilla asetetaan arvot muuttujille nimi, pituus sekä paino. Riveillä 12-14 komponentin funktioilla haetaan muuttujien arvot, tulostusta varten. Kuviossa 20 on nähtävissä tiedoston tuloste komponentin käytöstä.



KUVIO 20. Esimerkki ColdFusion-komponentin käytöstä saatavasta tulosteesta web-selaimessa.

4.5 ColdFusion-komponenttidokumentoija

ColdFusion-sovelluspalvelimen yhteyteen on rakennettu työkalu ColdFusion-komponettitiedostojen dokumentointia varten. Dokumentointi suoritetaan antamalla komponentille sekä komponentin funktioille attribuutteja haluttua informaatiota varten. Osa dokumentoijassa hyödynnettävistä attribuuteista on samoja, joiden avulla tapahtuu komponentin ja funktioiden määrittely. Taulukossa 2 on kuvattu attribuutit, joita komponenttidokumentoija hyödyntää muodostaessaan dokumentaatiota komponentista ja sen funktioista. (Gifford 2010, 47-50)

TAULUKKO 2. ColdFusion-komponettidokumentoijan käyttämät attribuutit ja selvitykset

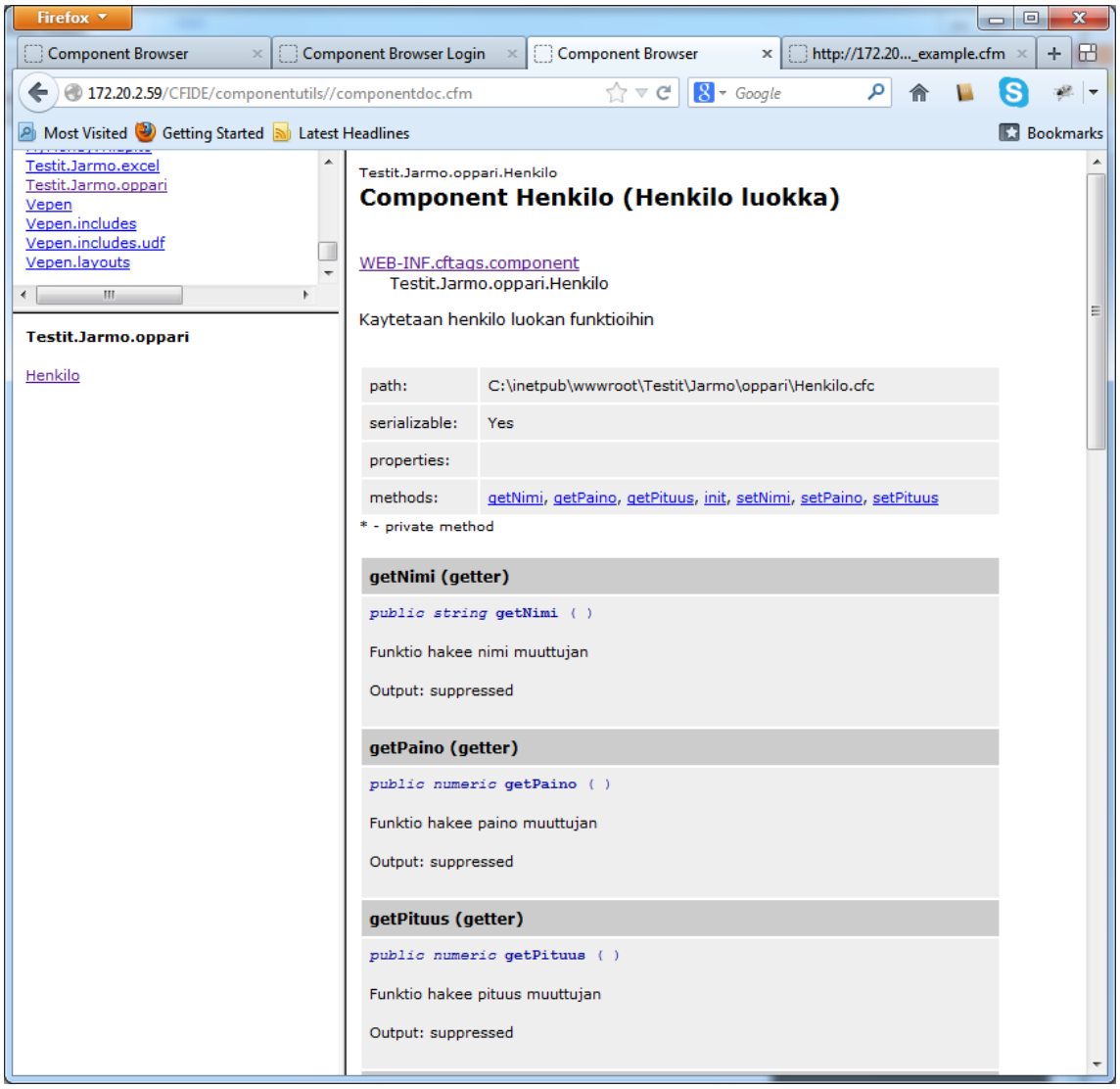
Attribuutti	Selvitys
Access	Funktion käyttöoikeus
Application	Sovelluksen nimi
Developer	Komponentin luoja
Displayname	Näyttönimi
Hint	Komponentin tai funktion kuvaus
Required	Onko funktion parametri pakollinen
Returntype	Funktion palautustyyppi
Type	Funktion parametrin tyyppi

Kuviossa 21 on esimerkki dokumentoidusta komponentista, jolle on annettu Taulukon 2 mukaisia dokumentointiattribuutteja. Komponentti on dokumentoitu käyttäen display-, hint-, Application- sekä Deleveloper-attribuutteja. Komponentin sisältämät funktiot on dokumentoitu käyttäen access-, displayname-, returntype- sekä hint-attribuutteja.

```
<cfcomponent displayname="Henkilo luokka" output="false"
    hint="Kaytetaan henkilo luokan funktioihin"
    Application="Testi Projekti" Developer="Jarmo Tolonen">
    <!--- Constructor--->
    <cffunction name="init" access="public" output="true" displayname="Constructor"
                returntype="void" hint="Luokan rakentaja">
        <cfscript>
            variables.ominaisuus = structNew();
            variables.ominaisuus.nimi = "";
            variables.ominaisuus.pituus = 0;
            variables.ominaisuus.paino = 0;
        </cfscript>

        <cfreturn this>

    </cffunction>
    <!--- get/set nimi --->
    <cffunction name="getNimi" access="public" output="false" displayname="getter"
                returntype="string" hint="Funktio hakee nimi muuttujan">
        <cfreturn variables.ominaisuus.nimi>
    </cffunction>

    <cffunction name="setNimi" access="public" output="false" displayname="setter"
                returntype="void" hint="Funktio asettaa nimi muuttujan">
        <cfargument name="nimi"  required="true" type="string">

        <cfset variables.ominaisuus.nimi = arguments.nimi>

    </cffunction>
    <!--- get/set pituus--->
    <cffunction name="getPituus" access="public" output="false" displayname="getter"
                returntype="numeric" hint="Funktio hakee pituus muuttujan">
        <cfreturn variables.ominaisuus.pituus>
    </cffunction>

    <cffunction name="setPituus" access="public" output="false" displayname="setter"
                returntype="void" hint="Funktio asettaa pituus muuttujan">
        <cfargument name="pituus" required="true" type="numeric">

        <cfset variables.ominaisuus.pituus = arguments.pituus>

    </cffunction>
```

KUVIO 21. Esimerkki komponentista, johon on lisätty attribuutit dokumentoinnin toteutusta varten.

Adobe ColdFusion 10 -sovelluspalvelimessa on sisäänrakennettuna dokumentoitujen komponenttien lukuohjelma CFC-Explorer. CFC-Explorer-ohjelmalla pystytään lukemaan kaikki kyseisellä palvelimella sijaitsevat ColdFusion-komponentit ja niiden sisältämät tiedot sekä ominaisuudet. CFC-Explorer-ohjelma toiminnaltaan selainpohjainen ja muistuttaa toiminnaltaan sekä ominaisuuksiltaan Java API -selainta (Kuvio 22.). CFC-Explorer-ohjelman lisäksi komponenttien tietoja voidaan hakea ohjelmakoodissa getComponentMetaData-fuktion avulla. (Gifford 2010, 48)



KUVIO 22. Näkymä CFC-Explorer-ohjelmassa valmiista komponenttidokumentaatiosta

5 WEB-SOVELLUKSEN KIRJAUTUMISKOMPONETTI

Web-sovelluksen kirjautumiskomponentti luvussa käydään läpi työssä toteutettavalle komponentille asetetut vaatimukset, komponentin toteutuksen suunnittelu sekä toteutus. Komponentin suunnittelua ja toteutusta kuvataan luvussa tietyiltä osin vain pääpiirteittäin, koska kehityksen kohteena olevaan web-sovellukseen kohdistuvan tietoturvariskin vuoksi on mahdotonta kuvata komponentin toimintaa täydellisesti.

5.1 Vaatimukset

Opinnäytetyön toimeksiantaja on kehittämässä oman elämän hallintaan sovellusta nuorille, yläasteen sekä toisen asteen opiskelijoille. Opinnäytetyön toimeksiantaja on alustavasti arvioinut sovelluksella olevan tulevaisuudessa useita tuhansia käyttäjiä. Kehitteillä olevan sovelluksen kehitystyö aloitetaan alusta, toteutettava kirjautumiskomponentti on ensimmäinen osa kehitteillä olevaa sovellusta.

Toteutettavalle sovellukselle on laadittu alustava vaatimusmäärittely, jossa on määritelty sovelluksen tietokantataulujen rakenne ja sovelluksen toimintakaavio sekä käyttäjärooleihin perustuvat käyttökaaviot (Liite 1). Sovelluksen käyttäjäroolit tulevat olemaan opiskelija, opettaja ja ylläpitäjä sekä FINPEC, jolla ei ole sovelluksessa käyttörajoituksia.

Opinnäytetyön toimeksiantajan pyynnöstä kirjautumiskomponenttiin tulisi toteutuksen yhteydessä sisällyttää kehitteillä olevalle sovellukselle ajanmukainen ja sovelluksen käyttötarkoitusta kuvaava etusivu, jossa käytössä olisi nykyisin suosittua animaatiota ja muita näyttäviä toiminnallisuuksia. Toteutettavaan etusivuun tulisi lisäksi sisällyttää kirjautumislomake sovellukseen.

Kehitteillä oleva sovellus toteutetaan Adobe ColdFusion 10 -sovelluspalvelinympäristössä. Toteuttavan kirjautumiskomponentin tulisi hyödyntää mahdollisimman hyvin sovelluspalvelimen tarjoamia ominaisuuksia kirjautumisen yhteydessä ja sovelluksen istunnonhallinnassa.

Sovelluksen käyttäjätiedot tullaan tallentamaan MySQL-tietokantaan sovelluksen vaatimusmäärittelyn mukaisesti. Sovelluksen kehittämisen alkuvaiheessa käyttäjien tiedoista

suojataan ainoastaan salasana. Suojauksen tulisi olla mahdollisimman tehokas mutta vähän järjestelmää kuormittava.

5.2 Suunnittelu

Annetun tehtävän suunnittelua aloittaessa kävi selväksi, että toteutettava kirjautumiskomponentti on kokonaisuuden kannalta järkevintä jakaa toiminnan mukaisesti omiksi kokonaisuuksiksi. Sovelluksen etusivulle luodaan oma komponentti, kirjautumiskomponentin toiminnallisuudelle oma komponentti sekä salasanojen suojauskäsittelylle oma komponentti. Tarkoituksena on selkeyttää toteutettavan komponenttikokonaisuuden rakennetta ja toimintaa.

Sovelluksen etusivu toteutetaan HTML-kielellä ja CSS-tyylimäärittelyjä hyväksi käyttäen. Rakenteeltaan HTML-dokumentti koostuu pääsääntöisesti div-elementeistä. HTML-dokumentissa käytetään kuvina mainostoimiston tekemiä kuvia. Animaatiot toteutetaan kirjautumislomakeosaan ja HTML-dokumentille luodaan muuttuva ilme kellonajan mukaan, riippuen onko vuorokaudenaika aamu, päivä vai ilta. Sovelluksen etusivu toteutetaan kokonaisuudessaan omaan komponenttiinsa.

Salasanojen suojaus toteutetaan omaan komponenttiinsa. Salasanojen suojaus toteutetaan SHA-512 -tiivistefunktiota käyttäen. Salasanojen suojausta vahvistetaan käyttämällä muuttuvaa suolausta salasanoissa. Käytettävä suojaustapa on yksisuuntainen, joten tallennettua salasanaa ei pureta vertailussa käyttäjän antamaan salasanaan. Tarkastusprosessi toteutetaan tekemällä käyttäjän antamalle salasanalle sama toiminto kuin salasanan luonnin yhteydessä.

Sovellukseen kirjautuminen, käyttäjätietojen tarkastaminen ja istunnon luonti toteutetaan omassa komponentissaan. Sovelluksen istunnonhallinta toteutetaan sovelluspalvelimen istunnonhallintaa hyväksi käyttäen. Sovelluspalvelimen istunnonhallintaa ohjataan toiminnallisuuskomponentista käsin sisäänkirjautumisen sekä uloskirjautumisen yhteydessä.

5.3 Toteutus

Annetun tehtävän toteutus koostuu useammasta ColdFusion-komponentista: Etusivu-komponentti, suojauskomponentti ja toiminnallisuuskomponentti.

Toteutettu toiminnallisuuskomponentti suorittaa sovelluksen kirjautumisen. Toiminnallisuuskomponentti vaatii toimintaansa käyttäjätunnuksen ja salasanan, jotka välitetään komponentille sovelluksen kirjautumislomakkeelta. Toiminnallisuuskomponentin toimintaa ei voida kuvata tarkasti tässä dokumentissa tietoturvariskin takia. Toimintaperiaatteeltaan komponentin suoritus aloitetaan tarkastamalla käyttäjätunnuksen ja salasanan muodollinen oikeellisuus. Tämän jälkeen salasana välitetään suojauskomponentille tiivistefunktion ja suolauksen suorittamista varten. Suojauskomponentilta saadun tuloksen perusteella suoritetaan tietokantahaku, jossa haetaan yhtäläisyyttä annetulle käyttäjätunnukselle ja salasanalle. Yhtäläisyyden löydyttyä toiminnallisuuskomponentissa luodaan istunnon sovelluksen käyttöä varten. Istunnon luontiin käytetään ColdFusion-sovelluspalvelimen ominaisuuksia.

Toteutettavan kirjautumiskomponentin yhtenä osana on toteuttaa kehitettävälle web-sovellukselle etusivu, joka samalla toimii kirjautumiskomponentin käyttöliittymänä. Etusivu toteutetaan yhteen cfm-tiedostoon (Kuvio 23.), käyttäen hyväksi sivunasettelulle luotua ColdFusion-komponenttia. Sivunasettelukomponentissa HTML-koodi on jaettu osiin ja funktioitettu. Kirjautumislomakkeelle sekä kirjautumislomakkeen ilmoitusosalle on luotu omat funktiot. Tiedostossa sivunasettelukomponentista luodaan olio, jonka avulla HTML-dokumentin osat saadaan näkyviin.

```
<cflogin>

    <cfset objEtusivu = CreateObject("component", "MyMoney.Komponentti.Etusivu")>

    <cfoutput>#objEtusivu.head('#REQUEST.teksti.mymoney_yleiset_finpec#')#</cfoutput>

    <cfoutput>#objEtusivu.body('#REQUEST.teksti.mymoney_yleiset_mymoney#')#</cfoutput>

    <cfoutput>#objEtusivu.startContent()#</cfoutput>

    <cfoutput>#objEtusivu.notification()#</cfoutput>

    <cfoutput>#objEtusivu.form()#</cfoutput>

    <cfoutput>#objEtusivu.endContent()#</cfoutput>

    <cfoutput>#objEtusivu.end()#</cfoutput>

</cflogin>

<!--- Jos cflogin-container on käytössä ja kirjautuneen käyttäjän rooli on 1-4,
        (1=Opiskelija,2=Opettaja,3=Ylläpitäjä,4=Finpec), ohjataan paavalikko sivulle. --->

<cfif IsUserInRole("1") or IsUserInRole("2") or IsUserInRole("3") or IsUserInRole("4")>
    <cflocation url="/MyMoney/paavalikko" addtoken="no">
</cfif>
```

KUVIO 23. Tulostus index.cfm-tiedostosta

Sovellukselle luodulle etusivulle tehdään toiminto, jossa etusivun värimaailmaa ja kuvia muutetaan sivun lataushetken kellonajan perusteella. Toiminnallisuus toteutetaan CSS-tyylimääritteiden sekä JavaScriptin avulla.

Etusivun teeman vaihto toteutetaan luomalla CSS-tyylimääritteihin luokkia, yhteensä kuusi erilaista, joista kolme on header-tunnisteella identifioidulle div-elementille ja kolme content-tunnisteella identifioidulle div-elementille (Kuvio 24.).

```
/******* asetaTausta()-funktion vaatimat luokat *******/
.aamu {
    background-color:#B9D6E6;
    color:#0B265B;
    font-family:Arial,Helvetica,sans-serif;
    height:100px;
    padding-top:15px;
    padding-bottom:0;
    width:1100px;
}
.aamuCont {
    background-color:#B9D6E6;
    background-image:url("../Images/karvapallo_aamu_m.jpg");
    background-repeat:no-repeat;
    background-position:right;
    font-family:Arial,Helvetica,sans-serif;
    float:none;
}
```

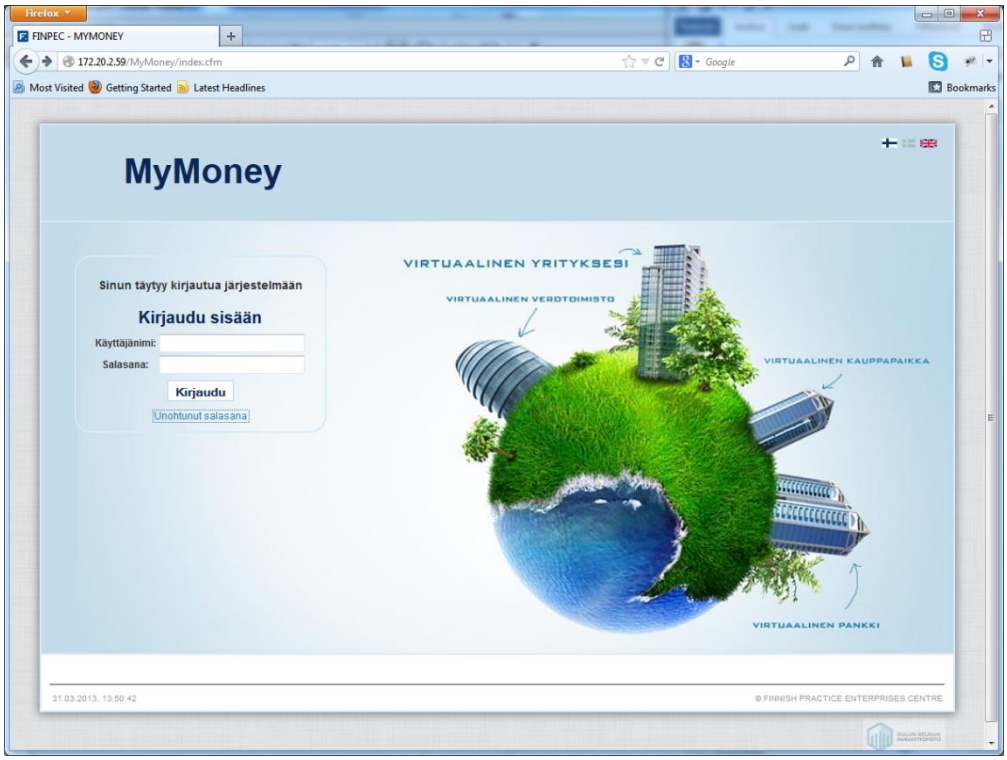
KUVIO 24. Ote taustan asettamiseen käytettävästä tyylimäärittelystä

Etusivun teeman asettamisen suorittaa JavaScript-funktio (Kuvio 25.). Funktiota kutsutaan HTML-dokumentin latauksen yhteydessä onLoad-tapahtumankäsittelijän kautta. Suoritettavan funktion toimintalogiikka on pyritty pitämään mahdollisimman yksinkertaisena. Funktion suoritus aloitetaan hakemalla suoritushetkellä olevan tunnin numero. Tunnin numeron perusteella asetetaan tietyn niminen CSS-tyylimäärittely luokka, header- ja content-tunnisteella identifioidulle div-elementeille.

```
function asetaTausta(){

    var paiva = new Date();
    var tunti = paiva.getHours();
    var ajankohta = "paiva";

    if(tunti >= 16 && tunti < 20 || tunti >=7 && tunti < 10) ajankohta = "aamu";

    if(tunti <= 6 || tunti >= 20) ajankohta = "ilta";

    document.getElementById("header").className = ajankohta;
    document.getElementById("content").className = ajankohta + 'Cont';
}
```

KUVIO 25. Taustan asettamiseen käytettävä JavaScript-funktio

Kuviossa 26 on nähtävillä sovelluksen etusivu valmiiksi latautuneena, jolloin kaikki animaatiot on suoritettu. Kuviossa nähtävässä sovelluksen etusivun väritys ja kuva on näkymä päivänäkymästä. Liitteessä 2 on nähtävänä kaksi muuta tilaa, jotka etusivu voi saada kellonajasta riippuen.



KUVIO 26. Näkymä sovelluksen kirjautumissivusta klo 10.00 – 16.00

Etusivu-komponentissa kirjautumislomakkeelle on luotu oma funktio. Funktio sisältää lomakkeen asettelun div-elementtien avulla. Div-elementit mahdollistavat lomakkeen animaation ja visuaalisen tyylien määrittelyn. Lomakkeen tiedot välitetään HTML:n post-metodia käyttäen toiminnallisuuskomponentin login-funktiolle, jotta välitettävät kirjautumistiedot eivät tule näkyviin selaimenosoiteriville. Kirjautumislomake on toteutettu HTML- sekä CFML-kielellä (Kuvio 27.).

```
<div id="login">
    <cfform action="Action.cfc?method=login" name="Form" method="post">
        <cfinput type="hidden" name="ip" value="#CGI.REMOTE_ADDR#">
        <div id="title">
            <cfoutput>#REQUEST.teksti.mymoney_yleiset_login_tervehdys#</cfoutput>
        </div>
    <div id="cont">
        <table>
            <tr>
                <th>
                    <cfoutput>#REQUEST.teksti.mymoney_yleiset_kayttajatunnus#</cfoutput>:
                </th>
                <td>
                    <cfinput id="username" type="text" name="username" maxlength="50" required="yes" onError="validointiVirhe('kt');">
                </td>
            </tr>
            <tr>
                <th>
                    <cfoutput>#REQUEST.teksti.mymoney_yleiset_salasana#</cfoutput>:
                </th>
                <td>
                    <cfinput id="password" type="password" name="password" maxlength="50" required="yes" onError="validointiVirhe('ss');">
                </td>
            </tr>
        </table>
    </div>
    <div id="button">
        <cfinput id="loginButton" name="loginButton" type="submit" value="#REQUEST.teksti.mymoney_painike_kirjaudu#"/>
    </div>
    </cfform>
    <div id="link">
        <a href="newpass"><cfoutput>#REQUEST.teksti.mymoney_yleiset_unohtunut_salasana#</cfoutput></a>
    </div>
</div>
```

KUVIO 27. Tulostus kirjautumislomakkeen ohjelmakoodista

Kuviossa 28 on näkymä kirjautumislomakekokonaisuudesta, joka muodostuu ilmoitusosasta ja kirjautumislomakeosasta. Ilmoitusosa on toteutettu Etusivu-komponentissa omana funktiona kirjautumislomakeosan tapaan. Peräkkäin suoritettuna funktiot muodostavat kirjautumislomakekokonaisuuden.

KUVIO 28. Näkymä sovelluksen kirjautumislomakkeesta

Kuviossa 29 on näkymä virheilmoituksesta epäonnistuneen kirjautumisen yhteydessä. Ilmoitus tulostetaan kirjautumislomakekokonaisuuden ilmoitusosassa. Ilmoitus annetaan toiminnallisuuskomponentista virheen tapahtuessa.

KUVIO 29. Näkymä sovelluksen kirjautumislomakkeen virheilmoituksesta

Kirjautumislomakkeen animaation toteutukseen käytetään jQuery JavaScript-kirjaston show-funktiota. Funktioiden suoritus määritetään alkamaan HTML-dokumentin ollessa latautuneen valmiiksi selaimeen. Lomakkeen animaatio aloitetaan pudottamalla notification-tunnisteella identifioitu div-elementti yhden sekunnin aikana näkyviin. Funktion suorituksen päätyttyä suoritetaan lomakkeen animaation seuraava vaihe, jossa login-tunnisteella identifioitu div-elementti liu'utetaan esiin yhden sekunnin aika (Kuvio 30.). Lisäksi samassa yhteydessä etusivun footer-elementissä oleva teksti tuodaan näkyviin yhden sekunnin aikana.

```
$(document).ready(function() {

    $('div##notification').show("drop", { direction: "up" }, 1000, function() {
        $('div##login').show("slide", { direction: "up" }, 1000);
    });

    $('div##footer').show();
    $('table##footertable').fadeIn(1000);

});
```

KUVIO 30. Kirjautumislomakkeen animaation toteutus jQuery-kirjaston avulla

Kirjautumiskomponentin toteutukseen on käytetty useita eri ohjelmistoja ja kehitystyökaluja. Toteutukseen käytetyt ohjelmistot ja kehitystyökalut (Taulukko 3.) koostuvat integroduista kehitysympäristöistä, web-selaimista, HTML- ja JavaScript-koodin debuggerista sekä tietokannanhallintatyökalusta.

TAULUKKO 3. Työssä käytetyt ohjelmistot ja kehitystyökalut

Ohjelmisto	Versio
Adobe Dreamweaver	5.5
SQLyog Community	9.6.0
Eclipse	Juno Release (4.2)
Firebug	Viimeisin versio
Internet Explorer	9.0
Mozilla Firefox	Viimeisin versio
Google Chrome	Viimeisin versio

6 POHDINTA

Opinnäytetyö onnistui mielestäni erittäin hyvin, opinnäytetyön toiminnallisen osan tavoitteena oli määritellä, suunnitella ja toteuttaa web-sovellukselle kirjautumisen mahdollistava kirjautumiskomponentti. Kirjautumiskomponentti piti sisällään kirjautumiseen vaadittavan käyttöliittymän joka toimii samalla kehitteillä olevan web-sovelluksen etusivuna. Sain sisällytettyä työhön kaikki toimeksiantajalta tulleet vaatimukset ja toiveet. Työ osoittautui haastavaksi mutta opinnäytetyön näkökulmasta tehtäviä oli sopivasti ja työn rajaus onnistui hyvin. Suurimpana haasteena toteutetussa työssä oli minulle uuden teknologian käyttö, jota en ollut aikaisemmin käyttänyt.

Opinnäytetyön toiminnallisen osan toteutin vaiheittain. Ensimmäisenä tutustuin sovelluspalvelimessa käytettäviin ohjelmointikieliin erilaisten tutoriaalien kautta. Käytettävien kielten omaksumista helpotti aikaisempi ohjelmointikokemus. Asensin itselleni oman Adobe ColdFusion -sovelluspalvelimen, jonka avulla opiskelin sovelluspalvelimen asetuksia ja ominaisuuksia. Opinnäytetyön toimeksiantajalla on ollut käytössä jo useamman vuoden ajan Adobe ColdFusion -teknologia. Sain mahdollisuuden tutustua ja käyttää hyväksi aikaisemmin toteutettuja sovelluksia ja niiden ratkaisuja. Tämä helpotti todella paljon sovellusten toimintarakenteen ymmärtämistä. Seuraavassa vaiheessa aloitin tutustumisen erilaisiin salaustekniikoihin. Vertaillessa kävin läpi tekniikoiden suojaavuutta ja käytettävyyttä. Samalla suunnittelin komponentin rakennetta ja lisäksi hahmottelin kirjautumissivun visuaalista ilmettä. Visuaalisen ilmeen suunnittelua pidin varsin haastavana, koska olen mielestäni enemmän teknisesti suuntautunut ihminen. Viimeisenä vaiheena työssä kokosin aikaisempien vaiheiden tulokset yhteen ja toteutin kirjautumiskomponentin vaadittavine ominaisuuksineen.

Parannettavaa jäi mielestäni työn aikataulutuksessa. Toiminnallinen osa työstä valmistui aikataulussa, kirjallisen osan ollessa jäljessä aikataulusta noin vuoden verran. Tätä selittänee toiminnallisen osan mielenkiintoisuus ja panostaminen sen toteutukseen. Kirjallisen osuuden tekeminen jäi hieman kesannolle toteutusvaiheen aikana. Toiminnallisen osan valmistuttua kirjallisen osan tekeminen siirtyi aina tuonnemmaksi. Kirjallisen osuuden valmistumista on myös osaltaan hidastanut ensimmäisen lapsen syntymä prosessin aikana.

Prosessina opinnäytetyö on ollut todella kehittävä. Olen päässyt opinnäytetyön kautta tutustumaan uuteen sovelluspalvelintekniikkaan sekä uusiin ohjelmointikieliin, joihin en todennäköisesti olisi muuten tutustunut. Lisäksi opin työn tekemisen seurauksena ymmärtämään web-sovellusten kirjautumiskäytänteistä sekä salasanojen suojauskäytänteistä todella paljon. Aiheena opinnäytetyö on ollut todella mielenkiintoinen ja ajankohtainen.

Työn onnistumisesta kertoo toimeksiantajan aikomus hyödyntää kirjautumiskomponentin toiminnallisuutta tulevaisuudessa myös muissa tarjoamissaan sovelluksissa.

LÄHTEET

365 Computer Security Training. 2013. What is Rainbow Table? | 365 Computur Security Training. Hakupäivä 9.5.2013, http://www.computer-network-security-training.com/what-is-a-rainbow-table/

Adobe. 2013.Build, develop, & manage rich internet applications | Adobe ColdFusion 10 Family. Hakupäivä 23.3.2013, http://www.adobe.com/fi/products/coldfusion-family.html

Farrar, J. 2010. ColdFusion 9 Developer Tutorial. Birmingham. Packt Publiching Ltd.

Gifford, M. 2010. Object-Oriented Programming in ColdFusion. Birmingham. Packt Publiching Ltd.

Gilmore, J. W. 2005. PHP & MySQL – Tehokas hallinta. Suom. Arto Kuvaja. Jyväskylä: Gummerus.

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Helsinki: Talentum.

if.fi. 2013. SSL-suojaus – if.fi. Hakupäivä 23.3.2013, http://www.if.fi/web/fi/henkiloasiakkaat/ifkansio/pages/ssl-suojaus.aspx

jQuery Board. 2012. jQuery: The Write Less, Do More, JavaScript Library. Hakupäivä 5.3 2012, http://jquery.com/.

Järvinen, P. 2003. Salausmenetelmät. Jyväskylä. Docendo.

Korpela, J. K. & Linjama, T. 2005. Web-suunnittelu. Jyväskylä: Docendo.

MacLees, N. 2012. jQuery for Desingners. Birmingham. Packt Publiching Ltd.

Mozilla Developer Network. 2013. JavaScript technologies overview - JavaScript | MDN. Hakupäivä 30.3.2013, https://developer.mozilla.org/en-US/docs/JavaScript/Guide/JavaScript_Overview

OpenID Fundation. 2013. OpenID Fundation website. Hakupäivä 30.3.2013, http://openid.net/

OWASP. 2013. OWASP, The Open Web Application Security Project – Session management Cheat Sheet. Hakupäivä 30.3.2013, https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

Peltomäki, J. & Nykänen, O. 2006. Web-selainohjelmointi. Jyväskylä: Docendo.

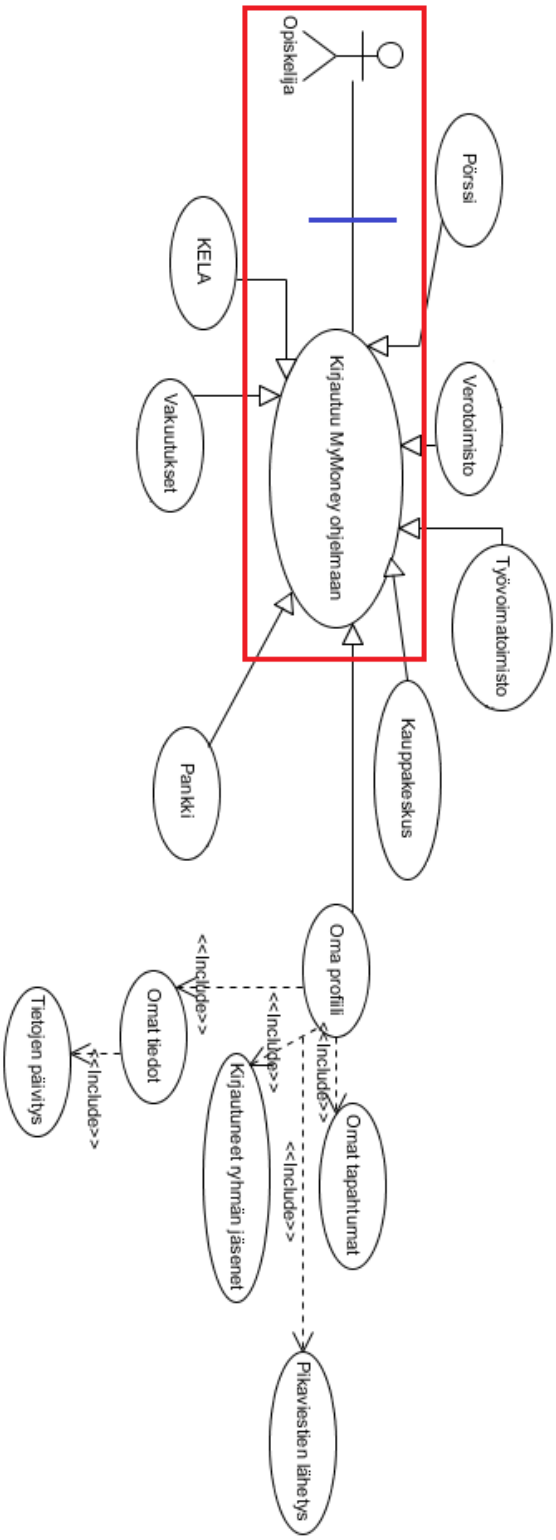
Rantala, A. 2005. Web-ohjelmointi. Jyväskylä: Docendo.

w3school 2013. JavaScript Tutorial. Hakupäivä 30.3.2013, http://www.w3schools.com/js/default.asp

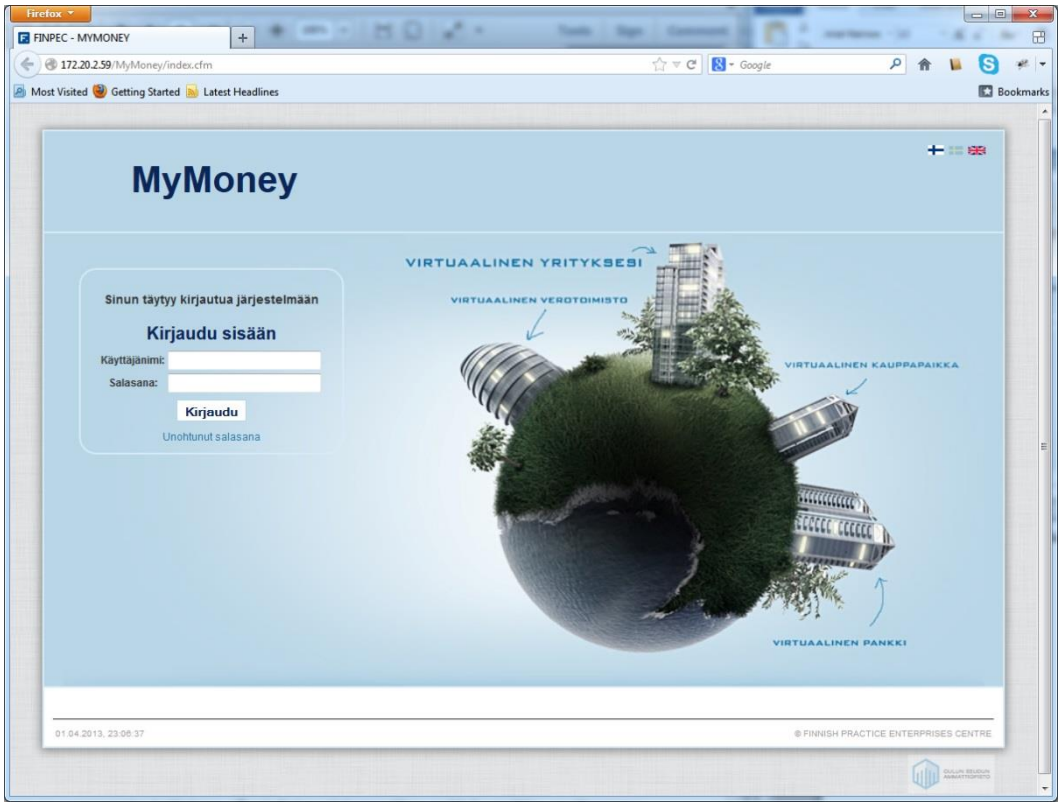
Wikipedia 2013. Adobe ColdFusion. Hakupäivä 23.3.2013, http://en.wikipedia.org/wiki/Adobe_ColdFusion

LIITTEET

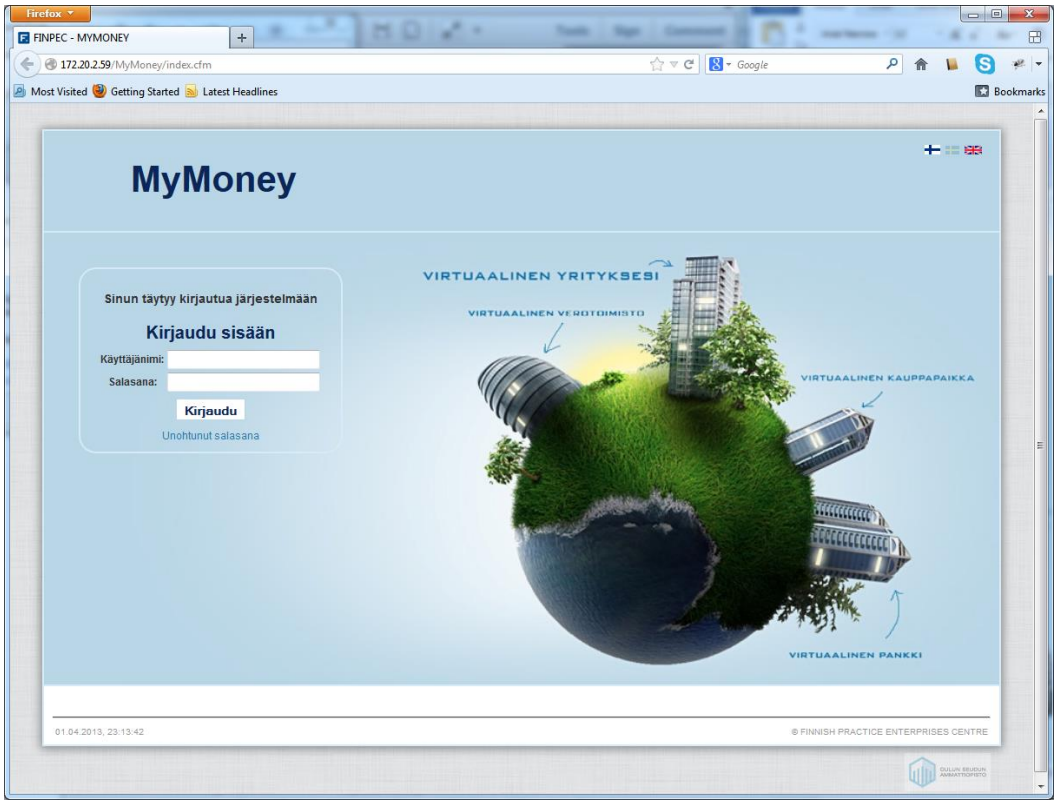
LIITE 1: OPISKELIJAN KÄYTTÖKAAVIO



LIITE 2. SOVELLUKSEN KIRJAUTUMISSIVUN NÄKYMÄT



KUVIO 1. Näkymä sovelluksen kirjautumissivusta klo 20.00 – 06.00



KUVIO 2. Näkymä sovelluksen kirjautumissivusta klo 06.00 – 10.00 ja 16.00 – 20.00