Dino Väärä

# Adobe ColdFusion tekniikkana yrityksen sisäisen järjestelmän luonnissa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

21.4.2013

Metropolia

| | |
|---|---|
| Tekijä(t)<br>Otsikko<br><br>Sivumäärä<br>Aika | Dino Väärä<br>Adobe ColdFusion tekniikkana yrityksen sisäisen järjestelmän luonnissa<br>41 sivua + 5 liitettä<br>21.4.2013 |
| Tutkinto | insinööri (AMK) |
| Koulutusohjelma | Tietotekniikan koulutusohjelma |
| Suuntautumisvaihtoehto | Ohjelmistotekniikka |
| Ohjaaja(t) | Lehtori Simo Silander<br>Lehtori Vesa Ollikainen |


Tässä opinnäytetyössä käsitellään ColdFusion-tekniikkaa ja sen mahdollistamia toteutuksia nykyaikaisessa web-ympäristössä. Työssä keskitytään Adoben tarjoamiin työkaluihin ColdFusionista.

Työn tavoitteena on antaa kattava kuva ColdFusion-tekniikasta ja siinä käytettävästä ColdFusion Markup Language -ohjelmointikielestä. Työssä käsitellään tarkemmin tietokantojen käsittelyä ja niiden käyttöä käytännön sivustoilla. Työn tarkoituksena on antaa tietoa ColdFusion-tekniikasta ja sen kehitysympäristöstä lukijalle.

Aluksi työssä käydään läpi ColdFusion-tekniikkaa ja sen käyttöä sekä minkälaisia ominaisuuksia se tarjoaa kehittäjille. Myös ColdFusion Markup Language -ohjelmointikieltä tarkastellaan alussa syntaksin ja rakenteen osalta. Alun esittelyn jälkeen työssä näytetään, kuinka ColdFusion-kehitysympäristö pystytetään ja millaisia vaihtoehtoja tähän on. Lopuksi työssä luodaan yksinkertainen tietokantasovellus ColdFusionilla.

Työn empiirisessä osassa pyritään antamaan selkeä kuva ColdFusionista ja siitä, kuinka käytetään. Työn loppuosa esittelee, miltä ColdFusion näyttää käytännössä.



| | |
|---|---|
| Avainsanat | ColdFusion, web-ympäristö, sovelluspalvelin |

| | |
|---|---|
| Author(s)<br>Title<br><br>Number of Pages<br>Date | Dino Väärä<br>Adobe ColdFusion as a method of creating an in-house system<br><br>41 pages + 5 appendices<br>21 April 2013 |
| Degree | Bachelor of Engineering |
| Degree Programme | Information Technology degree programme |
| Specialisation option | Software engineering |
| Instructor(s) | Simo Silander, Lecturer<br>Vesa Ollikainen, Lecturer |

This thesis deals with ColdFusion and its implementation in modern web development. Work will focus on the tools that are offered by Adobe for ColdFusion.

The aim is to provide a comprehensive view of the ColdFusion technology and of ColdFusion Markup Language which is used in in conjunction with ColdFusion. Working with databases in ColdFusion will be viewed in more detail. The purpose is to provide information about the ColdFusion technology and its development environment for the reader.

To start with there is some explanation of the ColdFusion in general and what kind of features it has to offer for developers. Also, there will be a overview of the programming language ColdFusion Markup Language in the beginning for the syntax and structure. Next setup of a ColdFusion development environment will be shown. Finally, a simple database application for ColdFusion is created.

The empirical section of the work is intended to provide a comprehensive view of ColdFusion and how it is used. The final part of the work presents what ColdFusion looks like when it is used in practice.

| | |
|---|---|
| Keywords | ColdFusion, web environment, application server |

## Sisällys

## Lyhenteet

| | |
|---|---|
| CFC | ColdFusion Component on .cfc-päätteinen tiedosto, joka voi sisältää dataa ja funktioita. |
| CFML | ColdFusion Markup Language on ohjelmointikieli, jota käytetään ColdFusionissa. |
| Olio | Olio on olio-ohjelmoinnin perusyksikkö, joka sisältää yhteenkuuluvia tietoja ja toiminnallisuuksia. |
| Funktio | Funktiot ovat ohjelmassa olevia koodipätkiä, jotka voivat muuttaa ohjelman ja datan tilaa tai muokata funktiolle annettuja ja sen tuntemia tietoja. |
| Layout | Layout tarkoittaa sivuston ulkoasua, layout:sta puhuttaessa web-maailmassa usein viitataan myös web-sivustojen tyylitiedostojen rakenteeseen. |

# 1 Johdanto

Web-maailma on muuttunut paljon vuosien saatossa, ja muutoksen vauhti vain kiihtyy. Nykyisin pelkkää HTML-koodia ja CSS-tyylejä osaa lukea ja ymmärtää suuri osa internetin käyttäjistä. Tämä on kuitenkin vain pintaraapaisu, sillä sivustojen alla vaikuttaa lähes väistämättä muita tekniikoita. Web-sivustojen käyttö on jokapäiväistynyt ja internetissä pyritään jatkuvasti tarjoamaan enemmän vaihtoehtoja kaupankäynnille, opiskelulle ja työnteolle. Samalla web-maailman täytyy pystyä tarjoamaan uusia tekniikoita ja suoritustapoja kasvavan tarpeen mukaan. Esimerkiksi nettikaupoissa voidaan dynaamisen ostoskorin käytön lisäksi käydä reaaliaikaista keskustelua asiakaspalvelijan kanssa. Tällainen sivusto vaatii käyttöönsä uusia tekniikoita, joita kehitetään jatkuvasti. Lähes jokaisella sivustolla tarvitaan tietokantoja sekä dynaaminen toteutus.

Tämän opinnäytetyön tarkoituksena on esitellä ColdFusion-tekniikkaa ja sen mahdollistamia toteutuksia nykyaikaisessa web-ympäristössä. ColdFusion on jo 1995 kehitetty nopean web-kehityksen sovellusalusta. Alkuperäistä tekniikkaa kehittää nykyisin Adobe, ja työssä erityisesti keskitytään Adoben tarjoamiin työkaluihin. Tavoitteena on antaa kattava kuva ColdFusion-tekniikasta ja siinä käytettävästä ColdFusion Markup Language -ohjelmointikielestä. Tarkemmin työssä käsitellään tietokantojen käsittelyä ja niiden käyttöä käytännön sivustoilla. Työn tarkoitus on tarjota lukijalle riittävä tietämys ColdFusion-tekniikasta ja sen kehitysympäristöstä, jotta yksinkertaisen web-sivuston, jolla käsitellään tietoja tietokannasta, luonti olisi mahdollista.

Aluksi selvitetään, mihin ColdFusion-tekniikka käytetään ja minkälaisia ominaisuuksia se tarjoaa kehittäjille yleisesti. Myös sovellusalustan kanssa käytettyä ohjelmointikieltä ColdFusion Markup Languagea käydään läpi ja tarkastellaan sen syntaksia ja rakennetta. Alun esittelyn jälkeen työssä näytetään, kuinka ColdFusion-kehitysympäristö pystytetään ja millaisia erilaisia vaihtoehtoja siihen on tarjolla. Lopuksi työssä luodaan esimerkkinä tietokantasovellus, jossa esitellään, kuinka ColdFusionia käytetään tietokantojen käsittelyyn ja millä lailla sen tarjoamat työkalut toimivat käytännössä.

# 2 ColdFusion-tekniikka

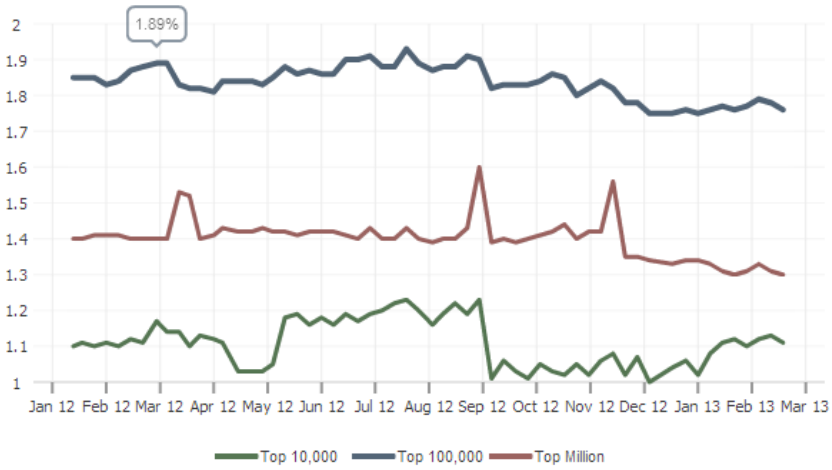
## 2.1 Käyttö

Kun sivustolta tarvitaan kykyä muuntautua käytön tai katselijan mukaan, puhutaan dynaamisesta sivustosta. Kun sivusto luodaan latauksen yhteydessä ja käytössä on esimerkiksi tietokanta ja kirjautuminen, tarvitaan palvelinpuolen koodia hoitamaan yhteys sivuston ja tietokannan välille (14, s. 1). Tällaisen toiminnallisuuden luomiseksi tarvitaan ohjelmointikieli jolla voidaan luoda palvelinpään käsittely. Tällaisia ohjelmointikieliä on useita, muun muassa

- PHP
- ASP
- Perl
- ColdFusion Markup Language.

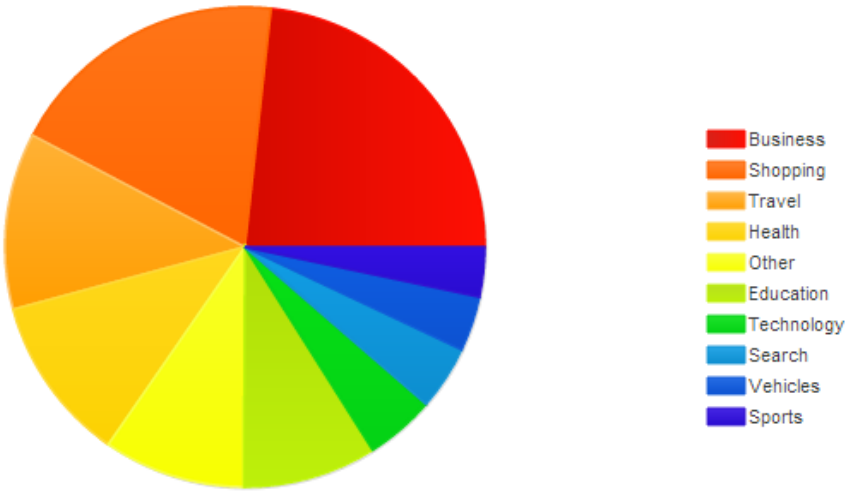
Kaikille näille kielille on yhteistä mahdollisuus yhdistää sivustoille useita resursseja, kuten tietokanta tai web service. Suurin ero kielten välillä on niiden syntaksi ja valmiit funktiot. Erona kuitenkin kaikkiin muihin ColdFusion mahdollistaa täyden hallinnan sen omaan lähdekoodiin ja mahdollistaa kehittämisen Javalla, jolla ColdFusion on kehitetty, uusien toiminnallisuuksien luomiseksi. ColdFusionissa käytetyn ColdFusion Markup Languagen syntaksi on myös erittäin helppo oppia ja sillä kehittämisen aloittaminen helppoa. (15, s. 1-4.)

ColdFusion on sovelluspalvelin, joka mahdollistaa nopean kehityksen, käyttöönoton sekä ylläpidon (1, s. 1). ColdFusionn kehitys on ilmaista, mutta ColdFusion Standard Edition, joka on suunnattu sivustolle keskiverto liikenteellä, maksaa jo 1600 € per kaksi konetta ja tästä seuraava versio ColdFusion Enterprise edition maksaa 7000 € per kaksi konetta (2, s. 1).Voidaan siis sanoa, että ColdFusion ei sovellu harrastesivustojen käyttöön ja on selkeästi suunnattu ammattikäyttöön. Seuraavaksi tarkastellaan käytön määrää sekä sen laatua.




**Kuva 1 Adobe ColdFusionin käyttö (3, s. 1)**

Kuvassa 1 näkyy historiaa ColdFusionin käytöstä. Y-akseli kuvaa prosentuaalista osaa web-sivuista, joissa ColdFusion on käytössä ja X-akseli aikaa. Käyrät kuvaavat Adobe ColdFusionin käyttöä sivustoilla (3, s.1). Kuvasta voidaan päätellä, että käytön osuus web-sivustoissa on pieni, myös kokonaisuudessaan sen käyttö laskevaa. Tähän asiaan palataan vielä luvussa 6.




**Kuva 2 10 Suosituinta käyttökohdetta 100 000 käytetyimmän sivun joukossa (3, s. 1)**

Piirakkakaavio kuvaa käyttökohteita. Kuvasta näkyy, että suurin osa käytöstä on kaupallista. Yleisesti ColdFusionin käyttö on kallista, sen osuus web-sivuista on pieni ja käyttö kaupallista.

## 2.2 Miksi valita ColdFusion?

Samankaltaisia tekniikoita on tarjolla useita. Miksi sitten valita ColdFusion? Ennen selvänä tekijänä ColdFusionin valintaa vastaan on ollut sen hinta. Vaikka työssä käsitellään suurimmaksi osaksi Adoben ColdFusionia, on otettava huomioon, että ColdFusionin käyttö ilmaiseksi on nyt mahdollista OpenBD:n tai Railon ansiosta (16, s. 1). OpenBD ja Railo ovat vapaan lähdekoodin CFML-moottoreita (ColdFusion Markup Language), joiden avulla voi ColdFusionia kehittää ilmaiseksi (23, s. 1;24 s.1). ColdFusion Markup Languagesta kerrotaan enemmän luvussa 3. Koodin määrä on suoraan verrannollinen ajan käyttöön. Kun koodi on optimoitua ja sen syntaksi mahdollistaa toiminnon kirjoittamisen pienemmällä määrällä rivejä, on selvää, että aikaa säästyy. ColdFusion on useisiin muihin kieliin verrattaessa nopeampaa kehittää juurikin tästä syystä (16, s 1). CFML-ohjelmointikieli on sen funktioiden ansiosta rikkaampi kuin monet muut kielet ja useita toimintoja on tehty helpommiksi tai valmiiksi kehittäjille (16, s. 1). Tuotantoon ajoa ColdFusionissa on helpotettu huomattavasti useilla mahdollisilla vaihtoehdoilla sovelluspalvelimissa ja tietokannoissa (16, s. 1; 17, s. 9).

Yhteenvetona voidaan sanoa, että ColdFusion kilpailee erityisesti sen nopeuden ja muuntautuvuuden avulla muita kieliä vastaan, ja sen käyttö säästää selvästi aikaa.

## 2.3 Ominaisuudet

ColdFusion tarjoaa paljon ominaisuuksia, joista tässä työssä käsitellään uusimpia, version Adobe ColdFusion 10 ominaisuuksia. Alla on lueteltu muutamia ominaisuuksia.

- Tietokantoja voi kätevästi käsitellä koodissa, jossa kyselyt ohjataan omien tietokanta-ajurien kautta oikeaan lähteeseen (4, s. 1).

- ColdFusion mahdollistaa tehokkaan käsittelyn PDF-tiedostoille, joita voidaan dynaamisesti muodostaa sovelluksen ajon aikana.

- Koodia voidaan parantaa käyttämällä CFC-komponentteja (Komponentit sisältävät funktioita ja dataan. Niistä selitetään enemmän luvussa 3.3), joilla mahdollistetaan tehokas tietokantakyselyjen käsittely (5, s. 1).

- On mahdollista käyttää sisäänrakennettua web-palvelinta, mutta myös erillisiä, kuten IIS ja Apache Web Server (6, s. 2).

- Media Player -ominaisuus pystyy toistamaan uusimpia videotyyppejä kuten HTML 5 -video, flash-videoita tai YouTube -videoita (6, s. 3).

Listassa on vain murto-osa ominaisuuksista, mutta työssä tullaan erityisesti törmäämään näihin ominaisuuksiin.

# 3 ColdFusion Markup Language

## 3.1 Yleistä

ColdFusion Markup Language (CFML) on jo vanha kieli. Se luotiin 1995 ja on ollut aktiivisessa käytössä siitä lähtien. Kieli oli vain kaupallisessa käytössä, mutta 2008 kaksi CFML-moottoria, Open BlueDragon ja Railo julkaisivat vapaan lähdekoodin versionsa kaikkien käyttöön (7, s. 1). Työssä kuitenkin keskitytään Adoben tuoteperheeseen ja sen kautta myös Adoben referenssiin CFML:stä.

## 3.2 Syntaksi

Seuraavassa taulukossa listataan muutamia merkintöjä, joita tullaan käyttämään työssä esitettävässä tietokantasovelluksessa myöhemmin.

**Taulukko 1. Syntaksiesimerkkejä (8, s. 1)**

| Yleisesti | ColdFusion Markup Languagessa | Selitys |
|---|---|---|
| IF-lause | <cfif> | Rakentaa IF-lausekkeen |
| Tietokantakysely | <cfquery> | Lähettää kyselyjä tai SQL-lauseita tiedonlähteeseen |
| - | <cfoutput> | Mahdollistaa ColdFusion-muuttujien ja -funktioiden käytön koodin sisällä. |
| Pysäytyskomento | <cfabort> | Pysäyttää ColdFusion-sivuston toiminnan merkinnän kohdalle. |
| Debug-työkalu | <cfdump> | Tulostaa lähes minkä tahansa objektin tai muuttujan arvon ruudulle. Käytännöllinen |

| | | |
|---|---|---|
| | | debug-työkalu. |

## 3.3 ColdFusion-komponentit

### 3.3.1 Yleisesti

CFC-komponentit ovat tiedostoja, jotka sisältävät funktioita ja dataa, jotka mahdollistavat oliopohjaisen ohjelmoinnin CFML:ssä. CFC-komponentit erottuvat .cfc-tiedostopäätteellä ja niitä käytetään samantapaisten funktioiden kokoamisen yhteen tiedostoon, mikä selventää koodia. CFC-tiedostossa voi kirjoittaa normaalisti CFML-koodia ja sen lisäksi käytössä on muutamia vain CFC-komponenteille käytössä olevia erikoismerkintöjä (9, s.1).

### 3.3.2 Käyttö

CFC-komponenttien tietoihin ja funktioihin käsiksi pääsyyn tarvitaan ilmentymä CFC-komponentista. Tapoja on muutamia erilaisia:

- Luodaan ilmentymä CFC-komponentista ja tätä kautta päästään käsiksi tietoihin ja funktioihin.

- On mahdollista käyttää metodi-kutsua <cfinvoke> luomatta ilmentymää CFC-komponentista, joka mahdollistaa pääsyn tietoihin sekä funktioihin. Tällöin ColdFusion luo ilmentymän komponentista vain siksi aikaa, kun metodin kutsu kestää.

- Ilmentymiä voi luoda myös pysyvästi. Jos ilmentymä luodaan pysyvään näkyvyysalueeseen, sitä voidaan käyttää esimerkiksi istunnon ajan tai pysyvästi koko sovelluksen ajan. (10, s. 1)

Tietokantasovelluksessa käytän viimeistä tapaa ja komponentit luodaan sovellukselle yhteiseksi.

## 3.4 Funktiot

Funktiot ovat ohjelmassa olevia koodinpätkiä, jotka voivat muuttaa ohjelman ja datan tilaa tai muokata funktiolle annettuja ja sen tuntemia tietoja. Yleinen funktio on matemaattinen, joka tekee laskennan sille annetuista arvoista ja palauttaa vastauksen, mutta ei muuta ohjelmassa itsessään mitään. Oliopohjaisessa ohjelmoinnissa funktioita usein kutsutaan metodeiksi. Olion metodeilla voidaan muokata tai pyytää oliolta dataa tai sen tilaa (26, s. 1).

Kun ohjelma jaetaan funktioihin, voidaan pienempiä kokonaisuuksia kirjoittaa nopeasti. Päällimmäisenä syynä funktioiden käyttöön on niiden uudelleenkäytettävyys. Tällä tarkoitetaan sitä, että samaa funktiota voidaan käyttää useassa kohdassa ohjelmaa, jolloin samaa koodia ei tarvitse uudelleenkirjoittaa, jolloin kehitys nopeutuu. Jos koodissa ei käytetä funktioita, joudutaan testaus suorittamaan koko ohjelmalle. Käytettäessä funktioita voidaan nopeuttaa testausta testaamalla funktiot erikseen (25, s. 3).

Funktioita CFML:ssä voidaan joko luoda itse tai käyttää jo kielessä valmiina olevia funktioita.

```
<cffunction
    name = "method name"
    access = "method access"
    description = "function description"
    displayName = "name"
    hint = "hint text"
    output = "yes|no"
    returnFormat = "not specified|JSON|plain|WDDX"
    returnType = "data type"
    roles = "securityRoles"
    secureJSON = "yes|no"
    verifyClient = "no|yes">
```

**Kuva 3. Funktion luonti (11, s. 1)**

Kuvassa 3 näkyy syntaksi funktion luonnille. Selitetään taulukossa lyhyesti, mitä merkinnät tarkoittavat:

**Taulukko 2. Funktion luonnin attribuutit (11, s. 1)**

| Attribuutti | Lyhyt selitys |
|---|---|
| name | Funktion nimi |
| access | Määrää, mistä funktiota voidaan kutsua |
| description | Lyhyt selitys funktiolle |
| displayName | CFC-metodeille, näkyy tarkastellessa CFC-komponentin tietoja |
| hint | CFC-metodeille, näkyy tarkastellessa CFC-komponentin tietoja |
| output | Voiko funktio tuottaa HTML-koodia |
| returnFormat | Palautettavan arvon muoto |
| returnType | Palautettavan arvon tyyppi |
| roles | Määrää, ketkä voivat käyttää funktiota käyttäjätasolla |
| secureJSON | Käytettäessä asettaa etuliitteet JSON-arvojen eteen |
| verifyClient | Käytettäessä vaatii valtuutuksen kutsuttaessa |

SecureJSON ja verifyClient ovat Ajax-turvallisuuteen liittyviä asetuksia. SecureJSON asettaa JSON-datan eteen merkkijonon, joka estää sivustolle tehtävät hyökkäykset JSON:n kautta (32, s. 1). VerifyClient vaatii käyttäjältä tietyn merkkijonon funktioiden käyttöön, jolloin funktioita ei voi kutsua ohjelman ulkopuolelta, vaan käyttäjän on oltava sivustolla käyttääkseen funktiota (33, s. 1).

Seuraavaksi esimerkki, millainen funktio voi olla.

```
<cfcomponent>
    <cffunction name="getEmp">
         <cfquery
                name="empQuery" datasource="ExampleApps" >
                SELECT FIRSTNAME, LASTNAME, EMAIL
                FROM tblEmployees
         </cfquery>
         <cfreturn empQuery>
    </cffunction>
    <cffunction name="getDept">
        <cfquery name="deptQuery" datasource="ExampleApps" >
         SELECT *
         FROM tblDepartments
         </cfquery>
         <cfreturn deptQuery>
    </cffunction>
</cfcomponent>
```

**Kuva 4. Esimerkki funktio (11, s. 2)**

Kuvassa on CFC-komponentti, jonka sisällä kaksi funktiota, jotka ovat getEmp ja getDept. Molemmat funktiot sisältävät tietokantakyselyn <cfquery> merkinnän sisällä. Ylempi hakee ExampleApps-tietolähteen tblEmployees-taulusta etunimen, sukunimen

ja sähköpostiosoitteen. Lopuksi funktio palauttaa empQueryksi nimetyn hakunsa. Alempi funktio toimii muuten samalla lailla, mutta hakee kaikki tiedot tblDepartments-taulusta ja palauttaa haun.

### 3.5 Sovelluskohtaiset asetukset

Sovelluskohtaiset asetukset ColdFusionissa tarkoittavat asetuksia, jotka ylittävät palvelinpuolella asetetut asetukset ja ovat voimassa vain tietyssä sovelluksessa. Sovelluskohtaisissa asetuksissa voidaan myös asettaa sovellukselle nimi ja määrittää sen funktiot (26, s. 1).

Sovelluskohtaiset asetukset määrätään joko application.cfc- tai application.cfm-tiedostoon, joista ColdFusion priorisoi application.cfc-tiedostoa. Mutta jos tätä ei löydy, käytetään application.cfm-tiedostoa. Tiedostot ladataan aina sivun latauksen yhteydessä ja liitetään pyydetyn sivun alkuun (12, s. 1). Työssä käytetään sovelluskohtaisten asetusten määrittämiseen application.cfc-tiedostoa. Esimerkissä (liite 1) asetetaan sivustolle sovelluskohtaisia asetuksia.

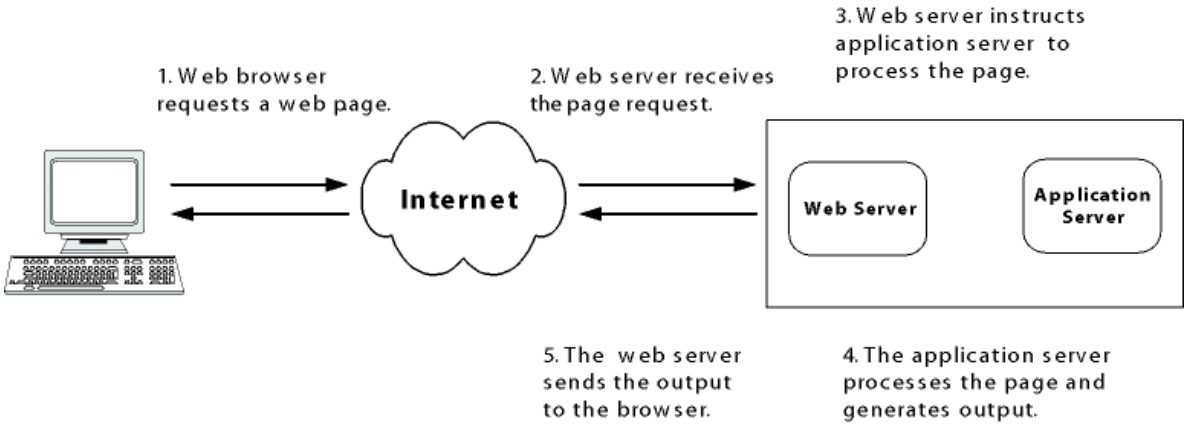
## 4 Kehitysympäristö

### 4.1 Yleisesti

Kehitysympäristön pystytyksessä on aluksi valittava työkalut, joilla toimitaan, koska ColdFusion mahdollistaa useiden sovelluspalvelimien ja tietokantojen käytön on valittava käytettävä ympäristö.

Aluksi on valittava web-palvelin, jonka päälle ColdFusion pystytetään. Adobe ColdFusion sisältää sisäänrakennetun web-palvelimen, mutta sen käyttöä suositellaan ainoastaan, jos työasemalle ei ole asennettu ulkoista web-palvelinta. Jos ulkoisella web-palvelimella on vielä vanha versio ColdFusionsta ja siihen päivitetään uutta, tällöin on mahdollista ajaa kahta eri versiota samalla työasemalla (18, s. 1).

Työssä asennetaan kehitysympäristö alusta alkaen, joten valitaan ulkoinen web-palvelin. Web-palvelin on käyttäjän ja sovelluspalvelimen (ColdFusion) välissä, ulkoinen data, kuten tietokannat ovat yhteydessä sovelluspalvelimeen.




**Kuva 5. Web-palvelimen ja web-sovelluspalvelimen toiminta (19, s. 1)**

Kuvassa 5 näkyy ympäristön toiminta, jossa on käytössä web-palvelin ja sovelluspalvelin. Kun käyttäjä pyytää sivustoa kirjoittamalla sen osoitteen selaimeen, lähetetään web-palvelimelle pyyntö. Jos kirjoitetun osoitteen pääte viittaa yksinkertaiseen web-sivustoon, esimerkiksi HTM- tai HTML- sivustoon, käsitellään pyyntö ja lähetetään sivustokäyttäjälle. Jos taas web-palvelin huomaa, että sovelluspalvelimen täytyy käsitellä pyyntö (CFM; CFML- tai CFC-päätteinen ColdFusion pyyntö), lähettää web-palvelin sovelluspalvelimelle pyynnön käsittelystä. Sovelluspalvelin käsittelee pyynnön, lähettää käsittelyn tuloksen web-palvelimelle, joka välittää sen käyttäjälle. Tämä mahdollistaa dynaamisen sivuston luonnin (19, s. 1).

Lopuksi on vielä valittava tietokanta, joka välittää tietoa sovelluspalvelimelle, jotta käytettävä tieto ympäristössä saadaan kulkemaan.

## 4.2 Asennus

Seuraavaksi aloitetaan asennus. Ympäristöksi olen valinnut seuraavanlaisen yhdistelmän:

- Apache Web Server 2.2.21

- ColdFusion 10

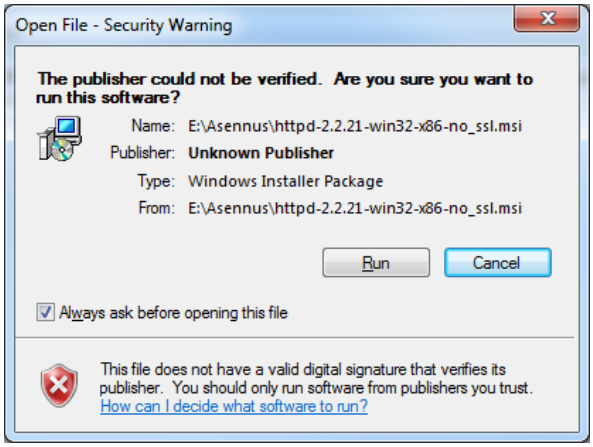
- Microsoft SQL Server 2008 R2.

Tämä on vain yksi mahdollinen ympäristö, mutta olen valinnut tämän, koska aiemman kokemukseni perusteella, tämä ympäristö on yhteensopiva.

Asennus aloitetaan Apache Web Serverin asennuksella, koska tämän jälkeen voidaan valita ColdFusion asennuksessa valmiiksi asennettu ulkoinen web-palvelin, jolloin se konfiguroidaan automaattisesti, asennuksen yhteydessä. Tämän jälkeen asennetaan ColdFusion ja lopuksi Microsoft SQL server.

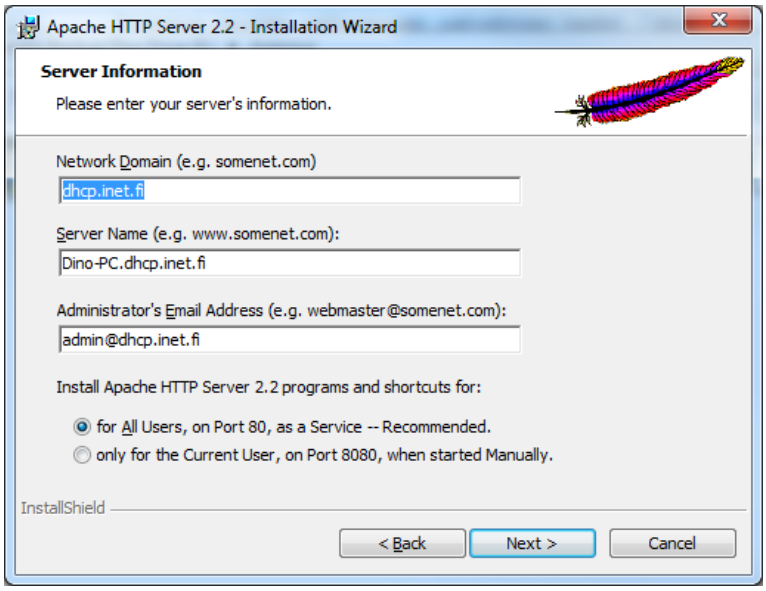
### 4.2.1 Apache Web Server

Aloitetaan asennus lataamalla oikea versio Apachen www-sivuilta ja käynnistetään asennus.



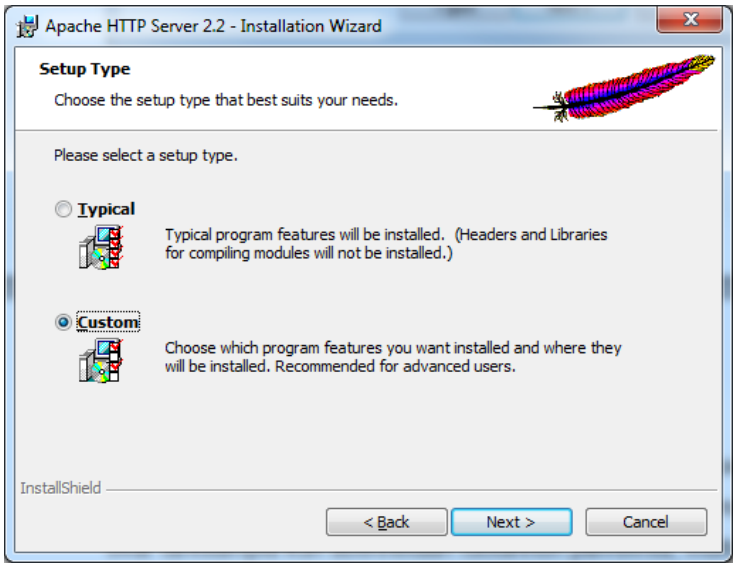

**Kuva 6. Apachen asennus**

Jatketaan asennusta kunnes päästään ruutuun, jossa asetetaan palvelimen asetuksia.



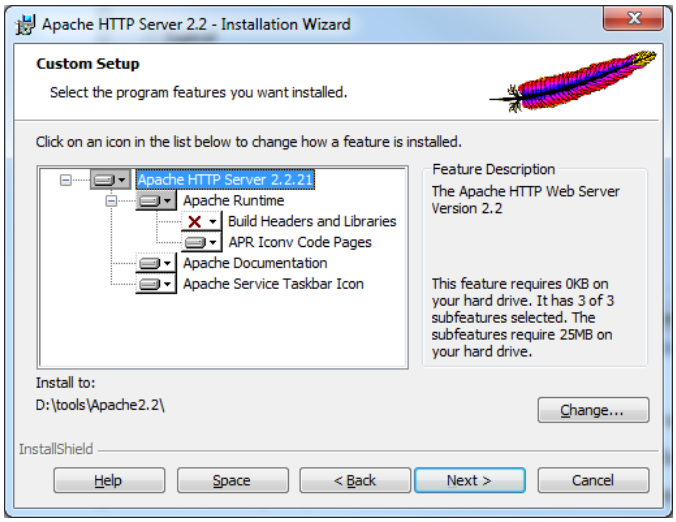

Kuva 7. Apachen palvelintiedot

Asennus pyytää valitsemaan palvelimen tiedot. Tässä voidaan jättää oletusasetukset Network Domain- ja Server Name -kohtaan paikallisessa ympäristössä. Nämä asetukset ovat tärkeämpiä, kun asennetaan tuotannon palvelinta, mutta tässä työssä ei käydä läpi tuotannon palvelinten asetuksia. Alimmat vaihtoehdot portin määrittämiseen ovat kuitenkin tärkeitä. Jos valitaan ensimmäinen vaihtoehto, jota suositellaan asennuksessa, on otettava huomioon, että portti 80 on käytössä useilla sovelluksilla, kuten Skypellä tai Internet Information Servicellä. On tärkeää, että nämä sovellukset tai palvelut eivät ole asennuksen aikana päällä, jotta Apache pystytään asentamaan ja käynnistämään (21, s. 2).



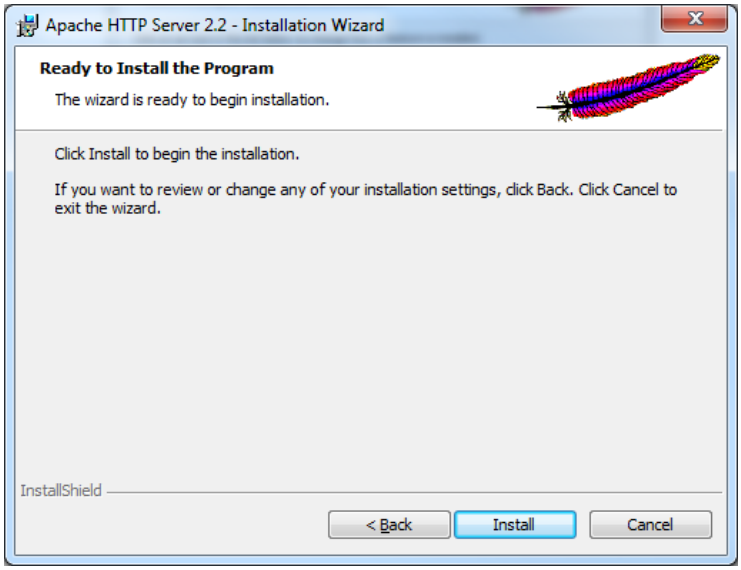

Kuva 8. Asennustyyppi

On tärkeää valita Custom-asennus, jotta päästään vaihtamaan asennuskansio. Asennuskansion alta löytyy htdocs-kansio, johon sovellukset sijoitetaan. Jos tämä polku on erittäin pitkä, on sinne turhauttava viitata selaimessa.




**Kuva 9. Kustomoitu asennus**

Oletuksena valitut paketit kelpaavat ja nyt asennukselle on valittu parempi sijainti.

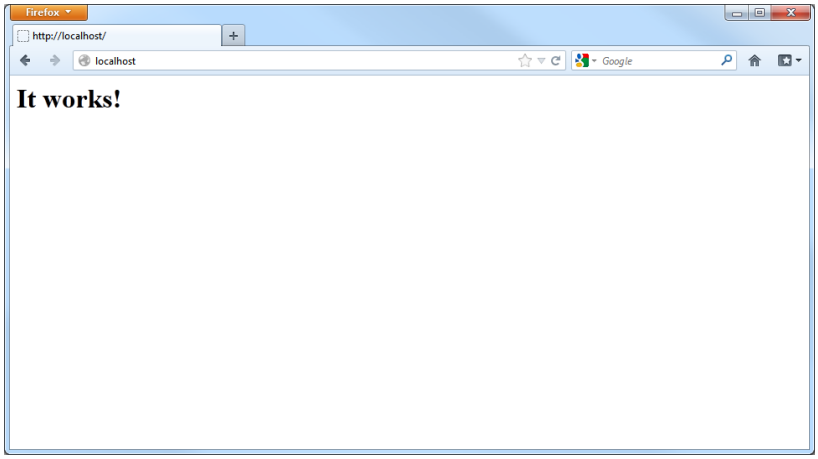



**Kuva 10. Valmis asentamaan**

Nyt Apache on valmis asennettavaksi. Valitaan Install, jos kaikki vaiheet ovat kunnossa.

Asennuksen jälkeen varmistetaan, toimiiko palvelin. Apachessa on valmiina testisivusto, joka löytyy millä tahansa selaimella http://localhost/-osoitteesta. Jos

kuitenkin asennuksessa olisi valittu esimerkiksi portti 8080 vakio-http-portti 80 sijaan, kirjoitettaisiin osoite http://localhost:8080.

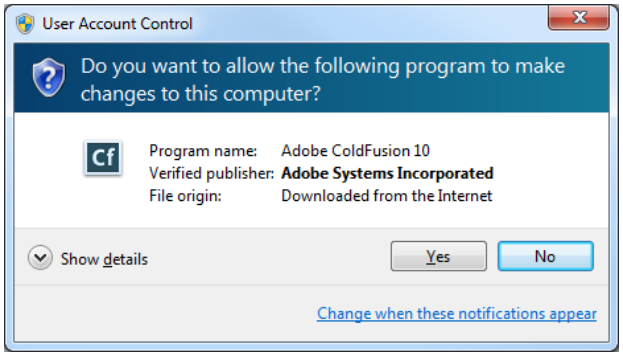



Kuva 11. Apachen testisivu

Testisivusto näyttää toimivan, joten Apache Web server on nyt asennettu, Voidaan siirtyä ColdFusion asennukseen.

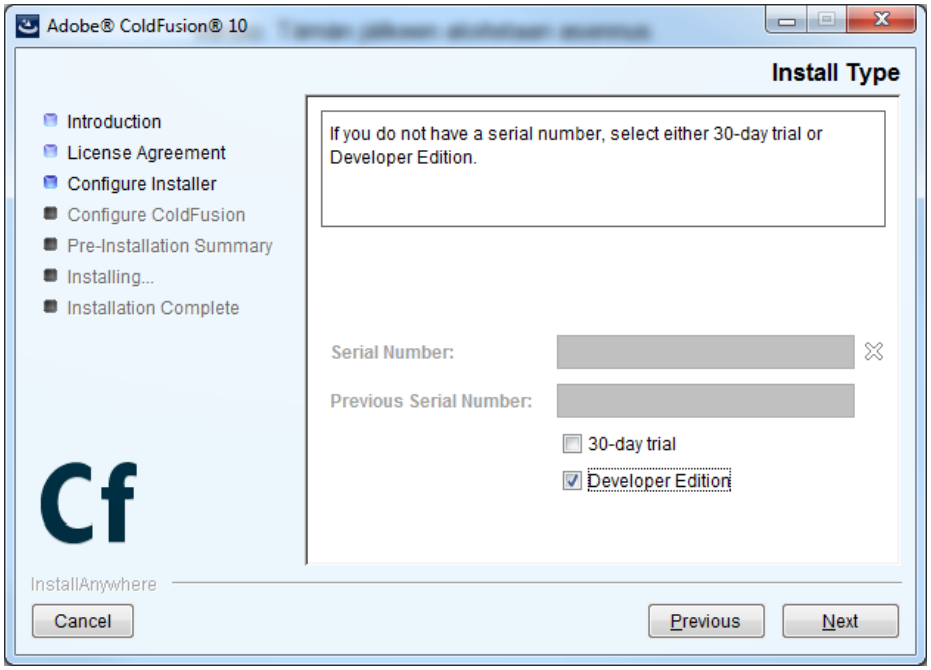
### 4.2.2 ColdFusion

Aloitetaan asennus hakemalla asennuspaketti Adoben sivustolta ja valitsemalla oikea versio. Tämän jälkeen aloitetaan asennus



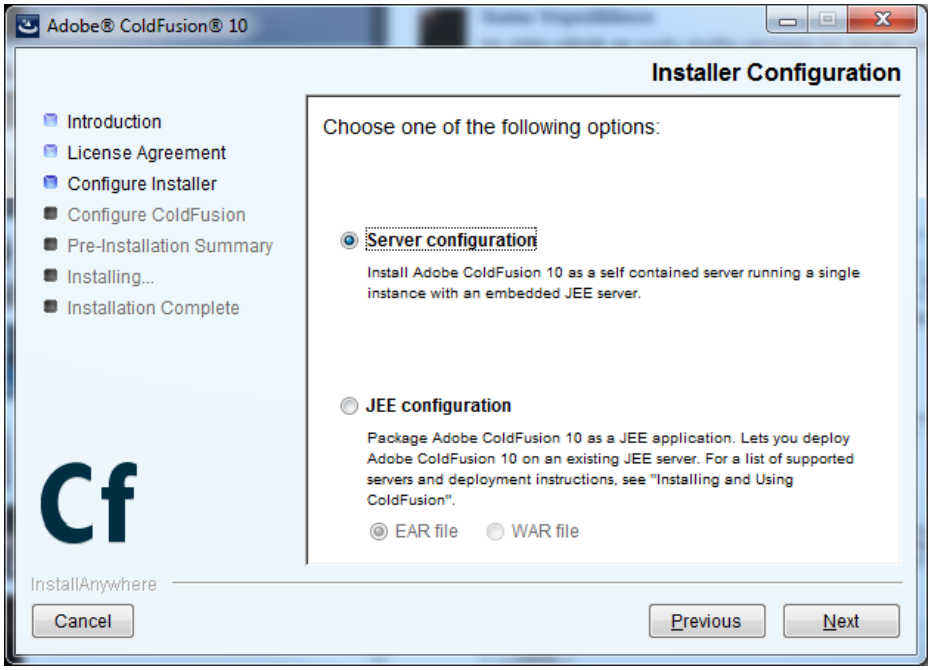

Kuva 12. ColdFusionin asennus

Voidaan jatkaa asennusta, kunnes päästään kohtaan, jossa valitaan asennuksen tyyppi.




Kuva 13. ColdFusion asennustyyppi

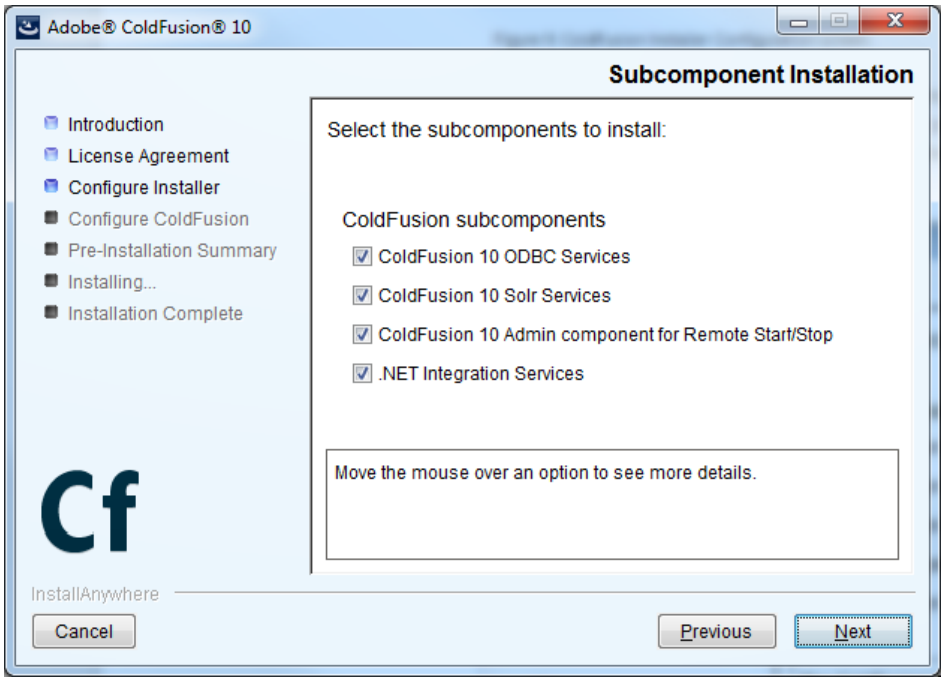
Tässä on erittäin tärkeää valita Developer Edition, joka on ilmainen paikalliseen kehitykseen asennettava versio. Tuotannossa tässä asetettaisiin sarjanumero asennuspaketille.




Kuva 14. ColdFusion asennuksen konfigurointi

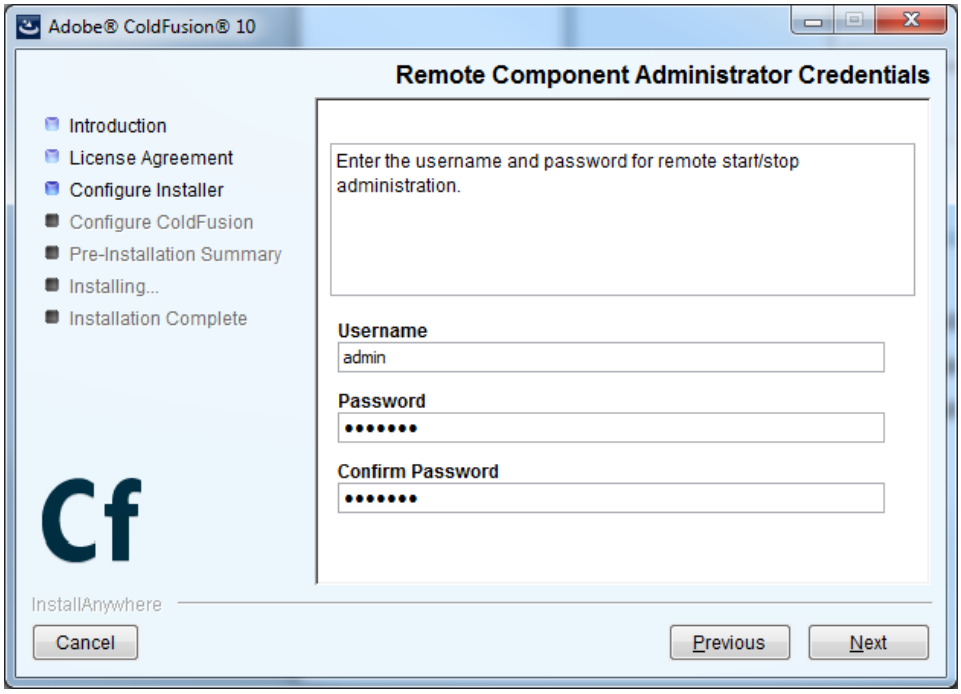
Seuraavaksi jatketaan oletuksena valitulla Server configurationin valinnalla. Tämä kelpaa paikallisen kehityksen ja tämän ympäristön toimintaan, koska käytössä ei ole

erillistä JEE-palvelinta. JEE (Java Enterprise Edition) mahdollistaa suurien java-projektien luonnin skaalautuvasti ja luotettavasti (31, s. 1), mutta tässä projektissa sitä ei käytetä.



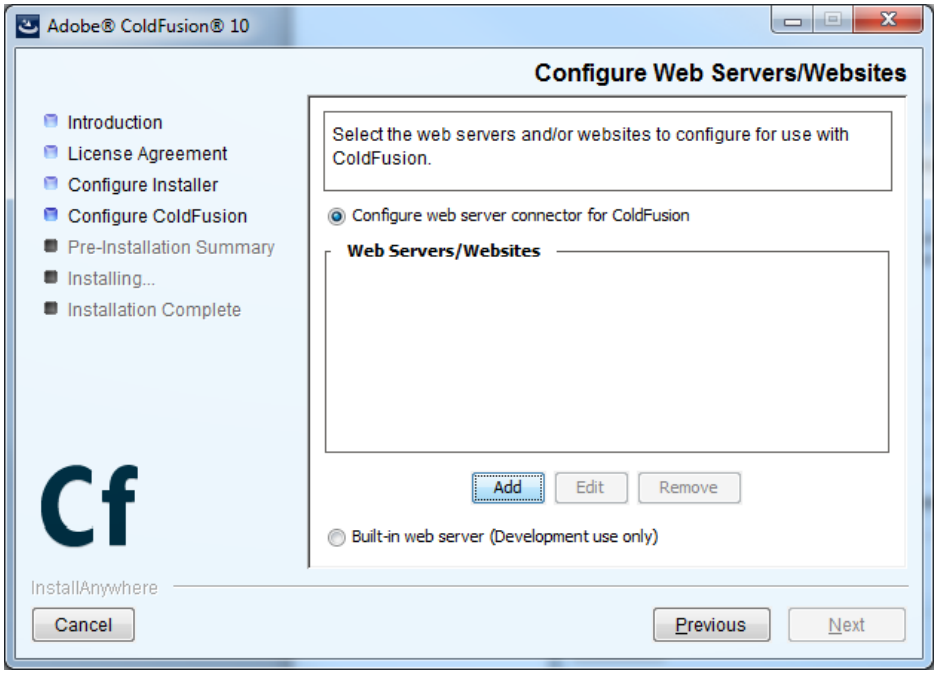

**Kuva 15. Komponenttien valinta**

Paikallisessa ympäristössä voidaan jättää kaikki toiminnot päälle. Tuotannossa kuitenkin olisi tärkeää, ettei ylimääräistä asennettaisi, joten tuotannossa tarkasteltaisiin tarkemmin, tarvitaanko kaikkia mahdollisia komponentteja. Seuraavaksi jatketaan oletusasetuksilla, kunnes päästään asettamaan salasana.



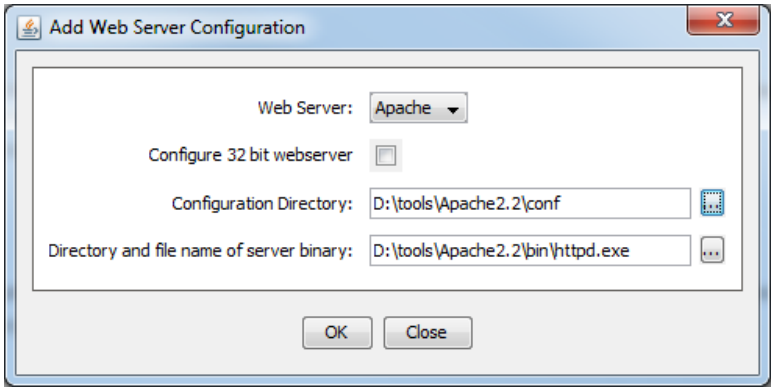

Kuva 16. ColdFusion admin -salasana

Tätä salasanaa käytetään ColdFusion admin -sivustolla. Jatketaan asennusta, kunnes päästään valitsemaan web-palvelin. Apache Web Server on asennettu tätä varten jo aiemmin, jotta se voidaan valita asennuksessa.



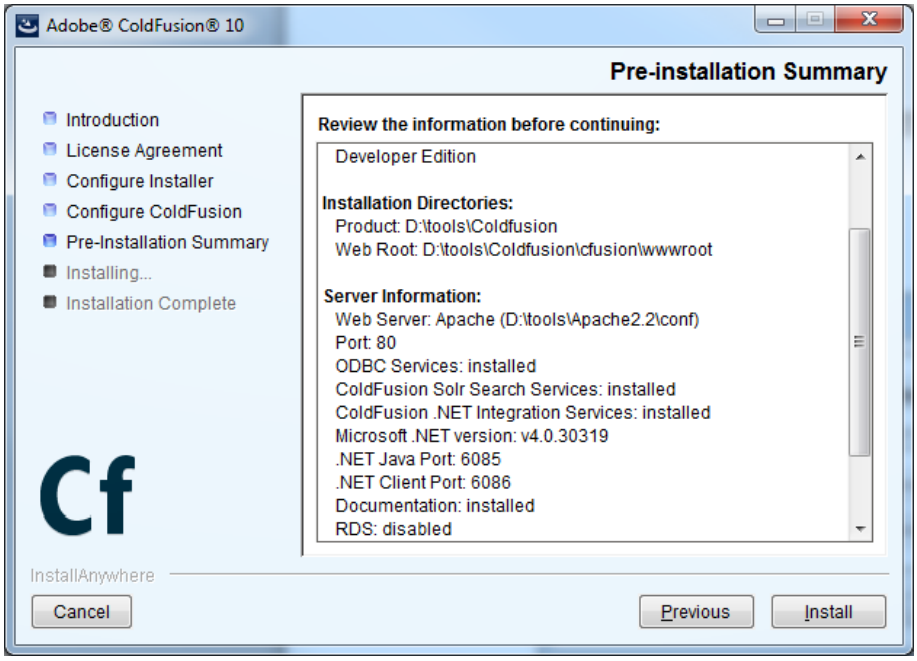

Kuva 17. ColdFusion liittäminen Apacheen

Valitaan add, johon asetetaan tiedot asentamastamme Apache Web Serveristä.



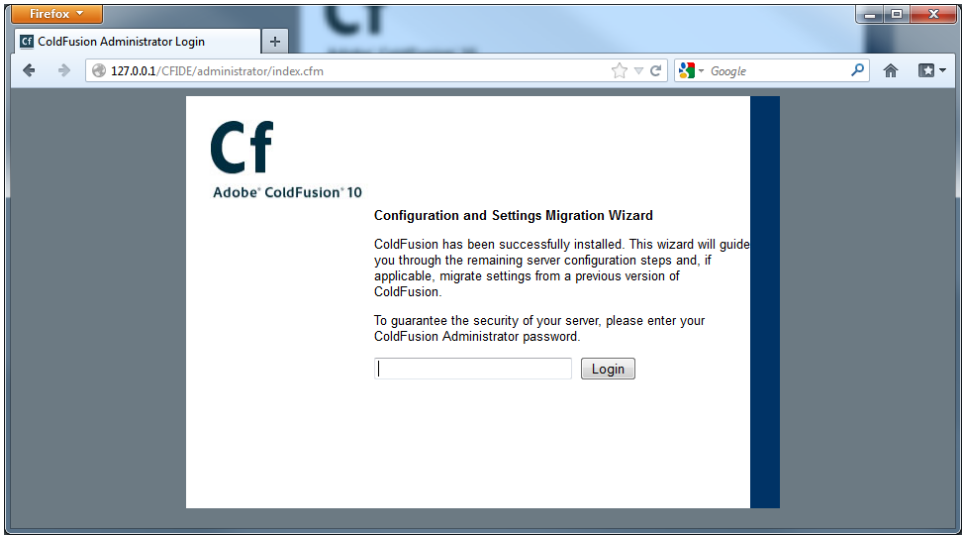

Kuva 18. Apache tiedot

Valitaan Web Server -valikosta Apache ja asetetaan sen tiedot. Nämä polut löytyvät Apachen kansiosta. Lisätään Apache painamalla OK ja jatketaan asennusta oletusasetuksin. Tarkistetaan lopuksi, että asennus näyttää oikealta.



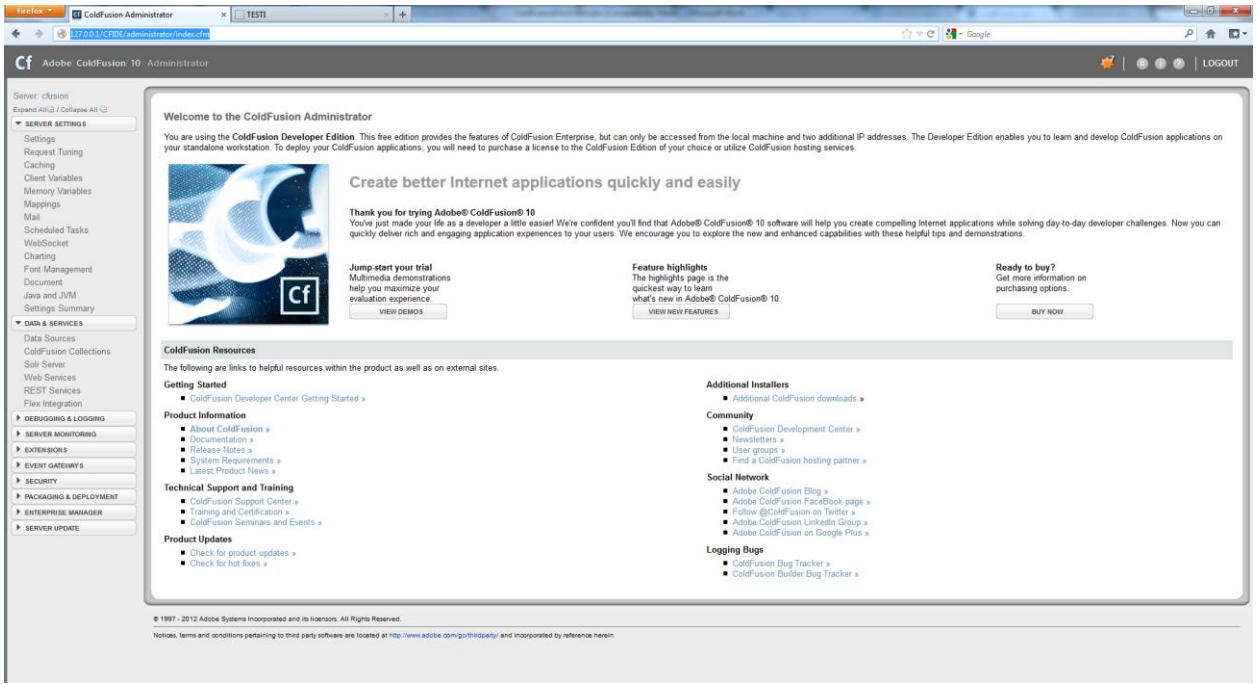

Kuva 19. ColdFusion asennuksen -yhteenveto

Seuraavaksi asennus käynnistää automaattisesti konfigurointiruudun, johon kirjaudutaan asennuksessa asetetulla salasanalla.




**Kuva 20. ColdFusion konfigurointi selaimessa**

Kun salasana on syötetty, konfiguroi ColdFusion-asetukset ja avaa admin-näkymän.

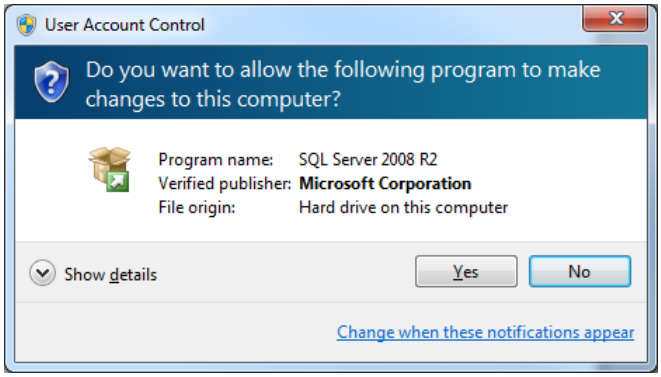



**Kuva 21. ColdFusionin admin-näkymä**

Palataan näkymään, kun Microsoft SQL Server on asennettu, jotta saadaan liitettyä viimeinen osa: tietokanta.

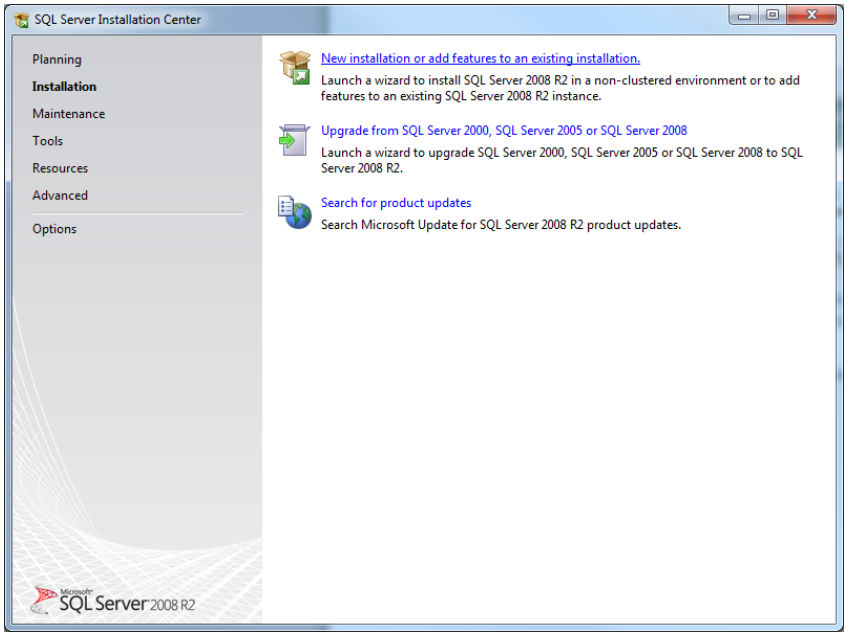
### 4.2.3 Microsoft SQL Server

Aloitetaan SQL-serverin asennus hakemalla asennuspaketti Microsoftin sivuilta ja käynnistetään asennus.



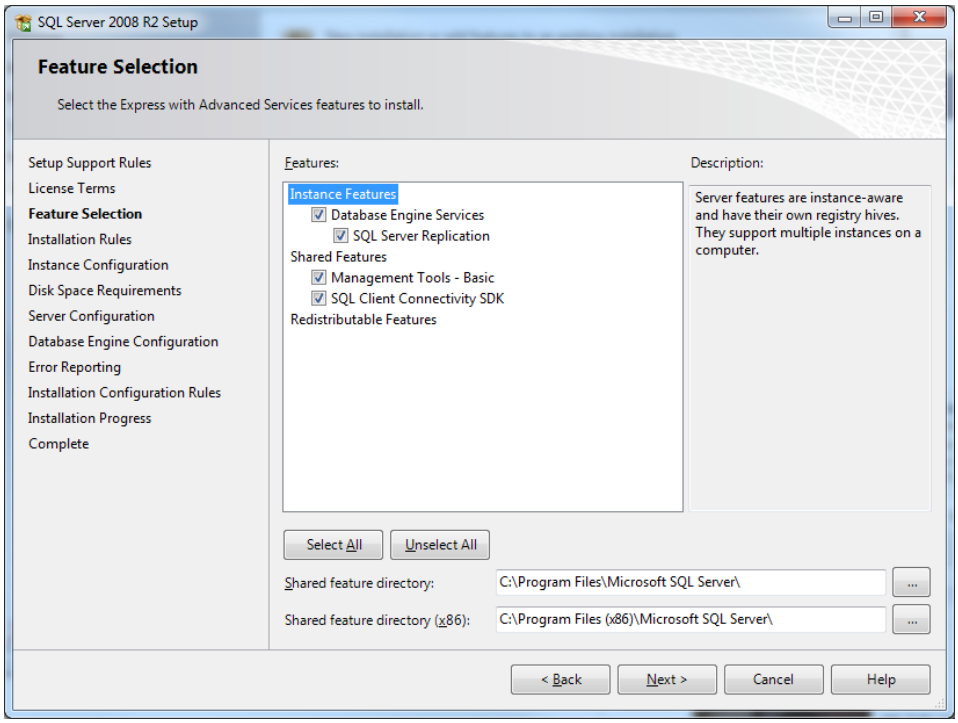

**Kuva 22. Microsoft SQL Server -asennuksen aloitus**

Asennuksen alussa on valittava asennuksen tyyppi.



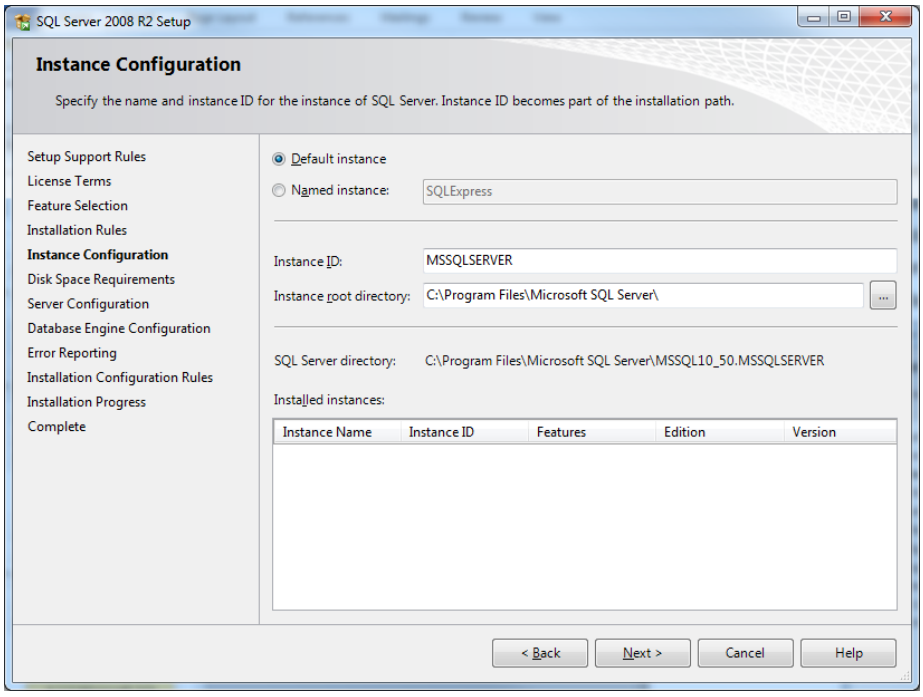

**Kuva 23. Microsoft SQL Server -asennustyyppi**

On tärkeää valita ylin vaihtoehto: uusi asennus, koska asennamme ympäristöä ensimmäistä kertaa, eikä pohjalla ole vanhoja palvelimia. Asennus tarkistaa nyt, onko asennusta varten tehtävä vielä joitakin toimenpiteitä, mutta ruutu ei tule näkyviin jos kaikki on kunnossa. Jatketaan asennusta, kunnes päästään ruutuun, jossa valitaan asennettavat toiminnot.



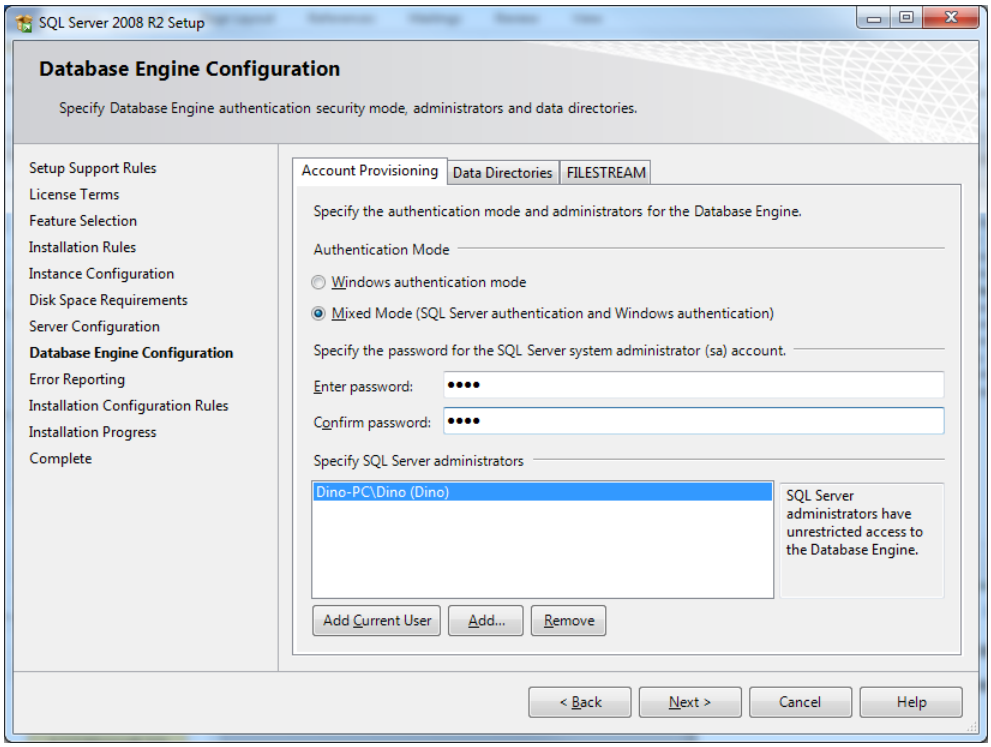

Kuva 24. Microsoft SQL Serverin toiminnot

Valitaan kaikki toiminnot listasta ja sopivat kansiot asennukselle.




Kuva 25. Instanssin valinta

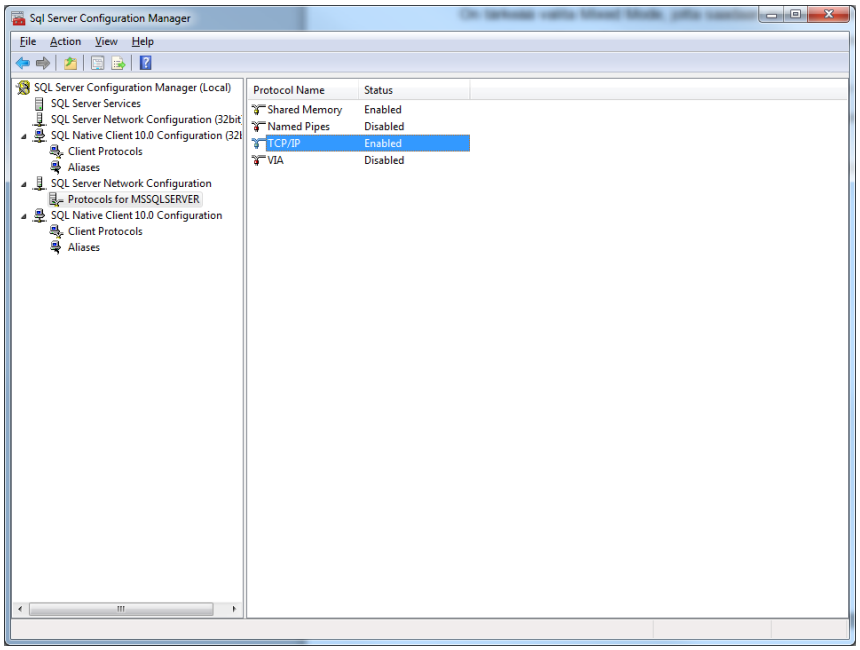
Seuraavaksi valitaan instanssi SQL-palvelimelle. Vaihdetaan oletusasetus Default instanceen. Jatketaan oletusasetuksilla, kunnes päästään ruutuun.




Kuva 26. SQL-palvelimen autentikointi

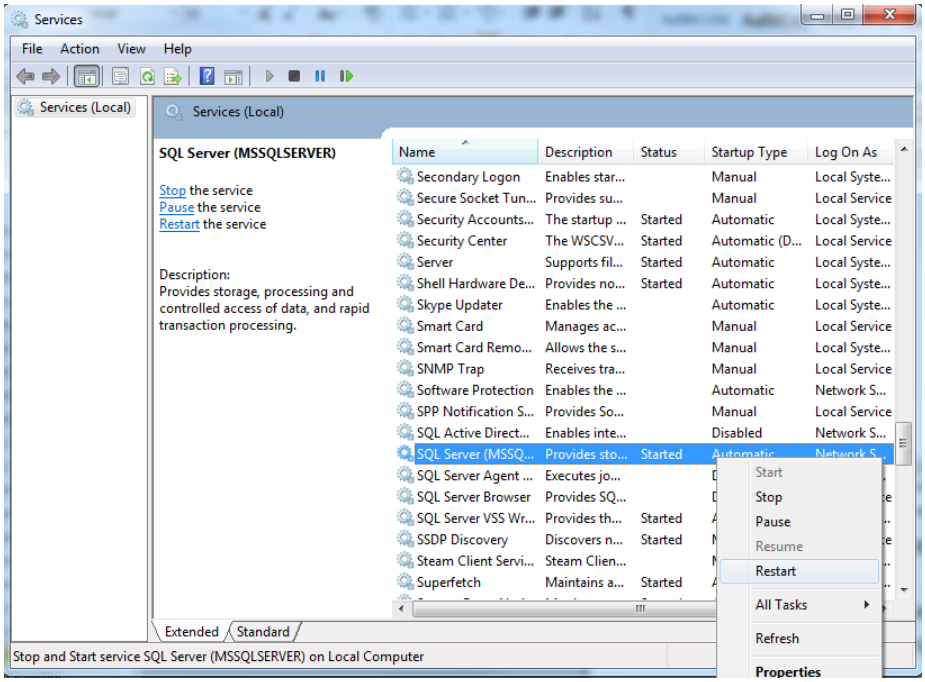
On tärkeää valita Mixed Mode, jotta saadaan asetettua nämä tunnukset ColdFusionille asennuksen jälkeen. Asetetaan salasana ja jatketaan asennus loppuun.

Asentamallemme instanssille täytyy nyt asettaa TCP/IP -protokolla päälle, jotta se saadaan otettua käyttöön. Tämä löytyy Sql Server Configuration Manager -ohjelmasta.



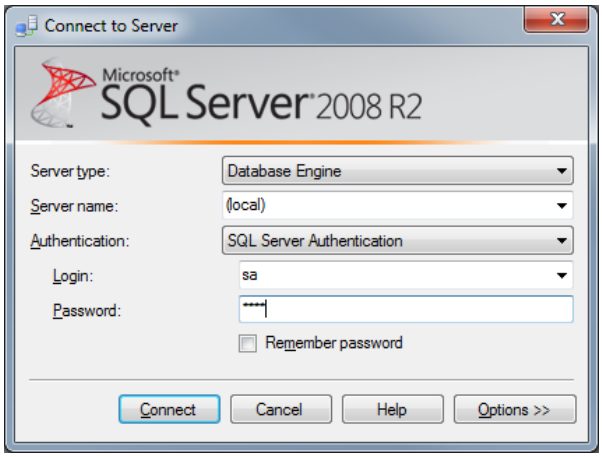
Kuva 27. Sql-palvelimen TCP/IP-protokollan asetus

Asetuksen muuttaminen vaatii palvelun uudelleen käynnistämistä. Käynnistetään käyttöjärjestelmän palveluista instanssi.



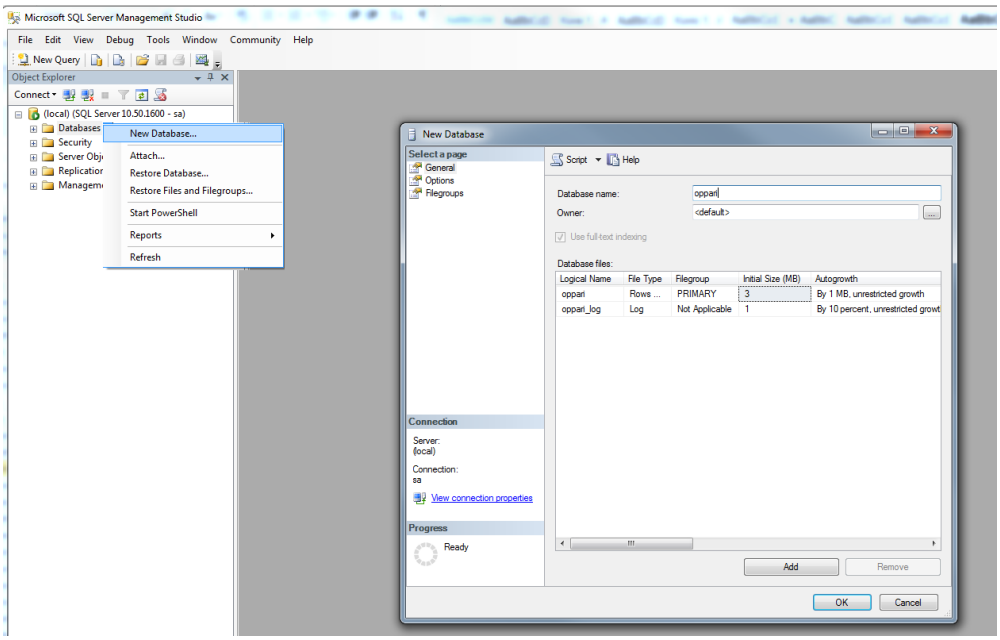

Kuva 28. Palvelun uudestaan käynnistys

Kun TCP/IP-protokolla on asennettu, käynnistetään Microsoft SQL Server Management Studio, johon kirjaudutaan seuraavanlaisilla asetuksilla:



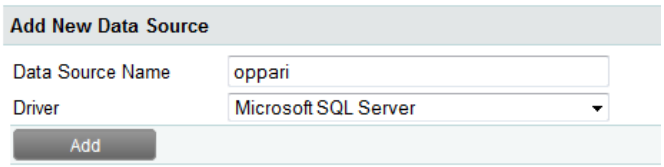

Kuva 29. SQL-palvelinkirjautuminen

Kun on kirjauduttu sisään, luodaan tai tuodaan tietokanta, jota halutaan käyttää. Tässä tapauksessa luomme uuden tietokannan klikkaamalla hiiren oikealla Databases-kohtaa, josta valitaan New Database, nimetään tietokanta ja tallennetaan painamalla OK.



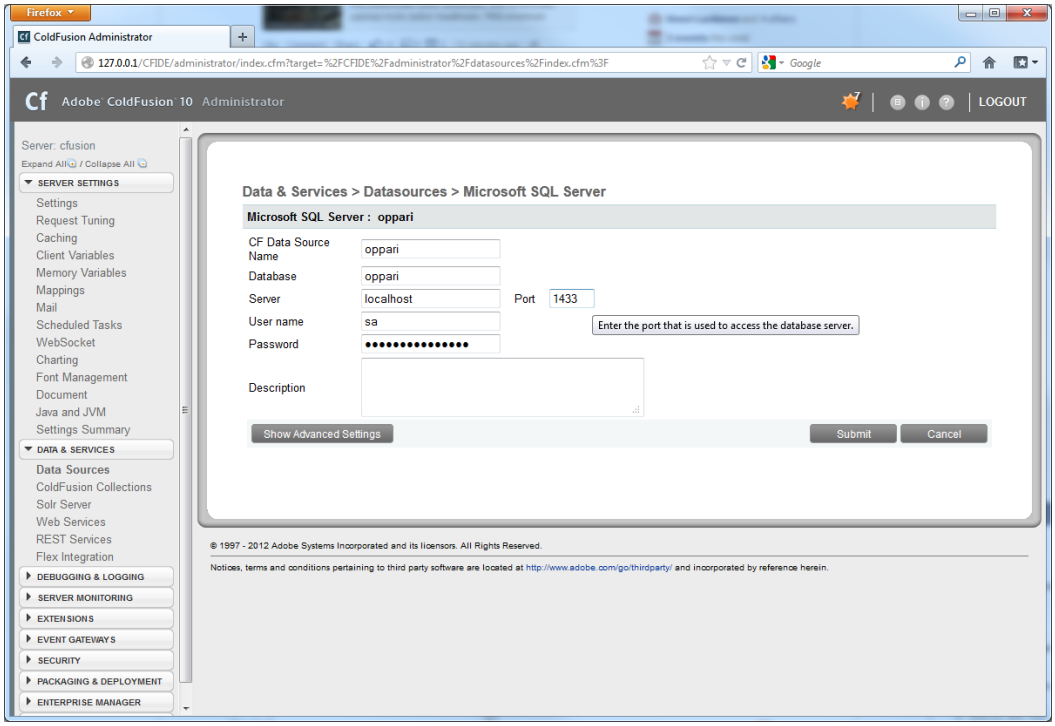

**Kuva 30. Tietokannan luonti**

Lopuksi yhdistetään tietokanta ColdFusioni kanssa admin-sivujen kautta. Valitaan Data Sources -välilehti, josta lisätään uusi tietokanta.



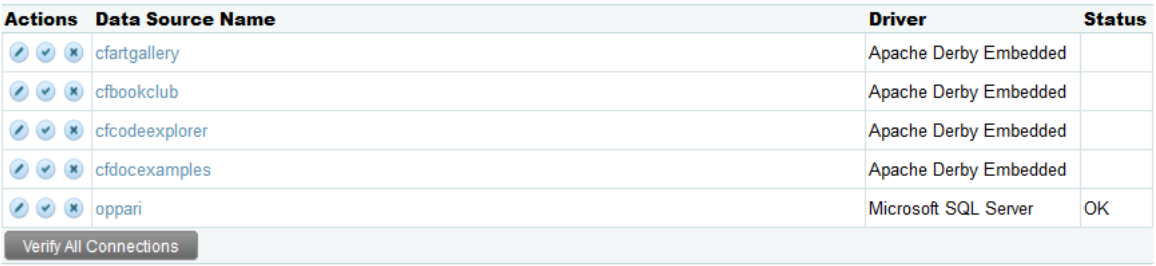

**Kuva 31. Tietokannan liitto ColdFusion-järjestelmään**

Annetaan Data Source Name -kohtaan nimi, jota tahdotaan käyttää koodissa, sekä asetetaan database-kohtaan tietokannan nimi, jonka loimme SQL-palvelimelle. Nyt kun TCP/IP-protokolla on päällä, voimme yhdistää localhost-osoitteen kautta 1433 portista, joka on SQL-palvelimen vakioportti (22, s. 1).




Kuva 32. SQL-palvelimen tiedot ColdFusion adminissa

Käyttäjänimi on vakiona SQL-palvelimella sa ja salasana asennuksessa asetettu. Kun tiedot on täytetty, painetaan Submit. Jos kaikki on kunnossa, näkyy Status kohdassa OK, kuten seuraavassa kuvassa.



| Actions | Data Source Name | Driver | Status |
|---|---|---|---|
| | cfartgallery | Apache Derby Embedded | |
| | cfbookclub | Apache Derby Embedded | |
| | cfcodeexplorer | Apache Derby Embedded | |
| | cfdocexamples | Apache Derby Embedded | |
| | oppari | Microsoft SQL Server | OK |

Verify All Connections

Kuva 33. SQL-palvelimen status

Nyt kaikki on asennettu ja liitetty toisiinsa sovelluksen luontia varten.

# 5 Tietokantasovellus

## 5.1 Vaatimukset

### 5.1.1 Käyttötarkoitus

Nykyisin lähes jokainen web-sivusto on tietokantasovellus. Käyttäjille näkymätön taso työskentelee jatkuvasti muokkaamalla tietokantaa, eikä käyttäjä ole välttämättä tietoinen sen toiminnasta lainkaan. Vaikka julkisessa web-maailmassa käytetäänkin tietokantoja usein, on niiden käyttö suurempaa yritysten käyttämissä web-työkaluissa. Tällaiset työkalut ovat räätälöityjä tuotteita yritystä varten ja tarjoavat yritykselle työkaluja web-pohjaisesti. Asennettavista ohjelmista pyritään eroon, ja kaikki tieto koetetaan siirtää verkkoon, jolloin käyttäjiä ei enää sido asennetut ohjelmistot, vaan käyttäjä pääsee käsiksi työkaluihin mistä tahansa. Yleinen luonnehdinta web-työkalusta on jonkinlainen järjestelmä, joka käsittelee tietoa tietokannasta tai tietokannoista. Järjestelmillä voidaan esimerkiksi hallita käyttäjiä, kirjata työtunteja, tehdä verkkojulkaisuja tai kirjata tuloksia. Näitä järjestelmiä yritysten työntekijät käyttävät päivittäisessä työssään, ja ne käsittelevät suuria määriä tietoa. Voidaan siis sanoa, että tällaisten järjestelmien täytyy toimia luotettavasti sekä nopeasti, jotta järjestelmien käyttö on perusteltua.

Opinnäytetyössä luodaan järjestelmä, jolla voidaan palkita työntekijöitä hyvästä työsuorituksesta. Palkintojen muoto on tulostettava lahjakortti. Järjestelmän tulee olla helppokäyttöinen ja yksinkertainen, jotta sen käyttöön ei tarvita erityistä tietotekniikan osaamista. Järjestelmässä voidaan lisätä uusia henkilöitä, jotka ovat tulleet palkituksi. Kun henkilö on lisätty järjestelmään, voidaan palkinto automaattisesti luoda henkilölle napin painalluksella, eikä järjestelmän käyttäjän tarvitse tietää mitään itse järjestelmän toiminnasta. Käyttäjä näkee käyttöliittymässä vain tarvittavat tiedot luodakseen palkittujen henkilöiden palkinnot ja pystyy toteuttamaan tämän vaivattomasti. Kaikki toiminnot löytyvät yhdeltä sivulta kootusti, jotta käyttö olisi mahdollisimman yksinkertaista.

### 5.1.2 Toiminnot

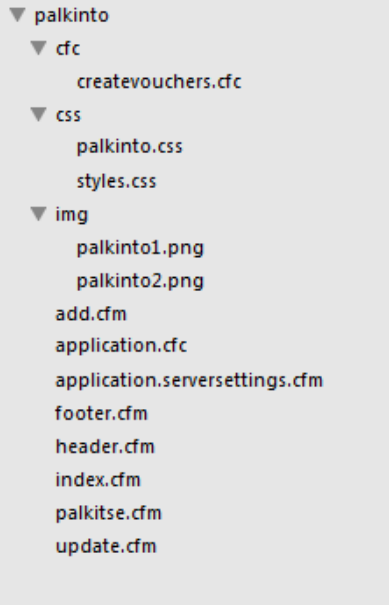
Järjestelmän pääominaisuudet ovat palkitun henkilön lisäys tietokantaan, mahdollisuus muuttaa palkittujen henkilöiden tietoja ja itse palkitseminen, joka nappia painamalla luo

valitulle henkilölle palkinnon ja tulostaa sen ruudulle, josta se voidaan käyttötarkoituksen mukaan esimerkiksi tulostaa tai tallentaa kovalevylle. Kaikki toiminnot löytyvät järjestelmän etusivulta ja henkilön lisäys sekä henkilön päivitystoiminnot vievät käyttäjän takaisin etusivulle, kun toiminto suoritetaan. Palkittaessa henkilö siirrytään kokonaan uuteen sivustoon, joka tulostaa palkinnon.

## 5.2 Ohjelmistoarkkitehtuuri

### 5.2.1 Rakenne

Ohjelmistojen rakenne vaihtelee paljon projektista riippuen, mutta usein projekteissa noudatetaan samankaltaista toimintojen erottelua projektin selkeyttämiseksi. Opinnäytetyössä luodussa järjestelmässäkin toiminnot on eritelty. Järjestelmän rakenne tiedostojen osalta selviää kuvasta 35.




**Kuva 34. Järjestelmän kansiorakenne**

Kansio cfc sisältää cfc-komponentin createvouchers.cfc, joka vastaa kaikista toiminnoista, joita suoritetaan tietokannassa. Tyylitiedostot palkinto.css ja styles.css löytyvät kansiosta css. Itse sivun ulkonäöstä vastaa tiedosto styles.css, jossa määritetään itse sivuston ulkonäkö ja asettelu. Asettelu otetaan käyttöön header.cfm-tiedostossa. Palkinto.css-tiedosto vastaa palkinnon ulkonäöstä ja asettelusta. Sitä

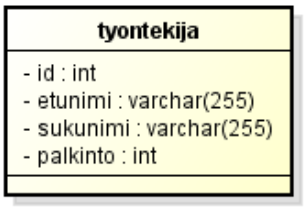
käytetään ainoastaan palkitse.cfm-tiedostossa, jossa luodaan palkinto. Img-kansiosta löytyvät kuvat ovat erilaisten palkintojen pohjia, joiden päälle syötetyt tiedot asetetaan. Sovelluskohtaiset asetukset löytyvät application.cfc-tiedostosta, joka sisällyttää itseensä application.serversettings.cfm-tiedoston. Sisällytettävä tiedosto on eritelty realistisen sivuston kuvainnollistamiseksi. Serversettings-tiedoston tilalla voisi olla jokaisella kehittäjällä oma personalsettings-tiedosto, jolloin käyttäjät voivat itse määrätä kehitysympäristönsä asetukset sovellukselle. Jos tällaisia henkilökohtaisia asetuksia ei löydetä, käytettäisiin serversettings-tiedostoa. Kaikki tiedostossa olevat asetukset voitaisiin suoraan kirjoittaa application.cfc-tiedostoon, mutta kehitystyön helpottamiseksi sen erittely poistaa turhan tiedoston muokkauksen, kun käytössä on aina kehitysympäristön oma asetustiedosto.

Järjestelmän etusivu index.cfm mahdollistaa toiminnallisuuksien käytön, jotka löytyvät add.cfm-, update.cfm- ja palkitse.cfm-tiedostoista. Erittelemällä ylä- ja alatunnisteet sivustojen koodista voidaan uudelleenkäyttää toistuvaa koodia useissa tiedostoissa. Molemmat sekä ylä- että ala-tunnisteet sisällytetään etusivulle sekä add- ja update-sivustoihin.

### 5.2.2 Tietokanta

Tietokantaa luotaessa on mietittävä tarkkaan tarvittavat kentät ja taulut. Kun rakenne on suunniteltu jo valmiiksi ja tiedetään tarvittavat toiminnallisuudet, on tietokannan luonti helppoa, eikä luotua tietokantaa todennäköisesti tarvitse myöhemmin muuttaa. Tietokantaan voidaan aina lisätä uusia tauluja, esimerkiksi silloin kun tietokantaa käyttävään sovellukseen tehdään uusia ominaisuuksia tai sitä muutetaan. On otettava kuitenkin huomioon, että jos tietokanta sisältää paljon tietoa, on tälläisen tietokannan muuttaminen aina hankalaa.

Työssä käytettävässä järjestelmässä on käytössä vain yksi tietokanta nimeltä oppari, jossa on yksi taulutyontekija, jonka rakenne selviää kuvasta 36.



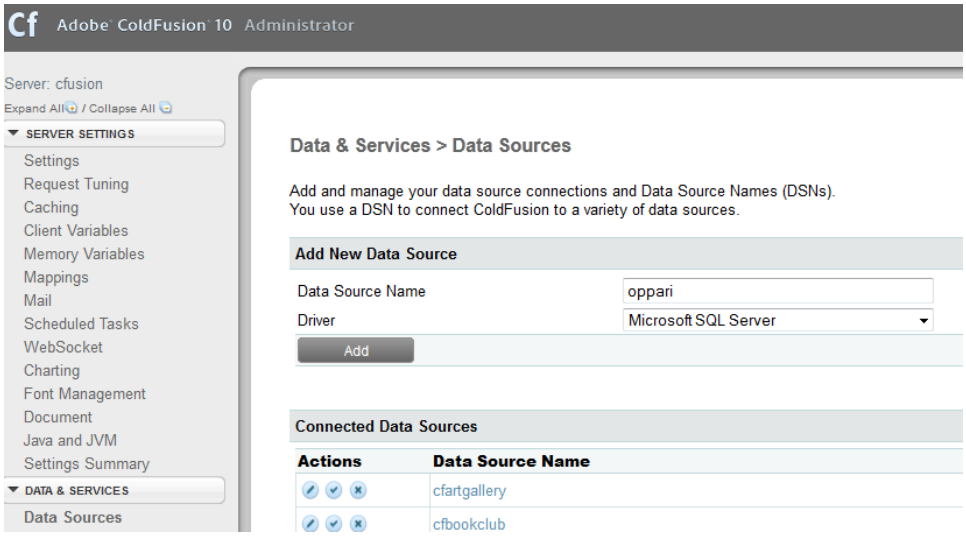

Kuva 35. Taulun rakenne

Taulu voidaan luoda antamalla kuvassa 37 näkyvä komento Microsoft SQL Server Management Studiossa

```
CREATE TABLE tyontekija
(
   id INT IDENTITY(1,1) PRIMARY KEY,
   etunimi VARCHAR(255),
   sukuknimi VARCHAR(255),
   palkinto INT
);
```

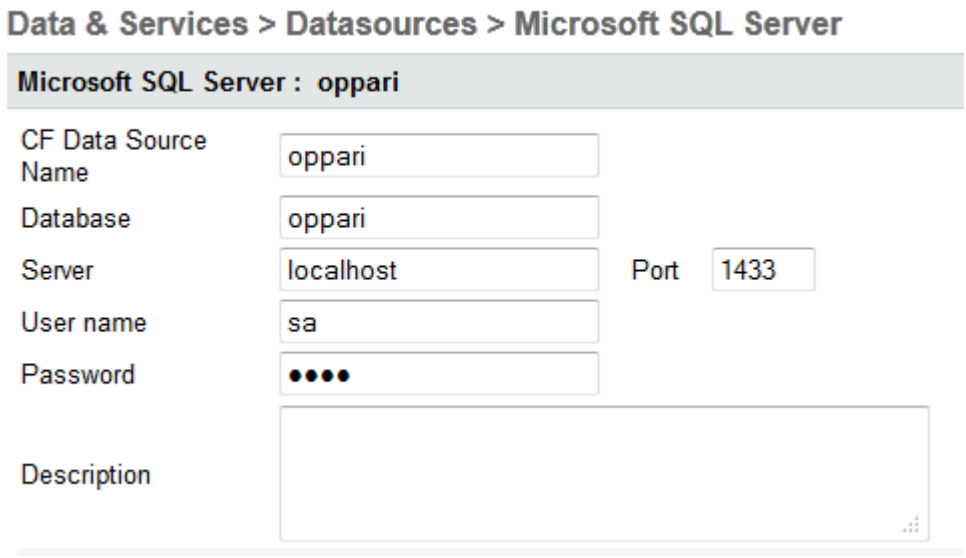
Kuva 36. Taulun luonti

Tyontekija-taulussa id-kenttä on automaattisesti päivittyvä luku, joka on yksilöllinen kullekin tietokantaan lisätyllä työntekijällä. Id-kenttää käytetään tunnisteena työntekijän hallinnoimiseksi. Työntekijällä on tiedot etunimi ja sukunimi, jotka ovat tyyppiä varchar(255) tarkoittaen merkkijonoja, joiden pituus on maksimissaan 255 merkkiä. Viimeisenä tietona työntekijällä on palkinto, joka on luku, jota käytetään järjestelmässä oikean palkinnon tunnistamiseksi. Yksinkertaistamisen vuoksi palkinto-kenttä ei viittaa toiseen tauluun, mutta suuremmassa järjestelmässä tämä tieto normaalisti viittaisi palkinto-taulun id-kenttään, josta voitaisiin hakea palkinnon tiedot. Työssä palkinnot ovat kovakoodattuja palkintoja. Järjestelmä hakee kannasta palkinnon numeron ja tämän tiedon perusteella luo lopullisen palkintosivun. Kun tietokanta on luotu, on se vielä otettava käyttöön ColdFusionin admin-näkymässä. Tietokanta otetaan käyttöön kuvasta 38 näkyvän Data & Services-valikon Data Sources -välilehdeltä




Kuva 37. ColdFusion admin, Data Sources

Välilehdellä asetetaan tietokannan nimi ja ajurityyppi. Tässä tapauksessa oppari ja Microsoft SQL Server. Painettaessa Add-painiketta päästään kuvassa 39 näkyvään ruutuun, jossa aluksi määritellään CF Data Source Name. Tähän viitataan koodissa, kun tahdotaan pääsy tietokantaan. Seuraava kenttä Database on Microsoft SQL Serverin puolella määritetty tietokannan nimi. Lopuksi määritetään osoite, joka paikallisessa ympäristössä on aina localhost. SQL:n vakioportti kelpaa käytettäväksi, jonka jälkeen jää vain käyttäjätiedot, jotka asetimme asennuksen yhteydessä Microsoft SQL Serverille.




Kuva 38. Tietokannan liitto ColdFusioniin

Nämä tiedot täytettynä on tietokanta valmis käytettäväksi koodissa.

### 5.2.3 ColdFusion-komponentti

ColdFusion-komponentti sisältää pyynnöt tietokantaan. Kun komponentilla erotellaan sql-pyynnöt cfm-tiedostoista, selkeytyy koodi huomattavasti. Järjestelmän cfc-komponentti createvouchers.cfc sisältää viisi funktiota, jotka hoitavat tietokannan käsittelyn koko järjestelmässä. Funktio init komponentin (liite 2) alussa alustavat komponentin. Alustuksessa määritetään tietokanta, jota käytetään kaikissa funktioissa ja joka antaa CFC-komponentin järjestelmän käyttöön.

Tietokannasta vain tietoa hakevat funktiot getPerson- ja getPersons-nimiensä mukaisesti hakevat henkilön tai henkilöt tietokannasta. Funktiossa getPerson-haku tapahtuu parametrinä annetun id:n mukaan. Kun id välitetään funktiolle, palauttaa funktio tietokannasta id:tä vastaavan työntekijän funktion kutsujalle. Kaikki työntekijät saadaan haettua getPersons-funktion avulla, joka ei tarvitse parametrejä käyttöönsä, vaan hakee kutsuttaessa kaikki työntekijät tietokannasta ja palauttaa listan, johon kaikki työntekijät on lisätty. Nämä funktiot ovat hyvin yksinkertaisia, sillä tietokantaan ei tarvitse tehdä muutoksia funktioita kutsuttaessa.

Tietokantaan muutoksia tekevät funktiot addPerson ja updatePerson ovat hieman monimutkaisempia. AddPerson-funktiossa otetaan vastaan työntekijän tiedot: etunimi, sukunimi ja palkinto. Vastaanotetun tiedon avulla lisätään tietokantaan uusi työntekijä, johon tiedot asetetaan, kuitenkin tarkastaen tietojen tyyppi luonnin yhteydessä, jottei tietokantaan päädy vääränlaista tietoa. UpdatePerson toimii lähes samalla tavalla, mutta addPerson-funktiossa vastaanotettujen tietojen lisäksi sille välitetään id, jonka mukaan tietokannasta etsitään muokattava tietue. Nyt lisäyksen sijaan, taululle muutetaan annetut arvot. Molemmat funktioista palauttavat, joko boolean-arvon success = true jos funktion suoritus onnistui tai success = false, jos taas funktion suoritus epäonnistui.

### 5.2.4 Sovelluskohtaiset asetukset

Sovelluskohtaisissa asetuksissa määritetään järjestelmälle yksilöllisiä asetuksia. Kuvassa 40 nähdään järjestelmän sovellusasetukset.

```
<cfcomponent output="false" >

    <cfset this.name = "palkinto">
    <cfset this.sessionmanagement = true>
    <cfset this.clientManagement = false>
    <cfset this.sessiontimeout = "#createtimespan(0,0,45,0)#">
    <cfset this.applicationtimeout = "#createtimespan(7,0,0,0)#">

    <cffunction name="onApplicationStart" output="false">

    </cffunction>

    <cffunction name="onRequestStart" output="false">
        <cfset request.charset = "UTF-8">
        <cfcontent type = "text/html; charset=#request.charset#">
        <cfset setEncoding("form", request.charset)>
        <cfset setEncoding("url", request.charset)>

        <cfinclude template="application.serversettings.cfm">

        <cfset variables.doSystemInit = true>

        <cfif variables.doSystemInit>
            <cfset systemInit()>
        </cfif>
    </cffunction>

    <cffunction name="systemInit" output="false" returntype="void">
        <cflock scope="application" timeout="20" type="exclusive">
            <cfset application.objCreateVouchers = createobject("component", "#request.cfcMap#/createVouchers").init(dbsource="#request.dbsource#")>
            <cfset application.appIsInited = true>

        </cflock>
    </cffunction>

</cfcomponent>
```

**Kuva 39. Application.cfc-tiedosto**

Tiedoston alussa määritellään yleisimpiä asetuksia, kuten nimi ja sessioiden käyttö. Sessionmanagement on otettu käyttöön, mikä tarkoittaa, että istunnolle voidaan esimerkiksi määrätä aika, jonka jälkeen käyttämättä ollut järjestelmä tyhjentää kyseisen istunnon näkyvyysalueen. Järjestelmässä istunnon aikaraja on 45 minuuttia. Myös application-näkyvyysalueelle on määritetty aika seitsemän sekuntia tyhjennykseen. Jälkimmäisen aika on huomattavasti lyhyempi, sillä tämän näkyvyysalueen toiminnot suoritetaan yleensä heti luotaessa, kun taas istunto voi olla taustalla pitempään, esimerkiksi tietoja haettaessa, jolloin tiedot eivät häviä tauon aikana. Edellä mainittujen alustusten jälkeen on onApplicationStart-funktio, mutta tämä on jätetty ohjelmassa tyhjäksi, sillä ohjelman käynnistyessä ei tarvitse tehdä erikseen toimenpiteitä, tässä kohtaa voisimme suorittaa esimerkiksi tietokannan saatavuuden tarkistuksen (28, s. 1). Kun kutsu sivuston hakemiseksi lähetetään ColdFusionin käsiteltäväksi, ajetaan funktio onRequestStart. Tässä funktiossa on määritelty käytettävä merkistökoodaus UTF-8, jolla saadaan käyttöön suomen kielessä käytettävät erikoismerkit. Seuraavaksi koodiin on liitetty jo aiemmin mainittu application.serversettings.cfm-tiedosto. Kuvassa 41

nähdään esimerkki, millainen koodi voisi olla useamman kehittäjän kehittämässä järjestelmässä.

```
<cftry>
    <cfinclude template="../kehitykseen/palkinto.application.personalsettings.cfm">
    <cfcatch>
        <cfinclude template="application.serversettings.cfm">
    </cfcatch>
</cftry>
```

Kuva 40. Kehitykseen tarkoitetut asetukset

Tällöin, jos kehittäjän omat asetukset löydettäisiin, käytetään niitä, muussa tapauksessa palvelimelle tarkoitettua application.serversettings.cfm-tiedostoa. Liitettävissä asetuksissa asetetaan kannan tai kantojen nimet sekä muuttujat, joilla voidaan aina viitata tiettyihin kansioihin, eivätkä ne ole riippuvaisia siitä, missä kansiossa käytettävässä tiedostossa niitä kutsutaan.

```
1  <cfset request.dbsource = "oppari">
2  <cfset request.rootMap = "/palkinto">
3  <cfset request.cfcMap = "#request.rootMap#/cfc">
4  <cfset request.cssMap = "#request.rootMap#/css">
5  <cfset request.imageUrl = "#request.rootMap#/img">
```

Kuva 41. Application.serversettings

Lopuksi on määritetty systemInit-funktion käyttöönotto. Funktiossa alustetaan cfc-komponentti createVouchers.cfc. Komponentti asetetaan muuttujaan application.objCreateVouchers, jossa sille luonnin yhteydessä määritetään käytettävä tietokanta. Tässä kohtaa alustettaisiin myös mahdolliset muut komponentit ohjelman käyttöön. Lopuksi määritetään, että ohjelma on alustettu ja valmis käytettäväksi.

### 5.2.5 Toiminnot

Sivuston toiminnot (liite 3) käyttäjälle mahdollistavat tiedostot index.cfm, add.cfm, update.cfm ja palkitse.cfm. Etusivulla index.cfm hallitaan koko järjestelmän toimintaa. Se vastaa sekä toimintoihin pääsystä, että niiden käsittelystä palattaessa takaisin.

Henkilön lisäys onnistuu add.cfm-tiedostossa, jossa lomakkeeseen täytetään palkittavan henkilön tiedot. Lähetettäessä lomake palataan index.cfm-tiedostoon. Lomakkeen mukana on määritetty type-kenttä, jonka index.cfm-tiedosto osaa käsitellä. Tunnistaessaan add-tyypin etusivu yrittää lisätä cfc-komponentin kautta taulussa annetut tiedot ja tulostaa sitten tiedon onnistumisesta etusivulle.

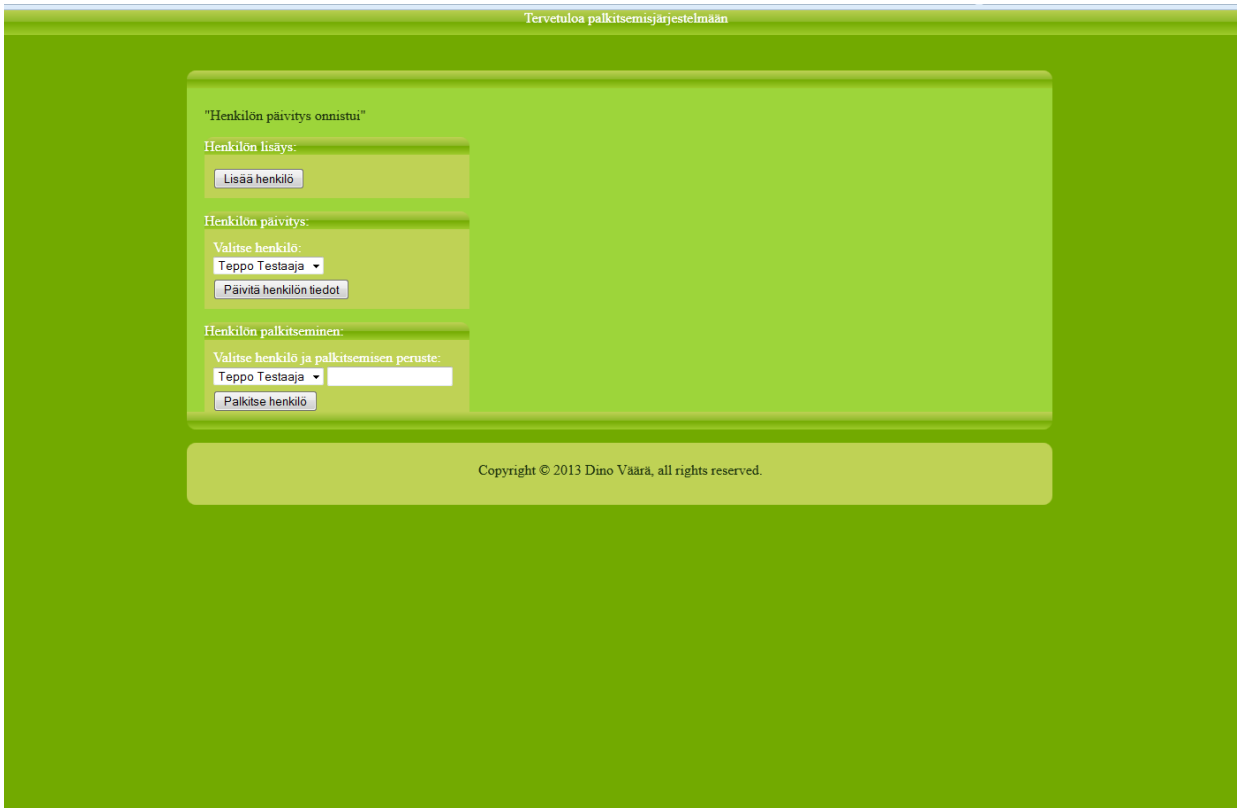
Henkilön muokkaus onnistuu update.cfm-tiedostossa, johon lähetetään etusivulla listasta valittu henkilö. Päivitys-sivulla haetaan cfc-komponentin kautta henkilön tiedot käyttäen etusivulta listasta valitun henkilön id-kenttää. Kun tiedot on täytetty ja lomake lähetetään takaisin etusivulle, käsitellään tiedot samalla tapaa kuin henkilöä lisätessä, mutta tällä kertaa type-kentässä annetulla arvolla update. Etusivu osaa käsitellä cfc-komponentin kautta henkilön tietojen päivityksen ja tulostaa jälleen tiedon onnistumisesta ruudulle.

Viimeisenä toimintona etusivulla nähdään palkitse-toiminto, jolla voidaan luoda itse palkinto. Tässä toiminnossa valitaan listasta palkittava henkilö ja annetaan palkitsemisen peruste. Järjestelmä lähettää tiedot palkitse.cfm-tiedostoon, jossa haetaan jälleen cfc-komponentista henkilö id:n perusteella. Palkitse-sivusto käsittelee henkilön palkitse-kentän ja tämän mukaan luo sivun pohjan ja täyttää pohjan päälle henkilön nimen sekä palkitsemisperusteen.

## 5.3 Käyttöliittymä

Käyttöliittymiin kiinnitetään nykyisin web-maailmassa enemmän huomiota aiempaan verrattuna. Yrityksen sisäiset sivut saavat uutta ulkoasua uusien tekniikoiden tullessa käytettäviksi uusien selaimien kanssa. Käyttöliittymän on tärkeää olla yksinkertainen sen käytön helpottamiseksi, mutta myös tyylikäs käyttömukavuutta parantaakseen.

Järjestelmän yleisilme on erittäin yksinkertainen, mutta värimaailmalla ja asettelulla on koetettu hakea nykyaikaista tyyliä. Kuvassa 43 näkyy kuvakaappaus etusivulta, kun henkilön tietoja on onnistuneesti päivitetty.




**Kuva 42. Etusivun layout**

Järjestelmässä etusivu, lisäys- ja päivitys-sivu käyttävät yhteistä tyylitiedostoa sekä footer- ja header-tiedostoja. Sivun layout pysyy samana koko käytön ajan, jolloin käyttäjälle ei tule yllättäviä muutoksia järjestelmän toimintoja käytettäessä. Lisäämällä sama header- ja footer-tiedosto säästytään koodin uudelleenkirjoittamiselta sekä lukitaan rakenne kiinteäksi. Palkitse-toiminnossa on käytössä oma tyylitiedosto, jossa määritetään palkinnon ulkoasu. Esimerkki palkinnon ulkoasusta voidaan nähdä liitteestä 4, jossa henkilö Teppo Testaaja on palkittu hyvästä työstä. Teppo Testaajan palkinto on tyyppiä 1, joka valittuna tulostaa palkintopohjan 50 €:n arvoiselle laatupalkinnolle.

Molemmissa tyylitiedostoissa käytetään CSS3-tekniikan mahdollistamia gradient-tyylejä, joilla palkkien pyöreä muoto on luotu. Kaikki koodit edellä mainituista tiedostoista voidaan nähdä liitteessä 5.

# 6 Vertailu

Web-palvelinympäristöjen ohjelmistokieliä on useita, mutta PHP on tällä hetkellä niistä suosituin (29, s. 1), joten vertaillaan ColdFusion-kieltä PHP-kieleen, jotta käsittäisimme miksi näin on.

Molempia kieliä käytetään jatkuvasti web-ohjelmoinnissa, mutta ColdFusionin käyttö on huomattavasti vähäisempää. Tämän asian pohjimmaisena syynä on varmastikin raha, koska ColdFusionin lisenssit ovat kalliita ja PHP:n halvempia. PHP:n suosion takia sen suosio kasvaa jatkuvasti. Koska PHP:ta kehittää niin suuri joukko, on selkeästi helpompi löytää valmiita ratkaisuja ongelmiin ja tietynlaisiin tarpeisiin. Kun taas ColdFusionin vähäisen suosion takia on kehittäjien osattava itse luoda suurin osa toiminnallisuuksista (30, s. 1). Vaikka ColdFusionista on nykyisin saatavilla vapaan lähdekoodin versioita, ovat nämä vielä niin tuoreita, että PHP, joka on aina ollut vapaan lähdekoodin kieli, on kasvattanut sillä suosiotaan. Lisenssien hinnoittelun myötä PHP:lla on myös suuri määrä palveluntarjoajia PHP-palvelimelle (31, s. 1).

PHP jakaa käytettäessä koodin pienempiin osioihin ja kääntää aina uudestaan käytettävät koodit. Tämä eroaa monista kielistä sillä, että koodia ei kokonaisuudessaan käännetä, vaan se pilkotaan ensiksi osiin. Tämä operaatio kuluttaa aikaa, mutta käyttäjän ei itse tarvitse kääntää koodeja vaan operaatio suoritetaan automaattisesti. Erona ColdFusioniin PHP ei suostu kääntämään koodia, jos syntaksissa on virhe, vaan vaatii tämän korjausta ennen kääntöä (15, s. 1). PHP:n kehityksessä koodin määrä on suurempi kuin ColdFusionissa, eikä sen oppiminen ole niin helppoa. Myös koodin lukeminen on haastavampaa (31, s.1). ColdFusionin admin-osio on helppo käyttää verrattuna PHP:n php.ini-tiedostoon, sillä ColdFusionin admin-osio on erillinen sivusto, jossa on selkeä ulkoasu (31, s. 1).

Yhteenvetona voidaan siis sanoa, että vaikka ColdFusionin kehittäminen on nopeaa ja helpompi aloittaa, on sen hinta kuitenkin niin suuri tekijä, että se vie paljon kehittäjiä PHP-puolelle. Vaikka PHP-koodi onkin vaikeampaa kirjoittaa, on ongelmiin myös helpompi löytää ratkaisuja suuremman yhteisön avulla. Jos raha ei ole ongelma kehitykselle vertailun puolesta, valitsisin ColdFusionin, mutta pienempään käyttöön PHP on oiva ratkaisu, koska sillä voidaan minimoida kuluja ja osaavien kehittäjien löytäminen on helppoa.

# 7 Yhteenveto

Opinnäytetyössä oli tarkoituksena esitellä ja selvittää, kuinka ColdFusion soveltuu tekniikkana web-ohjelmoinnin toteutuksiin. Työssä selvitettiin taustoja tekniikasta ja pureuduttiin erityisesti sen kehitysympäristön käyttöönottoon ja luotiin yksinkertainen esimerkkijärjestelmä yrityksen sisäiseen käyttöön. Työn kautta ColdFusionin syvempi rakenne tulikin tutuksi, ja sen toiminnallisuuden ydin sekä kehitysympäristöjen monimuotoisuus valottui. Kaikkia teoriaosuuden tekniikoita ei käytetty esimerkkijärjestelmässä, joten tällä tasolla järjestelmän lähdekoodia voitaisi vielä kehittää. Työssä käytettävien sovellusasetusten luonti osoittautui hankalaksi vähäisten esimerkkien johdosta. Tämä oli selkeä huomio järjestelmän luonnissa siitä, että ColdFusionin käyttäjäkunta on pieni, eikä valmiita esimerkkejä ole juuri tarjolla.

ColdFusion soveltuu monenlaisiin toteutuksiin web-ympäristössä ja tarjoaa käyttöön ison joukon valmiita funktioita sekä niiden muokattavuuden Java-pohjansa ansiosta. Työssä käytettiin valmiita funktioita, mutta niitä ei lähdetty räätälöimään järjestelmälle sen enempää. Jos havaittaisiin syitä järjestelmän myöhemmässä kehityksessä toiminnallisuuden muokkaukselle, voisi funktioiden muokkaus olla yksi ratkaisu.

Työssä onnistuttiin luomaan yksinkertainen esimerkkijärjestelmä, jota voisi käyttää pohjana joko suurempaan kokonaisuuteen liitettävänä järjestelmänä tai laajentaa omaksi isoksi järjestelmäksi tarpeiden mukaan. Jatkokehitystä helpottava rakenne nopeuttaa toteutuksien luomista, sillä järjestelmää luotaessa on otettu huomioon uudelleenkäytettävyys erottelemalla osioita toisistaan. Työn avulla päästään helposti perille ColdFusionin käytöstä tällaisten järjestelmien luontiin.

Tarkastellessa web-maailman tulevaisuutta on selvästi havaittavissa, että internetissä toimivat palvelut yleistyvät ja niiden kehityksen tahti kiihtyy. Työn avulla voimme havaita, että vaikka PHP pitää suosituinta sijaa web-palvelimien ohjelmointikielenä, on ColdFusion sille varteenotettava vaihtoehto. ColdFusionia kehitetään nyt useamman yhteisön toimin ja on todennäköistä, että sen suosio alkaa jälleen kasvaa avoimen lähdekoodin versioiden yleistyessä.

On mielenkiintoista seurata, kuinka web-palvelimien ohjelmointikielien tulevaisuus tulee muuttamaan tämänhetkistä näkemystä palvelujen tarjoamisesta internetissä ja kuinka ColdFusion pystyy pärjäämään tällä sektorilla. ColdFusionin käyttö tulee

todennäköisesti pysymään suurempien yritysten käytössä, mutta sen käyttö yleistynee lähiaikoina.

## Lähteet


1. ColdFusion. 2013. Tuotekuvaus. http://www.adobe.com/products/coldfusion-family.html. Luettu 15.3.2013
2. Usein kysytyt kysymykset. 2013. Hinnoittelu. http://www.adobe.com/products/coldfusion-standard/faq.html. Luettu 15.3.2013
3. ColdFusionin käyttö. 2013. Käyttötarkoitukset. http://trends.builtwith.com/framework/Adobe-ColdFusion. Luettu 15.3.2013
4. Tietokantojen käsittely. 2013. Kyselyt. http://helpx.adobe.com/coldfusion/kb/database-connections-handled-coldfusion.html. Luettu 15.3.2013
5. ColdFusion 10. 2013. Ominaisuudet. http://www.adobe.com/products/coldfusion-standard/features.html. Luettu 15.3.2013
6. Arehart, Charlie. 2012. Charlie Arehart's Ultimate List of 200+ New ColdFusion 10 Features [blog] 7. Maaliskuuta http://www.carehart.org/blog/client/index.cfm/2012/3/7/charlie_areharts_ultimate_cf10_new_features_list. Luettu 15.3.2013
7. What's CFML?. 2013. Historia. http://www.opencfmlfoundation.org/whats-cfml/. Luettu 16.3.2013
8. CFML refrenssi. 2013. Merkintöjen yhteenveto. http://help.adobe.com/en_US/ColdFusion/9.0/CFMLRef/WSc3ff6d0ea77859461172e0811cbec17576-7ffd.html. Luettu 16.3.2013
9. ColdFusion developer's guide. 2013. ColdFusion komponentit. http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=buildingComponents_01.html. Luettu 16.3.2013
10. ColdFusion developer's guide. 2013. ColdFusion komponenttien käyttö. http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=buildingComponents_15.html. Luettu 16.3.2013
11. ColdFusion 9 CFML Reference. 2013. cffunction. http://help.adobe.com/en_US/ColdFusion/9.0/CFMLRef/WSc3ff6d0ea77859461172e0811cbec22c24-7f5c.html. Luettu 18.3.2013
12. ColdFusion 9 CFML Reference. 2013. Structuring an application. http://help.adobe.com/en_US/ColdFusion/9.0/Developing/WSc3ff6d0ea77859461172e0811cbec22c24-7d6f.html. Luettu 18.3.2013

13. Defining the application and its event handlers in Application.cfc. 2013. Example: a complete Application.cfc. http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=appFramework_14.html. Luettu 18.3.2013
14. What is a Dynamic Web Page? 2013. http://www.doteasy.com/web-hosting-articles/what-is-a-dynamic-web-page.cfm. Luettu 20.3.2013
15. Difference between various scripting languages http://www.bench3.org/tech/difference-between-various-scripting-languages-php-perl-asp-jsp-coldfusion/. Luettu 20.3.2013
16. Why ColdFusion? http://thenitai.com/2012/02/04/why-coldfusion-cfml-has-its-place-and-is-worth-to-learn-it/. Luettu 22.3.2013
17. Application development. 2012. Adobe ColdFusion 10: Ten Reasons Developers Need to Use It. http://www.eweek.com/c/a/Application-Development/Adobe-ColdFusion-10-Eight-Reasons-Developers-Need-to-Use-It-825736/. Luettu 22.3.2013
18. Web server management. Using the built-in web server. http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=webservmgmt_3.html. Luettu 22.3.2013
19. Introducing ColdFusion. 2013. About Internet applications and web application servers. http://help.adobe.com/en_US/ColdFusion/10.0/Developing/WSc3ff6d0ea77859461172e0811cbec09883-7ffa.html#WSc3ff6d0ea77859461172e0811cbec09883-7ff8. Luettu 22.3.2013
20. ColdFusion Application Server. http://www.learn-coldfusion-tutorial.com/. Luettu 22.3.2013
21. Setting up your ColdFusion development environment for Windows. 2013. Installing Apache Web Server http://www.adobe.com/devnet/coldfusion/articles/setup_dev.html. Luettu 25.3.2013
22. Support. 2013. TCP/IP port numbers required to communicate to SQL over a firewall. http://support.microsoft.com/kb/287932. Luettu 25.3.2013
23. OpenBD. 2013. Tietoa http://openbd.org/about/. Luettu 4.4.2013
24. Why Railo? 2013. The language - for CFML Developers. http://www.getrailo.org/index.cfm/about-railo/why-railo/for-cfml-developers/. Luettu 4.4.2013

25. Hughes, John. 1990. Why functional programming matters. http://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf. Luettu 6.4.2013
26. Benngade, Christopher. 2010. What is Functional Programming? http://devlicio.us/blogs/christopher_bennage/archive/2010/09/06/what-is-functional-programming.aspx.Luettu 6.4.2013
27. Elements of a ColdFusion application. 2013. http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=appFramework_03.html. Luettu 6.4.2013
28. Application.CFC Reference. 2013. onApplicationStart. http://help.adobe.com/en_US/ColdFusion/9.0/CFMLRef/WSc3ff6d0ea77859461172e0811cbec22c24-7d48.html. Luettu 14.4.2013
29. TIOBE Programming Community Index for April 2013. 2013. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Luettu 14.4.2013
30. Are you undecided about ColdFusion vs PHP. 2010. http://igoesolutions.com/blog/2010/07/20/are-you-undecided-about-coldfusion-vs-php/. Luettu 14.4.2013
31. Differences between Java EE and Java SE. 2012. http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html. Luettu 21.4.2013
32. Using ColdFusion Ajax security features. 2013. http://www.adobe.com/devnet/coldfusion/articles/ajax_security.html. Luettu 21.4.2013
33. ColdFusion Tags & Functions. 2013. http://www.cfquickdocs.com/cf8/?getDoc=VerifyClient#VerifyClient. Luettu 21.4.2013

## Liite 1. Esimerkki application.cfc-tiedostosta

```
<cfcomponent>
<cfset This.name = "TestApplication">
<cfset This.Sessionmanagement=true>
<cfset This.Sessiontimeout="#createtimespan(0,0,10,0)#">
<cfset This.applicationtimeout="#createtimespan(5,0,0,0)#">


<cffunction name="onApplicationStart">
    <cftry>
        <!--- Test whether the DB that this application uses is accessible
                by selecting some data. --->
        <cfquery name="testDB" dataSource="cfdocexamples" maxrows="2">
            SELECT Emp_ID FROM employee
        </cfquery>
        <!--- If we get database error, report it to the user, log the error
            information, and do not start the application. --->
        <cfcatch type="database">
            <cfoutput>
                This application encountered an error.<br>
                Please contact support.
            </cfoutput>
            <cflog file="#This.Name#" type="error"
                text="cfdocexamples DB not available. message: #cfcatch.message#
                Detail: #cfcatch.detail# Native Error:
#cfcatch.NativeErrorCode#">
            <cfreturn False>
        </cfcatch>
    </cftry>
    <cflog file="#This.Name#" type="Information" text="Application Started">
    <!--- You do not have to lock code in the onApplicationStart method that
sets Application scope variables. --->
    <cfscript>
        Application.availableResources=0;
        Application.counter1=1;
        Application.sessions=0;
    </cfscript>
    <!--- You do not need to return True if you don't set the cffunction return-
type attribute. --->
```

```
 </cffunction>

<cffunction name="onApplicationEnd">
    <cfargument name="ApplicationScope" required=true/>
    <cflog file="#This.Name#" type="Information"
        text="Application #ApplicationScope.applicationname# Ended">
</cffunction>



<cffunction name="onRequestStart">
    <!--- Authentication code, generated by the Dreamweaver Login Wizard,
        makes sure that a user is logged in, and if not displays a login page. -
-->
    <cfinclude template="mm_wizard_application_include.cfm">
<!--- If it's time for maintenance, tell users to come back later. --->
    <cfscript>
        if ((Hour(now()) gt 1) and (Hour(now()) lt 3)) {
            WriteOutput("The system is undergoing periodic maintenance.
                Please return after 3:00 AM Eastern time.");
            return false;
        } else {
            this.start=now();
        }
    </cfscript>
</cffunction>

<cffunction name="onRequest">
    <cfargument name = "targetPage" type="String" required=true/>
    <cfsavecontent variable="content">
        <cfinclude template=#Arguments.targetPage#>
    </cfsavecontent>
    <!--- This is a minimal example of an onRequest filter. --->
    <cfoutput>
        #replace(content, "report", "MyCompany Quarterly Report", "all")#
    </cfoutput>
</cffunction>

<!--- Display a different footer for logged in users than for guest users or
         users who have not logged in. --->
```

```
<cffunction name="onRequestEnd">
    <cfargument type="String" name = "targetTemplate" required=true/>
    <cfset theAuthuser=getauthuser()>
    <cfif ((theAuthUser EQ "guest") OR (theAuthUser EQ ""))>
        <cfinclude template="noauthuserfooter.cfm">
    <cfelse>
        <cfinclude template="authuserfooter.cfm">
    </cfif>
</cffunction>

<cffunction name="onSessionStart">
    <cfscript>
        Session.started = now();
        Session.shoppingCart = StructNew();
        Session.shoppingCart.items =0;
    </cfscript>
    <cflock timeout="5" throwontimeout="No" type="EXCLUSIVE" scope="SESSION">
        <cfset Application.sessions = Application.sessions + 1>
    </cflock>
    <cflog file="#This.Name#" type="Information" text="Session:
        #Session.sessionid# started">
</cffunction>

<cffunction name="onSessionEnd">
    <cfargument name = "SessionScope" required=true/>
    <cflog file="#This.Name#" type="Information" text="Session:
            #arguments.SessionScope.sessionid# ended">
</cffunction>

<cffunction name="onError">
    <cfargument name="Exception" required=true/>
    <cfargument type="String" name = "EventName" required=true/>
    <!--- Log all errors. --->
    <cflog file="#This.Name#" type="error" text="Event Name: #Eventname#">
    <cflog file="#This.Name#" type="error" text="Message: #exception.message#">
    <!--- Some exceptions, including server-side validation errors, do not
             generate a rootcause structure. --->
    <cfif isdefined("exception.rootcause")>
```

```
        <cflog file="#This.Name#" type="error"
            text="Root Cause Message: #exception.rootcause.message#">
    </cfif>
    <!--- Display an error message if there is a page context. --->
    <cfif NOT (Arguments.EventName IS onSessionEnd) OR
            (Arguments.EventName IS onApplicationEnd)>
        <cfoutput>
            <h2>An unexpected error occurred.</h2>
            <p>Please provide the following information to technical sup-
port:</p>
            <p>Error Event: #EventName#</p>
            <p>Error details:<br>
            <cfdump var=#exception#></p>
        </cfoutput>
    </cfif>
 </cffunction>


</cfcomponent>
```

**Kuva 1. Esimerkki application.cfc-tiedostosta (13, s. 2)**

# Liite 2. Järjestelmän createvouchers.cfc-tiedosto

```
<cfcomponent output="false">

    <cffunction name="init" output="false" returntype="createVouchers">
        <cfargument name="dbsource" required="true" type="string">
        <cfset variables.dbsource = arguments.dbsource>
        <cfreturn this />
    </cffunction>

<cffunction name="getPerson" output="false" returntype="query">
    <cfargument name="personId" type="string" required="true">
    <cfset var locals = {}>
    <cfquery name="locals.get" datasource="#variables.dbsource#">
        select * from tyontekija where id = <cfqueryparam cfsqltype="cf_sql_integer" value="#arguments.personID#">
    </cfquery>

        <cfreturn locals.get>
</cffunction>

<cffunction name="getPersons" output="false" returntype="query">
    <cfset var locals = {}>
    <cfquery name="locals.get" datasource="#variables.dbsource#">
        select * from tyontekija
    </cfquery>

        <cfreturn locals.get>
</cffunction>

<cffunction name="addPerson" output="false" returntype="boolean">
    <cfargument name="firstName" type="string" required="true">
    <cfargument name="lastName" type="string" required="true">
    <cfargument name="reward" type="string" required="true">
    <cfset var locals = {}>
    <cfset locals.success = false>

<cftry>
    <cfquery name="locals.add" datasource="#variables.dbsource#">

        INSERT INTO tyontekija
                ( etunimi, sukunimi, palkinto)
                VALUES
                (
                    <cfqueryparam cfsqltype="cf_sql_varchar" maxlength="255" value="#trim(arguments.firstName)#">,
                    <cfqueryparam cfsqltype="cf_sql_varchar" maxlength="255" value="#trim(arguments.lastName)#">,
                    <cfqueryparam cfsqltype="cf_sql_integer" value="#arguments.reward#">
                )
    </cfquery>

        <cfset locals.success = true>
                <cfcatch>
                    <cfset locals.success = false>
                </cfcatch>

</cftry>
<cfreturn locals.success>
</cffunction>

<cffunction name="updatePerson" output="false" returntype="boolean">
<cfargument name="personId" type="string" required="true">
<cfargument name="firstName" type="string" required="true">
    <cfargument name="lastName" type="string" required="true">
    <cfargument name="reward" type="string" required="true">
    <cfset var locals = {}>
    <cfset locals.success = false>

<cftry>
    <cfquery name="locals.update" datasource="#variables.dbsource#">

        UPDATE tyontekija SET etunimi = <cfqueryparam cfsqltype="cf_sql_varchar" maxlength="255" value="#trim(arguments.firstName)#">,
         sukunimi=<cfqueryparam cfsqltype="cf_sql_varchar" maxlength="255" value="#trim(arguments.lastName)#">, palkinto = <cfqueryparam cfsqltype="cf_sql_integer" value="#arguments.reward#">
                WHERE
                id=<cfqueryparam cfsqltype="cf_sql_integer" value="#arguments.personID#">
    </cfquery>

        <cfset locals.success = true>
                <cfcatch>
                    <cfset locals.success = false>
                </cfcatch>

</cftry>
<cfreturn locals.success>
</cffunction>

</cfcomponent>
```

Kuva 1. Createvouchers.cfc-tiedosto

# Liite 3. Järjestelmän toiminnallisuudet

```
<cfinclude template = "header.cfm">
<div class = "contentwrapper">
        <div class="contenttopbar">
        </div>
    <div class = "content">
        <cfoutput>
            <cftry>
                <cfif #form.type# eq "update">
                    <cfset updateaction =  application.objCreateVouchers.updatePerson(personId=#form.reward#, firstname=#form.firstname#, lastname=#form.lastname#, reward=#form.reward#)>
                        <cfif "#updateaction#" eq true>
                            <cfoutput>"Henkilön päivitys onnistui"</cfoutput>
                        <cfelse>
                            <cfoutput>"Henkilön päivitys epäonnistui"</cfoutput>
                        </cfif>
                </cfif>
                <cfif #form.type# eq "add">
                    <cfset addaction =  application.objCreateVouchers.addPerson(firstname=#form.firstname#, lastname=#form.lastname#, reward=#form.reward#)>
                    <cfif "#addaction#" eq true>
                        <cfoutput>"Henkilön lisäys onnistui"</cfoutput>
                    <cfelse>
                        <cfoutput>"Henkilön lisäys epäonnistui"</cfoutput>
                    </cfif>
                </cfif>
                <cfcatch>
                    <cfset mode = "none">
                </cfcatch>
            </cftry>
                <br>
            <div class="actionwrapper">
                    <div class="contenttopbar">
                        Henkilön lisäys:
                    </div>
                <div class="action">
                    <form method="post" action="#request.rootMap#/add.cfm">
                        <button type="submit">Lisää henkilö</button>
                    </form>
                </div>
            </div>
            <div class="actionwrapper">
                <div class="contenttopbar">
                    Henkilön päivitys:
                    </div>
                <div class="action">
                    Valitse henkilö:
                    <form method="post" action="#request.rootMap#/update.cfm">
                        <select name="person">
                            <cfset list = application.objCreateVouchers.getPersons()>
                            <cfloop  query = #list#>
                                <option value=#list.Id#>#list.etunimi# #list.sukunimi#</option>
                            </cfloop>
                        </select>
                        <button type="submit">Päivitä henkilön tiedot</button>
                    </form>
                </div>
            </div>
            <div class="actionwrapper">
                <div class="contenttopbar">
                    Henkilön palkitseminen:
                    </div>
                <div class="action">
                    Valitse henkilö ja palkitsemisen peruste:
                    <form method="post" action="#request.rootMap#/palkitse.cfm">
                        <select name="person">
                            <cfset list = application.objCreateVouchers.getPersons()>
                            <cfloop  query = #list#>
                                <option value=#list.Id#>#list.etunimi# #list.sukunimi#</option>
                            </cfloop>
                        </select>
                        <input name="reason" type="text" size="20">
                        <button type="submit">Palkitse henkilö</button>
                    </form>
                </div>
            </div>
        </cfoutput>

    </div>
    <div class="clear">
    </div>
    <div class="contentbottombar">
    </div>
</div>
<cfinclude template = "footer.cfm">

```

**Kuva 1. Index.cfm-tiedosto**

```
<cfinclude template = "header.cfm">
<div class = "contentwrapper">
    <div class="contenttopbar">
    </div>
    <div class = "content">
        <div class="actionwrapper">
            <div class="contenttopbar">
                Henkilön lisäys
            </div>
            <div class="action">
                <form method="post" action="index.cfm">
                    <table>
                        <tr>
                            <td>Etunimi:</td>
                            <td><input name="firstname" type="text" size="20"></td>
                        </tr>
                        <tr>
                            <td>Sukunimi:</td>
                            <td><input name="lastname" type="text" size="20"></td>
                        </tr>
                        <tr>
                            <td>Palkinto</td>
                            <td>
                                <select name="reward">
                                    <option value="1">1</option>
                                    <option value="2">2</option>
                                </select>
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2" align="right">
                                <input name="type" value="add" hidden="true">
                                <input type="submit" value="Lisää henkilö">
                            </td>
                        </tr>
                    </table>
                </form>
            </div>
        </div>
    </div>
    <div class="clear">
    </div>
    <div class="contentbottombar">
    </div>
</div>
<cfinclude template = "footer.cfm">

```

**Kuva 2. add.cfm-tiedosto**

```
<cfinclude template = "header.cfm">
<div class = "contentwrapper">
  <div class="contenttopbar">
  </div>
  <div class = "content">
    <div class="actionwrapper">
      <div class="contenttopbar">
        Henkilön päivitys
      </div>
      <div class="action">
        <form method="post" action="index.cfm">
          <table>
          <cfoutput>
            <cfset person = application.objCreateVouchers.getPerson(personId=#form.person#)>
            <tr>
              <td>Etunimi:</td>
              <td><input name="firstname" type="text" value="#person.etunimi#" size="30"></td>
            </tr>
            <tr>
              <td>Sukunimi:</td>
              <td><input name="lastname" type="text" value="#person.sukunimi#" size="30"></td>
            </tr>
            <tr>
              <td>Palkinto</td>
              <td>
                <select name="reward" value="#person.palkinto#">
                  <option value="1"<cfif #person.palkinto# eq 1>
                  selected
                  </cfif>>1</option>
                  <option value="2" <cfif #person.palkinto# eq 2>
                  selected
                  </cfif>> 2</option>
                  <option value="3" <cfif #person.palkinto# eq 3>
                  selected
                  </cfif>> 3</option>
                </select>
              </td>
            </tr>
            <tr>
              <td colspan="2" align="right">
                <input name="type" value="update" hidden="true">
                <input type="submit" value="Päivitä tiedot">
              </td>
            </tr>
          </cfoutput>
          </table>
        </form>
      </div>
    </div>
  </div>
  <div class="clear">
  </div>
  <div class="contentbottombar">
  </div>
</div>
<cfinclude template = "footer.cfm">

```

Kuva 3. update.cfm-tiedosto

```
<html>
<head>
    <title></title>
    <cfoutput><link rel="stylesheet" href="#request.cssMap#/palkinto.css"></cfoutput>
</head>
<body>
    <cfoutput>
        <cfset person = application.objCreateVouchers.getPerson(personId=#form.person#)>
    <div id="content"><cfif #person.palkinto# eq 1>
        <img src="#request.imageUrl#/palkinto1.PNG" width="900" height="1200" ></div>
    <cfelse>
    <img src="#request.imageUrl#/palkinto2.PNG" width="900" height="1200" ></div>
    </cfif>
    <div id = "palkittu">#form.reason#</div>
    <div id="saaja">#person.etunimi# #person.sukunimi#</div>
</cfoutput>
</body>
</html>
```

Kuva 4. palkitse.cfm-tiedosto

## Liite 4. Esimerkki palkinnosta


Kuva 1. Palkinto

## Liite 5. Järjestelmän ulkoasun tiedostot

```
<!doctype html>

<html>
<head>
<meta name="author" content="Dino Väärä">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>
        Palkintojen jako
    </title>
    <cfoutput><link rel="stylesheet" href="#request.cssMap#/styles.css"></cfoutput>

</head>

<body>

    <header>
        <div class="header">
            <div class="headertext">
                Tervetuloa palkitsemisjärjestelmään
            </div>
        </div>

    </header>
    <div class = "heightwrapper">
```

Kuva 1. header.cfm

```
        <div class="clear"></div>
        <footer>
            <div class="footer"><br>
                <div class="footertext">
                    Copyright &copy; 2013 Dino Väärä, all rights reserved.
                </div>
            </div>
        </footer>
        </div>

</body>
</html>

```

Kuva 2. footer.cfm

```css
body{
    width: 100%;
    background-color: #72aa00;
    color:#1A350C;
}

.heightwrapper{
    margin-right: auto;
    margin-left:auto;
    width: 980px;
}
header{
    width: 100%;
    margin-top:-10px;
    margin-left:-8px;
    height: 30px;
    background: #bfd255; /* Old browsers */
background: -moz-linear-gradient(top,  #bfd255 0%, #8eb92a 50%, #72aa00 51%, #9ecb2d 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#bfd255), color-stop(50%,#8eb92a), color-stop(51%,#72aa00), color-stop(100%,#9ecb2d));
background: -webkit-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* IE10+ */
background: linear-gradient(to bottom,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#bfd255', endColorstr='#9ecb2d',GradientType=0 ); /* IE6-9 */

}

.contentwrapper{
    height:420px;
    margin-top:40px;
}
.contenttopbar{
    height: 20px;
-webkit-border-top-left-radius: 10px;
-webkit-border-top-right-radius: 10px;
-moz-border-radius-topleft: 10px;
-moz-border-radius-topright: 10px;
border-top-left-radius: 10px;
border-top-right-radius: 10px;
background: #bfd255; /* Old browsers */
background: -moz-linear-gradient(top,  #bfd255 0%, #8eb92a 50%, #72aa00 51%, #9ecb2d 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#bfd255), color-stop(50%,#8eb92a), color-stop(51%,#72aa00), color-stop(100%,#9ecb2d));
background: -webkit-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* IE10+ */
background: linear-gradient(to bottom,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#bfd255', endColorstr='#9ecb2d',GradientType=0 ); /* IE6-9 */
}
.footer{
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
border-radius: 10px;
background: #bfd255;
    height: 70px;

}
button{
    margin-top:5px;
}
.actionwrapper{
    margin-top:15px;
        color:#fff;
    width: 300px;
}
.action{

    padding:10px 10px 10px 10px;

    background: #bfd255;
}
.footertext{
    margin-right: auto;
    margin-left:auto;
    width: 400px;
    padding-left: 80px;
}
.headertext{
    margin-right: auto;
    margin-left:auto;
    width: 400px;
    padding-left:200px;
    color:#fff;
}
.content{
    background-color:#9dd53a;
    height: 325px;
    padding:20px 20px 20px 20px;
    color:#1A350C;
}

.contentbottombar{
    height: 20px;

    -webkit-border-bottom-right-radius: 10px;
-webkit-border-bottom-left-radius: 10px;
-moz-border-radius-bottomright: 10px;
-moz-border-radius-bottomleft: 10px;
border-bottom-right-radius: 10px;
border-bottom-left-radius: 10px;
background: #bfd255; /* Old browsers */
background: -moz-linear-gradient(top,  #bfd255 0%, #8eb92a 50%, #72aa00 51%, #9ecb2d 100%); /* FF3.6+ */
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#bfd255), color-stop(50%,#8eb92a), color-stop(51%,#72aa00), color-stop(100%,#9ecb2d));
background: -webkit-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Chrome10+,Safari5.1+ */
background: -o-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* Opera 11.10+ */
background: -ms-linear-gradient(top,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* IE10+ */
background: linear-gradient(to bottom,  #bfd255 0%,#8eb92a 50%,#72aa00 51%,#9ecb2d 100%); /* W3C */
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#bfd255', endColorstr='#9ecb2d',GradientType=0 ); /* IE6-9 */
}
.clear{
    clear:both;
}
```

**Kuva 3. styles.css-tiedosto**

```css
#palkittu {
    top:965px;
    left:470px;
    position:absolute;
    width: 200px;
    font-size: 18px;
    color:#999999;
    font-weight: bold;
    z-index: 10;
}
#saaja {
    top:1040px;
    left:470px;
    position:absolute;
    width: 200px;
    font-size: 18px;
    color:#999999;
    font-weight: bold;
    z-index: 10;
}
#content {
    position: relative;
}
```

Kuva 4. palkinto.css-tiedosto