TAMPEREEN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

WWW-KOULUTUSSIVUSTON KEHITTÄMINEN
Case Inspecta Oy

Janne Leinonen

Tietojenkäsittelyn koulutusohjelma
Tammikuu 2008
Työn ohjaaja: Paula Hietala

TAMPERE 2008

TAMPEREEN
AMMATTIKORKEAKOULU

Tekijä(t) Leinonen Janne

Koulutusohjelma(t) Tietojenkäsittely

Opinnäytetyön nimi Www-koulutussivuston kehittäminen Case Inspecta Oy

Työn valmistumis-kuukausi ja -vuosi Tammikuu 2008

Työn ohjaaja Hietala Paula

Sivumäärä: 50

TIIVISTELMÄ

Työn ja tuotannon tehostaminen on nykypäivänä tärkeitä teesejä yritysmaailmassa. Turhia työtunteja ja matkoja on syytä karsia, jotta saatu hyöty on suurempi kuin käytetyt resurssit. Vanhojen työtehtävien sähköistäminen on yksi tapa tehostaa työntekoa monella alalla erityisesti koulutuspuolella. Toimeksiantajanani toimi Inspecta Oy, joka on kansainvälinen tuotantolaitosten turvallisuustarkastusyritys. Inspecta Oy kouluttaa itse omat työntekijänsä, sekä järjestää koulutusta monilla eri tarkastusaloilla. Inspecta Oy:n tarkoituksena oli kehittää nykyistä koulutustapaansa tehokkaammaksi siirtämällä osa koulutukseen käytetystä ajasta virtuaaliseksi verkko-koulutukseksi. Sain tehtäväkseni rakentaa koulutussivuston, joka vastaa yrityksen erityistarpeita ja olla osana kehittämässä koulutuksen tehostamista valtakunnallisesti.

Koulutussivuston rakentaminen aloitettiin hankkimalla tietoa nykyaikaisista WWW-tekniikoista ja näiden soveltamisesta käytäntöön. Aineistona käytettiin pääasiassa painettua materiaalia, mutta myös sähköisen materiaalin osuus työssä on merkittävä johtuen sen ajankohtaisuudesta. Käytettyjä tekniikoita ovat: PHP, MySQL, JavaScript, Ajax sekä sivupohjajärjestelmä.

Raporttini kuvaa projektin kehitystä alkuideasta lopulliseksi tuotteeksi. Lopputuloksena esitellään valmis tuote ja kuinka kaikki on lopulta rakennettu. Esitelty työ on siinä vaiheessa missä se oli raportin kirjoitushetkellä 15.11.2007.

Avainsanat PHP Ajax Sivupohja Koulutussivusto Javascript

TAMPERE
POLYTECHNIC

Author(s)	Leinonen Janne
Degree Programme(s)	Business Information Systems
Title	Creating a Web based learning platform. Case Inspecta Oy
Month and year	January 2008
Supervisor	Hietala Paula

Pages: 49

ABSTRACT

Efficiency and productivity are important words nowdays in business world. Useless work hours and business trips should be cut to minimum, when the gain is less than the resources used. Rethinking old tasks in modern ways is one way to accomplish this especially in educational sector. My employer is Inspecta Ltd. which is a international production unit safety inspection corporation. The company trains all it´s workers by itself and also gives training to others. Inspecta Ltd´s main idea was to enhance their present training possibilities with a modern web based learning platform. I was given a task to design and develop the platform which was especially designed for company´s special demands nation-wide.

Project started with obtaining information about modern web techniques and adapting those techniques in use. Material was primary written literature but e-literature played also important role as it is more current than written. Web techniques which were used: PHP, MySQL, JavaScript, Ajax and template system.

My report describes how an idea was developed into a final product. As a result I introduce the product and how it was built. Because development is continuous is the introduced product as it was when this report was written in 15.11.2007.

Keywords PHP Ajax Template Learning platform Javascript

SISÄLLYSLUETTELO

Käsitteet ja lyhenteet

WWW	World Wide Web Digitaalinen tietoverkko, joka koostuu tuhansista pienemmistä verkoista ja miljoonista tietokoneista.
Selain	Tietokoneohjelma, jonka avulla käyttäjä voi katsella WWW-sivuja.
Session	Käyttäjän tietokoneelle palvelimen asettama data, jota vain kyseinen palvelin voi käyttää.
HTML	HyperText Markup Language Avoimesti standardoitu kuvauskieli, josta internetsivut rakentuvat.
DTD	Document Type Definition Määrittelee dokumentin rakenteen eli ns. kieliopin. Ei puutu sisältöön millään tavalla.
XML	Extended Markup Language Rakenteellinen kuvauskieli, jolla tiedon merkitys on kuvattavissa tiedon sekaan.
XHTML	Extensible HyperText Markup Language Kuin HTML, mutta sisältää XML-rakenteen.
CSS	Cascading Style Sheets Tyylitiedosto, joka erottaa internetsivujen ulkonäön sisällöstä.
PHP	Hypertext Preprocessor (ent. Personal HomePage tools). Palvelinpuolen komentokieli.
SQL	Structured Query Language IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
DHTML	Dynamic HyperText Markup Language Eri tekniikoita yhdistävä tapa tuottaa dynaamisia ja interaktiivisia WWW-sivuja.
JavaScript	Web-ympäristössä käytettävä komentosarjakieli.

DOM	Document Object Model Tapa viitata XML tai HTML elementtiin objektina.
AJAX	Asynchronous JavaScript And Xml Ohjelmointitekniikka, jota käytetään interaktiivisten www-sovellusten kehittämiseen.
Sivupohjajärjestelmä	Verkkosivujen esityslogiikan ja käyttölogiikan erottaja. engl. (”Template system”)
Sivupohjamoottori	Sivupohjajärjestelmän tärkein osa, joka hoitaa itse työn. Käytetään myös nimityksiä: (Template engine, sivumuunnin)
Sivupohja	Sivumoottori käyttää hyväkseen sivupohjia (template) sivun ulkoasun esittämiseen.

1 Johdanto

Internet on nykypäivänä jo kaikille tuttu käsite. Yritykset ja yksityiset henkilöt ovat linkittyneet tiedon valtatielle niin saumattomasti, että on jo vaikea kuvitella aikaa ennen internetiä. Kuitenkin suurin osa internetissä käytetystä ajasta käytetään enemmän huvi-, kuin hyötytarkoituksiin. Verkottuminen tarjoaisi kuitenkin monelle yritykselle sellaisia mahdollisuuksia, joita ennen ei ole ollut tarjolla; muutakin kuin kotisivut.

WWW-sivustoja pystyy nykyään lähes kuka tahansa tekemään hyvinkin pienellä vaivalla ja vaihtoehtoisia tapoja tuottaa niitä on lukemattomia. Kuitenkin oikeaoppisen ja standardien mukaisen sivujen luonti on haaste tottuneellekin tekijälle varsinkin jos toteutetaan laajempaa sivustoa.

Opinnäytetyöni aiheen sain harjoittelupaikastani Inspecta Oy:ssä, jossa alun perin oheisprojektina lähtenyt WWW-sivujen tekeminen, muuttui harjoittelun edetessä tärkeimmäksi ja mielenkiintoisimmaksi projektiksi. WWW-sivustoidea lähti käyntiin kouluttajan tarpeesta siirtää paperiset tentit sähköiseen muotoon, mutta projektin edetessä mielenkiinto sivustoa kohtaan kasvoi ja ominaisuuksia kehitettiin lisää. Täydellinen suunnanvaihdos projektissa tapahtui siinä vaiheessa, kun projektista kiinnostui useampikin kouluttaja, joilla jokaisella oli omat vaatimuksensa sivuston suhteen. Aloimme suunnitella kokonaista koulutussivustoa.

Toimeksiantaja Inspecta Oy

Toimeksiantajana toimi Inspecta Oy, joka on kansainväliset tuotantolaitosten turvallisuusvaatimukset tunteva Suomessa toimiva palveluyritys. Tarkastuslaitosperustansa Inspecta on luonut toimiessaan teknillisenä tarkastuskeskuksena vuodesta 1975 vuoteen 1998, jona aikana siitä oli kasvanut Suomen johtava paineastiatarkastuslaitos. Inspecta Oy osti Suomen johtavan NDT-tarkastuslaitoksen vuonna 2000.

Seuraava merkittävä kehitysaskel oli maan johtavan sertifiointiyrityksen SFS - Sertifioinnin liittäminen osaksi Inspectaa sen yksityistämisen yhteydessä lokakuussa 2002. Kehitys johti maan johtavan sähkötarkastusyhtiön Fimteknon ostoon keväällä 2004. Tätä laajentumiskehitystä ovat tukeneet useat liiketoimintakaupat, joilla on täydennetty palveluvalikoimaa.

Inspecta Oy:lla ei ole ennen ollut mitään vastaavaa sovellusta tai sivustoa ja vasta tänä kesänä (2007) Intranet uudistui nykyajan vaatimuksia vastaavaksi. Tästä johtuen ei toimeksiantajalla ollut oikein konkreettista näkemystä sivustosta, joten paljon työssä jäi tekijän omalle vastuulle. Hyvänä puolena Intranetin uudistus toi mukanaan upouudet palvelimet ja ohjelmistot, mikä tarkoitti sitä, että PHP:n ja MySQL:n uusimpien versioiden käyttö oli mahdollista.

Tavoite

Tämän opinnäytetyön tavoitteena on antaa lukijalle käsitys siitä, mitä asioita kannattaa miettiä ja ottaa huomioon tämän mittakaavan sivustoa luotaessa. Tapoja tehdä on lukuisia, mutta esittelen tässä niistä yhden, joka pyrkii kattamaan nykyaikaisen WWW-sivuston vaatimukset.

Opinnäytetyöni tavoitteena on luoda toimiva dynaaminen tietokantapohjainen koulutussivusto selainkäyttöliittymällä. Raportissa kuvaan miten toteuttamassani sivustossa on otettu nykyaikaisen WWW-sivuston tekniikat huomioon.

Lähteet

Lähteiden käytössä pyrin aina ensin käyttämään uusinta mahdollista kirjallista aineistoa, jota tuin mahdollisesti sähköisestä lähteestä löytyvällä lisätiedolla. Lähdeaineistoa löytyy runsaasti koskien WWW-sivujen tekemistä ja suunnittelua niin kirjallisesti kuin sähköisesti. Tärkeimpänä kirjallisena lähteenä käytin Tero Linjaman XHTML-kirjaa (2001).

Ajankohtaisinta tietoa saa oikeastaan ainoastaan sähköisenä, mutta perus WWW-tekniikoiden kohdalla tämä ei onneksi ole mikään ongelma, sillä perusteet ovat pysyneet samoina. Uusinta ja ajankohtaisinta tietoa löytyy W3C:n virallisilta sivuilta ja PHP.net sivustolta. AJAX- tekniikan osalta hyvä painettu lähde on ehdoton edellytys oppimiselle, mutta tekniikasta löytyy myös paljon vapaan lähdekoodin alaisia esimerkkejä oppimista tukemaan. Ainoastaan sivupohjajärjestelmän (template system) suomenkielinen dokumentointi on hyvin vajavaista, joten jouduin tässä valitsemaan termeistä yhden useimmin käytetyn vaihtoehdon. Muita käytettyjä termejä ovat: sivumuunnin järjestelmä, mallinnejärjestelmä tai templatejärjestelmä. Tärkein sivupohjajärjestelmän tiedonlähde oli www.wikipedia.org, sekä Rami Heinisuon ja Ilkka Raudan PHP ja MySQL Tietokantapohjaiset verkkopalvelut-kirja (2007). Työssäni käytän termejä sivupohjajärjestelmä, sivumoottori ja sivupohja.

2 Merkkauskielet

2.1 HTML

HTML-sivut ovat edelleen WWW-sivujen perusrakennusmateriaali, vaikka puhutaankin dynaamisista PHP- tai ASP-sivuista. Kaikki selaimissa näkyvä data esitetään perinteisenä staattisena HTML-sivuna, vaikka sisältö tuotettaisiinkin dynaamisesti. HTML-koodia voidaan kirjoittaa millä tahansa tekstieditorilla ja ideana on kertoa selaimelle tietyssä järjestyksessä, mitä sen tulee näytölle tulostaa. HTML-koodi sisältää tageja, jotka kertovat elementin alkamisesta ja päättymisestä. Perinteinen HTML-koodaus sisältää kaiken ulkoasun, rakenteen ja sisällön samassa tiedostossa. HTML-taitto-ohjelmia on olemassa lukuisia mm. Microsoft FrontPage tai Macromedia Dreamweaver, mutta tällä kertaa työ on tehty käyttäen apuna vain ilmaista Notepad++ tekstieditoria.

HTML-dokumentti

”HTML-dokumentti on dokumentti, joka sisältää HTML-kielen muotoilukomentoja ja joka muodostaa siten ohjeet internet-selaimessa esitettävien WWW-sivujen loogiseen tai fyysiseen muotoiluun. Tiedostopääte on .htm tai .html” (Linjama 2001:32).

HTML-kieli sisältää elementtejä, joilla on hierarkkinen rakenne. Juurielementtejä voi olla ainoastaan yksi ja muut elementit tulevat tämän juurielementin sisälle. Vaadittu rakenne HTML-sivuilla on, että se sisältää <html><head><body> -elementit.

```
<html>
	<head>
		<title>Esimerkkisivu</title>
	</head>
	<body>
		<p>Sisältöä tähän</p>
	</body>
</html>
```

HTML-kieli tulkataan vasta selaimessa ja usein selaimet esittävät lopputuloksen hieman toisistaan poikkeavassa muodossa. Tämä johtuu yleensä HTML-dokumentissa olevista virheistä, joita selain yrittää korjata oletuksen perusteella, mutta myös siitä, että eri selaimet noudattavat hieman eri standardeja. Esimerkiksi Internet Explorer käsittelee margin-, padding- ja border-määreitä hieman

eri tavalla kuin muut, joten sivun asettelussa se aiheuttaa yleensä ongelmia erilaisena lopputuloksena. Siksi onkin hyvä testata tuottamansa sivut aina useammalla selaimella tai asentaa sivuille käyttäjälle kehotus siirtyä laadukkaimpiin selaimiin (www.explorerdestroyer.com).

2.2 XML

”XML on SGML-kielestä johdettu kielioppi, joka perustuu uuteen, rakenteellisten dokumenttien laajennettavaan ajattelumalliin” (Linjama 2001:23). XML:n avulla luodaan muita XML-pohjaisia kieliä. Se itse määrittelee dokumentin yleisen rakenteen, mutta ei elementeille merkityksiä. Jokainen elementti sisältää kuitenkin aina aloitus-, sekä lopetuskomennot. XML-dokumentti on tekstipohjaista dataa, joten se on alustariippumatonta ja on ihmiselle helposti ymmärrettävää sellaisenaan.

```
<?xml version=”1.0” encoding=”uft-8”?>
<kirjat>
        <kirja>
                <nimi>Manuaali</nimi>
                <koko>120</koko>
        </kirja>
</kirjat>
```

Ensimmäisellä rivillä esitellään kieliversio ns. prologi. Juurielementtinä on Kirjat, joita voi olla dokumentissa vain yksi kappale.

2.3 XHTML

XHTML on merkkauskieli, jonka ideana on yhdistää HTML-kielen sisältö XML-tyyliseen kielioppiin. Käytännössä tämä tarkoittaa sitä, että kaikki luotu data esitettäisiin kaikissa selaimissa samanlaisena ja selainten rakennetta voitaisiin keventää. XHTML 1.0 on virallisesti esitelty tammikuussa 2000 ja sen on tarkoitus korvata vanhat HTML-standardit, joten W3C suosittelee dokumenttien muuttamista XHTML-standardin mukaisiksi (Linjama 2001:854). XHTML-dokumentti ei eroa perinteisestä HTML-dokumentistä kuin kielioppierojen osalta, joiden tarkoitus on tiukentaa yhteisiä käytäntöjä ja normeja.

2.3.1 XHTML-kriteerit

Yleisimmät kriteerit XHTML sivuja luotaessa ovat:

- Kaikki elementit on suljettava. Myös ns. tyhjät elementit. (
).
- Juurielementti on aina <html>
- Elementit ja attribuutit kirjoitetaan pienellä.
- Elementtien avaus ja sulkujärjestyksen tulee täsmätä.
- Attribuuteille tulee antaa jokin arvo.
- Attribuutin arvon tulee olla lainausmerkeissä.
- Entiteetit >, <, & yms. merkit tulee esittää muodossa &alt, &amp. Suositeltavaa on enkoodata kaikki erikoismerkit kuten skandinaaviset merkit.

2.3.2 XHTML-dokumentti

XHMTL dokumentin rakenne on seuraavanlainen:
(perus HTML-koodi lihavoituna eron huomaamiseksi.)

```
1. <?xml version=”1.0” encoding=”utf-8”?>

2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN" "http://WWW.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">

3. <html xmlns="http://WWW.w3.org/1999/xhtml"
xml:lang="fi">

4. <head>
        <title>Otsikko</title>
        <meta http-equiv="Content-Type"
        content="application/xhtml+xml; charset=utf-
        8" />
</head>
<body>
      <p>Sisältöä tähän</p>
</body>
</html>
```

Ensimmäisessä kohdassa ilmoitetaan XML versio, rakenne ja käytetty merkistö. Tämä ei ole pakollista, mutta on kuitenkin suositeltavaa. Toisessa kohdassa ilmoitetaan kieliversio ja siihen linkitetään XHTML.dtd, jota vasten sivujen oikeaoppinen rakenne tarkastetaan. Kolmannessa kohdassa määritellään juurielementti, joka on aina html. Juurielementissä kerrotaan kielityyppi xmlns-attribuutilla, jonka arvona on W3C XHTML -määrittelyn URL-osoite. Neljäs kohta on normaalia HTML:ää, paitsi meta elementti, jossa kerrotaan käytetty merkistö, sekä sivun tuottaman tiedon kuvaus.

3 CSS-tyylimääreet

CSS on yksi askel verkkosivujen tuottamisen tehostamisessa. Ideana on erottaa ulkoasua määrittävät tekijät täysin rakennetta määrittävistä tekijöistä. Tyylien tarkoituksena on kuvata sivuston ulkoasua ja näin ollen sivuston ulkoasun hallinta tehostuu. Tyylien käyttämisen hyötyjä sanotaan yleisesti olevan enemmän kuin haittoja. Suurimpana haittana pidetään sitä, että useat selaimet eivät näytä oikein tyylitiedostoja ja ”Jacob Nielsenin tekemän tutkimuksen mukaan ihmiset eivät kovin helposti asenna koneelleen uusinta selainta, jos vanha selain toimii heidän mielestään riittävän hyvin” (Linjama 2001:502).

Tyylimääreitä on mahdollista käyttää kolmella eri tapaa. Style-attribuutilla se voidaan kirjoittaa suoraan HTML/PHP-tiedostoon <head> elementin sisään. Toinen tapa on kirjoittaa se suoraan kohteena olevan elementin sisään tai erillisellä CSS-päätteisellä tiedostolla, joka tuodaan (import) kuhunkin HTML/PHP-tiedostoon. Luonnollisesti jälkimmäinen tapa on huomattavasti järkevämpi, sillä tämän avulla molempien tiedostojen muokkaaminen ja ymmärtäminen on helpompaa. Lisäksi suositellaan, että tyylitiedosto olisi vain suositustyylitiedosto ja käyttäjällä olisi mahdollisuus korvata tyylitiedosto omallaan näin halutessaan.

Esimerkki sisäisen tyylin määrittämisestä sivun head-osiossa.

```
<Style type=”text/css”>
body{
     background: #000;
}
</style>
```

Esimerkki tyylin määrittämisestä sivun elementille:

```
<p style=”background: #000;”>
Sisältöä
</p>
```

Sivustolle voidaan tuoda tyylejä myös useammasta kuin yhdestä tyylitiedostosta. Myöhemmin määritelty tyyli syrjäyttää aina aiemmin määritellyt tyylit. Tästä on hyötyä silloin, kun halutaan esimerkiksi tehdä erilaisia tyylejä eri selaimille. Toinen yleinen käyttötapa on luoda yksi tyylitiedosto yleiselle rakenteelle ja toinen tyylitiedosto, jolla käyttäjä voi muokata esimerkiksi värit haluamikseen.

CSS tiedosto (tyylit.css) linkitetään HEAD osioissa import-komennolla:

```
<link rel="stylesheet" type="text/css"
href="tyylit.css" />
```

Tämän jälkeen kaikki CSS-tiedostossa tehdyt muutokset vaikuttavat suoraan HTML/PHP-sivun esittämiseen.

3.1 Syntaksi

”CSS:n perussyntaksi muodostuu valitsimesta, ominaisuudesta ja arvosta” (Wikipedia).

```
Valitsin{
    ominaisuus: arvo;
}
```

Valitsin voi olla mikä tahansa HTML kielen syntaksi (b, strong, table, body, h1…), jonka ominaisuuksia (color, height, width…) voidaan muutella suhteellisilla tai absoluuttisilla pituuksilla (px, %, em, pt). Valitsinta voidaan käsitellä globaalisti tai kohdennettuna jonkin id:n (#-merkki) tai class:n (. –merkki) alle.

Esimerkiksi otsikoille h1 ja h2 voidaan globaalisti määritellä väri (musta) seuraavasti:

```
h1, h2{
    color: #000;
}
```

Tyyleille voidaan määritellä myös luokkia, jolloin voidaan luoda eri tilanteisiin sopivia samoja elementtejä. Aliluokka voidaan määritellä joko tietylle elementille tai täysin globaaliksi luokaksi, jota voidaan käyttää minkä tahansa elementin yhteydessä.

Esimerkki h1 otsikolle tehty aliluokka ”etusivu”:

```
h1{
    color: #000;
}

h1.etusivu{
    color: #000FFF;
}
```

Voidaan kirjoittaa myös luokka ”etusivu”, joka ei ole elementtiriippuvainen:

```
.etusivu{
    color: #000FFF;
}
```

Dokumentissa tyylien aliluokka tai luokka otetaan käyttöön class-attribuutilla:

```
<h1 class=”etusivu”>Otsikko 1</h1>
```

Vaihtoehtona class-attribuutin käytölle on id-attribuutti, mutta erona näillä on se, että tietty id-attribuutti voi esiintyä ainoastaan kerran dokumentissa. Yleensä id-attribuuttia käytetään määrittämään yleistä rakennetta (layout), kun taas class-attribuuttia käytetään erottamaan eri sisältöisiä elementtejä esimerkiksi:

```
table.ekasivu{
    color: blue
}
table.tokasivu{
    color: black
}
```

3.2 DIV-Elementit

DIV–elementti on monikäyttöinen kappaletason elementti. Sitä käytetään pääasiassa erottamaan sivun eri osat toisistaan (esimerkiksi navigoinnin ja sisällön erottaminen). DIV–elementillä ei ole itsessään mitään rakenteellista erityismerkitystä, joten sitä voidaan käyttää muista elementeistä luotujen lohkojen tekemiseen. Lisäksi DIV–elementtejä voi olla lukuisia sisäkkäisiä. Yleensä DIV–elementeillä pyritään korvaamaan hitaat ja kankeat TABLE–elementit, mutta esimerkiksi otsikko- ja tekstielementtejä sillä ei kannata korvata.

Esimerkki lohkoelementin käytöstä:

```
#banner{
    width: 800px;
    background-color: #000;
    color: #fff;
}

#content{
    width: 800px;
    background: url(images/bg.gif);
    border: solid thin #000;
}

<div id=”banner”>
    <a href=”index.php”>Linkki etusivulle</a>
</div>

<div id=”content”>
    <p>Tervetuloa Etusivulle</p>
</div>
```

Lopputulos näyttäisi mustan yläpalkin valkoisella tekstillä, jonka alapuolella olisi samanlevyinen sisältölohko, jossa olisi bg–niminen taustakuva ja tervetuloa-teksti.

3.3 Pseudoluokat

Pseudoluokat eli näennäisluokat ovat tyyliluokkien erityistapauksia. Ne ovat luokkia, joita ei ilmoiteta HTML-merkkauksessa, mutta joihin voi viitata vain tietyillä tyylin nimen jälkiliitteillä. Esimerkiksi linkit käyttävät pseudoluokkia, sillä näillä voidaan erottaa vieraillut linkit vierailemattomista.

```
a:link, a:active{
    color: red;
}
a:hover, a:visited{
    color: yellow;
}
```

Esimerkissä linkin väri on muuten punainen, mutta hiiri päällä ja vierailtuna se on keltainen.

3.4 CSS-esimerkki

Ennen tyylitiedostoja kaikki ulkoasua määrittävät tiedot olivat yhdistettynä HTML-tiedostoon esimerkkinä <table>:

```
<table width=”200px” bgcolor:”#000”> </table>
```

Tämän tavan etuna on tietenkin, että tekijä tietää koko ajan minkä näköistä elementtiä hän on nyt muokkaamassa. Haittapuolina on koodin vaikea päivitettävyys ja turha toisto.

Sama CSS tiedostolla:

tyylit.css

```
table{
    width: 200px;
    background-color: #000;
}
```

Nyt kaikki tiedostot, jotka käyttävät tyylit.css tiedostoa, näyttävät kaikki <table> elementtinsä kuin ne olisivat kirjoitettu:

```
<table width=”200px” bgcolor:”#000”> </table>
```

4 Palvelinpuolen ohjelmisto

Palvelinpuolen ohjelmointikielivaihtoehtoja ovat esimerkiksi ASP, PHP ja Java, jotka toimivat joko WWW-palvelimella tai sovelluspalvelimella. Palvelinpuolen ohjelmoinnin tarkoitus on luoda dynaamisuutta WWW-sivuille.

4.1 PHP

PHP on Rasmus Lerdorfin vuonna 1994 kehittämä palvelinpuolen skriptikieli, jonka uusin versio on versio PHP5. PHP on kieli, jota käytetään yleisimmin HTML/XHTML:n yhteydessä luomaan dynaamisia WWW-sivuja. PHP-koodi upotetaan HTML-koodin sekaan ja tulkataan palvelimella ja tämän jälkeen ladataan käyttäjän selaimelle. Käytännössä tämä tarkoittaa sitä, että kun selain pyytää palvelimelta PHP-koodia sisältävän sivun, se antaa sivun suoritettavaksi palvelimelle asennetulle PHP-tulkille. PHP-tulkki tutkii dokumentin rakenteen, suorittaa komennot ja palauttaa työn tuloksena syntyneen staattisen dokumentin palvelimelle, joka palauttaa sen lopulta takaisin selaimelle. Palautetut PHP-sivut ovat lopputuloksena kuin mitkä tahansa perus- HTML-sivut. PHP:n syntaksi perustuu C-kieleen, mutta sisältää myös piirteitä Perl-, Java ja C++ kielistä. PHP on avointa lähdekoodia ja se on saatavilla vapaasti internetissä.

PHP-esimerkki

Esimerkki PHP-koodista upotettuna HTML-koodiin:

```
<html>
    <head>
          <title>Esimerkkisivu</title>
    </head>
    <body>
           <?php echo ”<p>Sisältöä tähän</p>” ?>
    </body>
</html>
```

Esimerkki PHP-muuttujan upottaminen HTML-koodiin:

```
<?php
$otsikko = “Esimerkkisivu”;
$text = “Sisältöä tähän”;
?>
<html>
    <head>
          <title><?php echo $otsikko; ?></title>
    </head>
    <body>
          <p><?php echo $text; ?> </p>
    </body>
</html>
```

Sama lopputulos, mutta kaikki nyt PHP-koodina:

```
<?php
$body = ”
<html>
    <head>
          <title>Esimerkkisivu</title>
    </head>
    <body>
          <p>Sisältöä tähän</p>
    </body>
 </html>
 ”;
print $body;
?>
```

4.2 SQL-relaatiotietokanta

"Structured Query Language on IBM:n aluperin kehittämä kyselykieli, jolla relaatiokantaan voi tehdä erilaisia hakuja, muutoksia, sekä lisäyksiä" (Wikipedia). SQL tietokantoja on saatavana niin ilmaisia kuin maksullisia. Ilmaisista suosituin on MySQL-relaatiotietokanta, jota kehittää ruotsalainen MySQL AB yritys. Tietokantoja käytetään tietojen helppoon tallentamiseen ja nopeaan hakemiseen.

4.2.1 MySQL

MySQL:n on kehittänyt suomalainen Michael Widenius, sekä ruotsalainen David Axmark vuonna 1995. Sitä on saatavilla vapaalla GNU GPL -lisenssillä sekä kaupallisella lisenssillä ja uusin versio on tällä hetkellä 5.0.

MySQL:ää käytetään yleensä yhdessä Linux käyttöjärjestelmällä, Apache palvelimella, sekä PHP kielellä rakennetulla ohjelmalogiikalla. Tätä yhdistelmää kutsutaan yleensä LAMP-alustaksi. Myös muilla ohjelmointikielillä on mahdollista käyttää MySQL:ää. Merkittävimmät käyttäjät MySQL tietokannoilla ovat Google, Wikipedia ja Yahoo.

4.2.2 SQL-esimerkki

Tietokannan käyttö perustuu tauluihin, jotka sisältävät ennalta määritellyn tyyppistä dataa. Taulut indeksoidaan hakujen nopeuttamiseksi määrittelemällä näille yksilöllinen tunnus. Myös ristiviittauksen taulujen välillä ovat mahdollisia. Yksinkertaisimmillaan tietokantaan tehdään lisäys/poisto/muokkaus/haku-toimintoja. MySQL:ää voidaan hallita komentokehotteesta tai helpommin suositulla PhpMyAdmin-työkalulla.

Taulu luodaan komennolla CREATE-komennolla:

```
CREATE TABLE ”taulun nimi”(
 arvo int(10) NOT NULL,
primary key(arvo)
);
```

Tauluun lisätään tieto INSERT-komennolla:

```
INSERT INTO ”taulun nimi” (sarake)VALUES(arvo);
```

Taulusta poistetaan tieto DELETE-komennolla:

```
DELETE FROM ”taulun nimi” WHERE arvo = 1;
```

Taulua sisältöä voidaan päivittää UPDATE-komennolla:

```
UPDATE ”taulun nimi” SET arvo = 2 WHERE arvo = 1;
```

Taulun rakennetta voidaan muokata ALTER-komennolla:

```
ALTER ”taulun nimi” ADD arvo2 int(10) NOT NULL;
          tai
ALTER ”taulun nimi” DROP arvo;
```

Taulu tuhotaan DROP-komennolla

```
DROP ”taulun nimi”;
```

SQL sisältää lukuisia eri komentoja mutta näillä viidellä (create, insert, delete, update, alter) peruskäytössä pärjää jo pitkälle. WWW-sovelluksessa tietokantaan muodostetaan yhteys PHP:n avulla, jonka avulla myös SQL-kyselyt annetaan.

Esimerkki tietokannan käytöstä PHP:n avulla:

Yhteys:

```
1. $lnk =
mysql_connect(”$server”,”$user_name”,”$password”);
2. mysql_select_db($database, $lnk);
```

Kysely:

```
3. $query = mysql_query(“SELECT * FROM testi”);
4. $results = mysql_fetch_assoc($query);
```

1. Valitaan serveri ja annetaan sille käyttäjätunnus sekä salasana.
2. Valitaan yhteydelle käytettävä tietokanta.
3. Haetaan kaikki tiedot ”testi”-taulusta.
4. Haetaan results-tauluun tulokset, jossa avaimena toimii taulun sarakkeen nimi.

5 Selaintekniikat

5.1 DHTML

Microsoftin ja Netscapen julkistaessa selaimistaan neljännen version verkkosovellusten kehittäjille tarjottiin uusi tekniikka: dynaaminen HTML (DHTML). DHTML ei ole kuitenkaan W3C:n hyväksymä standardi. Todellisuudessa dynaamiseksi HTML:ksi tai DHTML:ksi kutsutaan kokoelmaa tekniikoita, joilla luodaan WWW-sivuille dynaamista tai interaktiivista toimintaa ja efektejä. Yleensä tämä toteutetaan yhdistämällä staattiset HTML-sivut, asiakaspuolen skriptikieli JavaScript, CSS-tyylimääreet, sekä DOM asiakaspuolen skriptikielellä. Yleisimmin DHTML:n avulla esitetään näyttäviä valikkoja.

5.2 DOM

Document Object Model määrittelee dokumentissa olevat elementit ja näiden välisen tiedonvälityksen sekä kuinka näihin elementteihin voidaan viitata. Tämä on mahdollista kuvaamalla dokumentin elementit hierarkisen puumallin mukaisena (XML tai HTML).

DOM-oliomalli on hierarkinen dokumenttipuu, joka koostuu rajoittamattomasta määrästä solmuja, joista jokainen on oma yksittäinen olionsa hierarkiassa. Dokumentti-olio (Document) on hierarkiassa ylimmällä tasolla. Hierarkinen puu rakentuu xml:n tapaan siten, että jokaisella solmulla on yksi ainoa juurisolmu (root), sekä ainakin äitisolmu (parent). Solmulla voi olla yksi tai useampi sisar- tai lapsisolmu.

DOM-mallin etu on se, että puussa voidaan liikkua vapaasti ja jokaiseen solmuun voidaan viitata itsenäisenä oliona, jolla on omat ominaisuutensa ja metodinsa.

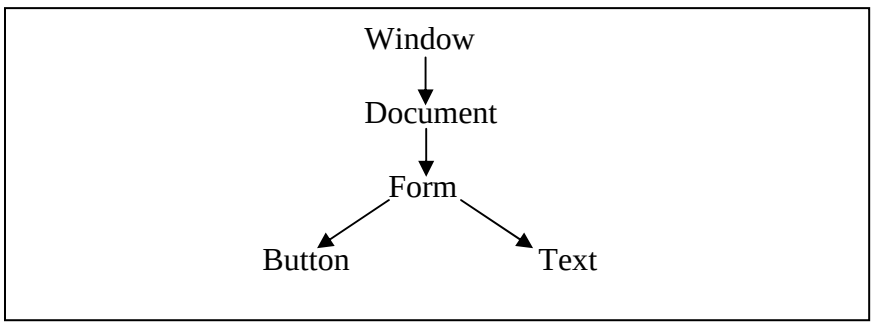
5.3 JavaScript

JavaScript on internetin suosituin komentosarjakieli ja sitä käytetään yleensä luomaan näppäriä käyttöliittymiä verkkosivuille. JavaScript kehitettiin alun perin helpottamaan HTML-kielen tagien dynaamista muokkaamista, jotta verkkosivut tuottaisivat käyttäjille aiempaa paremman käyttökokemuksen (Ryan Asleson, Nathaniel Schutta: 6). JavaScriptin toimintaidea on luoda dynaamista toiminnallisuutta sivuille, mutta sitä ei pidä kuitenkaan sekoittaa dynaamiseen sisällön tuottamiseen palvelinpuolella. JavaScript lataa tarvitsemansa tiedot selaimeen ja näin käyttäjä voi kokea interaktiivista toimintaa ilman, että palvelimeen otetaan välillä yhteyttä. JavaScript on kärsinyt tuen puutteesta koko olemassa olo aikansa ja varsinkin sen alkutaipaleella oli selaintuki erittäin puutteellista. Nykyäänkin JavaScriptin suorituksen voi käyttäjä sulkea selaimestaan näin halutessaan.

JavaScriptiä voi käyttää sovelluksessaan monella tapaa. Yksinkertaisin niistä on asettaa suoraan WWW-sivun objektille jokin JavaScript Event -käsittelijä, joka aktivoiduttuaan suorittaa jonkin sille annetun tehtävän. Eri objekteilla on eri Event-käsittelijöitä ja tässä listassa muutama yleisin:

Objekti	**Event-käsittelijä**	**Käsky, joka suoritetaan**
Button	onClick	painettaessa
Form	onSubmit	lähetettäessä
Link	onMouseOver	hiiren ollessa päällä
Select	onBlur	deaktivoituessa
Text	onChange	tekstin muuttuessa

WWW-sivun objekteja hallitaan JavaScriptin avulla Window-luokasta periytyneen Document-objektin avulla.



Eri objektit erotetaan toisistaan joko ID:n tai nimen perusteella. Esimerkiksi:

```
<input type='text' id='esim1' value='ok'/>
```

käsiteltäisiin:

```
document.getElementById('esim1')
```

Jos kyseisen kentän arvoa haluttaisiin muuttaa, tehtäisiin se seuraavasti:

```
<button
onClick="document.getElementById('esim1').value='canc
el'"/>
```

JavaScriptit ovat yleensä kuitenkin suurempia kokonaisuuksia, kuin vain nappeihin yhdistettyjä Eventtejä. Toinen tapa onkin kirjoittaa täysin oma JavaScript-lohko WWW-sivun sisään tyyliin:

```
<script type="text/javascript">
function startSivu() {
var teksti = "Hello World";
document.write = teksti;
}
window.onDomReady(startSivu);
</script>
```

Tässä esimerkissä tulostetaan ruudulle "Hello World" -teksti, kun sivu on ladattu. <script> tageilla ilmoitetaan HTML-sivulle, että kyseessä on skripti ja "text/javascript" tarkoittaa, että nimenomaan JavaScript.

Kolmas vaihtoehto JavaScriptin käytölle on ulkoisen JavaScript-tiedoston käyttäminen (.js -pääte). Tehtäessä suurempia JavaScript sovelluksia tai moduuleita on koodin uudelleen käytettävyyden kannalta järkevää tehdä siitä oma erillinen pakettinsa. Paketti kutsutaan HTML-sivun <head>-osiossa jolloin se toimii kaikkialla sivulla:

```
 <script src="scripts/javaEsim.js"
 type="text/javascript"></script>
```

Sama toiminto voidaan toteuttaa <body>-osiossa vastaavalla komennolla, jolloin toteutus on kirjoitettava <script>-tagien sisään.

Työssäni JavaScriptiä käytettiin ainoastaan menulinkkien esittämiseen, mutta saman lopputuloksen olisi voinut toteuttaa myös CSS-tyylitiedostolla.

5.4 AJAX

”AJAX on kokoelma tekniikoita, joiden ideana on vuorovaikutteisen verkkosovelluksen luominen” (Wikipedia). AJAX koostuu seuraavista tekniikoista:

- XHTML(tai HTML) informaation merkitsemiseen.
- CSS informaation muotoiluun.
- DOM asiakaspuolen skriptikieltä, dynaamisen informaation esittämiseen ja vuorovaikutukseen.
- XMLHttpRequest-objekti joka vaihtaa informaatiota asynkronisti verkkopalvelimen kanssa.
- XML:ää käytetään yleensä siirtämään dataa takaisin palvelimelta.

AJAX ei ole itsenäinen teknologia, vaan kuten DHTML, termi viittaa useiden teknologioiden käyttöön yhdessä. AJAX:in paras ominaisuus on mahdollisuus päivittää vain osa WWW-sivusta, jolloin säästetään huomattavasti resursseja. Toinen hyvä ominaisuus on mahdollisuus luoda ajastettu kutsujen virta, joka päivittää esimerkiksi sivun kerran sekunnissa ja näin reaaliaikaisen tiedon esittäminen on helpompaa, ilman että käyttäjän tarvitsee käyttää hakutoimintoa. Tunnetuin Ajaxin hyödyntäjä on varmasti Google, jonka Google Maps, Google Suggest ja Gmail on kaikille selaajalle tuttuja välineitä. Vaikka Ajaxissa ei olekaan varsinaisesti mitään uutta, on se merkittävä kehitysaskel internetin kehityksessä. ”Verkkosovellusten kehittäjät voivat vapaasti suunnitella sovellukset toimimaan asynkronisesti, jolloin monet sellaisista toiminnoista, jotka olivat aiemmin mahdollisia vain työpöytäohjelmistoissa, voidaan nyt tuoda helposti myös verkkosovelluksiin” (Ryan Asleson, Nathaniel Schutta 2007: 15)

Ajaxin tärkein objekti eli XMLHttpRequest ei ole W3C:n hyväksymä standardi, joten Internet Explorer ja muut selaimet käsittelevät sitä hieman toisistaan poikkeavasti. Internet Explorer toteuttaa sen ActiveX-komponenttina ja muut selaimet toteuttavat sen sisäisenä JavaScript-oliona. Tämä johtaa siihen, että jos haluaa tukea mahdollisimman montaa selainta, on tehtävä tarkistus käytetyn selaimen tyypistä. Onneksi Ajaxin kohdalla siitä selviää loppujen lopuksi pienellä määrällä koodia:

```
1.var xmlHttp:

2.function createXMLHttpRequest(){
3.       if(window.ActiveXObject){
4.       xmlHttp = new
ActiveXObject(”Microsoft.XMLHTTP”);
}
5.       else if(window.XMLHttpRequest){
6.       xmlHttp = new XMLHttpRequest();
}
}
```

1. Luodaan muuttuja xmlHttp.
2. Luodaan funktio createXMLHttpRequest, joka täytyy käynnistää tietenkin jossain kohtaa sivua.
3. Tarkistetaan löytyykö selaimesta ActiveX-objekti
4. Tehdään uusi ActiveX-olio.
5. Jos ActiveX-objektia ei löytynyt, tarkistetaan löytyykö XMLHttpRequest:ia.
6. Jos ActiveX-objekti löytyy tehdään siitä normaali JavaScript-olio.

Tämä ei tietenkään yksistään riitä toteuttamaan käyttäjän komentoja, vaan tarvitaan muitakin funktioita ottamaan yhteys esimerkiksi tietokantaan.

Esimerkki tapahtumankäsittelystä (Ryan Asleson, Nathaniel Schutta:30):

1. Luodaan asiakaspuolen tapahtumankäsittelijä, joka reagoi toimintaan:

```
<input type=”text” id=”email” name”email”
onblur=”validateEmail();”/>
```

2. Luodaan XMLHttpRequest-ilmentymä. Palvelimeen otetaan yhteys open()-metodia käyttämällä. Kutsun yhteydessä metodin URL-muuttujaan sijoitetaan haluttu resurssi ja samalla ilmoitetaan haluttu lähetystapa (normaalisti GET tai POST). Pyyntö toteutetaan send()-metodin avulla:

```
var xmlHttp:

function validateEmail(){
var email = document.getElementById(“email”);
var url = “validate?email=” +escape(email.value);

if(window.ActiveXObject){
xmlHttp = new ActiveXObject(”Microsoft.XMLHTTP”);
}
else if(window.XMLHttpRequest){
xmlHttp = new XMLHttpRequest();
}
xmlHttp.open(”GET”, url);
xmlHttp.onreadystatechange = callback;
xmlHttp.send(null);
}
```

3. Palvelimelle tehtävä pyyntö suoritetaan. Pyyntö saattaa kutsua palvelinsovelmaa, CGI-skriptiä tai jotain muuta palvelinpuolen tekniikkaa.
4. Palvelin suorittaa halutun tehtävän, tekee esimerkiksi tietokantahaun tai ohjaa pyynnön edelleen johonkin toiseen järjestelmään.
5. Selaimelle palautetaan vastaus. Content-Type on asetettu arvoon text/xml. XMLRequest voi käsitellä ainoastaan text/xml-tyyppisiä tuloksia.
6. Esimerkissä on määritelty kutsumaan callback()-funktiota, kun metodin suoritus päättyy. Funktio tarkistaa XMLRequest-objektin readyState ominaisuuden tilan ja tarkistaa seuraavaksi palvelimen tilakoodin. Jos kaikki on suorituksen kannalta kunnossa, callback()- funktio suorittaa asiakaspuolella jotain. Tyypillinen callback()-funtio näyttää seuraavalta:

```
function callback(){
if(xmlHttp.readystate == 4){
 if(xmlHttp.status == 200){
 //kaikki on suorituksen kannalta kunnossa
 //suoritetaan jotain kivaa.}
}
}
```

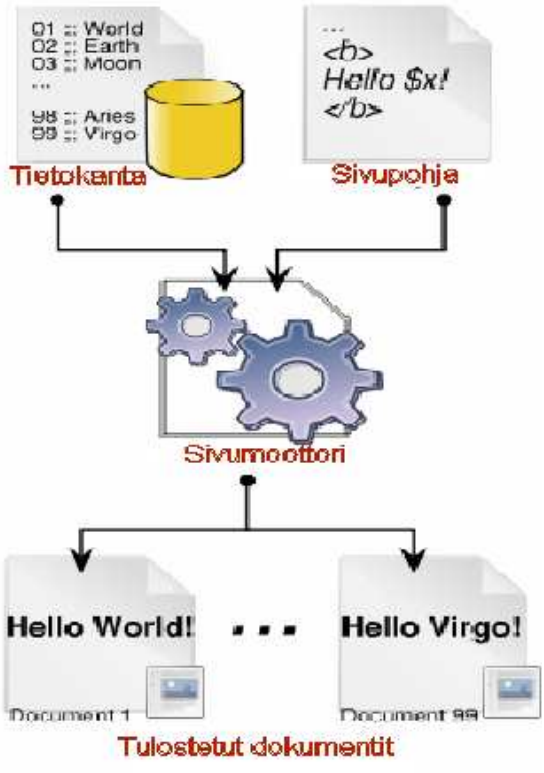
Seuraavassa lista Ajaxin perustekniikoista, joita sillä on mahdollista toteuttaa:

- tiedon validointi
- dynaamisesti täydentyvät valintalaatikot
- automaattisesti päivittyvän sivun luominen
- edistymispalkin esittäminen
- työkaluvihjeiden luominen
- verkkosivun dynaaminen päivitys
- web services -palveluiden käyttö
- automaattisen täydennyksen rakentaminen

Hyviä WWW-sivustoja, joissa pääsee nopeasti perille AJAX:in tarjoamista mahdollisuuksista, ovat www.ajaxrain.com ja www.ajaxdaddy.com. Näillä sivuilla kerätään demokokoelmaa erilaisista AJAX-tekniikkaa hyödyntävistä ”vempaimista”, joita käyttäjät voivat testata ja joissakin tapauksissa käyttää myös omilla sivuillaan. Sivustoista www.ajaxrain.com on huomattavasti laajempi ja saa uutta sisältöä lähes päivittäin. Tämä sivusto on ehdottomasti tutustumisen arvoinen kaikille AJAX-tekniikasta innostuneille.

6 Sivupohjajärjestelmä

Sivupohjajärjestelmä on nimitys järjestelmälle, jossa pyritään erottamaan verkkosivujen ulkoasun tuottavat sivut (esityslogiikka) sivujen sisällöstä vastaavista sivuista (toimintalogiikka). Sivupohjajärjestelmän sydän on sivumoottori, jonka tarkoitus on helpottaa sivujen hallintaa siten, että ulkoasua muutettaessa ei tarvitsisi osata taustalla olevaa dynaamista sisältöä tuottavaa kieltä ollenkaan (Kuva 1). Sivumoottorin voi jokainen ohjelmointitaitoinen tehdä myös itse, mutta saatavilla on myös runsaasti vapaasti käytettävissä olevia moottoreita. Näistä käsittelen lyhyesti kahta hieman toisistaan poikkeavaa koulukuntaa edustavaa sivupohjajärjestelmää, joista toisen valitsin oman sivustoni järjestelmäksi.



Kuva 1. Sivupohjajärjestelmän toiminta (Wikipedia)

Sivupohjajärjestelmä toimii siten, että luodaan sivupohja johon haetaan tietokannasta sisältö sivumoottorin avulla. Lopputulos on validia XHTML-koodia, mutta sisältö on tuotettu dynaamisesti. Tietoja lisäävän henkilön ei tarvitse siis osata HTML-koodia lainkaan voidakseen muokata tulostettavaa näkymää. ”Ulkoasu ja sovelluksen logiikka on järkevää erottaa toisistaan, sillä vaikka ulkoasu tehdään aina jollekin tietylle sovellukselle, ei kannata sitoa ulkoasua tiukasti yhteen sovelluksen logiikan ja sisällön kanssa” (Rami Heinisuoa, Ilkka Rauta:387).

Suurin hyöty sivupohjajärjestelmästä on, kun sivuja päivitetään useasti mutta rakenne pysyy aina samana, esimerkkinä uutispalvelut. Myös lokalisointi on huomattavasti helpompaa, kun käytetään Sivupohjajärjestelmää, sillä uuden kielen lisäämistä ei tarvitse koodata ollenkaan varsinaisiin sivutiedostoihin. Sivupohjajärjestelmää voidaan verrata XSL-transformaatioon, joka tehdään XML-tiedostolle, jotta sama sisältö voidaan esittää eri alustoilla niille sopivassa muodossa.

Sivupohjajärjestelmistä on olemassa hyvin vähän kirjallisuutta varsinkaan suomeksi, joten suurin osa oppimisesta on tapahtunut ”yritys ja erehdys”-menetelmällä. Rami Heinisuon ja Ilkka Raudan 2007 kirjoittama ”PHP ja MySQL Tietokantapohjaiset verkkopalvelut” -kirja sivuaa aihetta Smarty-sivupohjajärjestelmän kannalta.

6.1 Smarty

Smarty on yksi netin suosituimmista sivupohjajärjestelmistä. Se on erityisesti tarkoitettu erottamaan HTML-koodi muusta koodista ja tässä mielessä se toteuttaa ulkoasun ja sisällön erottamisen kirjaimellisesti. Smarty poikkeaa useista muista sivupohjajärjestelmistä siten, että Smarty tarkastaa aina sivupohjaa lukiessaan onko siitä saatavilla tuore käännetty versio. Jos tällaista ei löydy, Smarty kääntää sivupohjan puhtaaksi PHP-koodiksi ja sijoittaa koodin käännetyt-hakemistoon. Jos tuore käännetty versio taas löytyy, Smarty käynnistää sen. Näin sivumoottori toimii erittäin nopeasti, koska PHP:n ei tarvitse suorittaa lukuisia etsi-korvaa-toimintoja sijoittaessaan sisältöä muuttujiin. Myös ohjausrakenteet (toistot, ehtorakenteet) kääntyvät optimoiduksi PHP-koodiksi. Huonoina puolina pidetään sen massiivisuutta, monimutkaisuutta ja hitautta. Smarty käyttää omia funtioitaan ja metodejaan tulostukseen ja tietojen läpikäymiseen. Smarty-tagit ovat oletusarvoisesti { ja } –merkkien sisällä olevia merkintöjä, joten se eroaa selkeästi PHP- ja HTML-syntaksista.

Smartyn toimintalogiikka (esimerkki.php)

```
<?php
// esimerkki.php
define('SMARTY_DIR', 'smarty-2.6.9/' );
require_once(SMARTY_DIR . 'Smarty.class.php');

// luodaan Smarty-olio ja alustetaan se
$smarty = new Smarty;
$smarty->config_dir = SMARTY_DIR;
$smarty->template_dir = './template';
$smarty->compile_dir = './template/compiled';
$smarty->compile_check = TRUE;
$smarty->debugging = FALSE;

// Siirretään muuttujiin haluttu sisältö
$smarty->assign('title', 'Smartyllä
esimerkkiotsikko!');
$smarty->assign('body', 'Sisältöä tähän');

// Näytetään malline
$smarty->display('esimerkki.tpl');
?>
```

Smarty esityslogiikka (esimerkki.tpl):

```
<html>
<head>
    <title>{$title}</title>
</head>
<body>
    <p>{$body}</p>
</body>
</html>
```

6.2 PHP Savant3

Savant on kehitetty vaihtoehdoksi Smartylle ja suurimpina erona näiden kahden välillä on, että Savant ei käännä sivupohjia, ei sisällä välimuistitusta (caching), sekä sisältää puhdasta PHP-koodia sivupohjissa. Syy miksi PHP-koodi sallitaan pohjissa, on tekijöiden mielestä se, että koska PHP on jo itsessään mallinnekieli C:stä, on turhaa tehdä toinen mallinnekieli mallinnekielen sisälle Smartyn tapaan. Sen sijaan Savant käyttää hyödykseen kaikkia PHP:n ominaisuuksia ja on tästä syystä nopea ja tehokas sivupohjajärjestelmä.

Savant3 toimintalogiikka (esimerkki.php)

```
<?php
// Ladataan Savant3 luokka ja luodaan siitä instanssi
require_once 'Savant3.php';
$tpl =& new Savant3();

// Luodaan otsikko.
$title = "Savant esimerkkiotsikko";

// Sisältö
$body = ’sisältöä tähän’;

// siirretään haluttu sisältö muuttujiin.
$tpl->assign('title', $title);
$tpl->assign('body', $body);

// Näytetään malline
$tpl->display('esimerkki.tpl.php');
?>
```

Savant3 esityslogiikka (esimerkki.tpl.php)

```
<html>
    <head>
        <title><?php $this->eprint($this-
>title)?></title>
    </head>
    <body>
        <p>$this->eprint($this->body) ?></p>
    </body>
</html>
```

On mielipidekysymys kummasta tavasta pitää enemmän. Monen mielestä Smarty on ainoa oikea vaihtoehto nimenomaan siksi, että se erottaa PHP-komennot täysin HTML-syntaksista. Smartyllä saadaankin aikaan selkeä ero ohjaustoimintojen, rakenteen ja ulkoasun suhteen. Kuitenkin Smartyn opettelu vaatii aikansa, koska se sisältää omat loogiset- ja ohjausoperaattorinsa.

Savantin parhaana puolena voidaankin nähdä matala aloituskynnys. Kaikki PHP-käskyt ovat mukana Savantissa, jolloin myös kaikki PHP:n hienoudet ovat käytettävissä. Huonona puolena sivupohja sisältää edelleen PHP-tageja ja lopputulos ei näytä läheskään yhtä siistiltä kuin Smartyssa. Kuitenkin kaikki toimintalogiikka on oltava PHP-tiedostossa, joten toimintojen muokkaaminen on joka tapauksessa selkeää.

Käytetään sivupohjajärjestelmänä mitä tahansa valmista järjestelmää, tarjoaa sen käyttö kuitenkin suuria etuja normaaliin WWW-sivujen rakentamiseen verrattuna. Vaikka tiedostojen koodirivien määrä ei sivupohjajärjestelmää käyttämällä vähene, voi sivupohjia työstää henkilö, jolla ei ole niinkään syvällistä tuntemusta ohjelmoinnista. Lisäksi sivupohjat helpottavat myös koodin uudelleenkäyttämistä, koska yhtä sivupohjaa on mahdollista käyttää useasta paikasta käsin. Lisätietoa aiheesta löytyy järjestelmien omilta WWW-sivuilta, mutta kannattaa suhtautua varauksella sivujen sisältämiin markkinointikehuihin ja tutustua tuotteisiin itse ja tehdä niistä omat johtopäätöksensä.

7 Koulutussivustot

Internet-pohjaisia koulutussivustoja on tietenkin ollut jo olemassa ennen tätä sivustoa ja monet niistä perustuvat valmiisiin oppimisalustoihin. Esimerkiksi ”Moodle”, joka myös toimii SQL/PHP periaatteella, on monien oppilaitosten käyttämä oppimisalusta. Muita vastaavia ovat mm. ”Opintoverkko” sekä ”Verkkosalkku”.

7.1 Määrittely

”Opetusalusta eli verkko-oppimisympäristö on palvelinohjelmisto, jolla tuetaan johonkin pedagogiseen ajatukseen pohjautuvien oppimisprosessien hallinta. Ohjelma sisältää opettajalle työvälineitä opintokokonaisuuksien toteuttamiseksi ja hallinnoimiseksi verkossa ja oppilaalle työvälineitä opintoihin osallistumiseksi”. (Wikipedia)

Toisen tyyppinen ratkaisu monipuoliseen koulutuskäyttöön on erilaisten Wiki-sivustojen käyttäminen. Wikipedia on hyvä esimerkki sivustosta, jonka sisältö perustuu käyttäjien aktiivisuuteen ja josta monet hakevat tarvitsemaansa tietoa. Koulutuksen yksi alue eli itse tekeminen on kuitenkin Wikissä vaikeaa.

7.2 Inspecta Oy:n koulutussivusto

Inspecta Oy:n koulutussivustolla on kolme tärkeää tavoitetta:

- Saada tietoa eri paikkakuntien koulutustarpeista.
- Tarjota ajasta ja paikasta riippumaton mahdollisuus kouluttaa itseään.
- Saada selville yksittäisten henkilöiden tieto- sekä taitotaso.

Inspecta Testingin piiriin kuuluu neljä eri tarkastusalaa: silmämääräinen tarkastus, röntgen-, ultraääni- sekä pintatarkastus. Jokaisella alalla on oma koulutuksesta vastaava ylimmän tason tarkastajansa. Työntekijöistä jokainen osaa yhtä tai useampaa tarkastusalaa, joten koulutussivuston käyttäjäkuntaa ovat kaikki Inspecta Oy:n tarkastajat.

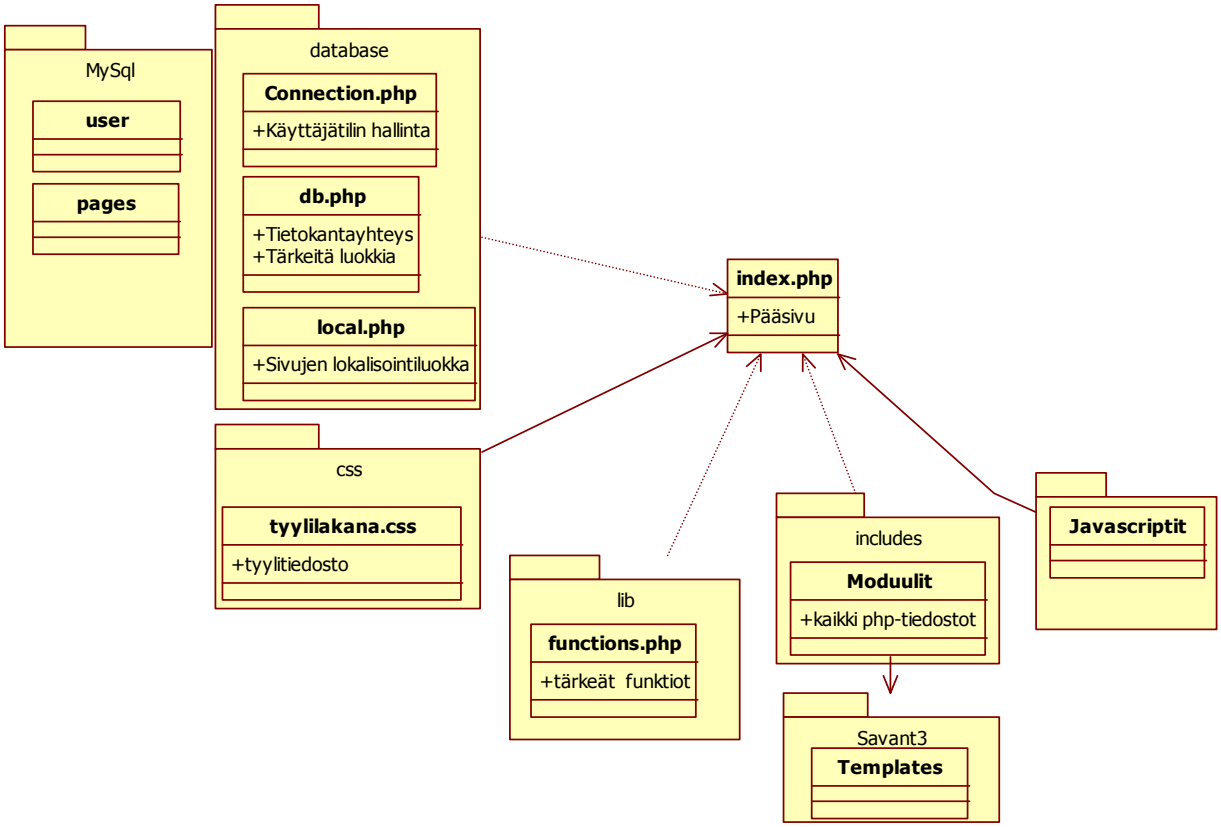
Tarkastajien koulutus on insinööritason korkeakoulututkinto, mutta keski-ikä on kuitenkin lähempänä 40:tä kuin 30:tä. Tietotekniset taidot ovat hyvin vaihtelevat, mutta kaikki joutuvat työssään käyttämään tietokonetta jossain määrin. Internet on jokaiselle tuttu työkalu.

Koska sivusto on tarkoitettu täsmälleen yrityksen omien tarpeiden täyttämiseen, on sivuston ulkoasun hyvä noudattaa yritykselle tuttuja periaatteita. Yrityksen Intranet uudistuksen myötä koulutussivusto päätettiin sijoittaa Intranetin yhteyteen.

Sivuston tarkat sisältövaatimukset olivat tärkein syy, miksi sivustoa haluttiin rakentaa itse, eikä valita valmista sivustopohjaa. Jokaisella tarkastusalalla on sen verran erityinen tenttinsä, että oikeastaan koko sivusto rakentuu näiden tenttien ympärille. Sivuston toiminta perustuu MySQL-relaatiotietokantaan, joka sisältää käyttäjätietojen lisäksi testien ja tenttien tiedot. Lisäksi tenttivastaukset tuloksineen tallennetaan kantaan.

Käyttäjän näkökulmasta sivut ovat kuin mitkä tahansa websivut, jossa linkkejä valitsemalla pääsee eteenpäin. Sivut on toteutettu PHP-tekniikalla, joka hakee dynaamisesti sisällön tietokannasta tarpeen mukaan ja tallentaa käyttäjän lähettämää dataa tietokantaan. Hyvän käyttöliittymäsuunnittelun mukaisesti käyttäjälle tarjotaan mahdollisimman selkeä kuva siitä, mitä hän on tekemässä ja mihin hänen toimintansa johtavat. Linkkitasoja ei sivustolla ole missään vaiheessa kahta enempää: tarkastusalan valinta ja tekemisen kohde. Tämän ansiosta ei käyttäjä pääse eksymään sivustolla, vaan tietää aina mitä on tekemässä.

Rakenteellisesti sivusto on toteutettu mahdollisimman helpoksi ymmärtää. Laajoja kokonaisuuksia on pilkottu pieniksi palasiksi, joita sitten tarpeen tullen liitetään rakenteeseen. Tällä tavalla palasia voidaan helposti muokata tai korvata uusilla ja uusia osia on helppo tehdä lisää (kuva 2).



Kuva 2. Rakennekuvaus

Käytännössä sivut toimivat siten, että index-sivu toimii alustana kaikelle toiminnalle ja hakee sisällön tietokannasta käyttäjän painamien linkkien mukaan. Jokaiselle sivulle on tietokannassa taulu, jonka perusteella sisältö valitaan näytettäväksi sivulle. Esimerkiksi etusivulla voi olla tieto: header.php, main.php, footer.php. Näillä tiedoilla index.php osaa sisällyttää oikeat moduulit includes-kansiosta, joka taas erottaa sisällön ulkoasusta vielä kertaalleen PHPSavant3 sivupohjamoottorin avulla.

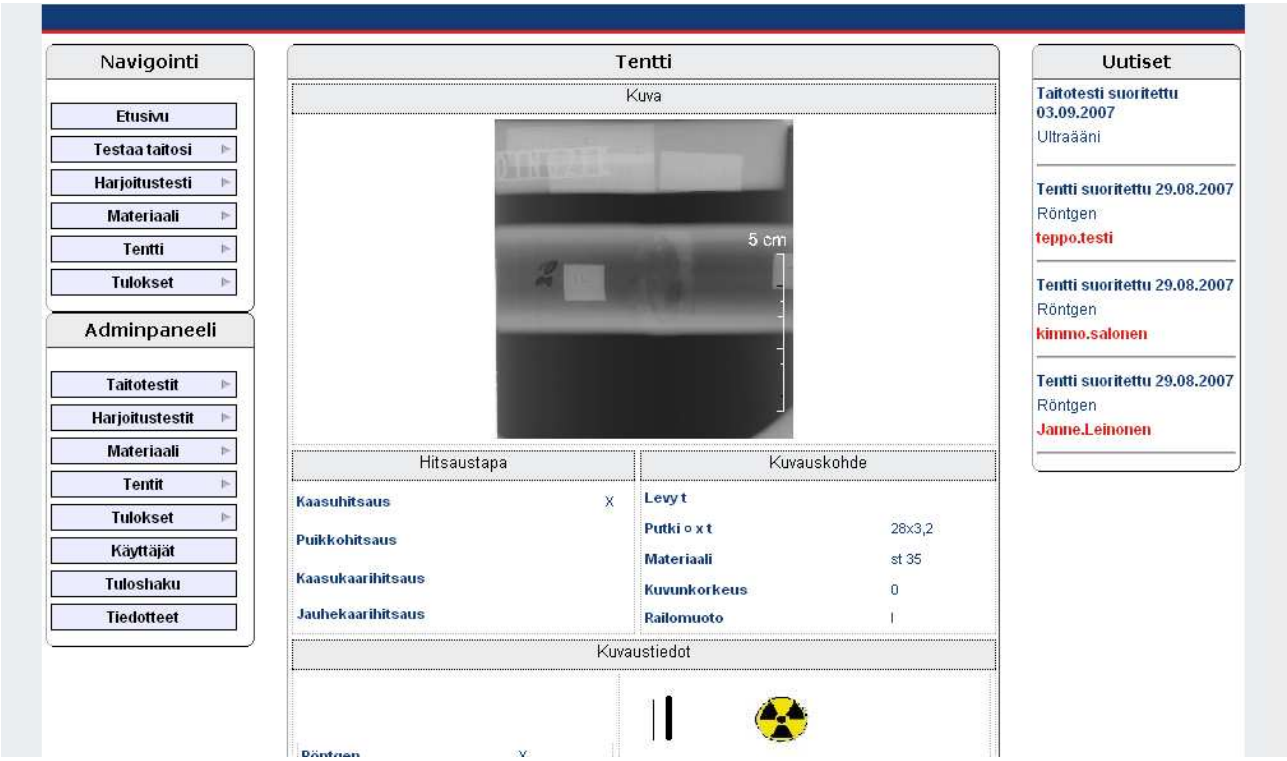
8 Toteutusprosessi

8.1 Kehitys

Projektissa toimi itseni lisäksi harjoittelun vastaavani, sekä idean isä, röntgenpuolen ylitarkastaja. Ylitarkastaja toimitti sisältöä ja ohjasi rakennetta oikeaan suuntaan. Itse olin vastuussa kaikesta ohjelmoinnista, rakenteesta sekä ulkoasusta. Harjoittelun vastaavani toimi yhdyshenkilönä eri sidoshenkilöihin. Toukokuussa ryhmään tuli ultraäänitarkastuksen ylitarkastaja tuomaan omia näkemyksiään projektiin. Tuotetta lähdettiin kehittämään kokeile ja erehdy
-menetelmällä, sillä kokemuksia WWW-sovellusten teosta oli hyvin vähän ja tarkoitus oli vain kokeilla taipuuko media haluttuihin tuloksiin.

8.1.1 Ensimmäinen kehitysaste

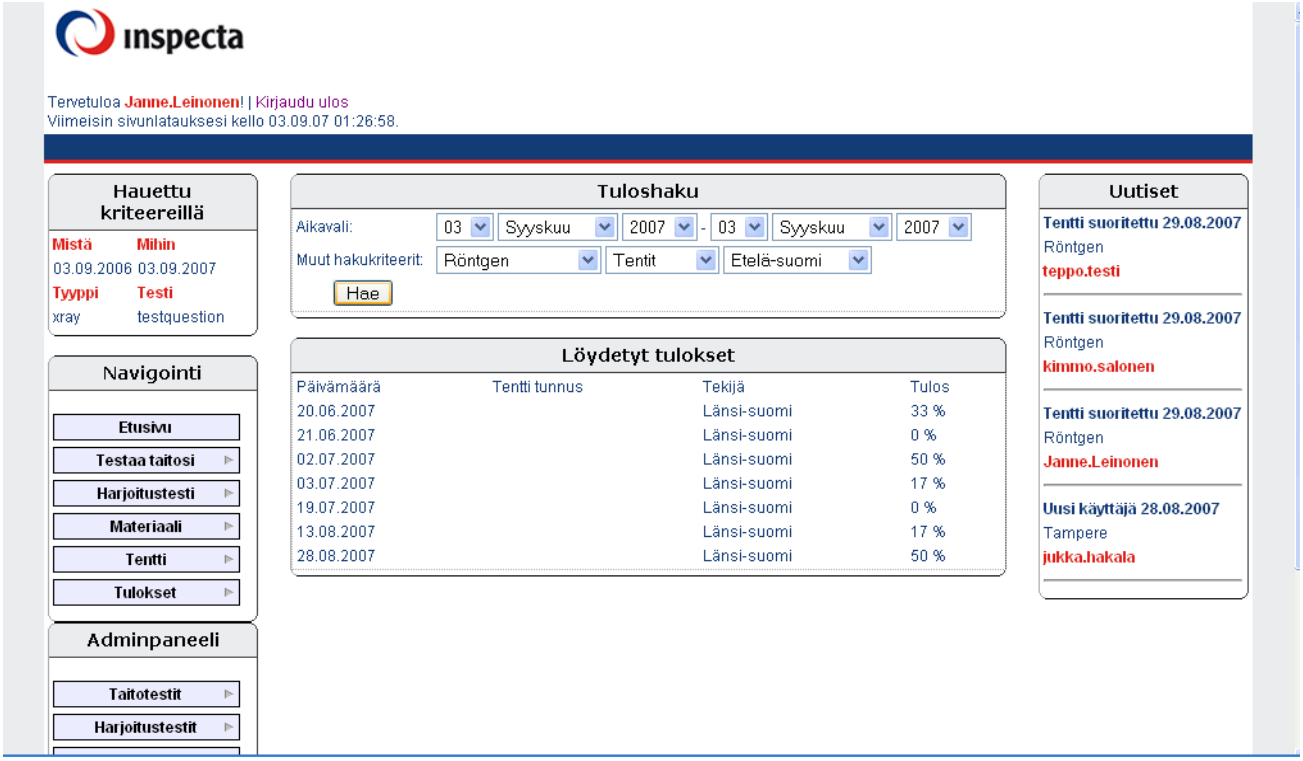
Taitotasoani alussa kuvaa se, että osasin sekä SQL:n, että PHP:n perusteet. Aluksi lähdimme toteuttamaan sivuja XML-pohjaisena, koska XML oli minulle tutumpi kieli. Mutta juuri kuin saimme tämän toteutuksen valmiiksi, innostui toinen tarkastaja projektista ja lähdimme toteuttamaan hänen visioitaan laajemmasta sivustosta. Tässä vaiheessa oli luovuttava XML:stä ja siirryttävä tukevampaan SQL-tietokantaan. Toimintaperiaatteiltaan ensimmäisen kehitysasteen tenttiosio pystyttiin kuitenkin sisällyttämään uuteen laajempaan rakenteeseen yllättävän kivuttomasti (Kuva 3). Tässä vaiheessa sivusto käsitti lomakkeen, joka haki testin tietokannasta ja vertasi käyttäjän vastauksia oikeisiin vastauksiin.



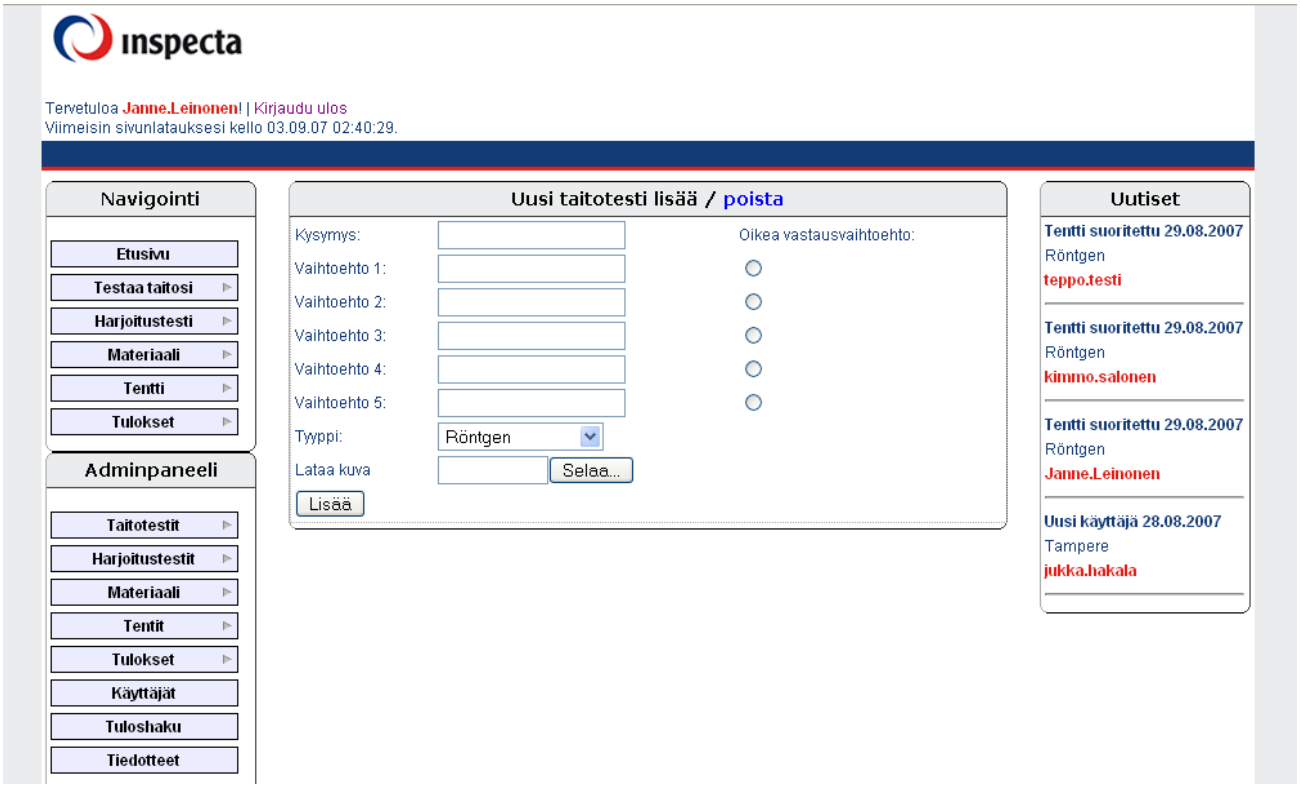
Kuva 3. Tenttinäkymä. Alhaalla vastauslomake. Kuvasta saa 1530x1639 suurennoksen.

8.1.2 Toinen kehitysaste

Uuteen sivustoon oli tarkoitus sisällyttää käyttäjien taitotason testaaminen, harjoitusten tekemismahdollisuus ennen tenttiä sekä lopullinen tentti. Itse halusin myös, että vanhoja koulutusmateriaaleja pystyisi lisäämään käyttäjien luettaviksi. Tiedon keruu käyttäjistä oli yksi vaatimus kouluttajien näkökulmasta, joten sivuston tuli sisältää myös oma sisäänkirjautuminen tunnuksilla. Koska ylemmän tason käyttäjiä oli nyt enemmän kuin yksi, tuli heille luoda oma admin-osio, jossa he pystyvät hallitsemaan materiaalia sekä tarkastelemaan tuloksia (Kuvat 4 ja 5).



Kuva 4. Admin-käyttäjän tuloshaku-näkymä



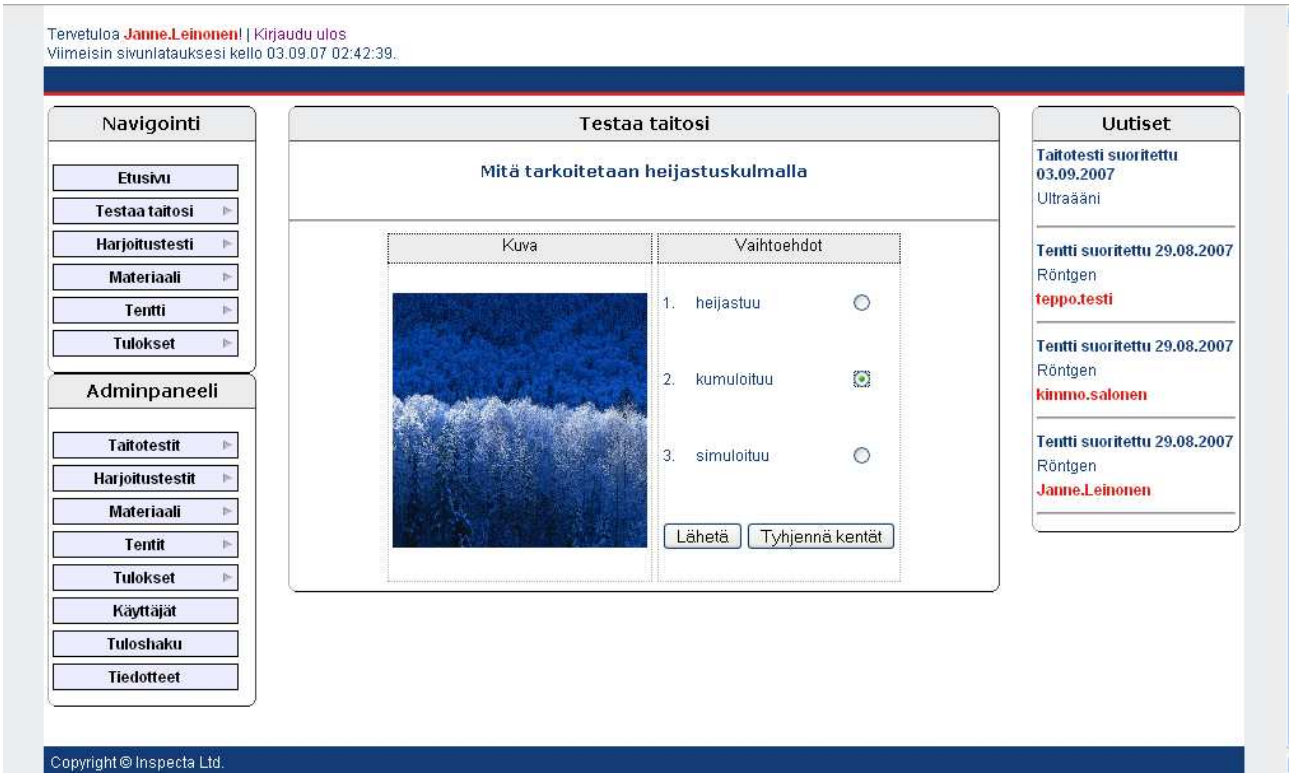
Kuva 5. Admin-käyttäjän lisää testi/harjoitustesti-näkymä

Tässä vaiheessa, kun sivusto alkoi laajeta, otin käyttöön sivupohjajärjestelmän Savant3:sen. Testasin aluksi toimintaa ystäväni suosittelemalla Smarty:llä, mutta koin sen asettamat rajoitukset PHP:n käytölle liian tiukaksi, joten päädyin toteuttamaan sivuston Savant3:sen päälle. Sivupohjajärjestelmän käytössä on ehdottoman tärkeää aloittaa sen käyttö jo projektin alkuvaiheissa, sillä kunnolla suunnittelemalla voi käyttää samoja sivupohjia eri sisällön esittämiseen.

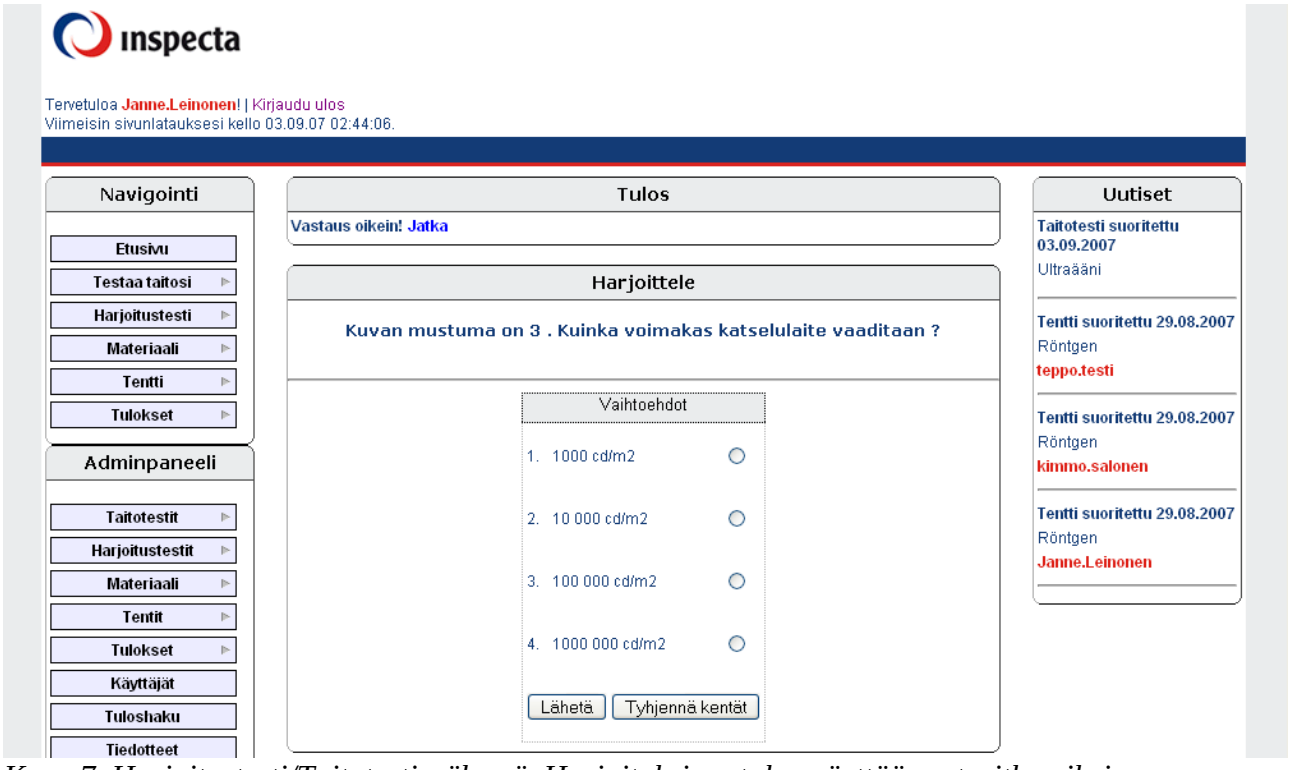
8.2 Esityslogiikka

Sivuston esityslogiikka ja toimintalogiikka on erotettu toisistaan käyttäen apuna sivupohjajärjestelmää. Tämä mahdollistaa sivuston helpon muokkaamisen sekä toimintojen seuraamisen. Kaikki toiminnot toteutetaan PHP-tiedostoissa, jotka kutsuvat tarpeen mukaan sivupohjia, joissa data esitetään.

Sivupohjat ovat pääasiassa PHP-tiedostokohtaisia, mutta myös joitakin yhdistelmärakenteita esiintyy. Sivupohjien yhtenä hienoutena on se, että jos muuttuja ei saakaan arvoa PHP-tiedostossa, mutta se lähetetään kuitenkin sivupohjaan, ei se aiheuta virhettä tulosteessa. Tämä helpottaa sivupohjien yhdistämistä moneen eri tarpeeseen (kuvat 6 ja 7). Toinen mukava puoli sivupohjissa on se, että esimerkiksi admineille tarkoitettua tietoa voi ohjata toiseen sivupohjaan tarkistuksen jälkeen ja lopuille näytetään riisuttu määrä tietoa toisessa sivupohjassa.



Kuva 6. Harjoitustesti/Taitotesti-näkymä. Vaihtoehtoja 1-5 ja kuva valinnainen



Kuva 7. Harjoitustesti/Taitotesti-näkymä. Harjoituksissa tulos näyttää vastasitko oikein.

Sivupohjia voi käyttää myös hyvin suurten kokonaisuuksien tulostamiseen PHP-tiedostosta, mutta itse päädyin käyttämään yksinkertaisinta ratkaisua eli yksi muuttuja PHP-tiedostossa vastaa yhtä muuttujaa sivupohjassa. Tämän takia eri sivupohjia tarvittiin runsaasti, mutta toisaalta sivuston mahdollinen jatkokehittäjä pääsee nopeasti selville sivuston rakenteesta ja toiminnasta.

8.3 Toimintalogiikka

Kaikki tietokantahaut sekä muuttujien määrittelyt tapahtuvat PHP-tiedostoissa. Näistä tiedostoista muuttujat lähetetään Savantin avulla sivupohjille ja lopputuloksena käyttäjälle tulostuu oikea HTML-sivu.

Sivusto sisältää istunto-tyyppisen (SESSION) sisäänkirjautumisen, joka tarkoittaa sitä, että jokaisen näytetyn sivun kohdalla on tarkistettava, onko käyttäjä kirjautunut sisään. Tämä on toteutettu siten, että aina esiintyvät tiedot eli <html><head><title><body> tagit on erotettu omaksi HEAD.php sivuksi, sekä lopetustagit </body></html> on erotettu omaksi BOTTOM.php sivuksi. Nämä sivut kutsutaan aina jokaisen sivun kohdalla ja ainoastaan keskiosan tiedot ovat muuttuvia.

Yksinkertaistettuna index.php sivu näyttää tältä:

```
<?php
 include ”head.php”;
 include ”body.php”;
 include ”bottom.php”;
?>
```

Tällöin tulostetaan aina alkupään rakenne. Muuttuva sisältö tulostetaan body.php sivulta, sekä korrektit lopetus rakenteet bottom.php sivulta.

Head.php sisältää yhteyden tietokantaan ja juuri tämän head.php:n avulla tarkastetaan, että käyttäjällä on oikeus selata sivuja.

Edellä kuvattu rakenne mahdollistaa erittäin helpon tavan pilkkoa sivuston rakennetta pienempiin ja helpommin hallittaviin kokonaisuuksiin. Esimerkiksi edellä mainittu body.php voitaisiin pilkkoa seuraavasti esimerkiksi admin-sivujen näyttämiseksi:

```
<?php
 if($user[’admin’]==1)
    include ”center-admin.php”;
 else
    include ”center.php”;
?>
```

$_GET ja $_POST

Ylläpidon helpottamiseksi sivuston sisältö luodaan pääasiassa $_GET ja $_POST muuttujien avulla. Molemmat ovat lomakkeessa lähetettäviä muuttujia, mutta $_GET-muuttujaa voidaan käyttää myös vaihtoehtoisesti lähettämällä tiedot linkkien kautta.
Esimerkki:

```
<a href=”index.php?sivu=etusivu”>Etusivu</a>
```

lähettää $_GET[‘sivu’] nimiseen muuttujaan arvon ”etusivu”.

Tämän avulla voidaan luoda sivusto, jonka käyttäjä näkee ainoastaan index.php sivun, mutta sisältö muuttuu linkkien mukaan. Tällaisen rakenteen haittapuoli on altistuminen tietoturvahyökkäyksille, mutta lähetteen sisällön tarkastaminen, sekä Savantin sisältämä eprint-metodi estävät näiden käytön. Lähetteiden mukaan haetaan tarvittavat tiedot tietokannasta tai vaihdetaan näytettävä sivu (include) body-osiossa.

Esimerkki:
linkki:

```
index.php?sivu=etusivu
```

php:

```
$sivu = $_GET[’sivu’];

//tulostaa etusivu.php
include ($sivu.”.php”);
```

Suositeltavaa on kuitenkin välttää suoria linkkikentän lähetteiden käyttämistä include-toiminnon kanssa. Parempi tapa on käyttää tunnisteita seuraavaan tapaan:

linkki:

```
index.php?sivu=1
```

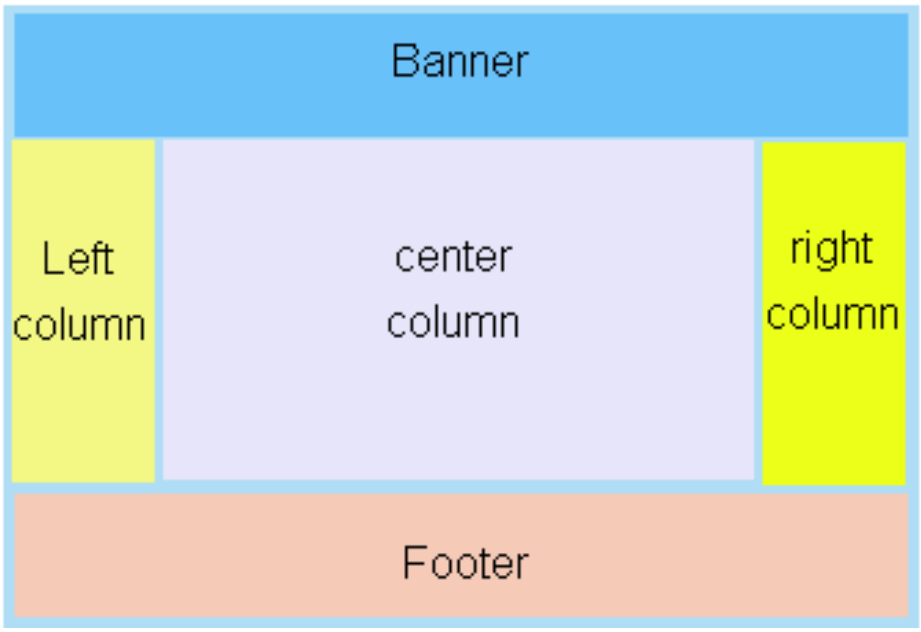
php:

```
$sivu = $_GET[’sivu’];
switch($sivu){
 case 1:
        include ”etusivu.php”;
 break;
 case 2:
        include ”tokasivu.php”;
 break;
}
```

Näin voidaan välttää tietoturvahyökkäys, jos käyttäjä kirjoittaakin linkkikenttään: index.php?sivu=c:/etusivu.php.

Ulkoasu

Ulkoasu on erotettu esityksestä ja toiminnasta CSS-tyylitiedoston avulla. Tyylitiedostoja on yksi, joka kontrolloi kaikkea ulkoasuun viittaavaa sivustolla. Layout on luotu taulukoiden sijasta DIV-rakenteilla, jotka määrittävät viiden eri elementin sijainnit ja suhteet (Kuva 8).



Kuva 8. Sivun layout eli rakenne.

- Banner osio sisältää sisään kirjautumiseen tarvittavan lomakkeen.
- Left column sisältää navigoinnin.
- Center column näyttää kaiken vaihtuvan sisällön käyttäjän valintojen mukaisesti.
- Right column näyttää päivittyvän ”uutiset” sivuston tapahtumista admineilla: esim testien tekemiset ja uudet käyttäjät, sekä ”tiedotteet” osion käyttäjille, joissa adminit voivat ilmoittaa tulevista tapahtumista.
- Footer sisältää tietokannan sulkemisen, sekä kaikki tarvittavat lopputagit rakenteen oikeaoppiseen sulkemiseen.

Lisäksi ulkoasua on tehostettu JavaScriptillä toimivalla valikolla, joka näyttää aina jatkovaihtoehdot ja piilottaa turhat linkit. Tällä on tarkoitus helpottaa käyttäjän valinnan vaikeutta ja varmistaa, että käyttäjä olisi koko ajan tietoinen mihin on nyt menossa.

Ainoa Ajaxia hyödyntävä kohta sivustossa on datan esittäminen taulukossa, jossa järjestely tapahtuu Ajaxin avulla (Kuva 9). Jos halutaan järjestää taulukko nimen perusteella, tehdään uusi haku Ajaxilla ja tulokset järjestetään uuden hakutuloksen mukaan.

Kuva 9. Käyttäjät-näkymä. Taulukko toteutettu AJAX:lla.

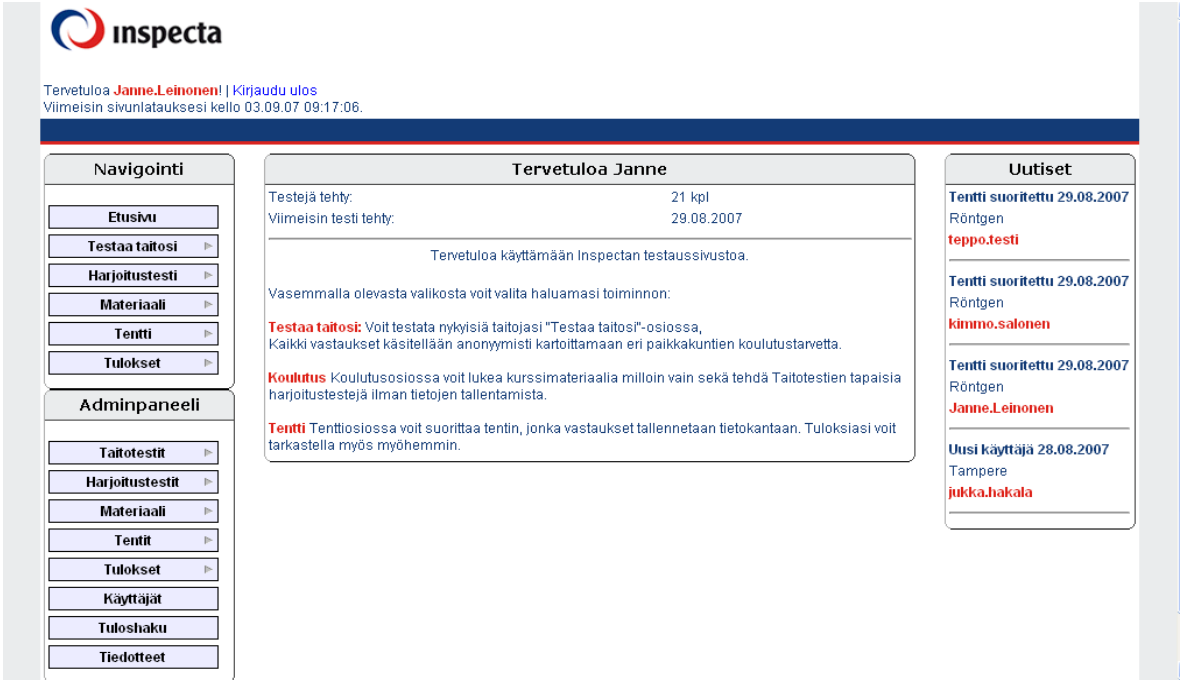
Rajoitteet

Projektin alussa ei annettu mitään rajoitteita sivuston ulkoasun suhteen, mutta kun sivusto päätettiin sisällyttää Intranetiin, oli selvää, että ulkoasun tulisi mukailla tätä (Kuva 10). Koulutussivusto onkin värimaailmaltaan ja mittasuhteiltaan lähes identtinen Intranetin kanssa ja ainoastaan muutamassa kohdassa (left- ja right column) halusin tehostaa näiden osioiden erottuvuutta keskiöstä (center column).

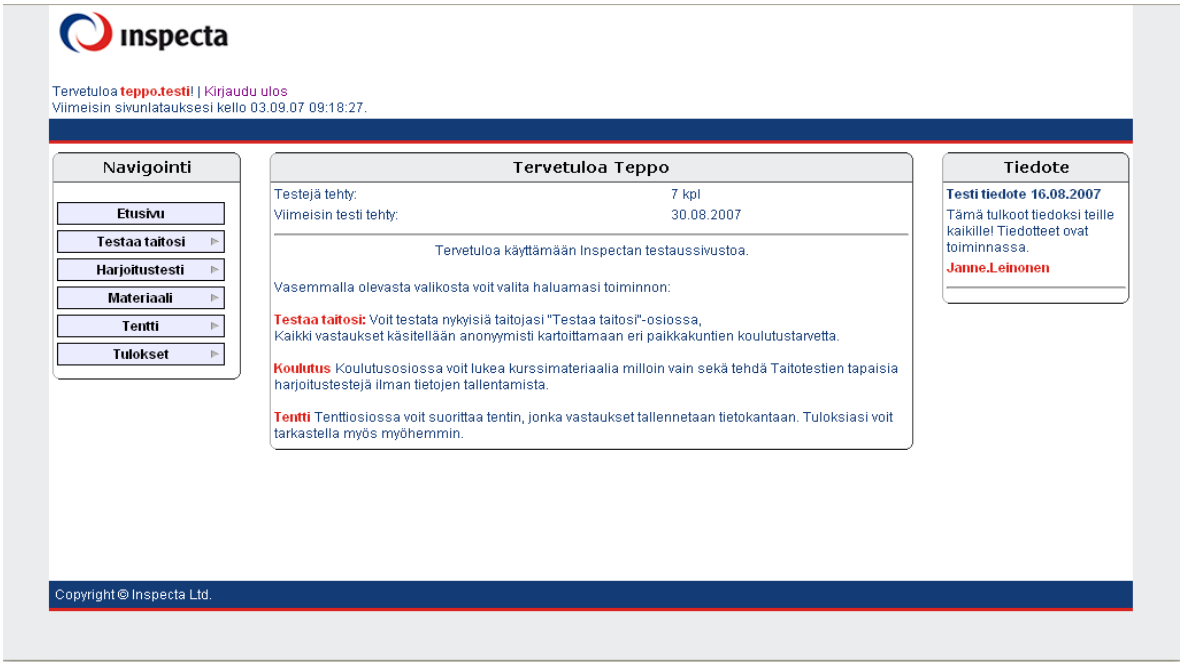
Kuva 10. Sivuston ulkoasu on yhtenäinen yrityksen Intranetin kanssa. (Etusivu)

Käyttäjät

Eri käyttäjiä sivustolla on peruskäyttäjät, sekä admin-tason käyttäjät. Molempien kohdalla pidin toimintalogiikan samana kirjautumisen ja selauksen suhteen ja ainoastaan esitettävään sisältöön vaikutti käyttäjätaso (Kuva 11). Tämän avulla pyrin saavuttamaan sivuston nopean omaksumisen myös admin-tason käyttäjille, koska heidän täytyy ensin luoda materiaali, jota oppilaat voivat myöhemmin opiskella. Käytännössä molemmilla ryhmillä navigointi tapahtuu samoista linkeistä samoilla tavoilla ainoastaan eri toiminnoin (Kuva 12).



Kuva 11. Ulkoasu pyritty luomaan yksinkertaiseksi ymmärtää. (Admin-sivu Testit-osio)



Kuva 12. Peruskäyttäjän etusivunäkymä.

9 Pohdinta

9.1 Yhteenveto

Aikataulullisista ongelmista huolimatta projekti onnistui hyvin. Vaikka käyttäjäkokemukset eivät ehtineet tutkimukseen mukaan, oli lopputulos kuitenkin ilmeisen onnistunut. Projekti opetti, kuinka tärkeää on saada asiakkaalta selkeä suunnitelma tai toivelista niistä asioista, mitä sivuille halutaan, sillä liiallinen vapaus kaikessa on lopulta hyvin turhauttavaa. Oppimiskokemuksena projekti opetti aivan valtavasti hieman laajemman internetsivuston luomisesta ja siitä, kuinka monella tavalla on mahdollista toteuttaa sama asia. Taitotasoni sanoisin kasvaneen useita satoja prosentteja siitä, mitä se oli projektia aloitettaessa.

Mitä nyt tekisin toisin?

On vaikea verrata tilannetta nyt ja projektin alussa: mitä tekisin toisin, jos nyt aloittaisin saman projektin uudestaan. Silloin kun aloimme kehittää sivustoa, ei kukaan meistä vielä tiennyt, mitä kaikkea sivusto tulisi sisältämään. Jos kuitenkin oletetaan, että vastaava projekti aloitettaisiin ja tilaaja olisi laatinut selkeän toivelistan, laatisin sille heti alkuun vedenpitävän suunnitelman kaikesta, mitä sen tulee sisältää ja sitouttaisin tilaajan paremmin mukaan projektiin.

AJAX oli tekniikka, johon tutustuin vasta projektin loppuvaiheessa ja sen täysimittainen hyödyntäminen olisi vaatinut koko sivuston uudelleen suunnittelun. Kuitenkin mahdollisuuksia keksin lähes rajattomasti tämän tekniikan pohjalta ja tulevissa projekteissani aion sitä ehdottomasti käyttää heti alusta alkaen. AJAX:lla olisi ollut mahdollista toteuttaa eräs testitoive, jossa kiilannäköistä kuvaa olisi pitänyt siirtää toisen kuvan päällä mielestään oikeaan kohtaa ja tallentaa tämä kuva vastauksena. Muutenkin testit olisi voinut luoda käyttäen hyväksi AJAX:ia ja näin säästyä suurelta määrältä POST ja GET tietojen lähettelyä.

Toinen oppimisen arvoinen tekniikka oli sivupohjajärjestelmä, joka myös alusta asti käytettynä tuo huomattavia ylläpidollisia helpotuksia vähänkään isommissa projekteissa. Esimerkiksi useamman kielen tuki on helppo heti alkuvaiheessa rakentaa mukaan ja myöhemmin niin halutessaan lisätä mukaan muita kieliä. Esimerkkejä ja oppimateriaalia sivupohjajärjestelmistä ei työn kirjoitushetkellä löytynyt juuri yhtään, mutta hyvä ja sopivan helppo tutustumiskeino on tutkia avoimen lähdekoodin phpBB keskustelupalstan lähdekoodia, jossa on käytetty Smartya sivupohjajärjestelmänä.

9.2 Tulosten arviointi ja tulevaisuus

Sivuston sain määräaikaan mennessä valmiiksi ja testauksen oli tarkoitus olla syyskuun aikana. Varsinaista palautekierrosta en työstä voinut kuitenkaan tehdä, sillä sivuston testaaminen on edelleen toimeksiantajalla (15.11.2007) kesken. Selvää kiinnostusta sivustoa kohtaan kuitenkin yrityksellä on ja minut palkattiin jatkokehittämään sitä vuoden loppuun saakka ja koko ajan keksitään sivustolle uusia ideoita. Voidaan siis arvioida lopputuloksen olleen ainakin toimeksiantajan mieleen.

Itse olen lopputulokseen tyytyväinen. Työtä tehdessä ajauduin kuitenkin lievään oravanpyörään, sillä aina kun opin jotain uutta, tuntuivat monet vanhat ratkaisut kummallisilta. Tälläkin hetkellä, jos aloittaisin projektin täysin alusta, tekisin monta asiaa rakenteessa toisella tavalla. Silti en sano, että nykyiset ratkaisut ovat huonoja vaan, että nyt olisi helpompi rakentaa kokonaisuus, kun tietää asioista enemmän.

Uusista tekniikoista olen erityisen innoissani AJAXista ja sivupohjajärjestelmästä. PHP:n ja SQL:n avulla osaan jo rakentaa asiakkaalle mieleisen palvelun, mutta oikeastaan vasta AJAX tuo siihen lisäetua normaaliin ohjelmointiin verrattuna. Pitkällä tähtäimellä sivupohjajärjestelmän käyttö tuo sellaisia ylläpidollisia etuja, että uskon tulevaisuudessakin sellaista käyttäväni. Voin myös hyvällä omatunnolla suositella kyseisiä sivupohjajärjestelmiä kenelle tahansa kokeiltavaksi, mutta toivoisin samalla, että näistä löytäisi pian jo jotain lähdemateriaalia.

Tulevaisuus koulutussivuston osalta näyttää valoisalta ja käyttöön se pääsee varmasti vuoden 2008 alkupuolella. Jatkokehitys ja lisäosion kehittäminen voidaan tehdä modulaarisesti, joten nyt kun runko on kunnossa, on siihen palkitsevaa keksiä uusia asioita lisää. Tällä hetkellä sivustolle on jo suunniteltu rakenteellisia muutoksia olio-ohjelmoinnin muodossa, sekä AJAX:in käyttämistä hyväksi testeissä. Ajaxia olisi voinut hyödyntää enemmänkin sivustolla, mutta tekniikkana se tuli niin myöhäisessä vaiheessa mukaan, ettei siitä saanut tarpeeksi luotettavaa lopulliseksi tekniikaksi. Sivustoa kehitetään kuitenkin edelleen ja Ajaxilla on tulevaisuudessa vahva rooli tässä kehityksessä.

On mahdollista, että Inspecta Oy ei jää ainoaksi tätä sivustoa tarvitsevaksi yritykseksi. Koska puhutaan teollisuuden vaatimista tarkastuksista ja nämä tarkastukset on tehtävä standardien mukaisesti, on tarve työkalulle, joka toimisi yleisen taitotason mittarina. On siis mahdollista, että sivusto saatetaan laajempaan käyttöön, sekä että sitä tultaisiin laajentamaan. Mikään ei ole tietenkään kehittäjän kannalta mieluisampaa, kuin että hänen tuotettaan käytetään.

Lähteet

PAINETUT LÄHTEET

Asleson Ryan, Schutta Nathaniel T. 2007. AJAX tehokas hallinta.
Saarijärvi: Gummerus Kirjapaino Oy.

Linjama 2001. XHTML. Jyväskylä: Docendo

Rami Heinisuo, Ilkka Rauta 2007. PHP ja MySQL - Tietokantapohjaiset verkkopalvelut.
Gummerus Kirjapaino Oy.

David Flanagan 1997. JavaScript - Tehokäyttäjän opas. Gummerus Kirjapaino Oy.

VERKKOLÄHTEET

Smarty Wikipedia [online] [viitattu 20.06.2007] http://fi.wikipedia.org/wiki/Smarty (20.06.2007)

Smarty.net [online] [viitattu 20.6.2007] http://smarty.PHP.net/

Savant 3 [online] [viitattu 21.6.2007] http://PHPsavant.com/yawiki/

Template System Wikipedia [online] [viitattu 19.06.2007]
http://en.wikipedia.org/wiki/Web_template_system
Article Template Engines [online] [viitattu 19.06.2007]
http://WWW.massassi.com/PHP/articles/template_engines/

Sgl Wikipedia [online] [viitattu 14.06.2007] http://fi.wikipedia.org/wiki/SQL

MySQL Wikipedia [online][viitattu 14.06.2007] http: //fi.wikipedia.org/wiki/MySQL