



**Tampereen
ammattikorkeakoulu**

Opinnäytetyö

**TEKSTIVIESTIENPALVELUN KEHITTÄMINEN PHP:N JA
XML:N AVULLA**

Helena Heikkilä

Tietojenkäsittelyn koulutusohjelma
joulukuu 2006
Työn ohjaaja: Petri Heliniemi

Tampere 2006



Tekijä(t)	Helena Heikkilä	
Koulutusohjelma(t)	Tietojenkäsittely	
Opinnäytetyön nimi	Tekstiviestipalvelun kehittäminen PHP:n ja XML:n avulla	
Työn valmistumis- kuukausi ja -vuosi	joulukuu 2006	
Työn ohjaaja	Petri Heliniemi	Sivumäärä: 31

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee toimeksiantajani WM-datan tekstiviestipalvelun kehittämistä. Tekstiviestipalvelun käyttäjänä on vuokratyövoimaa välittävä yritys, joka ilmoittaa vapaista työpaikoista asiakkailleen tekstiviestillä. Palvelu on toiminut tähän mennessä yhdensuuntaisena, eli yrityksestä on pysytty lähettämään asiakkaalle tekstiviesti, mutta asiakas ei ole pystynyt vastaamaan viestiin, sillä toimeksiantajan palvelimella ei ole sovellusta, joka poimisi asiakkaan viestin tiedot talteen. Palvelusta haluttiin kahdensuuntainen, jotta olisi mahdollista saada asiakkaalta vastaus työtarjoukseen tekstiviestinä.

Tekstiviestipalveluun haluttiin lisäominaisuuksina palvelimelle saapuvien tekstiviestien tietojen tallennus, tallennuksessa esiintyvien mahdollisten virheiden raportointi virhelokitiedostoon sekä tapahtumaloki, johon kirjataan tallennettujen tekstiviestien tiedostonimet ja tallennusajankohdat. Tekstiviestien tiedot, esimerkiksi lähettäjän matkapuhelinnumero, viestin avainsana ja varsinaisen tekstiviestin sisältö, tallennetaan toimeksiantajan palvelimelle niin, että jokaisesta viestistä luodaan oma tiedosto. Palvelun kehittäminen toteutettiin PHP- ja XML-tekniikoiden avulla.

Työssä on käytetty lähteinä enimmäkseen PHP:tä ja XML:ää käsitteleviä www-sivuja, mutta myös toimeksiantajan QLA Messaging Server-yhdyskäytäväpalvelimen manuaalia sekä PHP:ta ja XML:ää käsittelevää kirjallisuutta.

Opinnäytetyön tuloksena syntyi toimiva PHP-sovellus, joka ottaa talteen palvelimelle saapuvista tekstiviesteistä tunnistetietoja ja tekee jokaisen viestin tiedoista oman XML-tiedoston. Tulevaisuudessa toimeksiantajan palvelimella oleva PHP-versio päivitetään uudempaan, joten tehtäviini kuului myös raportoida mahdollisista sovellusmuutoksista PHP:n päivityksen yhteydessä.



Author(s)	Helena Heikkilä	
Degree Programme(s)	Business Information Systems	
Title	Developing SMS-service with PHP and XML	
Month and year	December 2006	
Supervisor	Petri Heliniemi	Pages: 31

ABSTRACT

This thesis is about the development of my employer WM-data's SMS service. The final user of the service is a company, which provides staffing services. Company informs its clients about free jobs via SMS message. The service was based on 1-way SMS gateway, but my employer wanted the service to be 2-way, so that the client could reply the job offer via SMS message. The idea is to create an application, which saves the information of the SMS messages to my employer's server.

My employer wanted to develop the SMS service with a couple of additional services: the saving of incoming SMS messages, logging possible error situations during the saving and creating an event log, which contains the names of created SMS message files and the time of the file creation. The information of every SMS message, e.g. GSM number of the sender, keyword and the actual SMS message, is saved to employer's server as an individual file. The development of the service was implemented with PHP and XML.

The sources of this thesis consist mainly of websites that concern PHP or XML, but also the manual of QLA Messaging Server and literature of PHP and XML.

As a result of this thesis I managed to create a functioning PHP application, which picks up the information of incoming SMS messages and saves the information of each message to an individual XML file. In the future PHP version on my employer's server will be updated, so I also had to report possible changes to the application during the updating process.

Käsitteet

DOM	Document Object Model, eli dokumenttiobjektimali, W3C:n standardoima ohjelmointirajapinta
DTD	Document Type Definition, dokumenttityypin määrittely
GET	http-palvelupyynnön metodi
HTML	HyperText Markup Language, standardoitu sivukuvauskieli
http	HyperText Transfer Protocol, hypertekstin siirtoprotokolla
Merkistö	Sopimus, joka määrittelee, miten eri bittiyhdistelmät tulkitaan eri merkeiksi
PHP	PHP Hypertext Preprocessor, skriptikieli
POST	http-palvelupyynnön metodi
QMS	QLA Messaging Server, Kuulalaakeri Oy:n kehittämä yhdyskäytäväpalvelin
Schema	W3C:n kehittämä skeemakieli, mahdollistaa DTD:tä tarkemman rakennemäärittelyksen
SGML	Standard Generalized Markup Language, metakieli, jolla voidaan määrittellä merkkaukieliä
Skriptikieli	Ohjelmointikieli, joka ei vaadi kääntämistä
SMS	Short Message Service, tekstiviesti
URL	Uniform Resource Locator, dokumentin osoite Internetissä
W3C	World Wide Web Consortium, kansainvälinen yritysten ja yhteisöjen yhteenliittymä joka kehittää WWW-standardeja
WWW	World Wide Web, Internetissä toimiva hyperteksti-järjestelmä
XML	eXtensible Markup Language, metakieli, jolla voidaan määrittellä merkkaukieliä
Yhdyskäytäväpalvelin	Palvelin, joka välittää tekstiviestit eteenpäin

Sisällysluettelo

1 JOHDANTO	6
1.1 TOIMEKSIANTAJAN ESITTELY	6
1.2 TYÖN ESITTELY JA TAVOITTEET.....	6
1.3 KÄYTETYISTÄ LÄHTEISTÄ	7
2 XML	8
2.1 MITÄ ON XML?.....	8
2.3 DOKUMENTTIPUU-ESITYSMALLI	9
2.4 HYVIN MUODOSTETTU JA VALIDI DOKUMENTTI.....	10
2.5 DTD-MÄÄRITYS	11
2.6 SCHEMA-MÄÄRITYS.....	12
3 PHP.....	14
3.1 MITÄ ON PHP?	14
3.2 PHP:N SYNTAKSI.....	14
3.3 PHP:N DOM-LAAJENNUS	15
4 TEKSTIViestIPALVELUN KULKU	17
4.1 QLA MESSAGING SERVER.....	18
4.2 TEKSTIViestIN LÄHETYS PALVELIMELTA	18
4.3 TEKSTIViestIIN VASTAAMINEN	18
4.4 TEKSTIViestIN OHJAAMINEN AVAINSANAN AVULLA PHP-SOVELLUKSEEN	18
4.5 TUNNISTETIETOJEN TALLENNUS	19
4.5.1 Tunnistetietojen tarkempi esittely	19
4.5.2 XML-dokumentin luonti	20
4.5.3 Tunnistetietojen vieminen muuttujiin	20
4.5.4 Elementtien luonti ja tunnistetietojen vieminen elementteihin.....	21
4.5.5 Dokumentin tallennus	21
4.5.6 Virheiden käsittely	22
4.5.7 Virhelokin luonti	23
4.5.8 Tapahtumalokin luonti	25
4.6 MUUTOKSET PHP-SOVELLUKSEEN PHP-VERSION PÄIVITYKSEN YHTEYDESSÄ	25
5 SOVELLUKSEN TYÖVAIHEET JA TESTAUS	27
5.1 SOVELLUKSESSA KÄYTETTÄVIEN TEKNIKOIDEN VALINTA.....	27
5.2 TYÖYMPÄRISTÖ JA TESTAAMINEN	27
6 YHTEENVETO.....	29
6.1 ONGELMAT.....	29
6.2 LOPPUTULOS JA ARVIOINTI.....	29
LÄHTEET.....	31

1 Johdanto

1.1 Toimeksiantajan esittely

Työn toimeksiantajana toimii WM-data, joka on pohjoismaiden johtava informaatiotekniikan palveluyritys. Tämä opinnäytetyö liittyy toimeksiantajan henkilöstöohjauspalveluihin. WM-datan henkilöstöohjauksen ratkaisut tukevat nykyaikaista henkilöstöjohtamista ja auttavat henkilöstöressurssien ja niiden kustannusten tehokkaassa hallinnassa. (Henkilöstöhallinto, WM-data Oy 2006.)

WM-data-konserni on perustettu 1969, ja tällä hetkellä sen palveluksessa on noin 9000 ammattilaista. WM-data on osa kansainvälistä IT-palveluyritystä, LogicaCMG:tä. LogicaCMG on listattu Lontoon ja Amsterdamin pörseissä. Suomessa WM-datalla on ollut toimintaa vuodesta 1983. Työntekijöitä WM-datalla on Suomessa noin 2400. (Fakta, WM-data Oy 2006.)

1.2 Työn esittely ja tavoitteet

Toimeksiantajalla on asiakasyritys, joka on vuokratyövoimavälittäjä. Toimeksiantajan tehtävänä on tarjota asiakasyritykselleen tekstiviestipalvelu, jonka avulla voidaan ilmoittaa avoimena olevista työpaikoista työnhakijoille. Työnhakijan tulisi pystyä vastaamaan tekstiviestiin, jos on kiinnostunut kyseisestä työpaikasta. Työnhakijoilta tulevat tekstiviestit tulisi pystyä tallettamaan toimeksiantajan palvelimelle, jotta niissä olevat tiedot voidaan toisen sovelluksen avulla hakea asiakasyrityksen tietokantaan.

Tekstiviestipalvelu toimi aikaisemmin yhdensuuntaisesti, eli työnhakijoille pystyttiin lähettämään tekstiviesti palvelunumerosta, mutta ei ollut olemassa sovellusta, joka käsittelisi palvelunumeroon tulevat tekstiviestit ja tallettaisi niissä olevat tiedot ylös.

Toimeksiantajalta saadun tehtävän tavoitteena on kehittää toimeksiantajan tekstiviestipalvelu kahdensuuntaiseksi. Tarkoituksena on tehdä sovellus, joka poimii tekstiviestistä tunnistetiedot ja tallettaa jokaisen viestin tiedot omaan tiedostoonsa. Jos tallennuksessa tapahtuu virhe, raportoidaan siitä erilliseen virhelokiin. Sovellukseen kuuluu myös tapahtumalokin luonti. Tapahtumalokiin kirjataan tallennetun XML-tiedoston nimi ja tallennusajankohta. Sovellus tehdään PHP-skriptikielellä ja tekstiviesteistä luodaan XML-tiedostot. Tämän osuuden jälkeen toimeksiantaja kehittää työtä eteenpäin

niin, että tallennetut XML-tiedostot haetaan automaattisesti tietyn väliajoin asiakkaan palvelimelle ja tiedot siirretään toisen palveluksen avulla asiakkaan tietokantaan.

Toimeksiantajalla on käytössä QLA Messaging Server-yhdyskäytäväpalvelin. Yhdyskäytäväpalvelimelle on mahdollista räätälöidä palveluita useiden ohjelmointikielien avulla käyttäen http-rajapintaa. Tekstiviestit lähetetään QLA-yhdyskäytäväpalvelimen kautta asiakkaille. Kun asiakas vastaa viestiin, ohjataan viestit QLA-palvelimen kautta PHP-sovellukseen.

Teoriaosuudessa esitellään tekstiviestipalvelun kehittämisessä käytettyjä tekniikoita, PHP-skriptikieltä ja XML-merkkäuskieltä. Luvussa neljä esitellään tekstiviestipalvelun kulku ja kerrotaan miten toimeksiantajalle tehty sovellus toimii osana tekstiviestipalvelua.

1.3 Käytetyistä lähteistä

Olen käyttänyt opinnäytetyössäni lähteinä enimmäkseen PHP:ta ja XML:ää käsitteleviä WWW-sivuja. Olisin mieluummin käyttänyt enemmän kirjallisia lähteitä, mutta esimerkiksi XML-dokumenttien käsittelystä PHP:n avulla en löytänyt muualta tietoa, kun Php.net WWW-sivuilta.

Wikipedian käyttö XML-teoriaosuuden lähteenä tuntui aluksi arveluttavalta, mutta huomasi, että XML:n kuvaus Wikipediassa on kirjoitettu hyvin ytimekkäästi ja mielestäni kuvaus on asiasisällöltään oikein.

2 XML

Tässä luvussa kerrotaan XML-merkkaukielestä, sen syntaksista ja XML-dokumentin rakennemäärittämisestä.

2.1 Mitä on XML?

XML (eXtensible Markup Language) on metakieli. Sen avulla voidaan kuvata tiedon rakennetta sekä luoda omia merkkaukikieliä. Tieto eli data koostuu yleensä tekstistä ja kuvista, jotka on järjestetty jonkin periaatteen mukaan, säännöllisesti (regular data) tai epäsäännöllisesti (irregular data). XML:n avulla voidaan esittää tiedon hierarkia sekä kapseloida data. (XML Johdanto, Koulutus ja konsultointipalvelu 2006.)

XML kieltä käytetään karkeasti jaettuna joko formaattina tiedonvälitykseen järjestelmien välillä tai formaattina dokumenttien tallennukseen.

Wikipedian (XML, Wikipedia 2006) mukaan XML-dokumenttien käytöllä tavoitellaan mm.

- alustariippumatonta tiedonsiirtoa
- sisältöjen yhdenmukaisempaa tallennusmuotoa
- sisältövirheiden välttämistä
- tiedon hakemisen helpottamista
- tiedon pitkäaikaissäilyvyyden parantamista.

2.2 XML:n syntaksi

XML muistuttaa HTML-kieltä, jolla WWW-sivut kirjoitetaan, ja ne kummatkin ovat SGML-kielen yksinkertaistettuja osajoukkoja. XML-kieli ei kuitenkaan ole tarkoitettu sivunkuvauskieleksi kuten HTML, vaan sillä kuvataan tiedon rakenne ilman ennalta määrättyjä merkkaukiskoodeja. XML-kielillä voi muodostaa uusia merkkaukiskoodeja, joiden avulla voidaan luoda dokumentteja hyvinkin erilaisiin ja erityisiin tarkoituksiin. (XML, Wikipedia 2006.)

XML-dokumentti muodostuu hierarkkisesta elementtirakenteesta. Elementti merkitään HTML:stä tutuilla < ja > merkeillä. Dokumentin elementeillä voi olla myös ominaisuuksia, eli attribuutteja. Mahdolliset attribuutit määritellään aloituskomennon yhteydessä seuraavasti:

<elementti attribuutti="arvo">.

Kuvassa 1 nähdään, miltä XML-tiedoston rakenne näyttää esimerkiksi Internet-selaimen avulla katseltuna.

```

<viestit>
  <viesti id="1234">
    <sisältö>Tämä on viestin sisältö.</sisältö>
    <osoitetiedot>
      <vastaanottaja>joku@jotain.com</vastaanottaja>
      <lähettäjä>jokutoinen@jotain.com</lähettäjä>
    </osoitetiedot>
  </viesti>
</viestit>

```

Kuva 1 XML-dokumentin rakenne

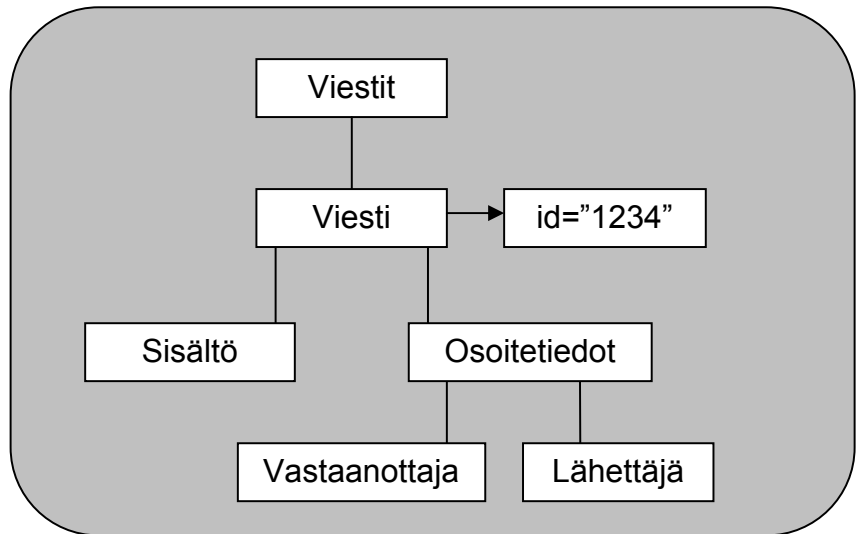
Wikipedian (XML, Wikipedia 2006) mukaan XML-dokumentin rakenteessa on muutamia kielioppisääntöjä:

- XML-dokumentti on tekstiä.
- XML-dokumentissa on yksi, ja vain yksi juurielementti.
- Elementteillä, joissa on sisältöä, tulee olla aloitus- ja lopetustagi.
- Jokainen attribuutti on lainausmerkkien sisällä.
- Elementit voivat olla sisäkkäisiä, mutta ne eivät saa mennä ristiin toisten elementtien kanssa.

2.3 Dokumenttipuu-esitysmalli

Dokumenttipuu on esitysmalli, jonka avulla asiakirjan rakenne esitetään ja tulkitaan. Kaikki dokumentissa käytetyt elementit ovat osa dokumenttipuuta. DOM-oliomallin (Document Object Model) mukaisesti XML-asiakirja on hierarkkinen dokumenttipuu, joka koostuu etukäteen rajoittamattomasta määrästä solmuja (nodes). Jokainen solmu on yksittäinen olio hierarkiassa. Dokumenttiolio on koko hierarkian ylimmällä tasolla. Jokaisella solmulla voi olla useita lapsisolmuja (child nodes, children). Vastaavasti kaikilla solmuilla paitsi juurisolmulla (body, root node) on äitisolmu (parent node). (XML Johdanto, Koulutus ja konsultointipalvelu 2006.)

Kuva 2 havainnollistaa XML-dokumentin dokumenttipuu-esitysmallia.



Kuva 2 XML-dokumentin dokumenttipuu-esitysmalli

Koulutus ja konsultointipalvelu KK Mediat (XML Rakenne, Koulutus ja konsultointipalvelu 2006) Internet-sivuilla esitellään, kuinka dokumenttipuun rakenne muistuttaa paljolti oikeaa puuta:

- Puussa on juuri; dokumenttipuussa on runko (body, root node).
- Puussa on oksia; dokumenttipuussa on lapsielementtejä (children, child node).
- Puussa on lehtiä; dokumenttipuussa on lehtiä (nodes), joilla ei ole enää omia lapsielementtejä.

2.4 Hyvin muodostettu ja validi dokumentti

XML-dokumentin on oltava syntaksiltaan oikein muotoiltu. Käytännössä jokainen pienikin kirjoitus- tai rakennevirhe johtaa käytökeltöttömyyteen tiedostoon. Syntaksin oikeellisuutta voidaan valvoa kahdella mittarilla. XML-dokumentti voi olla joko hyvin muodostettu (well formed) tai validi (valid).

Hyvin muodostettu dokumentti on XML-kieliopin mukainen ja täyttää XML-määrittelyn mukaiset ehdot. Dokumentissa on ainoastaan yksi juurielementti ja kaikilla elementeillä on aloitus- ja lopetustagit ja ne suljetaan oikeassa järjestyksessä.

Jotta dokumentti olisi validi, on sen sisällettävä DTD-määrittely (Document Type Definition) tai schema-määrittely, joka määrittää

XML-dokumentin tarkan rakenteen ja kieliopin. Dokumentin rakennetta verrataan määriteltyyn rakenteeseen. Mikäli se täyttää määritellyt ehdot, on dokumentti validi. (XML Rakenne, Koulutus ja konsultointipalvelu 2006.)

2.5 DTD-määrittäminen

Eräs rakenteisen dokumentin tärkeimpiä ominaisuuksia on mahdollisuus pakottaa looginen rakenne jonkin tietyn mallin mukaiseksi. Mahdollisuus määrätä elementit ja niiden esiintymisjärjestys helpottaa dokumentin laatimista. Kun dokumenttiin liitetään dokumenttityypin määrittäminen eli DTD, kirjoitetaan dokumentti sen mukaisesti ja määrätty rakenne säilyy.

DTD voi olla dokumentin sisäinen tai ulkoinen määrittäminen. Sisäinen DTD sijaitsee XML-dokumentissa hakasulkujen sisällä. Ulkoinen DTD sijaitsee omassa tiedostossaan XML-dokumentin ulkopuolella. (Tuikka & Kanala 2001: 23.)

Kuvassa 3 on esimerkki XML-dokumentin dokumenttityypin määrittämisestä. Kuvassa 4 nähdään kuinka DTD-määrittäminen voidaan liittää ulkoisena tiedostona XML-dokumenttiin.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE viestit [
<!ELEMENT viesti (sisältö, osoitetiedot)
<!ELEMENT sisältö (#PCDATA)>
<!ELEMENT osoitetiedot (vastaanottaja, lähettäjä)
<!ELEMENT vastaanottaja (#PCDATA)>
<!ELEMENT lähettäjä (#PCDATA)>
]>

```

Kuva 3 Esimerkki DTD-määrittämisestä

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE viestit SYSTEM "viestit.dtd">
<viestit>
  <viesti>
    <sisältö>
      ....
      ....

```

Kuva 4 Ulkoisen DTD-määrittämisliittimen liittäminen XML-dokumenttiin

2.6 Schema-määrittäminen

Schema on W3C:n (World Wide Web Consortium) kehittämä skeemakieli, joka mahdollistaa DTD:tä tarkemman rakennemäärittämyksen. Se on laajennettavissa tulevia lisäyksiä varten ja se tukee tietotyyppiä (esim. int, string, date) sekä erilaisia nimiavaruuksia. Myös XML-skeema on XML-dokumentti. Sen tiedostopäätte on .xsd ja juurielementti on <schema>. Kaikkien elementtien edessä esiintyy xsd:, sillä skeema noudattaa xsd-nimiavaruutta, ja ilman tunnistetta se voitaisiin sekoittaa tavalliseen XML-dokumenttiin. (Tuikka & Kanala 2001: 30.)

Kuva 5 esittää dokumentin tarkennettua rakennemäärittäystä schema-kielen avulla.

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="viestit" type="viestitType"/>

<xsd:complexType name="viestitType">
  <xsd:sequence>
    <xsd:element name="viesti" type="viestiType"/>
  </xsd:sequence>

<xsd:complexType name="viestiType">
  <xsd:sequence>
    <xsd:element name="sisalto" type="xsd:string"/>
    <xsd:element name="osoitetiedot" ty-
pe="osoiteType"/>
  </xsd:sequence>

<xsd:complexType name="osoiteType">
  <xsd:sequence>
    <xsd:element name="vastaanottaja"
type="xsd:string"/>
    <xsd:element name="lahettaja"
type="xsd:string"/>
  </xsd:sequence>
</xsd:schema>
```

Kuva 5 Esimerkki schema-määrittelystä

3 PHP

Tässä luvussa kerrotaan yleisesti PHP-skriptikielestä, PHP:n syntaksista ja PHP:n DOM-funktioista, joita sovelluksessa on käytetty.

3.1 Mitä on PHP?

PHP eli PHP: Hypertext Preprocessor (hypertekstin esikäsittelijä) on suosittu yleiskäyttöinen skriptikieli, joka sopii erityisesti Web-sovelluskehitykseen, koska se voidaan liittää suoraan HTML:ään. Lisäksi PHP:n lähdekoodi on avointa, eli sitä saa lisenssin sallimissa rajoissa tutkia, muokata ja levittää. Kieli on syntaksiltaan helppo, se pohjautuu suurimmilta osin C:hen, Javaan ja Perliin. (Käsikirja, Php.net 2006.)

Toisin kuin tavallinen HTML-sivu, PHP-skriptiä ei lähetetä suoraan asiakkaalle, vaan se jäsenellään PHP-ohjelman tai -moduulin toimesta. Skriptissä olevat HTML-elementit jätetään käsittelemättä, mutta PHP-koodi tulkitaan ja suoritetaan. Skriptissä oleva PHP-koodi osaa tehdä kyselyjä tietokannasta, luoda kuvia, lukea tiedostoja ja kirjoittaa tiedostoihin sekä keskustella etäpalvelimien kanssa. Mahdollisuudet ovat rajattomat. PHP-koodin tulostus yhdistetään skriptissä olevaan HTML-koodiin ja tulos lähetetään käyttäjälle. (Zandstra 2001: 20.)

3.2 PHP:n syntaksi

PHP-dokumentin tiedostotunniste on tärkeä, koska se kertoo palvelimelle, että sen tulee käsitellä tiedostoa PHP-koodina. PHP-dokumentin oletustunnus on .php. (Zandstra 2001: 36.)

PHP-koodi upotetaan WWW-sivulle aloitus- ja lopetustagien sisään.

Aloitustagi	Lopetustagi
<?php	?>

Kuvassa 6 on esimerkki, kuinka PHP-koodia voidaan upottaa HTML-koodin sekaan.

```
<html>
  <head>
    <title>Esimerkki</title>
  </head>
  <body>
    <?php
      echo "Hei, olen PHP skripti!";
    ?>
  </body>
</html>
```

Kuva 6 PHP-koodiesimerkki

3.3 PHP:n DOM-laajennus

Mitä DOM tarkoittaa?

DOM (Document Object Model) eli dokumenttiobjektimalli on W3C:n standardoima ohjelmointirajapinta dokumenttien käsitteelyyn. DOM määrittää tavan kohdella dokumenttia solmupuuna. Tässä mallissa jokainen erillinen tietoalkio on solmu, jolloin lapsielementeistä ja sisällytetystä tekstistä tulee alisolmuja. (Holzner 2001: 307.)

DOM-laajennus ja DOM XML -laajennus

DOM-laajennus (DOM extension) on joukko funktioita, jotka mahdollistavat XML-dokumenttien käsittelyn PHP:ssa DOM-rajapinnan kanssa. Funktioiden käyttö ei vaadi erillisten ohjelmien asentamista, vaan ne toimitetaan PHP:n mukana. (DOM functions, Php.net 2006.)

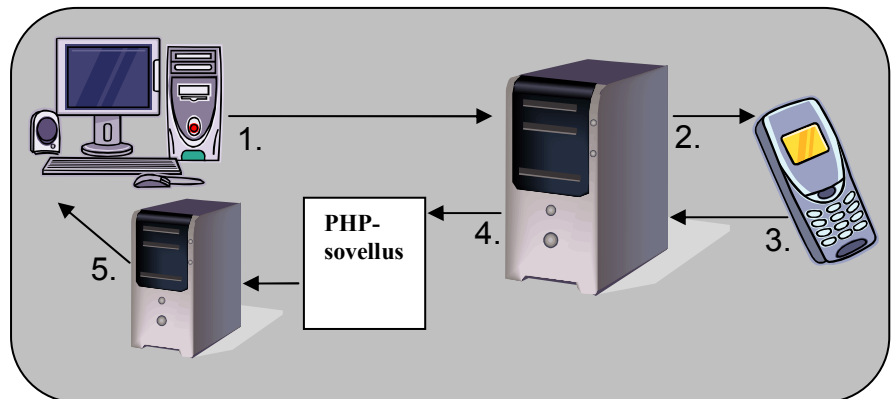
PHP:n DOM-funktioiden avulla voidaan mm. luoda XML-dokumentteja, dokumenttien elementteihin voidaan tallettaa tietoa, tietoja voidaan tarvittaessa muokata ja dokumentteihin voidaan tehdä hakuja. XML-dokumentteja voidaan käyttää tietokannan tavoin ja PHP:n avulla dokumentteja voidaan käsitellä.

Toimeksiantajalle tehdyssä PHP-sovelluksessa ei ollut mahdollista käyttää PHP 5:n DOM-funktioita XML-tiedostojen luomiseen ja tekstiviestien tietojen tallennukseen, sillä toimeksiantajan palvelimella oli PHP:n versio 4.3.2. Päivitys uudempaan PHP-versioon ei

ollut sillä hetkellä mahdollista, sillä palvelimella sijaitsee monia muita sovelluksia, jotka käyttävät PHP:ta ja joidenkin sovellusten toiminta olisi saattanut häiriintyä päivityksen yhteydessä. PHP 4:ssä laajennus kulkee nimellä DOM XML extension ja syntaksi on erilaista verrattuna PHP 5:n DOM-funktioihin. DOM XML-laajennus on uudemmissa PHP-versioissa korvattu DOM-laajennuksella, koska on haluttu parantaa yhteensopivuutta DOM-standardin kanssa. Monet PHP 4:n DOM XML-funktiot eivät toimi PHP 5:ssä.

4 Tekstiviestipalvelun kulku

Tekstiviestipalvelun käyttäjänä toimii vuokratyövoimaa välittävä yritys. Aikaisemmin avoimista työpaikoista on ilmoitettu asiakkaille joko puhelimitse tai sähköpostilla. Nyt sama voidaan hoitaa nopeasti tekstiviestipalvelun avulla. Kuva 7 selventää tekstiviestipalvelun kulkua.



Kuva 7 Tekstiviestipalvelun kulku

1. Vuokratyövoiman välittäjä valitsee tietokannastaan työnhakijat, joille lähetetään tekstiviesti avoimesta työpaikasta. Tieto viestien lähetyksestä kulkee QMS-palvelimelle.
2. Tekstiviestit lähetetään QMS-palvelimen kautta työnhakijoille.
3. Jos työnhakija on kiinnostunut työpaikasta, hän vastaa tekstiviestiin lähettämällä viestin sellaisenaan palvelunumeroon, josta viesti lähetettiin. Jos hän ei ole valmis vastaanottamaan työpaikkaa, ei viestiin tarvitse vastata ollenkaan.
4. Työnhakijan lähettämä vastausviesti ohjataan avainsanan perusteella QMS-palvelimelta PHP-sovellukseen ja PHP-sovellus poimii tekstiviestistä tiedot ja tallentaa ne toiselle palvelimelle XML-tiedostoiksi.
5. Vuokratyövoiman välittäjällä on ohjelmisto, joka hakee minuutin välein palvelimelle tallentuneet XML-tiedostot yrityksen tietokantaan.

Näin saadaan nopeasti tieto siitä, kuka on vastannut tekstiviestiin, eli kuka on valmis vastaanottamaan työpaikan.

4.1 QLA Messaging Server

QLA Messaging Server (QMS) on Kuulalaakeri Oy:n kehittämä operaattoririippumaton viestiliikenteen yhdyskäytäväpalvelin. Sen avulla on helppo toteuttaa monipuolisia yhden- tai kahdensuuntaisia teksti- ja multimediaviestipalveluja. Yhdyskäytäväpalvelin on liitettävissä Internetin kautta matkapuhelinoperaattoreiden teksti- ja multimediaviestikeskuksiin. Palvelin tukee kaikkia tärkeimpiä viestikeskustyyppjejä. (Kuulalaakeri Oy 2004.)

4.2 Tekstiviestin lähetys palvelimelta

Toimeksiantajalla on käytössään ohjelmisto, jolla voidaan Internet-selaimen avulla valita tietokannasta tietyt henkilöt, joille lähetetään tekstiviesti avoimesta työpaikasta. Samalla tietokantaan tallennetaan tieto siitä, kenelle viestit on lähetetty ja kyseisen tekstiviestin tunnistetiedot. Tekstiviestiin pyydetään vastaamaan, jos työnhakija on halukas ottamaan työpaikan vastaan. Tilausjärjestelmä toimii yrityksen sisäverkossa olevalla palvelimella eikä siihen sallita ulkoapäin yhteydenottoja.

4.3 Tekstiviestiin vastaaminen

Kun viesti saapuu työnhakijalle, lähettäjänä on jokin tietty palvelunumero, esimerkiksi 16123. Asiakas vastaa viestiin esim. lähettämällä saapuneen tekstiviestin sellaisenaan palvelunumeroon, josta tekstiviesti hänelle lähetettiin. Jos työnhakija ei ole kiinnostunut työpaikasta, hänen ei tarvitse vastata viestiin ollenkaan.

4.4 Tekstiviestin ohjaaminen avainsanan avulla PHP-sovellukseen

Tekstiviestin mukana kulkee erilaisia tunnistetietoja. Näitä ovat lähettäjän numero, vastaanottajan numero, avainsana, avainsanan jälkeinen tekstiviestiosa ja koko tekstiviestin sisältö. Avainsanan mukaan viesti voidaan ohjata QLA Messaging Serverille saapuesaan http-pyyntöä avulla tiettyyn URL-osoitteeseen. Tässä osoitteessa sijaitsee PHP-sovellus, jonka avulla asiakkaan lähettämästä viestistä poimitaan tunnistetiedot ylös ja tallennetaan omaan XML-tiedostoon. Tekstiviesti alkaa aina avainsanalla, ja tekstiviesti voi olla pituudeltaan korkeintaan 160 merkkiä.

Esimerkiksi toimeksiantajan palvelimelta voidaan lähettää seuraavanlainen tekstiviesti asiakkaalle:

”POPULUS 1232 Kahvila-apulaisen työpaikka avoinna alkaen 11.12.2006.”

Avainsana on tässä tekstiviestissä ”POPULUS”, ja QMS-palvelimella on määritelty tälle avainsanalle tietty URL-osoite, johon viesti ohjataan QMS-palvelimelta.

4.5 Tunnistetietojen tallennus

4.5.1 Tunnistetietojen tarkempi esittely

Työnhakijan lähettämä viesti ohjataan viestin ensimmäisen sanan, eli avainsanan mukaan QMS-palvelimelta URL-osoitteeseen, jossa PHP-sovellus sijaitsee. Kaikki tekstiviestin parametreina olevat tunnistetiedot ovat:

- sms_source; numero, josta tekstiviesti on lähetetty
- sms_dest; tekstiviestin vastaanottajan numero
- sms_command; tekstiviestin avainsana
- sms_params; tekstiviestin avainsanan jälkeinen osa
- sms_text; koko tekstiviesti alkuperäisessä muodossa.

GSM-numerot kulkevat tunnistetiedoissa aina kansainvälisessä muodossa.

Esimerkiksi yritys voi lähettää seuraavanlaisen tekstiviestin työnhakijalle:

Viesti lähtee palvelunumerosta 16123 ja vastaanottajan GSM-numero on 0401234567.

” POPULUS 1234 Firma Oy:llä sihteerin paikka avoinna 7.12.2006 alkaen ”

Tällöin tunnistetiedot ovat:

sms_source	16123
sms_dest	+358401234567
sms_command	POPULUS
sms_params	1234 Firma Oy:llä sihteerin paikka avoinna 7.12.2006 alkaen.
sms_text	POPULUS 1234 Firma Oy:llä sihteerin paikka avoinna 7.12.2006 alkaen

4.5.2 XML-dokumentin luonti

PHP-sovelluksen tiedostotyyppi, eli Content-Type täytyy asettaa sovelluksen koodissa "text/plain"-arvoksi. Sovelluksen koodissa ei saa olla yhtään tyhjää merkkiä ennen php-aloitustagia, sillä muuten tiedostotyyppiä ei voida asettaa oikein. (Pesonen & Löytömäki 2003.)

PHP-sovelluksen alussa luodaan muuttuja \$xml, jonka arvoksi laitetaan XML-dokumentin alkuun tarvittava prologi ja XML-dokumentin juurielementti <sms_client>.

```
$xml = "<?xml version='1.0' encoding='utf-8'?">";
$xml .= "<sms_client> </sms_client>";
```

Tämän jälkeen sovelluksessa luodaan muuttuja \$doc, jonka arvoksi ladataan muistista muuttujan \$xml arvo. Nyt luotavaa XML-dokumenttia käsitellään solmupuuna.

```
$doc = domxml_open_mem($xml);
```

4.5.3 Tunnistetietojen vieminen muuttujiin

Tekstiviestin tunnistetiedot kulkevat viestin mukana parametreina. Sovelluksessa luodaan PHP-muuttujat jokaiselle tunnistetiedolle, ja tunnistetiedot viedään muuttujiin GET-metodin avulla. Metodiksi valittiin nimenomaan GET eikä POST, sillä GET-metodin avulla sovellusta voidaan testata kirjoittamalla tunnistetiedot parametreina selaimen osoiteriville sovelluksen URL-osoitteen perään.

```
$sms_source=$_GET['sms_source'];
$sms_dest=$_GET['sms_dest'];
```

4.5.4 Elementtien luonti ja tunnistetietojen vieminen elementteihin

Sovelluksessa luodaan muuttuja `$root`, jonka arvoksi tulee luotavan XML-dokumentin juurielementti, joka määriteltiin aikaisemmin elementiksi `<sms_client>`.

```
$root = $doc->document_element();
```

Seuraavaksi täytyy luoda elementit tekstiviestin tunnistetiedoille. Tunnistetieto-muuttujista luodaan tekstisolmut, joiden arvoiksi tulevat tekstiviestin parametrien oikeat arvot, esimerkiksi vastaanottajan numero tai viestin avainsana. Nämä tekstisolmut viedään luotujen tunnistetietoelementtien arvoiksi ja tunnistetietoelementit viedään dokumentin juurielementin eli `<sms_client>` alle.

```
$sele = $doc->create_element("sms_source");
$sele = $root->append_child($sele);
$txt = $doc->create_text_node($sms_source);
$txt = $sele->append_child($txt);
```

```
$sele = $doc->create_element("sms_dest");
$sele = $root->append_child($sele);
$txt = $doc->create_text_node($sms_dest);
$txt = $sele->append_child($txt);
```

jne.

4.5.5 Dokumentin tallennus

XML-dokumenttien nimien tulee olla uniikkeja ja kuitenkin toisistaan erottuvia. Tämä hoidettiin niin, että jokaisen XML-dokumentin nimi koostuu uniikista merkkijonosta sekä päivämäärästä, jolloin dokumentti on luotu.

Ensin luodaan muuttuja `$id`, jonka arvoksi tulee `uniqid()`- ja `rand()`-funktioiden avulla uniikki id-tunnus.

```
$id = uniqid(rand());
```

Tämän jälkeen luodaan muuttuja `$time`, jonka arvoksi tulee sen hetkinen päivämäärä niin, että ensin on vuosi, sitten kuukausi ja lopuksi päivä, esim 20060929.

```
$time = date("Ymd");
```

Tämän jälkeen luodaan muuttuja `$xmlfile`, jonka arvoksi tulee polku, johon XML-dokumentit palvelimella tallennetaan, sekä XML-tiedoston nimi. `$xmlfile` muuttujalle on sovelluksen alussa määritetty toimeksiantajan palvelimella kansio, johon tekstiviestien XML-tiedostot tallennetaan.

```
$xmlfile = "$xmlfile_dir/$time"."_id.xml";
```

Tallennus suoritetaan loppuun `dump_file()` funktion avulla, joka muuntaa käsiteltävän XML-solmupuumallin takaisin XML-tiedostoksi.

```
$doc->dump_file($xmlfile, false);
```

Tallentuneen XML-tiedoston rakenne on seuraavanlainen:

```
<sms_client>
  <sms_source>16123</sms_source>
  <sms_dest>+358401234567</sms_dest>
  <sms_command>POPULUS</sms_command>
  <sms_params> 1234 Firma Oy:llä sihteerin paikka avoimna
  7.12.2006 alkaen</sms_params>
  <sms_text>POPULUS 1234 Firma Oy:llä sihteerin paikka
  avoimna 7.12.2006 alkaen</sms_text>
</sms_client>
```

4.5.6 Virheiden käsittely

XML-tiedostojen luonti ja tekstiviestien tunnistetietojen tallennus saattaa jostain syystä epäonnistua. Tällöin työnhakijan lähettämä tekstiviesti ei tallennu mihinkään, eikä yritys tiedä, vaikka työnhakija olisikin kiinnostunut kyseessä olevasta työpaikasta.

Tämän vuoksi on tärkeää, että mahdollisista virheistä muodostetaan toimeksiantajan palvelimelle virhelokitiedosto. Virhelokitiedostossa tulisi näkyä, mikä on virheen laatu, missä yhteydessä virhe tapahtui ja mihin aikaan virhe tapahtui.

Tätä tarkoitusta varten sovellukseen otettiin mukaan oma virheenkäsittelyfunktio, jossa määritellään virhelokia varten tarvittavat tiedot.

Kun QLA Messaging Server lähettää http-pyynnön PHP-sovellukselle, välittyy PHP-sovelluksen mukana otsikkotietoja. Tämän sovelluksen otsikkotietoina välitetään sisällön tyyppi ja käytetty

merkistö. Sisällön tyyppinä täytyy olla text/plain ja merkistönä utf-8, jotta sovellus toimii palvelimella oikein.

On tärkeää, että sovellus ei välitä otsikkotietoja, ennen kuin on todettu, että tekstiviestin tunnistetietojen tallennuksessa ei ole havaittu virhettä. Jos otsikkotiedot välitetään ennen virheiden käsittelyä ja sovelluksessa tapahtuu virhe, ei virhelokiin tallennu mitään. Virhe lähtee vastauksena QLA Messaging Serverille, ja QMS välittää sen edelleen tekstiviestin lähettäjälle. Myöskään tekstiviestin tunnistetiedot eivät tässä tapauksessa tallennu XML-tiedostoon.

Virhetarkastuksen ajaksi voidaan ns. puskuroida tulostus eli tallentaa se muistiin, ennen kuin tulostus lähetetään mihinkään. Tämä tapahtuu funktion `ob_start()` avulla. Puskurointi aloitetaan ennen otsikkotietojen lähetystä. Aloittamisen jälkeen voidaan suorittaa normaalia tulostavaa koodia. Lopuksi puskurointi poistetaan käytöstä funktion `ob_end_clean()` avulla. (Tulostuksen Puskurointi PHP:ssä 2006.)

4.5.7 Virhelokin luonti

Virhelokin luonti aloitetaan asettamalla oma virheidenkäsittelijä `set_error_handler()` funktiolla.

```
set_error_handler('userErrorHandler');
```

Tämän jälkeen määritellään oman funktion `userErrorHandler` toiminta. Oman virhekäsittelijäfunktion tulee hyväksyä ainakin kaksi parametria. Ensimmäinen on virheen koodi ja toinen virheen viesti, joka kuvailee virheen laatua. On myös olemassa kolme muuta parametria, jotka voidaan liittää virhekäsittelijään: tiedoston nimi ja polku, jonka käsittelyn yhteydessä virhe tapahtui (`$filename`), koodirivin numero, missä virhe tapahtui (`$linenum`) ja asiayhteys, missä virhe tapahtui (`$vars`). (Error handler, Php.net 2006.)

```
function userErrorHandler($errno, $errmsg, $filename, $linenum, $vars)
```

Virhelokin alkuun ensimmäiseksi tiedoksi laitetaan päivämäärä ja kellonaika `date()` funktion avulla.

```
$dt = date('Y-m-d H:i:s (T)');
```

Seuraavaksi selvennetään virhetyyppejä, jotta niitä on helpompi tulkita virhelokista. Zend-sivuston mukaan (Error Handling and Logging Functions 2006) virhetyypeistä voidaan luoda taulukko, jossa virhetyypeille voidaan antaa oma nimi.

```
$errortype = array (
    E_ERROR => 'Error',
    E_WARNING => 'Warning',
    E_PARSE => 'Parsing Error',
    E_NOTICE => 'Notice',
    E_CORE_ERROR => 'Core Error',
    E_CORE_WARNING => 'Core Warning',
    E_COMPILE_ERROR => 'Compile Error',
    E_COMPILE_WARNING => 'Compile Warning',
    E_USER_ERROR => 'User Error',
    E_USER_WARNING => 'User Warning',
    E_USER_NOTICE => 'User Notice',
    E_STRICT => 'Runtime Notice'
);
```

Virhelokiin tallennetaan myös tekstiviestin lähettäjän numero, jos virhe ei ole tapahtunut sms_source parametrin yhteydessä.

Jos tunnistetietojen tallennuksessa esiintyy virhe, tallennetaan tieto siitä virhelokitiedostoon omaan lohkoonsa, joka muodostuu XML-tiedoston tapaisesta tyyppirakenteesta.

Alla on esimerkki virhelohkosta virhelokitiedostossa:

```
<errorentry>
    <datetime>2006-10-04 12:42:06 (FLE Daylight Ti-
    me)</datetime>
    <sms_source>+358401234567</sms_source>
    <errornum>8</errornum>
    <errortype>Notice</errortype>
    <errormsg>Undefined index: sms_text</errormsg>
    <scriptname>C:\sms\sms.php</scriptname>
    <scriptlinenum>92</scriptlinenum>
</errorentry>
```

Virheloki voidaan tallentaa palvelimelle error_log() funktiolla. Funktion mukana annetaan kolme parametria, jotka määrittävät virheviestin, virheen ilmoitustavan sekä virhelokin nimen ja tallennuskohteen.

```
error_log($err, 3, 'sms/error_log.log');
```


Ensimmäiseen parametriin, muuttujaan \$err on tallennettu koko virheviesti, esimerkiksi päivämäärä, virheen tyyppi ja tiedoston nimi, jonka käsittelyn yhteydessä virhe tapahtui. Toinen parametri, funktion tyyppinumero ilmoittaa, millä tavalla virheestä raportoidaan. Tässä tapauksessa numero 3 tarkoittaa, että virhe tallennetaan tiedostoon. Viimeinen parametri määrittää virhelokin tiedoston nimen ja tallennuspolun. (Error log, Php.net 2006.)

4.5.8 Tapahtumalokin luonti

Tekstiviestipalveluun haluttiin myös toiminto, jossa XML-tiedostojen luominen kirjataan erilliseen tapahtumalokitiedostoon varmuuden vuoksi. Lokitiedosto on tavallinen tekstitiedosto, ja siihen merkitään XML-tiedoston nimi, sekä luomisajankohdan päivämäärä ja kellonaika.

Lokimerkinnän luominen tapahtuu niin, että avataan yhteys palvelimella olevaan tapahtumalokitiedostoon ja annetaan tiedostolle kirjoitusoikeus. Yhteyden avaaminen tapahtuu funktiolla *fopen()*.

Luotavan XML-tiedoston nimi viedään muuttujaan, ja kirjoitetaan tapahtumalokitiedostoon yhdessä luomisajankohdan kanssa. Tämän jälkeen yhteys tiedostoon suljetaan funktiolla *fclose()*.

4.6 Muutokset PHP-sovellukseen PHP-version päivityksen yhteydessä

Toimeksiantajan palvelimella oleva PHP-versio on 4.3.2. Toimeksiantajalle suunniteltu ja toteutettu sovellus on tehty PHP 4:n syntaksin mukaisesti. Tulevaisuudessa toimeksiantajan palvelimella sijaitseva PHP-versio kuitenkin tullaan päivittämään uudempaan.

PHP 4:n DOM XML -laajennus on korvattu uudemmissa PHP-versioissa DOM-laajennuksella (DOM Functions, Php.net 2006). DOM-funktiot ovat syntaksiltaan erilaisia verrattuna DOM XML -funktioihin, eivätkä vanhat funktiot enää toimi uudemmissa PHP-versioissa. Tehtäviini kuului myös dokumentoida tarvittavat muutokset sovellukseen, kun PHP-versio päivitetään tulevaisuudessa uudempaan.

Olen kommentoinut PHP-sovellukseni koodiin jokaisen DOM XML -funktion yhteyteen, miten funktiota tulee muuttaa, jotta sovellus tulee olemaan uudemman syntaksin mukaista.

Esimerkiksi seuraava funktio

```
$doc = domxml_open_mem($xml);
```

korvataan funktioilla

```
$doc = new domDocument();  
$doc->loadXML($xml);
```

ja elementtien luonti ja arvojen tallennus elementteihin tapahtuu seuraavanlaisesti:

```
$source = $doc->createElement('sms_source',  
$_GET['sms_source']);  
$doc->documentElement->appendChild($source);
```

5 Sovelluksen työvaiheet ja testaus

Tehtävänäni oli suunnitella ja toteuttaa toimeksiantajalle sovellus, jonka avulla asiakasyrityksen työnhakijoiden tekstiviestit voitaisiin tallentaa toimeksiantajan palvelimelle. Toimeksiantajalla oli toive, että sovellus olisi mahdollisimman yksinkertainen ja talletettavat tekstiviestit olisivat mahdollisimman yksinkertaisessa muodossa.

5.1 Sovelluksessa käytettävien tekniikoiden valinta

Mietimme toimeksiantajan yhteyshenkilön kanssa, millä tekniikoilla tehtävä olisi toimivuuden kannalta paras toteuttaa. Itse ehdotin ratkaisuksi PHP-sovellusta, sillä minulla on PHP-ohjelmoinnista ennestään kokemusta ja yhteyshenkilöni otti selvää, että toimeksiantajan palvelimella on PHP-tulkki. Ehdotin tekstiviestin tallennusmuodoksi XML-tiedostoa, sillä XML on alustariippumaton tallennusformaatti. Tallennetut tekstiviestit voitaisiin mahdollisesti viedä sellaisinaan johonkin tietokantaan.

Yhteyshenkilö lähetti minulle sähköpostilla QMS-manuaalin sekä lyhyehkön kuvauksen siitä, miten tekstiviestit pitäisi palvelimelle tallentua. QMS-manuaalia tutkimalla selvitin, miten tekstiviestistä voidaan poimia tunnistetietoja PHP-skriptikielellä. Sovelluksen suunnittelu ja toteutus oli mielenkiintoista ja haastavaa. Toteutuksessa ilmenevien ongelmien ratkaisu oli välillä hankalaa ja aikaa vievää, mutta harvoin mikään projekti sujuu täysin ilman vastoinkäymisiä.

5.2 Työympäristö ja testaaminen

Työympäristönä toimi oma kotitietokone ja tehtävää varten asensin tietokoneelleni Xitami-palvelimen, sekä PHP-versiot 4 ja 5. PHP-sovellusta työstin ilmaisen tekstieditorin avulla. Perustin oman hakemiston Xitami-palvelimen juureen PHP-sovellukselle, XML-tiedostoille ja lokitiedostoille. PHP-sovelluksessa on käytetty GET-metodia http-pyyntöä välittämiseen, joten pystyin testaamaan sovellusta Internet-selaimen avulla. Kirjoitin sovellukseni URL-osoitteen selainriville ja osoitteen perään tunnistetietojen parametrit, jolloin ne välittyvät URL-osoitteen mukana.

Esim.

http://localhost/sms/home/sms.php?sms_source=16123&sms_dest=+358401234567&sms_command=POPULUS&sms_params...

Kun sain omalla tietokoneellani valmiiksi toimivan version PHP-sovelluksesta, lähetin sen sähköpostilla yhteyshenkilölleni toimeksiantajalle. Hän perusti toimeksiantajan palvelimelle tarvittavat hakemistot sovellusta varten ja tallensi PHP-sovelluksen palvelimelle. Hän myös konfiguroi QMS-palvelinta niin, että POPULUS-avainsanalla alkavat tekstiviestit ohjataan URL-osoitteeseen, jossa PHP-sovellus sijaitsee. Yhteyshenkilö testasi PHP-sovellusta toimeksiantajan palvelimella niin, että hän lähetti palvelimelta POPULUS-avainsanalla alkavan tekstiviestin itselleen ja vastasi viestiin lähettämällä sen sellaisenaan takaisin palvelunumeroon, mistä alkuperäinen tekstiviesti lähetettiin. Sovellus tallentaa tällöin vastaanotetusta tekstiviestistä tunnistetiedot XML-tiedostoon.

Valitut tekniikat, PHP ja XML, osoittautuivat hyviksi. Palvelua on kehitetty eteenpäin niin, että palvelimelle syntyvät XML-tiedostot noudetaan toisen ohjelman avustuksella tasaisin väliajoin asiakasyrityksen tietokantaan. XML-tiedostot voidaan viedä tietokantaan suoraan, sillä asiakkaan käyttämä Progress-tietokanta tukee XML:n tietorakennetta.

6 Yhteenveto

6.1 Ongelmat

Sovellusta rakennettaessa tuli eteen muutamia erilaisia ongelmia. Ensimmäinen ongelma oli se, että lähdin tekemään sovellusta PHP 5:n DOM-funktioiden avulla ja kuulin vasta myöhemmin, että PHP-versio toimeksiantajan palvelimella on vanhempi ja sama koodi ei toimisi kyseisellä palvelimella. Alkutyö ei mennyt kuitenkaan hukkaan, sillä jouduin palaamaan DOM-funktioihin, kun lopuksi kommentoin sovellukseeni muutokset, jotka täytyy tehdä päivitettäessä PHP-versio uudempaan.

Yksi sovelluksen ehto oli se, että sovellus ei saa tulostaa mitään merkkejä selaimen, ainoastaan otsikkotiedot ovat sallittuja. Jos sovellus tulostaa jotain, lähtee tuloste automaattisesti QMS-palvelimelta vastauksena viestin lähettäjälle. Alkuun sovellusta testattaessa lähti tekstiviestin lähettäjälle aina tyhjä vastausviesti. Sain tutkia koodia melko pitkän aikaa, ennen kuin ymmärsin, minkä vuoksi tyhjä viesti lähtee tekstiviestin lähettäjälle. Koodissa oli yksi tyhjä välilyönti otsikkotietojen tulostuksen jälkeen ja se aiheutti vastausviestin lähtemisen.

Alussa XML-tiedostojen luonti ei onnistunut oikealla tavalla. Tekstiviestin tunnistetiedoista muodostetun XML-dokumentin alussa täytyy olla prologi, jossa kerrotaan XML-versio (version="1.0") ja dokumentissa käytetty merkistö. Tekstiviestin merkistön tuli olla QLA Messaging Server manuaalin-mukaan ISO-8859-1, joten samoin XML-tiedoston merkistön tuli olla sama, jotta viestissä näkyisi myös skandinaaviset kirjaimet oikein. Jostain syystä skandinaaviset kirjaimet kuitenkin sekoittivat sovelluksen ja testatessa tekstiviestin lähetystä, XML-tiedostoon tallentui vain joukko epämääräisiä merkkejä. Kun ongelma ei poistunut lukuisista koodimuutoksista huolimatta, kokeilin vaihtaa XML-tiedostojen merkistöksi utf-8, jonka jälkeen sovellus toimi normaalisti.

6.2 Lopputulos ja arviointi

Toimeksiantajalta saadun tehtävän tuloksena syntyi toimiva PHP-sovellus, joka poimii palvelimelle saapuvasta tekstiviestistä tunnistetiedot omaan XML-tiedostoon. Sovellus kirjaa myös XML-tiedostojen luonnit tapahtumalokiin ja mahdolliset XML-tiedoston tallennuksenaikaiset virheet virhelokiin.

PHP-sovellus on kommentoitu huolellisesti, jotta sovelluksen toiminta on helposti ymmärrettävissä ja siihen on helppo tehdä tarvittaessa muutoksia. Sovellukseen on myös kommentoitu muutokset, jotka täytyy tehdä, kun PHP-versio toimeksiantajan palvelimella päivitetään uudempaan.

Olen myös toimittanut sovelluksesta erillisen dokumentoinnin toimeksiantajalle. Dokumentoinnissa painopisteenä olivat sovelluksen asennus ja konfigurointi, ei niinkään jokaisen funktion toiminnan tarkka selitys.

Työn tavoitteena oli kehittää toimeksiantajan tekstiviestipalvelu kahdensuuntaisesti. Tavoite onnistui, ja nyt toimeksiantajan asiakasyritys voi lähettää työnhakijoille tekstiviestejä avoimista työpaikoista ja työnhakijat voivat vastata tekstiviesteihin, jolloin hakukuus työpaikan vastaanottamisesta voidaan kirjata asiakasyrityksen tietokantaan.

Suoriuduin projektista mielestäni hyvin. Toimeksiantaja oli tyytyväinen sovelluksen toimivuuteen sekä sovelluksen dokumentointiin. Työn tavoite saavutettiin ja olen tyytyväinen että osasin toteuttaa projektin, vaikka alussa olin hieman epävarma sovelluksen onnistumisesta. Tekstiviestipalvelu on otettu onnistuneesti käyttöön pääkaupunkiseudulla toimivassa vuokratyövoimaa välittäväsä yrityksessä.

Lähteet

- DOM functions, Php.net 2006. [online][viitattu 9.9.2006]. <http://fi2.php.net/dom>
- Error handler, Php.net 2006. [online][viitattu 12.9.2006]. http://fi2.php.net/set_error_handler
- Error Handling and Logging Functions, Zend 2006. [online][viitattu 23.10.2006].
<http://www.zend.com/manual/ref.errorfunc.php>
- Error log, Php.net 2006. [online][viitattu 12.9.2006]. http://fi2.php.net/error_log
- Fakta, WM-data Oy 2006. [online][viitattu 23.10.2006].
<http://www.wmdata.fi/wmwebb/media/fakta.asp>
- Henkilöstöhallinto, WM-data Oy 2006. [online][viitattu 23.10.2006].
<http://www.wmdata.fi/wmwebb/Netscape/default.asp>
- Holzner, Steven 2001. Inside XML. Jyväskylä: Gummerus kirjapaino Oy.
- Kuulalaakeri Oy – www-sivut 2004. [online][viitattu 9.9.2006].
<http://www.kuulalaakeri.fi/products.html>
- Käsikirja, Php.net 2006. [online][viitattu 9.9.2006]. <http://fi2.php.net/manual/fi/preface.php>
- Pesonen, Juhon & Löytömäki, Mikko 2003. QLA Messaging Server 2.1.4 User's Manual.
- Tuikka, Tommi & Kanala, Sari 2001. XML Ohjelmoinnin perusteet. Helsinki: Oy Edita Ab.
- Tulostuksen Puskurointi PHP:ssä, Mureakuha 2006. [online][viitattu 23.10.2006].
http://wiki.mureakuha.com/wiki/Tulostuksen_puskurointi_PHP:ssä
- XML, Johdanto, Koulutus ja konsultointipalvelu KK Mediat – www-sivut 2006
[online][viitattu 13.10.2006]. <http://www.2kmediat.com/xml/johdanto.asp>
- XML, Rakenne, Koulutus ja konsultointipalvelu KK Mediat – www-sivut 2006
[online][viitattu 13.10.2006]. <http://www.2kmediat.com/xml/syntaksi.asp>
- XML, Wikipedia 2006. [online][viitattu 23.10.2006]. <http://fi.wikipedia.org/wiki/XML>
- Zandstra, Matt 2001. PHP Trainer Kit. Helsinki: Oy Edita Ab.