TIETOKANTAPOHJAISTEN WWW-SOVELLUSTEN KEHITYSYMPÄRISTÖT

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Teknisen visualisoinnin suuntautumisvaihtoehto
Opinnäytetyö
9.5.2006
Jaakko Teittinen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

TEITTINEN, JAAKKO: Tietokantapohjaisten www-sovellusten kehitysympäristöt

Teknisen visualisoinnin opinnäytetyö, 45 sivua, 2 liitesivua

Kevät 2006

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee tietoturvallisen ja dynaamisen verkkosovelluksen suunnittelua ja toteutusta.

Teoriaosassa tutkitaan verkkopalvelun tuotantoa, dynaamisen web-sovelluksen tekemiseen käytettäviä tekniikoita, tietoturvaratkaisuita ja tietokannan suunnittelua yleisesti sekä henkilöstöhallintajärjestelmän tarpeiden pohjalta. Teoriaosuuden tarkoituksena on antaa kokonaiskäsitys koko web-projektin läpiviemisestä, siihen kuuluvista vaiheista ja sen asettamista vaatimuksista.

Opinnäytetyön case-osuudessa toteutettiin tietokantapohjainen henkilöstöhallintajärjestelmä huomioiden tutkitut verkkosovelluksen erikoistarpeet ja tietoturvan asettamat edellytykset ja rajoitukset. Toteutuksen kannalta oleellisimmat tekniikat olivat palvelinpuolen tekniikat, pääasiassa PHP ja SQL, joita myös teoriaosuudessa käydään läpi. Myös standardin mukaiseen HTML- ja CSS-merkintä otettiin huomioon, vaikkeivät ne sovelluksen luonteesta johtuen olleetkaan etusijalla, eikä niihin (tai muihin asiakaspuolen ohjelmointitekniikoihin) erikseen työssä perehdytä.

Toteutuksen aikana käytetyistä tekniikoista kerättiin kokemuksia, joiden avulla ohjelmaa voidaan kehittää jatkossa.

Avainsanat: PHP, tietokannat, SQL, henkilöstöhallinta

Lahti Polytechnic
Information Technology Studies

TEITTINEN, JAAKKO: Development environments for database-based www-applications

Research of Visualization Technology, 45 pages, 2 appendices

Spring 2006

ABSTRACT

This research deals with project and realization of secure and dynamic web application.

Theory part examines the production and techniques used in making of dynamic web-applications, as well as takes notes on security issues and designing of well structured database, in general and also based on needs of human resources management software. The purpose of theory part is to give comprehensive view of stages and requirements needed to carry out a web-project.

Case-study consists the building of human resources management software, based on investigation in theory part. Special needs for web-application and security issues are taken into account. Most relevant techniques used in the making of application are server side techniques, mainly PHP and SQL, which are also covered in theory part. Standard HTML and CSS marking is also taken into account, although client side techniques weren't the priority of the study, and therefore are not covered in the theory part.

During the realization of the application experiences and ideas were collected to help future development of the software.

Keywords: PHP, MySQL, databases, SQL, human resource management

SISÄLLYS

1 JOHDANTO

Verkkopalvelu voidaan määritellä selaimella käytettäväksi palveluksi tai palveluiden kokonaisuudeksi, jonka avulla käyttäjät haluavat saavuttaa tietyn päämäärän (Parkkinen, 2002). Laadukas verkkopalvelu täyttää tämän tarpeen, ehkä jo ennen kuin käyttäjä osaa sitä määritellä. Laatu on yleensä välillisesti mitattavissa oleva palvelun ominaisuus, jonka kriteerejä ovat esimerkiksi tarkoituksenmukaisuus, toimivuus, yhteensopivuus, kestävyys, vakaus, miellyttävyys, saavutettavuus, helppokäyttöisyys ja virheettömyys. Laatuominaisuus ilmenee suhteessa tuotteen tai palvelun käyttäjän tarpeisiin ja toiveisiin. Laadukkaan verkkosovelluksen toteutuksen keskeinen lähtökohta onkin asiakkaan tarpeiden ymmärtäminen ja niiden tyydyttäminen. (Nykänen, 2003.)

Opinnäytetyön teoriaosa jakaantuu neljään kokonaisuuteen. Ensimmäisessä perehdytään henkilöstöhallintaan ja tutkitaan henkilöstöhallintajärjestelmän erityisvaatimuksia ja rakennetta. Toinen kokonaisuus perehtyy dynaamisen WWW-sovellusprojektin suunnitteluun, tutkii tietoturvamenetelmiä ja vertailee mahdollisia toteutusarkkitehtuureita. Kaksi viimeistä osaa käsittelevät tarkemmin toteutuksessa käytettäviä tekniikoita, PHP:tä ja SQL:ää, sekä perehdyttää tietokannan suunnitteluun ja toteutukseen.

Työn tarkoituksena on paitsi toteuttaa laadukas sovellus asiakkaan käyttöön, myös kertoa mahdollisista vaihtoehdoista verkkopalvelun toteutuksessa sekä tarjota kokonaiskäsitys siitä, mitä vaiheita web-projektin läpivieminen pitää sisällään palvelinympäristön valitsemisesta aina tehokkaaseen lopputuotteeseen saakka.

Opinnäytetyössä ei käsitellä esiteltävien ohjelmointikielten perussyntaksia tai tutkita käytettäviä metodeja pintaa syvemmältä, vaan esitellään niiden soveltuvuutta tutkittavana olevaan aiheeseen.

2 HENKILÖSTÖHALLINTAJÄRJESTELMÄ

2.1 Henkilöstöhallinta

Henkilöstö on yksi organisaation keskeisistä tuotannontekijöistä. Yrityksen tehokkuus ja tuloksellisuus ovat ensisijassa riippuvaisia henkilöstön osaamisesta, innovatiivisuudesta ja motivaatiosta. Henkilöstöhallinta on entistä tärkeämpää, kun kilpailu osaavista henkilöresursseista kiristyy entisestään lähivuosina. (Järvinen, 1996.)

Henkilöstöhallinta voidaan jakaa karkeasti kahteen osaan, henkilöstöhallintoon ja henkilöstövoimavarojen käyttöön. Henkilöstöhallinnolla tarkoitetaan niitä toimenpiteitä, joilla henkilöstövoimavaroja ohjataan palvelemaan organisaation toiminnallisia tavoitteita. Henkilöstöhallintoon kuuluu myös henkilöstötoimen perustehtävien, kuten palkanmaksun, työnohjauksen, työterveyshuollon, työsuojelun ja henkilöstöpalveluiden suorittamista.

Henkilöstövoimavarojen johtamisella tarkoitetaan organisaatiossa toimivien ihmisten hankintaa, motivointia, kehittämistä ja sitouttamista organisaation toimintaan (Kauhanen, 2001). Henkilöstövoimavarojen johtamiseen kuuluu työtehtäviin sopivan henkilöstön valinta ja rekrytointi. Henkilöstövoimavarojen johtamiseen sisältyy myös työntekijöiden jatkuva motivointi työtehtävien tekemiseen, työssä viihtymiseen ja työssä pysymiseen. Työntekijöiden työhön sitouttamisen ja motivoinnin kannalta merkityksellisiä toimia ovat tarpeellisen koulutuksen sekä urakehityksen takaaminen. Siirryttäessä organisaatiokeskeisestä ajattelutavasta yksilökeskeisempään suuntaan, työntekijöiden henkilökohtaisten ominaisuuksien sekä yksilöllisyyden tunnistaminen ja huomioon ottaminen on myös tärkeää. Työntekijöiden työhön sitoutumista parantavat myös olosuhteet työpaikalla. (DeCenzo & Robbins, 1999.)

2.2 Henkilöstöhallintajärjestelmän vaatimukset

Internet mahdollistaa tavallisen pöytäkoneen sovellusmaailmaa monipuolisemman toimintaympäristön. WWW-sovellusten yleisiä ominaispiirteitä ja vaatimuksia voidaan soveltaa myös verkkoon siirret-

tävän henkilöstöhallintajärjestelmän suunnittelussa.

Sovelluksen elinkaaren alussa käyttö- ja tietomäärä on pientä, mutta parhaassa tapauksessa se kasvaa piankin käytön myötä. Järjestelmän pitää pystyä sopeutumaan kasvavaan käyttäjä- ja tietomäärään ilman, että sitä täytyy suunnitella uudestaan tai että sen toimintavarmuus heikkenee. Tietoturvamurtojen yleistyessä myös tietoturvan merkitys on suuri, sillä liiketoiminnan ollessa entistä tiukemmin kiinni Internetissä tietohyökkäysten vaikutus yrityksen kannalta voi olla kohtalokasta. Hyvä saatavuus takaa puolestaan palvelun käytettävyyden ympäri vuorokauden. Pitkät katkokset palvelussa vaikuttavat haitallisesti liiketoimintaan ja aiheuttavat töiden seisomista yrityksessä. Tärkeää on myös järjestelmän luotettavuus, sillä epävakaa ympäristö vaikuttaa palvelun saatavuuteen. Järjestelmän on oltava myös helposti hallittavissa, eikä muutosten tekeminen saa olla liian vaikeaa, sillä uusia vaatimuksia ja kehitysideoita tulee esiin koko ajan. (Ahonen ym., 2004.)

Yhteenvetona voidaan listata edellä esitellyt yleiset vaatimukset verkkopalveluille, kuten henkilöstöhallintajärjestelmälle:

- Skaalautuvuuden avulla palvelu pystyy sopeutumaan niin pieniin kun isoihinkin tietomääriin
- Tietoturva pitää huolen, ettei palvelun sisältämiä tietoja pääsee ulkopuolisten käsiin
- Saatavuus takaa, että palvelu on käytettävissä koko ajan
- Luotettavuus varmistaa toimintavakaan ympäristön
- Hallittavuus mahdollistaa palvelun monipuolisen käytön

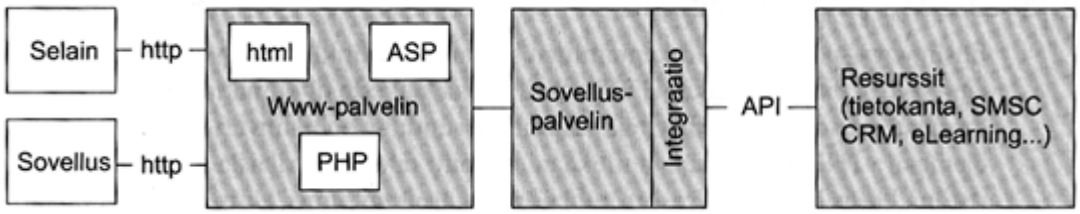
2.3 Henkilöstöhallintajärjestelmän rakenne

Julkisella palvelimella sijaitseva henkilöstöhallintajärjestelmä on rakenteeltaan samantapainen kuin yleisesti käytettävät hajautetut järjestelmät, kuten sisällönhallintajärjestelmät. Se pohjautuu kolmitasoiseen malliin, joka koostuu seuraavista tasoista:

- asiakastaso (selain tai sovellus)
- sovelluslogiikkataso (sovelluspalvelin)
- tietokanta- ja resurssitasosta (integraatio- ja resurssitasot)

Asiakastaso tarjoaa käyttäjälle näkyvän rajapinnan sovellukseen, eli käyttöliittymän. Sovelluslogiikkataso sisältää sovelluksen käyttämän logiikan, eli sovelluksen skriptit, ja tietokanta- ja resurssitaso tarjoaa tiedon säilytyksen. (Ahonen ym., 2004.)

Kolmitasoinen malli tarjoaa joustavammat mahdollisuuden tehdä muutoksia sovelluksessa; muutoksen yhdessä tasossa eivät vaikuta muihin tasoihin. Sovelluslogiikkatasolla se mahdollistaa myös paremman skaalautuvuuden, sillä skriptit eivät ole riippuvaisia asiakas- tai tietokantatasosta. Monipuolistuneiden www-palveluiden ansiosta niiden arkkitehtuurista rakennetta kehitetään edelleen modulaarisempaan ja hallittavampaan suuntaan. N-tasoarkkitehtuuri on kolmitasoisen arkkitehtuurin kehittyneempi muoto, jossa sovellus on jaettu edelleen pienempiin osiin. (Ahonen, 2004.)



Kuva 1: N-tasoarkkitehtuuri www-palveluissa

2.4 Henkilöstöhallintajärjestelmän tietoturva

Yrityksen Internetin kautta käytettävän verkkopalvelun tietoturva on erityisen tärkeää, jos palvelu sisältää informaatiota, joka ei saa joutua ulkopuolisten käsiin. Yrityksen liikesalaisuuksien säilyttämisen lisäksi henkilörekisterin pitäjien asemaa säätelee Suomen henkilötietolaki, joka velvoittaa henkilörekisterinpitäjää huolehtimaan, etteivät rekisterissä olevien henkilöiden arkaluonteiset tiedot joudu ulkopuolisten käsiin (Henkilötietolaki, 5 §).

3 DYNAAMISET WWW-SOVELLUKSET

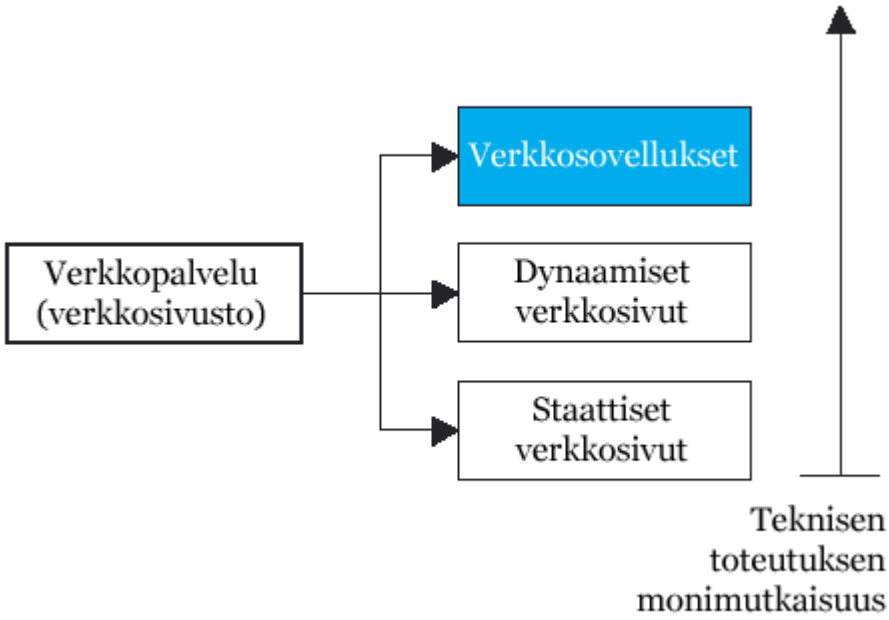
3.1 Verkkopalvelut

3.1.1 Verkkopalvelun määritelmä

Verkkopalvelu on kokonaisuus, jonka on määrä toimia käyttäjän apuvälineenä jonkin tietyn tehtävän suorittamisessa. Se on palveluiden kokonaisuus, jota käytettään halutun päämärän saavuttamiseen. Ennen kuin verkkopalvelua aletaan toteuttaa, on suunnitteluvaiheessa tarpeen miettiä:

- Kuka on palvelun käyttäjä?
- Millaisia tehtäviä käyttäjällä on?
- Mitkä ovat palvelun tavoitteet?
- Mitä palvelu sisältää?

Jos kysymyksiin ei löydy järkeviä vastauksia, palvelun tarpeellisuutta on syytä vielä miettiä. (Parkkinen, 2002.)



Kuva 2: Verkkopalvelu muodostuu erilaisista verkkosivuista

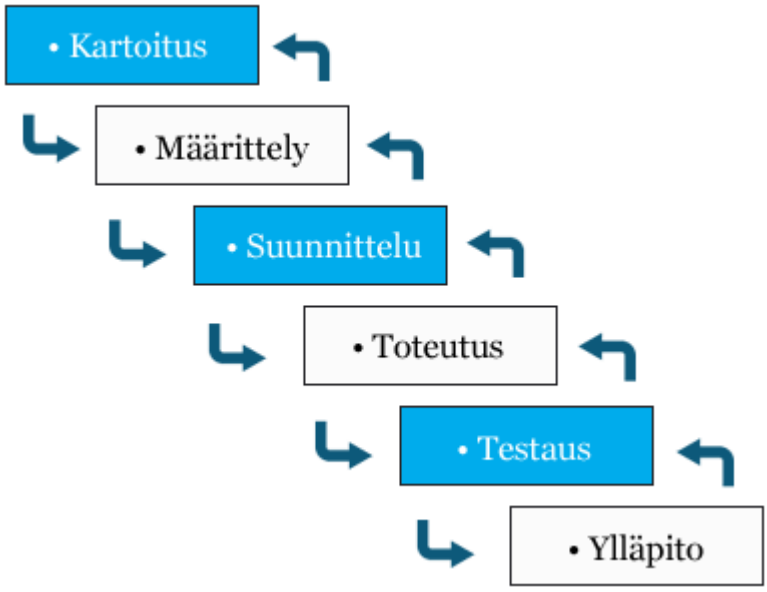
Verkkopalvelun perustaminen ja ylläpito on tavoitteellista toimintaa. Pääasiassa sitä käytetään taloudellisten tavoitteiden toteuttamiseen, esimerkiksi kustannusten alentamiseen, työn tehostamiseen tai markkinoiden laajentamiseen. Taloudellisten tavoitteiden lisäksi on syytä asettaa myös laadullisia ja kehitystavoitteita, joita voivat olla

esimerkiksi parempi asiakaspalvelu tai parempi verkkopalvelu. Tavoitteiden märittely on tärkeä osa verkkopalvelun suunnittelua, sillä perustavoitteen puuttuminen johtaa hyvin todennäköisesti päämäärättömään toteutukseen. Tavoitemäärittely käsittää palvelun tavoitteet, asiakasmäärittelyn, palveltavat prosessit ja toteutustavoitteet.

Verkkopalvelu on selkeä ja näkyvä osa organisaation toimintaprosessia. Siksi palvelulla tulee olla hyväksytyt tavoitteet, jotka tukevat organisaation toiminnan kokonaistavoitetta.

3.1.2 Tuotantoprosessi

Verkkopalvelun tuotantoprosessi voidaan kuvata esimerkiksi vesiputousmallin mukaisesti. Pienemmissä web-projekteissa ei ole välttämättä tarpeellista noudattaa kaikkia yleisiä projektinhallinnan työkaluja, kuten kustannusseurantaa, markkina-analyyseja, kampanjointia tai julkistamista. Tuotantoprosessin tärkeimmät vaiheet on kuitenkin tarpeen toteuttaa huolella.



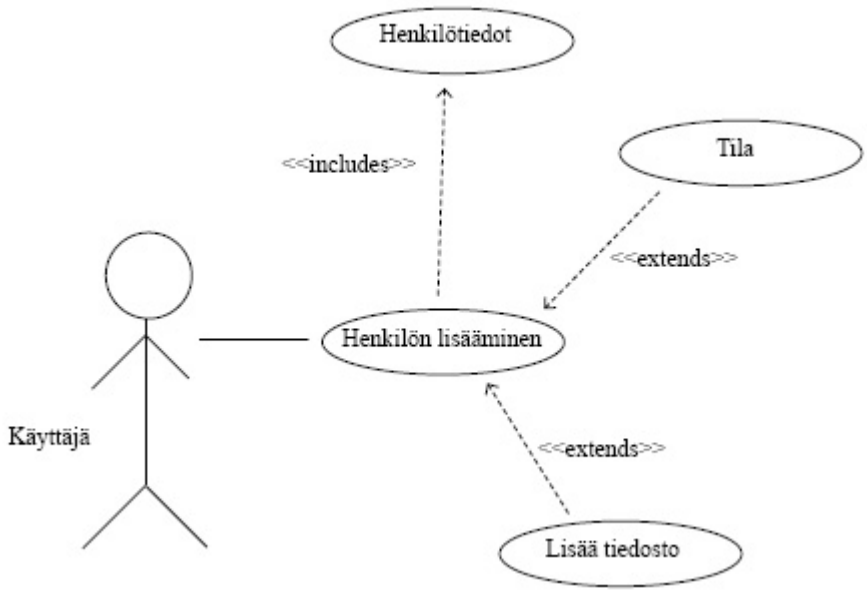
Kuva 3: Esimerkki yksinkertaisesta web-projektin vesiputousmallista

Web-palvelut ovat viimeisten vuosien aikana muuttuneet tavoitteettomista ja aikatauluttomista luomisprosesseista laajoiksi projekteiksi, joiden läpiviemiseksi tarvitaan tarkkaa projektinhallintaa ja tuo-

tannollista ajattelua. Web-palveluiden sulautuessa yhä enemmän yrityksen muuhun toimintaan, kuten markkinointiviestintään, niiden suunnittelussa on ymmärrettävä yhä enemmän asiakkaan liiketoimintastrategioita. (Mielonen ym., 1998.)

3.1.3 Sovellussuunnittelu ja rakenne

Ennen sovelluksen teknistä toteutusta on tarpeen suunnitella sovelluksen rakennetta ja sisältöä. UML (Unified Model Language) on standardoitu visuaalinen mallinnuskieli sovelluksen määrittelyyn, havainnollistamiseen ja toimintalogiikan kuvaukseen ennen kuin varsinainen sovellus on käytössä. UML:n käyttö on suosittua etenkin oliopohjaisten sovellusten mallintamisessa. (Erikson, 2000.)



Kuva 4: UML-käyttötapauskaavio uuden henkilön lisäämisestä järjestelmään. Includes on ns. pakollinen käyttötapaus, valinnaiset extends-käyttötapaukset täydentävät henkilötietoja.

Verkkopalvelun rakennetta suunniteltaessa on tarpeen määritellä kuinka tiedot organisoidaan, jotta palvelun käyttäjä löytää etsimänsä nopeasti ja helposti. Palveluissa tarvitaan aina looginen rakenne, jonka rungon määrittelee sivuston sisällön organisointi. Siksi on tärkeää, että koko sivuston pystyy hahmottamaan kerralla. Koko sivuston rungoksi voidaan suunnitella sivustokartta, josta näkee pääpiir-

teissään sivuston toiminnan. Sisällön ja sivukartan määritelmän jälkeen voidaan laatia sivun tarkkuudella tehty käsikirjoitus, josta käy ilmi kunkin sivun sisältö ja käytettävät elementit. (Goto ym., 2003.)

3.1.4 Yleiset käytettävyystekijät

Käyttäjän ensikokemus palvelusta syntyy käyttöliittymästä, siitä miltä se tuntuu ja näyttää. Käyttöliittymällä tarkoitetaan tässä verkkosovelluksen ja käyttäjän välistä rajapintaa, siksi käyttöliittymällä ja palvelun käytettävyydellä on suuri merkitys siihen, kuinka tehokkaasti palvelua käytetään.

Käytettävyys on iso osa minkä tahansa laitteen tai sovelluksen käyttökelpoisuutta. Tunnetun käytettävyystutkijan Jacob Nielsenin määritelmän mukaan käytettävyys jakautuu viiteen osaan: opittavuuteen, tehokkuuteen, muistettavuuteen, virheettömyyteen ja miellyttävyyteen. Opittavuudella tarkoitetaan sitä, kuinka nopeasti ja helposti uusi käyttäjä oppii sovelluksen toimintalogiikan ja käyttämisen. Tehokkuudella tarkoitetaan käyttäjän mahdollisuuksia saada sovelluksesta enemmän irti opittuaan sen perustoimintalogiikan. Siksi sovelluksen pitää pystyä tarjoamaan erilaisia vaihtoehtoja tehtävien suorittamiseen. Muistettavuudella tarkoitetaan sitä, miten helppoa jo aiemmin sovelluksen käytön oppineen henkilön on palauttaa mieleen sovelluksen käyttö ja sen toiminnallisuus. Tähän vaikuttaa muun muassa suunnittelun pysyvyys ja visualiset elementit. Virheettömyydellä tarkoitetaan käyttäjän suorittamissa toimenpiteissä tapahtuvien virheiden määrää, virheiden tekeminen tulee tehdä mahdollisimman vaikeaksi. Miellyttävyydellä tarkoitetaan sovelluksen käytön mukavuutta, sekä käyttäjän tyytyväisyyttä järjestelmän käyttöön ja sen vuorovaikutukseen. (Parkkinen, 2002.)

Toinen lähestymistapa käytettävyyteen on Kansainvälisen standardoimisliiton määritelmä ISO 9241-11, joka on Nielsenin kaupalliseksi kutsuttua näkökulmaa tieteellisempi. Sen mukaan yleistä määritelmää hyvälle käytettävyydelle ei ole, vaan se riippuu aina käyttäjästä ja käyttötilanteesta, eli kontekstista. Jos konteksti on jossain määrin tunnettu, voidaan käytettävyyden avulla mitata työn tehokkuutta, taloudellisuutta ja miellyttävyyttä. (Parkkinen, 2002.)

3.1.5 Käytettävyysongelmat

Käytettävyysongelmia voidaan havaita ja poistaa soveltamalla esimerkiksi yleisesti käytettyä Nielsenin 10 kohdan tarkistuslistaa. (Laakso, 2002.)

- Näkyvyys
  Käyttäjän on hyvä tietää missä tilassa järjestelmä on. Järjestelmän on tarjottava myös mahdollisuus palautteelle.

- Luonnollinen ilmaisutapa
  Käytetään käyttäjälle tuttua kieltä ja selviä käsitteitä. Tieto on loogisessa järjestyksessä.

- Kontrolli käyttäjällä
  Käyttäjä on vapaa liikkumaan tilasta toiseen, myös perumismahdollisuus oltava.

- Yhdenmukaisuus ja standardit
  Termien käytössä on pyrittävä johdonmukaisuuteen. Lisäksi noudatetaan toteutusympäristön standardeja.

- Virheiden estäminen
  Käyttäjän virheet pyritään välttämään jo etukäteen.

- Tunnistaminen ja muistaminen
  Järjestelmän suorittamat toimenpiteet ovat näkyviä. Käytetään hyödyksi järjestelmä mahdollisuutta tallentaa tietoja käyttäjän sijaan.

- Käytön tehokkuus
  Oikopolkuja on luotava tehokkaamman käytön mahdollistamiseksi. Käyttäjälle voi myös tarjota mahdollisuuden muokata usein käyttämiään toimintoja.

- Minimalistinen suunnittelu
  Turha tieto poistetaan käyttöliittymästä ja harvemmin tarvit-

tavat tiedot laitetaan vähemmän selkeästi esille.

- Virheistä toipuminen
  Virheilmoitusten oltava ymmärrettäviä niin, että ne kuvaavat ongelman ja esittävät siihen ratkaisun.

- Opastus ja käyttöohjeet
  Yleensä on tarkoituksenmukaisempaa, että järjestelmää voi käyttää myös ilman käyttöohjeita. Käyttöohjeet eivät saa olla liian laajoja ja toimenpiteiden tulee olla hyvin jäsennellyt.

Samanlaiset käytettävyysongelmat toistuvat palvelusta toiseen, koska suunnittelutavat ovat usein hyvin samankaltaisia. Käytettävyystestaus on tehokas keino palvelun kehitykseen jo ennen sen julkaisemista. Jos käyttäjä eivät osaa hyödyntää palvelua, eivät he tule takaisin. (Goto ym., 2003.)

3.1.6 Sisältö

Huolellisesti suunniteltu ja teknisesti hyvin toteutettu sovellus ei vielä sinänsä takaa hyvää verkkopalvelua. Tarvitaan sisältöä, ylläpitoa ja sovelluksen jatkokehitystä.

Verkkopalvelun sisällön tulisi olla ajankohtaista ja kattavaa, sillä tutkitusti ihmiset saadaan parhaiten palaamaan palveluun nimenomaan laadukkaan sisällön ansiosta (Goto ym., 2003). Sisällöntuotannon on oltava hyvin suunniteltu loogisen rakenteen säilyttämiseksi. Palvelun ylläpito ja kehittäminen on jatkuva prosessi, palvelun sisältöä on tarkistettava jatkuvasti ja pidettävä se ajan tasalla. Verkkopalvelujen hyödyt saavutetaan aina varsinaisen tuotantokäytön aikana, siksi on tärkeää, että palvelun ylläpito ja kehitys suunnitellaan huolellisesti aina pidemmäksi ajaksi, esimerkiksi kahdeksi vuodeksi, kerrallaan. Hyviä perusteina verkkopalvelun kehittämispäätökselle ovat asiakaspalaute, edeltävien palveluiden ongelmat ja kehittämistarpeet. (Juhta, 2005; Goto ym., 2003.)

3.2 Tietoturva

3.2.1 Tietoturvan perusperiaatteet

Organisaatiot siirtävät palveluitaan yhä enemmän Internetiin hajautetun arkkitehtuurin suuntaan, jossa palvelimet ja työasemat kommunikoivat keskenään verkon välityksellä. Tietoverkot ovat mullistaneet tavan, jolla yritykset harjoittavat liiketoimintaa, mutta samalla verkkoihin kohdistuvat hyökkäykset synnyttävät valtavia riskejä, ja voivat aiheuttaa rahallisia ja ajallisia, sekä tuotteisiin, maineeseen että luottamuksellisiin tietoihin kohdistuvia menetyksiä. (Allen, 2002.)

Tietoturvan tavoitteena on suojata verkossa liikkuvaa tietoa, käytössä olevia laitteita ja yhteyksiä erilaisia uhkia vastaan. Dynaamisten verkkosovellusten lisääntyessä myös tietoturvan merkitys kasvaa, sillä verkon yli kulkee entistä enemmän tietoja käyttäjien ja palvelinten välillä. Osa tietoturvauhista on teknisiä, esimerkiksi laitteiston hajoamisesta aiheutuvia ongelmia, mutta suurin osa tietoturvauhista aiheutuu ihmisen omasta toiminnasta. Uhka käsitetään yleensä ulkopuolelta tulevaksi, jolloin esimerkiksi luottamuksellista tietoa voi joutua hyökkääjän käsiin, mutta myös omien työntekijöiden tekemät virheet ja huolimattomuus aiheuttavat ongelmia. Siksi tietoturvan kannalta myös henkilöstön opastaminen ja koulutus sovellusten käytössä on vähintään yhtä tärkeää kuin tiedon tekninen ja fyysinen suojaaminen.

Tietoturvallisuus rakentuu tiedon kolmen ominaisuuden eli luottamuksellisuuden, eheyden ja saatavuuden turvaamisella. Tietoturvaan liittyy lisäksi kiistämättömyyden, pääsynvalvonnan ja todennuksen tekninen toteuttaminen. (Kerttula, 2000.)

- Luottamuksellisuus
  Luottamuksellisuudella tarkoitetaan sitä, että tietoon on pääsy vain siihen oikeutetulla henkilöllä. Jos esimerkiksi salaiseksi luokiteltu tieto joutuu sellaisen henkilön käsiin, jolla siihen ei ole oikeutta, tiedon luottamuksellisuus on menetetty. Luottamuksellisuus edellyttää tiedoston salausta siirron ja säilytyksen aikana.

- Eheys
  Tiedon eheys tarkoittaa, että tieto pysyy muuttumattomana siirron sekä tallennuksen aikana, eli sitä ei ole muutettu, poistettu tai ettei siihen ole lisätty mitään. Tiedon eheyden varmistamiseen liittyy aina lähettäjän todennus. Eheyden rikkominen ei vaaranna tiedon luottamuksellisuutta, mutta muuttaa sen sisältöä.

- Saatavuus
  Tiedon saatavuus käsittää teknisten laitteiden toimivuuden, ja sen että tieto on niiden käytettävissä, joille se on tarkoitettu, ja juuri silloin kun sitä tarvitaan.

- Pääsynvalvonta
  Pääsynvalvonnan tarkoitus on varmistaa osaltaan tiedon luottamuksellisuus ja eheys ja sallia käyttäjälle pääsy vain ennalta määrättyihin tietoihin tunnistuksen jälkeen.

- Kiistämättömyys
  Kiistämättömyydellä tarkoitetaan, että tiedon lähettäjä tai vastaanottaja tai tietoon liittyvä tapahtuma voidaan varmistaa luotettavasti myös jälkikäteen. Kiistämättömyyteen voidaan pyrkiä käyttämällä esimerkiksi kolmannen osapuolen varmenteita.

- Todennus
  Tärkein tietoturvaperiaate on todennus, jonka tehtävänä on varmistaa, että osapuolet ovat niitä, joita sanovat olevansa. Ilman todennusta muilla tietoturvaominaisuuksilla ei ole paljoakaan merkitystä.

Tiedon luottamuksellisuutta ja eheyttä pyritään lisäämään erilaisilla todennus- ja salausmenetelmillä, jotka osaltaan tukevat tietoturvaan vaikuttavien kuuden perusperiaatteen toteutumista. Yhdessä tai erikseen käytettynä näiden menetelmien avulla saadaan luotua käyttäjän ja WWW-palvelimen välille luottamussuhde, jolloin asianosaiset voivat luottaa olevansa tekemisissä oikean osapuolen kanssa. (Allen, 2002.)

3.2.2 Osoitepohjainen todennus

Palvelinohjelmistot tukevat usein IP-osoitteisiin ja DNS-tietoihin perustuvia käyttörajoitetoimintoja. Osoitepohjainen todennus määrittelee sallitut osoitteet ja koneet, jotka voivat sovellusta käyttää. Ongelmana on kuitenkin IP- ja DNS-spoofing, eli IP-osoitteen ja DNS-tietojen väärentäminen, joka rajoittaa menetelmän tehokkuutta.
Osoitepohjaisen todennuksen sijaista tai tueksi suositellaan käytettäväksi jotain salausmenetelmää. Tietoon käsiksi pääsyä voidaan rajoittaa neljällä yleisiin standardeihin perustuvalla tekniikalla, joita ovat: perustodennus, Secure Socket Layer (SSL), Secure HTTP (S/HTTP) ja Secure Electronic Transaction (SET).

3.2.3 Perustodennus

Perustodennuksen avulla suojataan www-palvelimien sisältöä käyttäjätunnusten ja salasanojen avulla. Turvallisuuden kannalta perustodennuksen huono puoli on se, että salasanatiedot eivät ole salattuja, vaan ne pystytään varsin helposti tulkitsemaan. Myös sivun sisältö liikkuu verkossa salaamattomana selkokielisessä muodossa. Perustodennusta tukevat kaikki standardin mukaiset WWW-selaimet, ja se voidaan toteuttaa esimerkiksi käytössä olevan ohjelmointitekniikan avulla. (Allen, 2002.)

3.2.4 Secure Socket Layer (SSL)

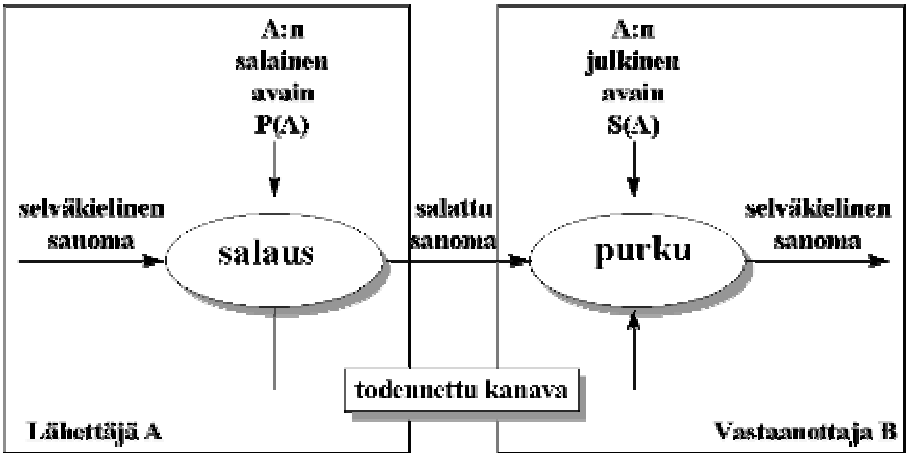
SSL on Netscapen kehittämä, maailmanlaajuisesti hyväksytty protokolla, joka takaa salatun kommunikoinnin asiakkaan ja palvelimen välillä. SSL:ää käytetään laajasti eri selaimissa ja WWW-palvelimilla, mutta sitä voidaan hyödyntää myös esimerkiksi FTP- tai telnet-yhteyksillä. SSL on suositeltavin menetelmä, jos tarvitaan luotettavaa tiedonsiirto- ja salausmenetelmää, sillä se tarjoaa tärkeimmät turvapalvelut, eli luottamuksellisuuden, todentamisen ja eheyden.

SSL koostuu kahdesta aliprotokollasta, eli SSL-siirtoprotokollasta, jota käytetään tietoa siirrettäessä, sekä SSL-kättelyprotokollasta, jol-

la hallitaan asiakkaan ja palvelimen välisiä yhteyksiä. Kättelyprotokolla on kaksivaiheinen, siihen kuuluu palvelimen tunnistaminen sekä tarvittaessa myös käyttäjän tunnistaminen. Kättelyn jälkeen SSL-siirtoprotokollaa käytetään molempiin suuntiin liikkuvan tiedon salaukseen ja avaamiseen. Järvinen, 2003.)

SSL käyttää toisen osapuolen identiteetin varmennukseen varmentimia. Varmenne eli sertifikaatti on sähköinen todistus, joka sisältää joukon varmenteen myöntäjän tarkistamia ja oikeaksi todentamia tietoja. Varmenne voi sisältää muun muassa henkilön julkisen avaimen, henkilötiedot, varmenteen voimassaolopäiväyksen sekä varmenteen myöntäjän sähköisen allekirjoituksen. (Järvinen, 2003.)

Avaimiin perustuvassa todennuksessa voidaan käyttää joko symmetristä tai epäsymmetristä salausta. Symmetrisessä salauksessa tieto salataan ja puretaan samalla avaimella. Epäsymmetrisessä salauksessa käytetään avainparia, joista toinen avain on julkinen ja toinen yksityinen. Julkisella avaimella salattu viesti voidaan avata avainparin yksityisellä avaimella ja päinvastoin. SSL-suojauksessa käytetään julkisen avaimen tekniikkaa, eli epäsymmetristä salausta.



Kuva 5: Epäsymmetrisen salauksen käyttö lähettäjän todentamisessa

3.2.5 Secure HTTP (S/HTTP)

Vuonna 1993-1994 kehitetty Secure HTTP on toiminnoiltaan laajempi kuin SSL, mutta se vaatii käyttäjältä tietynlaista selainta, ja on siksi jäänyt taka-alalle. S/HTTP on HTTP-protokollassa käytettävä

salausmenetelmä, joka sisältää neljä ominaisuutta: Todennus, salaus, kryptograafiset tarkistussummat ja digitaaliset allekirjoitukset. (Allen, 2002.)

3.2.6 Secure Electronic Transaction (SET)

SET on kahden tärkeimmän luottokorttiyhtiön kehittämä menetelmä, joka tarjoaa protokollan lisäksi myös infrastruktuurimäärityksen, joka mahdollistaa pankkimaksutoiminnot. SET-protokollan käyttö vaatii, että käyttäjä hankkii luottokorttiyhtiöltään ohjelmiston ja muuta tukimateriaalia. (Allen, 2002.)

3.2.7 Muita menetelmiä

Phil Zimmermannin kehittämä ohjelmistopaketti PGP (Pretty Good Privacy) on yleistynyt sähköpostien turvaratkaisuna. PGP tarjoaa turvapalveluina luottamuksellisuuden ja digitaalisen allekirjoituksen, ja se perustuu julkiseen avaimeen. (Kinnunen, 2001.)

3.3 Dynaamisten WWW-sovellusten arkkitehtuurit

3.3.1 Palvelinpohjainen ohjelmointi

Palvelinpuolen ohjelmointikielien avulla on mahdollista luoda ja suorittaa dynaamisia, vuorovaikutteisia sovelluksia, joita voi kehittää ja muokata joustavasti. Palvelinpohjaisuus tarkoittaa sitä, että koodi suoritetaan käyttäjän pyynnön jälkeen aina palvelimella, jolloin se ei vaadi mitään erityistä tukea käyttäjän selaimelta. Se tarkoittaa myös sitä, että sovelluksella on pääsy esimerkiksi palvelimen tiedostoihin ja tietokantoihin. Tästä johtuen dynaamiset WWW-sovellukset toteutetaan lähes poikkeuksetta palvelinpohjaisilla tekniikoilla.

Palvelinpuolen ohjelmoinnin (server-side programming) vastakohta on asiakaspuolen ohjelmointi (client-side programming), joka suoritetaan aina käyttäjän omalla koneella. Se mahdollistaa HTML-

dokumentin yksinkertaisen vuorovaikutteisuuden ilman palvelimen tukea, mutta varsinaista dynaamisuutta sillä ei saada aikaiseksi. Asiakaspuolen ohjelmointikielistä yleisiä ovat HTML, JavaScript ja CSS. (Rantala, 2002.)

Dynaamiset WWW-palvelut voidaan karkeasti toteuttaa kahdella eri tapaa, joko käyttäen palvelimelle asennettua CGI-ohjelmaa tai hyödyntäen upotettuja tekniikoita. Upotetut tekniikat tarkoittavat HTML-dokumenttien sekaan kirjoitettuja tai lisättyjä skriptejä, jotka toimivat usein WWW-palvelimien laajennuksina. (Rantala, 2002.)

3.3.2 Web-palvelimet

Palvelimella voidaan tarkoittaa verkossa kiinni olevaa fyysistä tietokonetta tai asiakasohjelmilta tietoa vastaanottavaa tietokoneohjelmaa. Web-palvelimella tarkoitetaan pääsääntöisesti sellaista ohjelmistoa, joka vastaa asiakkaiden eli WWW-selaimien lähettämiin tiedostopyyntöihin. Web-palvelimena voi toimia lähes mikä tahansa laite, sillä sopivia ohjelmistoja löytyy lähes kaikille mahdollisille käyttöjärjestelmille. Verkossa olevalla tietokoneella voi olla käynnissä myös muita palvelinohjelmia, kuten MySQL-tietokantapalvelin. (Heinisuo, 2001.)

Apache on Unix-alustalla tavanomaisin palvelinohjelmisto, joka tosin on saatavilla myös Windowsille ja useille muille käyttöjärjestelmille. Apache perustuu avoimeen lähdekoodiin, se on erittäin luotettava, nopea sekä ilmainen. Apache onkin tällä hetkellä suosituin web-palvelinohjelmisto, jota esimerkiksi Internet-palveluntarjoajat käyttävät kotisivupalveluiden tarjoamiseen. (KK Mediat, 2000f)

Internet Information Server (ISS) on Microsoftin kehittämä palvelinohjelmistokokonaisuus ja suosituin palvelinohjelmisto Windows-ympäristössä. IIS on käytössä lähinnä yritysten Windows NT -palvelimissa.

Www-palvelinohjelmistoja löytyy paljon muitakin, mutta niille on vain vähän tarvetta, sillä kaksi suosituinta ovat ilmaisuutensa vuoksi lähes kaikkien ulottuvilla. (Heikniemi)

3.3.3 CGI-arkkitehtuuri

Palvelimella sijaitsevien ohjelmien kommunikointi verkkosivujen kanssa tapahtuu CGI-standardin (Common Gateway Interface) avulla. Sen tehtävänä on määritellä missä muodossa parametrit välitetään palvelimella olevalle sovellukselle, ja miten sovelluksen prosessoima tuloste välitetään takaisin asiakkaalle (Heinisuo, 2001). CGI oli ensimmäisiä käyttökelpoisia ja yleisesti hyödynnettyjä tekniikoita vuorovaikutteisten web-sivujen luomisessa, ja se avasi paljon uusia mahdollisuuksia WWW-sovellusten alalla. Sen käyttö yleistyi hyvin nopeasti, ja monet palvelimet alkoivat tukea sitä. (Ahonen ym., 2004.)

Palvelimilla sijaitsevia ohjelmia kutsutaan CGI-ohjelmiksi. CGI-ohjelmia voi kirjoittaa millä tahansa komentoriviltä ajettavalla ohjelmointikielellä. Tällaisia ohjelmointikieliä ovat muun muassa Perl, C/C++ ja Pascal. Myös esimerkiksi PHP voi toimia CGI:n osana, joskaan se ei ole kovinkaan yleistä. Palvelimen näkökulmasta CGI toimii siten, että tietyt osoitteet on määritetty suoritettavaksi ajettavina ohjelmina (esimerkiksi http://www.esimerkki.com/testi.cgi). Kun tätä URL:ia haetaan selaimella, kutsutaan vastaava (testi.cgi) ohjelma. Ohjelman tuottama tuloste kerätään web-palvelimelle ja lähetetään asiakasohjelmalle.

Lähes kaikki web-palvelimet tukevat nykyään CGI-rajapintaa, joten sen saatavuus on varsin hyvällä mallilla. CGI:n merkittävimpiä puutteita on sen huono suorituskyky. Kun WWW-selaimelta tulee CGI-sovellukselle ohjattava palvelupyyntö, tulee CGI-sovelluksen ajamiseksi ja perille viemiseksi avata aina uusi prosessi. Palvelin joutuu siis muodostamaan uuden prosessin aina uuden palvelupyynnön tullessa, mikä kuluttaa aikaa ja palvelimen resursseja. Open Market –niminen yritys on kehittänyt FastCGI vaihtoehdon, joka luo yhden pysyvän prosessin jokaista FastCGI-sovellusta kohden, ja näin ollen parantaa tehokkuutta. (Ahonen ym., 2004.)

CGI:llä on ongelmia myös tietoturvan kanssa, sillä se jää aina ohjelmoijan vastuulle. Sovellukset suoritetaan web-palvelimessa ylläpitä-

jän asettamilla oikeuksilla, jolloin vähäisilläkin oikeuksilla CGI-sovellukset pääsevät käsiksi kaikkiin web-palvelimen tiedostoihin, joita ei ole erikseen suojattu.
Kaikkien CGI-sovellusten yhteinen piirre on HTML:n tai tulosten kirjoittaminen ohjelmakoodin tulostuksissa, jolloin sovelluksen hallittavuus edellyttää valitun ohjelmointikielen jonkinlaista osaamista, joka puolestaan rajoittaa hallittavuutta. (Ahonen ym., 2004.)

3.3.4 ASP-arkkitehtuuri

ASP (Active Server Pages) on Microsoftin kehittämä palvelinpohjainen teknologia, joka yhdistää HTML:n, skriptikielen, sekä palvelinpuolen tietokannat toisiinsa. Sen avulla voi luoda tehokkaita Web-pohjaisia sovelluksia. Tavallisemmin ASP sivuissa on käytetty ohjelmointikielenä VBScript- tai JScript-kieliä. Perussyntaksiltaan ASP on varsin helppo, sillä tarvittava koodi upotetaan HTML-koodin sekaan, mikä luo hyvät edellytykset helpolle hallittavuudelle. (Peltomäki ym., 2000.) (KK Mediat 2000c.)

Vanhempaan CGI-menetelmään verrattuna ASP on huomattavasti yksinkertaisempi ja tehokkaampi. Omia COM (Component Object Model) –komponentteja luomalla sovellus voidaan jakaa moduuleiksi, joita pystytään käyttämään uudelleen esimerkiksi toisesta komponentista tai ohjelmasta. (KK Mediat 2000a.)

Pääasiallisesti yrityskäytössä olevia ASP -sivuja voidaan ajaa lähinnä Microsoft Internet Information Server –palvelimelta. Tämä sitoo sen osaksi yhtenäistä palvelinjärjestelmää, mutta toisaalta tekee sen palvelinriippuvaiseksi (Ahonen ym., 2004). Kaupallisen sovelluksen heikko soveltuvuus erilaisiin ympäristöihin voidaan laskea ASP:n heikkoudeksi. Myös Microsoftin IIS –palvelinten tietoturvaongelmat ovat tunnettuja (ITviikko, 2003).

ASP:n uusin versio on Microsoftin .net-arkkitehtuurin perustuva, täysin uudelleenkirjoitettu ohjelmisto ASP.net, joka olio- ja tapahtumapohjaisena muistuttaa hyvin paljon Windows-ohjelmointia. ASP.net alustalla voidaan käyttää mitä tahansa .net-alustan tukemaa kieltä, joista tavallisempia ovat VBScript, Jscript.net, Visual Ba-

sic.net ja C#. Sen vahvuuksia ovat parannettu skaalautuvuus ja suorituskyky, sekä uudistetut tietoturvaominaisuudet.

3.3.5 PHP teknologia

PHP (Hypertext Preprocessor) on avoimeen lähdekoodiin perustuva tulkattava skriptikieli, joka oli alun perin joukko WWW-pohjaisten sovellusten tekemistä helpottavia skriptikokoelmia. Tulkattavuus tarkoittaa, että PHP-koodi ajetaan joka kerta, kun sitä palvelimelta pyydetään. PHP-tulkki toimii usein Apachen web-palvelimen sisäisenä moduulina, jolloin PHP-koodia sisältävien tiedostojen tulkkaaminen on suhteellisen nopeaa. PHP-tulkki ja -moduuli ovat saatavilla lähes kaikille käyttöjärjestelmille ja web-palvelimille, joten se on alustasta riippumaton. Lisäksi se on täysin ilmainen. Näiden ominaisuuksien johdosta siitä on tullut yksi suosituimmista web-sovellusten toteutusarkkitehtuureista. (Heinisuo, 2001.) (Rantala, 2002.)

Syntaksiltaan PHP on Perlin ja C -kielen risteytys. PHP-koodi voidaan kirjoittaa suoraan HTML-kielen sekaan, mikä tekee siitä helposti ymmärrettävää ja käytettävää. PHP on erittäin monipuolinen ja laajennettavissa oleva ohjelmointialusta, jonka moduulirakenne mahdollistaa erilaisten lisätoiminnallisuuksien liittämisen PHP:hen. Tällainen on esimerkiksi GD-grafiikkakirjasto, joka mahdollistaa alkeelliset kuvankäsittelytoiminnot palvelinpäässä. Web-kehityksen kannalta PHP:n tärkeimpiä piirteitä ovat lisäksi suora tuki useille yleisimmistä tietokantapalvelimista (kuten MySQL, Oracle, Sybase, mSQL ja PostgreSQL) ja viestintäprotokollille (kuten IMAP, POP3 ja HTTP) (KK Mediat 2000b). PHPN täydellinen olio-ohjelmointituki mahdollistaa lisäksi ASP:ista tuttujen komponenttien käyttämisen.

Koska PHP ja usein sen kanssa käytettävät ohjelmistot, esimerkiksi Apachen web-palvelin, ovat täysin ilmaisia, niille löytyy nykyään runsaasti tukea. Lähes kaikki web-hotellipalvelut esimerkiksi tarjoavat mahdollisuuden PHP-sovellusten käyttöön. Suosionsa johdosta tarjolla on myös paljon dokumentaatiota ja oppaita, joten PHP:n käytön aloittaminen on varsin helppoa.

PHP:n heikkoudet ovat sen suorituskyvyssä ja tietoturvassa. Kun PHP-sovellukset tulkataan aina jokaisen palvelupyynnön yhteydessä, vasteaika kasvaa. Ongelman ratkaisemiseksi PHP:n kehityksessä vahvasti mukana oleva Zend on kehittänyt kaupallisia sovelluksia, jotka tehostavat PHP:n suorituskykyä tallentamalla kertaalleen suoritetut sovellukset palvelimen muistiin. PHP:ssä on useita hyödyllisiä funktioita valmiina tietoturvan parantamiseksi, mutta niiden hyödyntäminen jää kokonaan ohjelmoijan vastuulle. Tästä johtuen tietoturva PHP-sovelluksilla voi olla hyvinkin heikko. Myös skriptikielen nopeaa kehitystä voidaan pitää jonkinlaisena ongelmana, sillä se voi aiheuttaa turvallisuus- ja toimimattomuusongelmia. Isommissa verkkosovelluksissa myös PHP:n huono virheenkäsittelykyky voidaan laskea heikkoudeksi, sillä virheiden etsiminen ja niiden korjaaminen runsaasta koodimäärästä on vaivalloista. Voidaankin ajatella PHP:n sopivan parhaiten pienten sekä yksinkertaisten keskisuurten dynaamisten web-sovellusten alustaksi.

3.3.6 Muita teknologioita

Java Server Pages (JSP) on Sun Microsystemsin kehittämä teknologia, jonka tarkoitus ASP:n ja PHP:n tavoin on luoda informaatiosisältöisiä ja dynaamisia www-sivuja. Se on PHP:n lailla täysin alustariippumaton, mutta sen leviäminen on ollut paljon hitaampaa, sillä ilman Javan ja palvelinpuolen ohjelmoinnin perusteiden syvällistä tuntemista JSP:n opetteleminen voi olla erittäin vaikeaa.
JSP on tärkeä osa laajempaa J2EE (Java 2 Enterprise Edition) määrittelyä, joten voidaan ajatella, että se on suunnattu lähinnä suuren mittaluokan palveluiden rakentajille. (Ahonen ym., 2004.)

Server Side Includes (SSI) on CGI:n ohella eräs vanhimmista palvelinpuolen tekniikoista. Sen avulla voi helposti lisätä sivuille toistuvia koodinpalasia sekä noutaa palvelimelta perustietoja. Sen käyttö on kuitenkin vähentynyt palvelinpuolten skriptikielten yleistyttyä. (KK Mediat 2000d.)

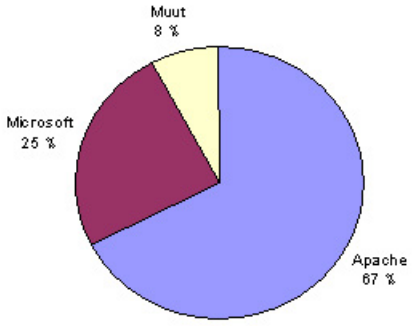
ColdFusion on nykyisin Macromedian omistuksessa olevan Allairen kehittämä nopea ja tehokas skriptikieli. Se on suunniteltu pääsääntöisesti Microsoft-alustalle, mutta toimii myös Linux- ja Unix-

alustoilla. ColdFusion käyttää kielenään CFML:ää (Cold Fusion Markup Language), joka sisältää paljon valmiita funktioita ja komponentteja. Se on syntaksiltaan hyvin HTML:n kaltainen, mutta tehokkuudestaan ja selkeydestään sen käyttö ei ole yleistynyt lähinnä tekniikan osaajien puutteen takia. (KK Mediat 2000e.)

3.3.7 Arkkitehtuurien vertailua

Henkilöstöhallintajärjestelmän kaltaista suhteellisen yksinkertaista mutta runsaan tietomäärän sisältävää sovellusta suunniteltaessa skriptikielen tärkeimmäksi vaatimukseksi nousevat hallittavuus ja skaalautuvuus sekä tuen saatavuus. Suorituskykyyn vaikuttaa tietokantapainotteisessa sovelluksessa enemmän tietokannan ja tietokantahakujen tehokkuus, ja riittävään tietoturvan takaamiseksi joudutaan joka tapauksessa turvautumaan johonkin salausmenetelmään.

Näillä ehdoilla varteenotettavat ohjelmointikielet PHP ja ASP ovat hyvin samantyyppisiä. Molempia käytetään suoraan HTML-koodiin upotettuna, ja molemmat kielet toimivat erityisen hyvin tietokantojen yhteydessä. Ohjelmointikieli valitaan lähinnä palvelinympäristöstä ja käyttötarkoituksesta riippuen. ASP:ta suositellaan käytettäväksi Microsoftin palvelimilla, kun taas PHP:tä Unix- ja Linux-palvelimilla. PHP-tukea on edullisuutensa ansiosta huomattavasti paremmin saatavilla, ja se onkin vajaassa kymmenessä vuodessa noussut suosituimmaksi ohjelmointikieleksi, mikä näkyy esimerkiksi PHP:tä tukevien Apache web-palvelinten suosiossa.



Kuva 6: PHP:tä tukevien web-palvelimien osuus on lähes 70% kaikista palvelimista

4 PHP TEKNOLOGIA

4.1 PHP:n kehitys

Ensimmäisen Perl-skripteistä koostuneen PHP-version (tuolloin vielä nimellä *Personal Home Page Tools*) julkaisi Rasmus Lerdorf vuonna 1995. Pian ilmestynyt PHP/FI (Personal Home Page / Forms Interpreter) -niminen laajennus sisälsi muun muassa mahdollisuuden kommunikoida tietokantojen kanssa. Marraskuussa 1997, kun PHP/FI 2.0 –versio julkaistiin, käyttäjiä oli jo useita tuhansia ympäri maailmaa.

Samoihin aikoihin avoimeen projektiin tuli mukaan muitakin ohjelmoijia. Seuraavaan PHP 3 versioon kirjoitettiin lähes koko lähdekoodi uusiksi, se onkin ensimmäinen versio, joka muistuttaa hyvin paljon nykyisin käytössä olevaa versiota. PHP 3 julkaistiin kesäkuussa 1998, samalla lopetettiin PHP/FI:n kehittäminen ja aloitettiin PHP:n (PHP: Hypertext Preprocessor) jatkokehitys. PHP 3:n mukana tullut hyvä laajennettavuus houkutteli projektiin mukaan lisää ohjelmoijia, ja sen suosion kasvi jatkui edelleen. Vuoden 1998 lopussa PHP:n käyttäjiä oli jo kymmeniä tuhansia, ja 10% kaikista Internetin palvelimista sisälsi tuen PHP:lle.

Maaliskuussa 2000 julkaistun PHP 4:n käänteentekevä muutos oli uusi Zend Engine -ydin, joka tuki kolmansien osapuolten ohjelmointirajapintoja. PHP sai tuen muun muassa useille webpalvelinohjelmistoille, HTTP-istunnoille ja käyttäjiltä tulevien syötteiden turvallisemmalle käsittelylle. Heinäkuussa 2004 julkaistun PHP 5:n ytimenä toimii Zend Engine II, joka tukee muun muassa olio-ohjelmointia täydellisesti sekä sisältää sisäänrakennetun tietokantamoottorin.

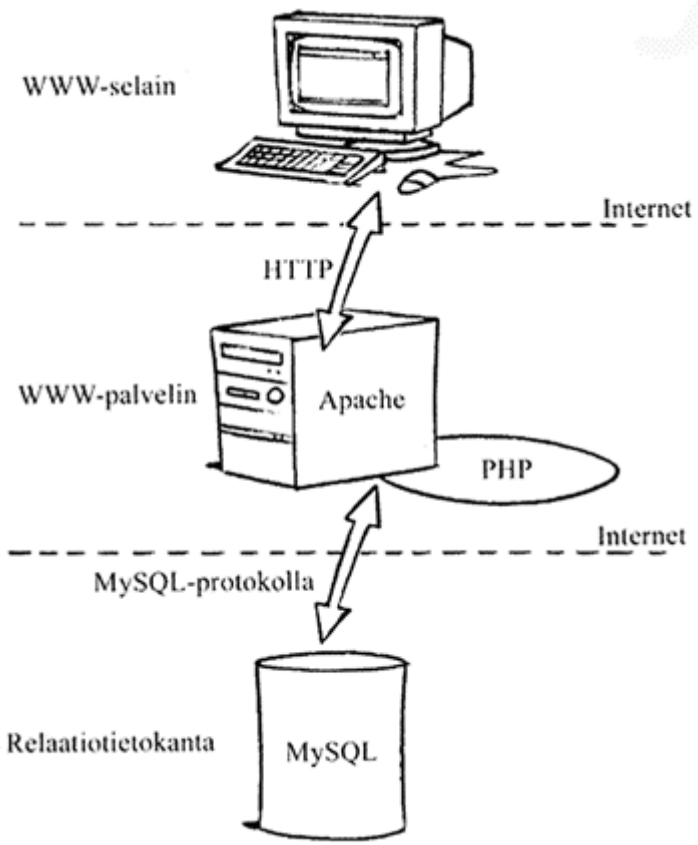
PHP:stä on siis varsin nopeasti kasvanut täysiverinen ohjelmointikieli, jonka avulla voidaan hallita laajojakin tietokantapohjaisia sovelluksia.

4.2 PHP:n toimintaperiaate

PHP:n suosioon on olemassa useita syitä. Avoimeen lähdekoodiin

perustuvana se on paitsi ilmainen käyttää, sen taustalla toimii laaja ja aktiivinen yhteisö, eli kieli kehittyy jatkuvasti. PHP on myös helpommin opittavissa oleva tekniikka kuin muut vastaavaan tarkoitukseen kehitetyt menetelmät, ja laajennettavuutensa ansiosta pystyy vaativampiinkin tehtäviin. PHP:n upotettavuus helpottaa toiminnallisuuden lisäämistä tavallisen jo olemassa olevan koodin sekaan. PHP sisältää myös tuen useiden protokollien (POP3, IMAP, HTTP) ja rajapintojen käsittelyyn.

PHP on useille eri alustoille sopiva skriptikieli, jota voidaan ajaa Windows-käyttöjärjestelmissä, useimmissa Unix-koneissa Linux mukaan lukien, ja jopa Macintosh koneissa. Varsinkin webhotellikäytössä suosituin yhdistelmä on Linux, Apache, MySQL, PHP. PHP:tä voidaan web-sovellusten kehittämisen lisäksi käyttää myös itsenäisenä skriptityökaluna esimerkiksi CGI-ohjelmissa. PHP on kaiken kaikkiaan alustasta riippumaton skriptikieli. (Zandstra, 2001.)



Kuva 7: Käyttäjän, www-palvelimen, PHP:n ja MySQL:n suhde toisiinsa

PHP:n tärkeimpiä ominaisuuksia on sen toiminta tietokantojen yh-

teydessä, mikä edesauttaa sen käyttöä kehittyneiden Web-sovellusten kehittämisessä. PHP tukee suoraan toistakymmentä erilaista tietokantaa, joista käytetyimmät ovat MySQL ja PostgreSQL. (Heinisuo, 2001.)

PHP koostuu Zend Enginestä, moduuleista ja rajapinnasta WWW-palvelimeen. Zend Engine sisältää esikääntäjän ja jäsentäjän, joiden avulla lähdekoodi jäsennetään ja suoritetaan. Moduuleiden avulla PHP voi kommunikoida ulkomaailmansa, esimerkiksi tietokantojen, kanssa. (Zend, 2004.)

4.3 PHP:n perusrakenne

PHP-kieliset komennot upotetaan HTML-dokumenttiin erottamalla ne HTML-koodista erillisisllä script-kielen erottimilla. Näistä erottimista PHP-tulkki havaitsee suoritettavan koodin. Tällaisia alueita voidaan upottaa mihin tahansa HTML-dokumentin kohtaan. Merkintätapoja on useita, mutta suositeltavin on täydellinen tapa:

```
<?php print "Tulostaa tekstiä"; ?>
```

Tietotyyppi

Tietotyyppi määrittää muuttujan sisältämän datan luonnetta, muistista tarvittavaa tilaa ja sitä, millaisia operaatiota siihen voidaan kohdistaa. Tietotyypit voidaan jakaa yksinkertaisiin, joita ovat muun muassa kokonaisluvut ja merkkijonot, sekä rakenteisiin tietotyyppeihin, joita voivat olla esimerkiksi taulukot (array) ja oliot.
PHP on heikosti tyypitetty kieli, mikä tarkoittaa sitä että muuttujien tyyppejä ei voi muutamaa poikkeusta lukuun ottamatta erikseen määrätä. Muuttujan tyyppi määräytyy automaattisesti sen mukaan millaista dataa siihen sijoitetaan. Tämä mahdollistaa nopean sovelluskehityksen, mutta vaatii myös erityistä huomiota tyyppien käytössä.

Muuttujat

Muuttujat ovat tiedon tallennuspaikkoja, PHP-kielessä muuttujiin

viitataan aina $-merkillä. Normaalisti muuttujat näkyvät vain siinä ohjelmalohkossa tai funktiossa missä ne on määritelty. Tällaiset paikalliset (local) muuttujat hävitetään lohkon loppuessa, eivätkä ole käytettävissä muualla koodissa. Ulkopuolella määriteltyjä muuttujia nimitetään globaaleiksi (global) muuttujiksi, joiden käyttö on kuitenkin harvemmin perusteltua. PHP-skripteillä on käytössä myös käyttöympäristöstä, Web-palvelimesta ja PHP:n versiosta riippuvia etukäteen määriteltyjä muuttujia, joilla saadaan tietoon esimerkiksi käyttäjät selain tai IP-osoite.

Operaattorit

PHP:ssä lausekkeet koostuvat operaattoreista, joita on viidenlaisia. Aritmeettisten operaattoreiden avulla voidaan suorittaa yksinkertaisia matemaattisia tehtäviä, sijoitusoperaattoreiden avulla voidaan muuttujia yhdistää tai sijoittaa niihin arvoja, vertailuoperaattorien avulla verrataan kahden lausekkeen arvoja keskenään (josta saadaan tulokseksi tosi tai epätosi), kasvattavat/pienentävät operaattorit lisäävät tai poistavat arvoja muuttujasta, ja loogiset operaattorit määrittävät ehtojen tilanteen ja suorittavat koodia sen perusteella, onko ehdot tosia vai epätosia.

Ohjausrakenteet

Ohjausrakenteiden avulla määritellään miten PHP-skriptin sisältämät lauseet suoritetaan. Rakenteisen ohjelmoinnin kolme perusrakennetta ovat ehtolauseet, toistolauseet, valintalause. Ehtolauseita ovat if…elseif…else, toistolauseita while, for, foreach ja do…while, sekä valintalauseita switch.

Funktiot

PHP-kielinen ohjelma koostuu yhdestä tai useammasta ohjelmalohkosta, joita ovat pääohjelma ja funktiot (aliohjelmat). Funktio on erikseen kutsuttava koodikokonaisuus, ohjelmointikielen rakenne, joka mahdollistaa toistuvien tehtävien kokoamisen yhteen paikkaan sovelluskoodissa. Funktion voi määritellä missä tahansa PHP-koodin kohdassa, ja sitä voidaan käyttää miten monta kertaa ja missä kohtaa tahansa ohjelmaa.
Funktiot voivat olla valmiita PHP:n sisäänrakennettuja, tai käyttäjän

määritelmiä funktioita. Funktio on pieni ohjelma, joka suorittaa tietyn rajatun tehtävän ja palauttaa aina jonkin arvon. Funktion tietotyyppi määräytyy automaattisesti paluuarvon tyypin mukaan.

4.4 PHP ja olio-ohjelmointi

Ennen versiota neljä PHP ei sisältänyt mahdollisuutta olio-ohjelmointiin. PHP 4 sisälsi pienen tuen olio-ohjelmoinnille, mutta PHP 5:sta lähtien olio-ohjelmointi on ollut täydellisesti tuettuna. Olio-ohjelmointi on yleisesti tuettu ja käytetty esimerkiksi C++, C# ja Visual Basic .NET –kielillä.

Oliopohjaisen ohjelmointikielen keskeisiä tavoitteita ovat ohjelmakoodin uudelleenkäytettävyyden saavuttaminen ja tietojen kokoaminen eli kapselointi yksiköihin. *Olio* (object) on olio-ohjelmoinnin keskeinen käsite. Olio on tietyn mallin (*luokka*) ilmentymä, joka sisältää sekä muuttujia että funktioita. Luokka määrittelee olioiden rajat ja toiminnallisuuden, muttei niiden sisältämää dataa muuten kuin tietotyypeillä. Ennalta määrätystä luokasta voidaan siis luoda eri ilmentymiä, olioita, joilla on keskenään sama rakenne, mutta niiden sisältämä tieto voi olla eri.

Koodin uudelleenkäytettävyys

Olio-ohjelmoinnin selkeimmistä eduista on koodin uudelleenkäytettävyys. Kertaalleen yhteen luokkaan kirjoitettua koodia voidaan hyödyntää periyttämällä siitä luokasta uusi luokka, joka sisältää kaikki vanhan luokan toiminnallisuudet sekä lisää siihen uusia. Samasta luokasta tehtyjä olioita voidaan käyttää eripuolilla sovellusta, joka tekee sovelluksen rakenteesta tiiviimmän ja sen ylläpidosta helpompaa. (Zandstra, 2001.)

Kapselointi

Kapselointi-termiä käytetään lähinnä erilaisen tiedon kokoamisesta yhteen yksikköön, olioon. Termiä voidaan käyttää myös tiedonpiilotuksen yhteydessä, jolloin tietoja ”kätketään” olion sisälle. Olion sisäiset muuttujat on mahdollista luokitella siten, ettei niihin päästä

suoraan olion ulkopuolelta käsiksi. Ohjelmointivirheiden määrä vähenee ja tieturva paranee.

Oliopohjainen ohjelmointi ei ole PHP-kielen tehokkaan hyödyntämisen kannalta välttämätöntä, sillä useat toiminnot voidaan toteuttaa muillakin menetelmillä. PHP:n käyttäjän kannalta olioiden edut liittyvät koodien parempaan ylläpitoon, organisointiin, laajennettavuuteen ja uudelleenkäytettävyyteen. Olio-ohjelmointia voi siis käyttää esimerkiksi yhtenä osana kokonaisratkaisua.

4.5 Tietokannan käyttö PHP:llä

PHP:n käyttöön liittyy yleensä oleellisesti tietokannat, ja kuten aiemmin on todettu, se on ennen kaikkea suunniteltu käytettäväksi tietokantojen yhteydessä. PHP:n valmiit funktiot sisältävät tuen useimpien yleisesti käytettyjen tietokantajärjestelmien, kuten MySQL:n, PostgreSQL:n ja Oraclen, käyttöön. Tämän lisäksi normaalin ODBC-yhteyden (ODBC on Microsoftin kehittämä rajapinta tietokannan ja sovelluksen välille) kautta voidaan ottaa yhteys vaikkapa Accessiin ja Exceliin, jotka ovat tutumpia peruskäyttäjälle. (Zandstra, 2001.)

Yleisin vaihtoehto on kuitenkin relaatiotietokantojen hallintajärjestelmä nimeltään MySQL. Se on PHP:n tavoin ilmainen tavalliselle käyttäjälle, eikä suorituskyvyssä häviä juurikaan kaupallisille vaihtoehdoille (Meloni, 2003) vaativissakaan tehtävissä. MySQL-versioita on lisäksi saatavilla useille eri alustoille. (Zandstra, 2001.)

5 TIETOKANNAT

5.1 Tietokantojen historia

Tietokannalla tarkoitetaan yksinkertaisesti järjestettyä informaatiojoukkoa. Olennaista on, että informaatiolla on jonkinlainen järjestys tai rakenne, joka mahdollistaa tietojen haun ja tietojen muokkaamisen.

Yksinkertainen tietokanta voi olla esimerkiksi yrityksen asiakasrekisteri tai puhelinluettelo. Tietokantojen sovelluskohteiden määrä kasvaa jatkuvasti ja lähes jokaisesta yrityksestä löytyy tänä päivänä jonkinlainen tietokantaratkaisu, jonne yrityksen toimintaan liittyviä tietoja tallennetaan. Teknologian kehitys ja halventuminen on tuonut tietokannat myös yksityishenkilöiden ulottuville, jotka käyttävä kotikoneillaan tietokantoja esimerkiksi omien cd-levykokoelmiensa ylläpitämiseen.

Tietokantoja on ollut käytössä jo pitkään, tosin WWW on paljolti edesauttanut niiden tunnettavuutta ja tuotteiden kehitystä. Tietokantojen historia juontaa 1950-luvun lopulle ja 1960-luvun alkuun. Ensimmäiset ”tietokannat” olivat hierarkkisia ja yksinkertaisia tiedostojärjestelmiä, joista puuttui keskitetty hallinta. 60- ja 70-lukujen vaihteessa tulivat verkkotietokannat, mutta varsinainen mullistus tapahtui 1972 relaatiotietokantojen saadessa alkusysäyksen. Relaatiotietokannat olivat edeltäjiään yksinkertaisempia ja helppokäyttöisempiä, jonka mahdollisti varta vasten kehitetty SQL-hakukieli.

Nykyisin Internetissä on käytössä erilaisia tietokantoja, muun muassa tekstitiedostopohjaiset tekstitietokannat, UNIX – alustalle tarkoitetut DB-kannat, XML-kieleen perustuvat XML-tietokannat, sekä tällä hetkellä ylivoimaisesti suosituimmat relaatiotietokannat.

5.2 Relaatiomalli

Matemaatikko E. F. Codd julkaisi vuonna 1972 relaatiomallin, joka määrittelee relaatiotietokannan teoreettisen pohjan. Relaatiomalli perustuu joukko-oppiin, joka tutkii joukkoja, niiden alkioita ja joukkojen välisiä suhteita. Ensimmäiset kaupalliset toteutukset tulivat markkinoille 1970-luvun lopussa, jonka jälkeen relaatiotietokantojen suosio on kasvanut tasaisesti. Nykyisin tunnettuja relaatiomalliin perustuvia ohjelmistoja ovat muun muassa Access, Oracle, SQL-Server, DB2, Informix ja MySQL. (Hovi, 1999.)

Relaatiomallin mukainen tietokanta muodostuu useasta eri taulukosta eli relaatiosta, joihin tieto on tallennettu. Eri taulujen väliset riippuvuudet hoidetaan perusavaimilla ja viiteavaimilla. Kukin tau-

lukko sisältää rivejä ja sarakkeita eli attribuutteja (ominaisuuksia). Relaatiotietokantoja on helppo muokata, olemassa olevien taulujen lisäksi voidaan perustaa uusia tauluja ja vanhoihin tauluihin lisätä uusia sarakkeita. Perinteisiin kantoihin nähden relaatiotietokantojen suurimpia etuja on tietoriippumattomuus, joka mahdollistaa taulun muokkaamisen vaikuttamatta samalla jo olemassa oleviin ohjelmiin.

Täydellisessä relaatiomallissa tieto tallennetaan vain yhteen paikkaan, josta tieto tarvittaessa voidaan hakea. Virheiden mahdollisuus pienenee, kun muutoksia tehdessä uudet ja muuttuneet tiedot täytyy päivittää ainoastaan yhteen paikkaan.

5.3 SQL-kieli

SQL on standardoitu kieli relaatiotietokantojen käsittelyyn ja määrittelyyn. SQL – kieli sai alkunsa IBM:n kehittäessä relaatiotietokannan System R – nimistä prototyyppiä vuosina 1972–1973. Aluksi kyselykieli kulki nimellä SEQUEL (Structured English QUEry Language), mutta myöhemmissä kehitysvaiheissa se muuttui SQL (Structured Query Language) -kieleksi.

Ensimmäinen ANSIn (American National Standards Institute) määrittelemä virallinen SQL standardi julkistettiin 1986. Siinä määriteltiin kielen käyttö ohjelmiin upotettuna, mutta ei sen vuorovaikutteista käyttöä. Vuotta myöhemmin myös ISO (International Organization for Standardization) hyväksyi standardin. ANSI ja ISO päättivät tehdä kielen määritykseen laajennuksen ja näin syntyi vuonna 1989 uusi standardi SQL-89. Laajennuksen yhteydessä kieleen lisättiin muun muassa käsitteet perus- ja vierasavaimista ja säännöt, jotka vaikuttivat käyttäjän määrittelemiin viite-eheyssääntöihin. (Hovi, 1999.)

1990-luvulla määriteltiin SQL-92 (myös SQL2), joka on huomattavasti edeltäjiään laajempi standardi, johon on lisätty paljon käytännössä tarvittavia osia. SQL-92:n laajuudesta huolimatta standardin kehittäminen on yhä jatkunut. Vuonna 1996 hyväksyttiin ohjelmointirajapinta (SQL/CLI, call level interface) ja tietokantaproseduurien (SLQ/PSM, persistent stored modules) määrittelykielen standardi.

Uusi SQL-99 (myös SQL3) pitää sisällään esimerkiksi automaattisesti käynnistyvät tietokantaproseduurit sekä oliopiirteitä. Nykyään SQL-kieli on kaikkien modernien tietokannanhallintajärjestelmien taustalla, mutta vaikka tuotteiden pohjana onkin standardoitu SQL, ovat ne silti ominaisuuksiltaan varsin erilaisia ja poikkeavat monissa kohdin standardista. Siksi yksikään tuote ei toteuta täydellisesti parissakymmenessä vuodessa vajaan sadan sivun kirjasesta yli 1000 sivun opukseksi paisunutta standardia. (Hovi, 1999.)

Käytännössä SQL-kieli on englantia muistuttava kyselykieli, jolla tietokannalle voidaan lähettää komentoja. Nimestään huolimatta SQL ei ole pelkästään kyselykieli, vaan sillä voi myös päivittää tietokantaa, ohjata tapahtumia sekä määritellä tietokannan rakennetta ja muuttaa sitä. Näin ollen SQL-kielen hallitseminen on osoittautunut myös ylläpitäjille suositeltavaksi.

Internetissä SQL-kieltä käytetään upotettuna ohjelmointikieleen. Tätä kutsutaan upotetuksi SQL:ksi. Yleisin tällainen liitos on PHP:n ja SQL-kielen yhdistäminen. Tällöin tietokantahaun tulokset siirretään ohjelmointikielen muuttujille, mikä mahdollistaa SQL:n dynaamisen käytön.

5.4 MySQL

Useista eri web-sovelluksissa käytetyistä tietokannoista yksi on yli muiden. MySQL tietokanta on laajimmin käytetty avoimen lähdekoodin tietokanta, ja sillä on useita miljoonia käyttäjiä. MySQL on monipuolinen, joustava ja suorituskykyinen relaatiotietokanta, jota käytetään niin suurten kuin pienienkin www-palvelujen taustalla. MySQL noudattaa asiakas-palvelin-arkkitehtuuria, jossa sovellukset eivät koskaan käsittele tietokantaa suoraan vaan käsittely tapahtuu aina palvelinohjelman kautta. MySQL on yhteensopiva käytännössä minkä tahansa ohjelmointikielen kanssa, esimerkkeinä PHP-, Java/JDBC-, Perl-, Python- ja Tcl – kielet. Etuihin kuuluu myös helppo siirrettävyys käyttöjärjestelmien välillä.

MySQL on helppo asentaa ja ylläpitää, eikä se vaadi täysipäiväistä huolenpitoa siihen tapaan kuin kalliimmat kaupalliset tietokantaoh-

jelmistot. MySQL soveltuu näin myös pienempien www-palvelujen taustatietokannaksi. MySQL ei sisällä kaikkia niitä ominaisuuksia, joita kalliilta kaupallisilta tietokannoilta on totuttu odottamaan ja vaatimaan. Toisaalta taas MySQL on pullollaan sellaisia ominaisuuksia, jotka tekevät siitä erinomaisen valinnan useimpien www-palvelujen taustalle. MySQL:n avulla on toteutettu niin julkaisujärjestelmiä, lomakepalveluita ja sähköisen kaupan järjestelmiä kuin yritysten intranet-palveluitakin.

Yhdellä MySQL-palvelimella voi olla useita tietokantoja, joissa kussakin voi olla useita tauluja. Tietokannan sisäiset käyttöoikeusasetukset mahdollistavat monimutkaistenkin sovellusten vaatimat käyttöoikeusmäärittelyt, kuten SQL-tietokannoissa yleensäkin. Tietokantapalvelimelle voidaan luoda käytännössä rajaton määrä käyttäjätunnuksia, joilla kullakin voi olla eritasoisia oikeuksia tietokantoihin ja tauluihin. Myös MySQL -tietokannan kyselykielenä on SQL, mutta muiden ohjelmien tapaan MySQL ei noudata standardia kovinkaan perusteellisesti peruskomentoja lukuun ottamatta. MySQL myös laajentaa SQL-komentokantaa omalta osaltaan, joten kyseessä on MySQL:n omiin tarpeisiin räätälöity versio standardi-SQL:stä.

MySQL muuttui hiljattain käyttöjärjestelmästä riippumatta kokonaan ilmaiseksi ohjelmaksi. Linuxiin ja muihin Unix-käyttöjärjestelmiin MySQL on aina ollut käytännössä ilmainen.

5.5 Tietokannan suunnittelu

5.5.1 Tietokantasuunnittelun merkitys

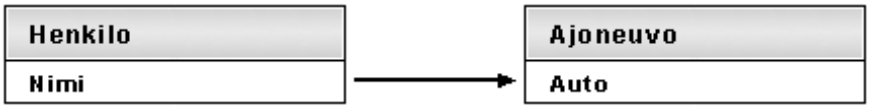
Hyvä tietokantasuunnittelu on tärkeää, jotta sovellukset toimisivat saumattomasti ja jotta tietokannasta saataisiin kaikki mahdollinen hyöty ja teho irti. Hyvä suunnittelu helpottaa myöhemmässä vaiheessa myös tietokannan ylläpitoa. Tietokanta on hyvä suunnitella siten, että toistuvia arvoja on mahdollisimman vähän, jos ollenkaan. Tällöin datan muutosta ei tarvitse tehdä useampaan esiintymään, vaan riittää sen muokkaaminen yhdessä tietueessa. Suunnittelussa tärkeään osaan nousevat avaimet ja niiden mahdollisuuksien hyödyntäminen. Avaimien avulla luotuja taulujen välisiä suhteita on kolmenlaisia: yksi yhteen, yksi moneen ja moni moneen. Suhteiden

käyttämistä datan järjestämiseksi kutsutaan normalisoinniksi. Normalisointi on joukko sääntöjä, joita seuraamalla tietokannasta tulee hyvin organisoitu ja hakeminen tietokannasta nopeutuu, koska tiedot haetaan vain yhdestä relaatiosta. Tarkkaan mietityt suhteet taulujen välillä ja huolellinen normalisointi helpottaa huomattavasti myös tietokantojen ylläpitoa.

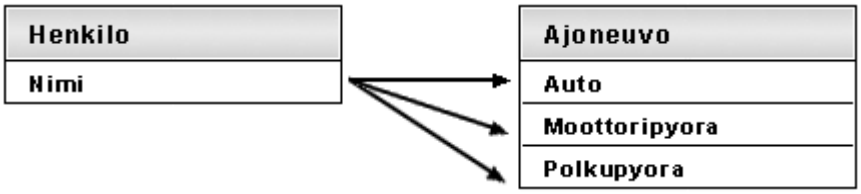
On sanomattakin selvää, että ajan kanssa tehty huolellinen suhteiden ja normalisoinnin suunnittelu projektin alkuvaiheessa säästää paljon työtä sen loppupuolella. Usein käy myös niin, että tietokannan suunnittelua aletaan toden teolla miettiä vasta sovelluksen julkistamisen jälkeen, jolloin aikaa ja voimavaroja on mennyt hukkaan.

5.5.2 Taulujen väliset suhteet

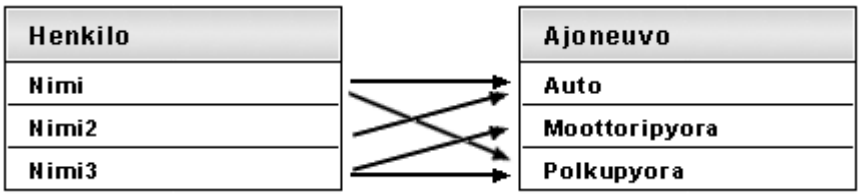
Taulut liitetään toisiinsa avaimilla, kuten edellä on kerrottu. Näin syntyviä suhteita on kolmenlaisia; yksi yhteen, yksi moneen ja moni moneen -suhteet. Yksi yhteen – suhteessa avain esiintyy vain kerran taulussa, johon viitataan. Yksi moneen – suhteessa puolestaan avaimet yhdestä taulusta esiintyvät useaan kertaan taulussa, johon viitataan. Moni moneen – suhteet ovat usein niin ongelmallisia, että niitä kuvataan usein useamman yksi moneen – suhteen yhdistelmällä. Monissa tällaisissa suhteissa yhden taulun avaimen arvo voi esiintyä monta kertaa taulussa, johon viitataan.



Kuva 8a: Yksi yhteen -suhde: henkilöllä on yksi ajoneuvo



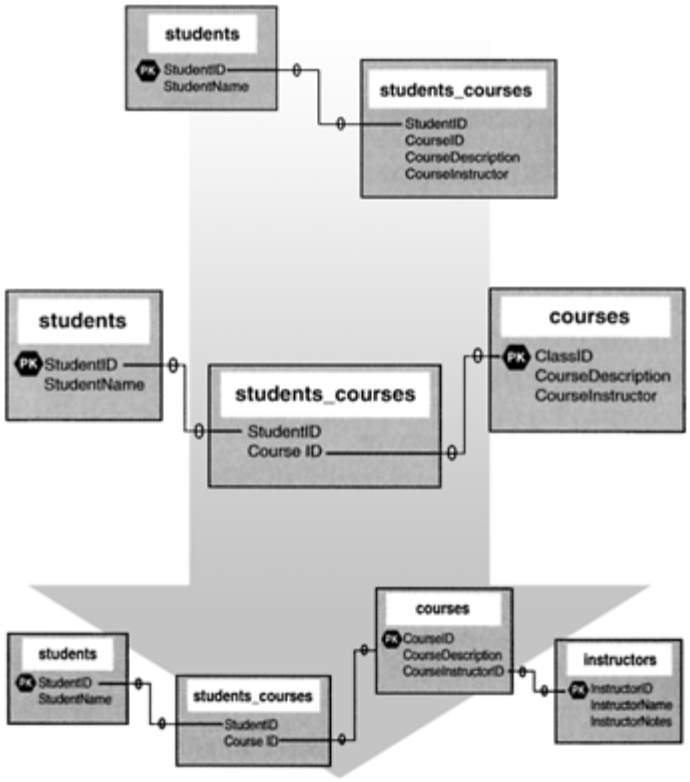
Kuva 8b: Yksi moneen -suhde: henkilöllä on monta ajoneuvoa



Kuva 8c: Moni moneen - suhde: useampi kuin yksi henkilö omistaa saman ajoneuvon

5.5.3 Normalisointi

Normalisoinnissa käytettyjä sääntöjä kutsutaan normaalimuodoiksi (normal forms). Olennaisimpia normaalimuotoja on kolme erilaista.



Kuva 9: Normalisoinnin kolme vaihetta

Ensimmäisen normaalimuodon sääntöjä on esimerkiksi tietojen toisteisuuden poistaminen ja erillisten taulujen luominen muille kuin asiaan välittömästi liittyville tiedoille.

Toisen normaalimuodon tarkoitus on, ettei mikään ei-avain-attribuutti riipu ensisijaisen avaimen osasta. Toisin sanoen taulussa olevat ei-avain-attribuutit muutetaan omaksi kentäkseen.

Kolmannen normaalimuodon sääntö on, etteivät mitkään attribuutit riipu toisista ei-avain-attribuuteista. Kolmas normaalimuoto on paras keino toisteisuuden poistoon, tosin normalisointia tehdessä kaikki kolme olennaisinta normaalimuotoa tulee yleensä käytyä järjestyksessä lävitse.

5.5.4 Suunnitteluprosessi

Tietokantapohjaisten sovelluksien suunnitteluun kuuluu perusteellinen arvio siitä, mitä tietoja tietokannan tulisi sisältää, miten tiedot ovat suhteessa toisiinsa ja miten tietokanta on laajennettavissa. Suurimmat ongelmat syntyvät silloin, kun kokonaisuutta ei ole mietitty tarpeeksi etukäteen, vaan on lähdetty toteuttamaan sovellusta takaperoisesti; heti idean synnyttyä lähdetään rakentamaan sovellusta, jonka jälkeen aletaan vasta miettiä, minkälaiset tietokantaratkaisut siihen sopivat.

5.6 Tietokannan toteutus

5.6.1 Tavoitteiden määrittely

Ensimmäisenä tulee määrätä tavoitteet ja päämäärät sovellukselle. Jos sovelluksen tavoitteet eivät ole selvillä, sitä ei ole myöskään järkeä alkaa toteuttamaan.
Asiakaslähtöisissä projekteissa asiakkailla on yleensä toiveita mahdollisista ominaisuuksista, joskin usein heidän käsityksensä tietokantojen mahdollisuuksista on rajallinen, jolloin on hyvä tuoda heti alussa esille myös laajempia ratkaisumalleja kuin mitä asiakas on aluksi määritellyt. Usein projektin edetessä asiakkaalta alkaa tulla enemmän vaatimuksia ja ideoita tarvittavista ominaisuuksista, jotka pahimmassa tapauksessa tekevät kokonaan turhaksi hyvin alkaneen tietokannan suunnittelun.

5.6.2 Taulukoiden hahmottelu

Kun sovelluksen toiminnot ovat selvillä, seuraava vaihe on datarakenteiden suunnittelu.
Tietokannan mallintaminen voidaan aloittaa tekemällä suuria ”litteitä” tauluja, eli kirjoitetaan ylös tarvittavat sarakkeet vain muutamaa taulua käyttäen. Tällöin tauluihin tulee paljon turhaa tietoa, mutta kun taulujen rakenne alkaa hahmottua, voi suunnittelussa alkaa hyödyntämään jo tutuksi tullutta tietokannan normalisointia ja löytää tietokannasta mahdolliset turhat asiat ja hahmottamaan sen suhteet. Paras työkalu tässä työvaiheessa on lyijykynä ja paperi.

Taulukot hahmotellaan kolmanteen normaalimuotoon ensimmäisestä lähtien, eli poistetaan toisteisuudet ja luodaan erillisiä tauluja asiaan liittyvälle datalle. Kun tiedot on jaettu useisiin pienempiin tauluihin, on vuorossa toinen normalisointivaihe, eli taulun suhteiden määrittäminen.

5.6.3 Suhteiden määrittäminen

Taulukoiden hahmottelun jälkeen niiden rakenne on optimaalinen, mutta niiden väliset suhteet puuttuvat. Jokainen taulu on määritelty vähintään yhdellä ID -kentällä, mutta keskinäistä yhteyttä niillä ei ole. Datan jaottelun yhteydessä yhdestä taulusta tulee ”päätaulu”, johon kaikki muut taulut ovat suhteessa. Suhteen luomiseksi muihin tauluihin lisätään yksi kenttä joka viittaa päätauluun.

Suhteiden luomisen jälkeen tietokanta on toisessa normaalimuodossa. Operaatiota voisi jatkaa vielä pidemmälle pilkkomalla informaatiota erillisiksi tauluiksi pelkästään tehokkuuden ja ylläpidon vuoksi, jolloin kyseeseen on kolmas normaalimuoto. (Meloni, 2003.)

6 CASE: GO ON HENKILÖSTÖHALLINTAOHJELMA

6.1 Projektin lähtökohdat

Ajatus Go On Yhtiöiden henkilöstöhallintaohjelmasta syntyi käytännön tarpeesta. Yrityksellä oli jo käytössään MySQL- ja PHP-pohjainen järjestelmä, joka toimi kuitenkin vain paikallisesti yhdessä yrityksen koneista.

Henkilöstörekisteri haluttiin kaikkien sitä tarvitsevien työntekijöiden käyttöön, myös mahdollisesti toimiston ulkopuolella. Lisäksi ohjelmaan tahdottiin paremmat hakuominaisuudet sekä mahdollisesti muita henkilöstöhallintoon liittyviä ominaisuuksia, kuten esimerkiksi työvuorolistat ja työtodistukset. Ohjelmaa toivottiin myös tehokkaammin osaksi työhönottoprosessia, joka pitää sisällään Internetin kautta esitäytettävät hakulomakkeet sekä paikanpäällä tapahtuvat työhaastattelut.

Oman ohjelmointikokemuksen ja saatavilla olevien tekniikoiden perusteella PHP ja MySQL olivat selkeät vaihtoehdot kehitysympäristöksi. Projekti tarjosi kuitenkin paljon uusia haasteita jo hyvin tutuksi tulleiden tekniikoiden osalta, kuten ohjelmaan tarvittavan laajan tietokannan suunnittelu mahdollisimman nopeaksi ja helposti laajennettavissa olevaksi. Haaste oli myös tietokannan kanssa yhtä joustavan rajapinnan luominen tietokannan ja käyttäjän välille PHP:n avulla.

6.2 Tarpeiden määrittely

Projekti lähti liikkeelle asiakasyrityksen kanssa pide, missä yhdessä määriteltiin sovellukselle aikataululliset ja toiminnalliset tavoitteet. Tavoitteita tarkennettiin ja lisättiin parin viikon välein, tai aina tarpeen mukaan. Aikataulu oli tiukka, joten sovelluksen rakenteen määrittely ja lukkoon lyöminen mahdollisimman aikaisessa vaiheessa oli tärkeää. Tämä vähentää turhan työn tarvetta, kun muutosehdotuksia ja toiveita ei ilmene enää jälkikäteen.

Palvelun tavoitteet ovat henkilöstöresurssien paremmassa hallitta-

vuudessa. Sovelluksen pääpaino on siis työntekijän henkilötietojen tehokkaassa haussa ja hallinnassa. Näiden tarpeiden mukaan sovelluksen kaksi tärkeintä osiota ovat henkilötietojen hakulomake ja henkilötietojen muokkauslomake, joiden ympärille koko ohjelma on rakennettu.

Yhteisen ideoinnin ansiosta mukaan saatiin alusta asti ominaisuuksia, joita asiakas ei olisi itse osannut toivoa. Sovellukseen saatiin vuorovaikutusta ja jatkuvaa, huomaamatonta kehitystä käyttäjän voidessa lisätä ominaisuuksia ohjelman käytön aikana esimerkiksi haastattelun yhteydessä. Visuaalisen ilmeen suunnittelua ei pidetty kovinkaan tärkeänä, sen sijaan sovelluksen käytettävyyteen ja johdonmukaisuuteen panostettiin, samoin kuin käyttöympäristöstä riippumattomaan ulkoasuun. Ohjelman hyvä käytettävyys oli erityisen tarpeellinen ominaisuus haastattelutilanteita silmällä pitäen.

6.3 Tietokannan suunnittelu

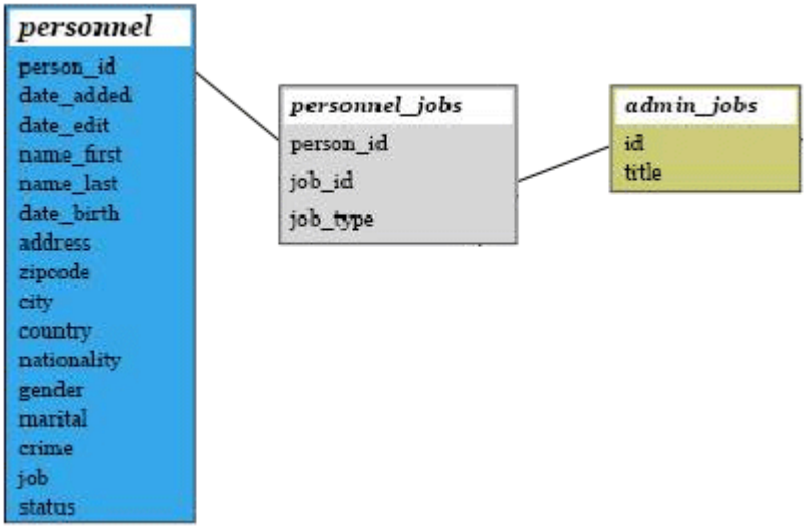
Tietokantaa suunnitellessa oli ensisijaisesti otettava huomioon laajennettavuus, ohjelman on kyettävä käsittelemään tuhansien ihmisten henkilötietoja. Henkilötiedot jaettiin mahdollisimman moneen tauluun toisteisuuden poistamiseksi ja hakujen nopeuttamiseksi. Tämän lisäksi ennalta määrätyt ominaisuudet jaettiin omaksi kokonaisuudekseen, joka mahdollisti niiden monipuolisemman käytön. Täten tietokanta muodostui itse päätaulun, eli henkilötietotaulun, lisäksi kahdentyyppisistä tauluista, ylläpito- ja henkilöstötauluista.

Ylläpitotaulut sisältävät ennalta määrättyjä arvoja, joita käyttäjä voi käytön yhteydessä lisätä tietyin rajoituksin. Tällaisia tietoja ovat esimerkiksi työtehtävät, kielet ja erikoisosaamiset. Ylläpitotaulut eivät ole riippuvaisia henkilöstötauluista tai päätauluista. Ylläpitotauluista tulostetaan tietoja suoraan lomakkeisiin tai henkilötietoihin henkilöstötaulujen avulla.

Henkilöstötaulut sisältävät henkilön lisäominaisuuksia, joiden sisältö määritellään joko ennalta määrätyillä numeroarvoilla tai ylläpitotaulujen avulla. Henkilöstötaulut ovat suoraan yhteydessä henkilön perustiedot sisältävään päätauluun, joten se on riippuvainen päätau-

lusta ja tietyissä tapauksissa myös ylläpitotauluista.

Koko tietokannan rakenne on nähtävissä liitteessä kaksi, alla olevasta kuvasta näkyy työntekijän työhistorian suhde päätauluun ja ylläpitotauluun.



Kuva 10: Ylläpitotalun (admin_jobs) ja henkilöstötaulun (personnel_jobs) suhde päätauluun (personnel).

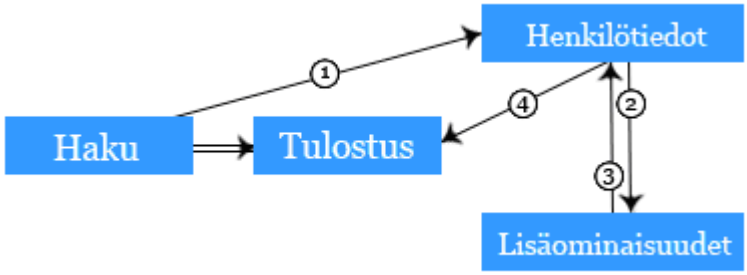
6.4 Tekninen toteutus

PHP:n käyttö mahdollisti sovelluksen skaalautuvuuden ja hallittavuuden. Julkaisun jälkeen ohjelmaan onkin lisätty useita ominaisuuksia, mikä on ennen kaikkea hyvän suunnitelman ja joustavan toteutusarkkitehtuurin ansiota.

Ohjelman rungon muodostavat erilaiset olioit, kuten *Person*-olio, joka hakee *Search*-oliosta henkilön tiedot, jotka *Output*-olio pyydettäessä tulostaa. Yksi hyödyllisimmistä luokista on *Form*-luokka, jonka avulla voidaan tehdä erillisiä lomake-olioita sovellukseen. Tämä on hyvä esimerkki koodin uudelleenkäytettävyydestä, sillä ohjelman useaa ja varsin mittavaa lomaketta varten on luotu yksi luokka alifunktioineen, josta erilaisia lomake-elementtejä voidaan tulostaa tarpeen mukaan. Olioiden hyödyntäminen jäi tässä vaiheessa aiheen vierauden takia kuitenkin lähinnä tiedon organisoinnin ja hallittavuuden asteelle, sillä esimerkiksi hyödyllistä periyttämistä ei ole käy-

tetty lainkaan.

Sovelluksen saatavuus ja luotettavuus saatiin sijoittamalla oma palvelin luotettavan toimittajan tiloihin.



Kuva 11: Haku-toiminnon logiikka. Nuolet kuvaavat tiedon liikkumista eri olioiden välillä, tuplanuoli käyttäjälle näkyvää tapahtumaa.

6.5 Tietoturva

Tietoturva on, kuten jo moneen kertaan todettua, yksi henkilöstöhallintaohjelman tärkeimpiä seikkoja. Tietoturvan maksimoimiseksi päädyttiin ottamaan täysin erillinen palvelinhotellipalvelu luotettavalta toimittajalta. Tämä takasi tietojen fyysisen turvallisuuden, sillä palvelin sijaitsee tarkkaan vartioidussa tilassa, jonne ei ulkopuolisilla ole asiaa. Lisäksi varmistus ja ylläpito ovat huippuluokkaa. Oma palvelin takaa muista käyttäjistä riippumattoman palvelinympäristön, jolla varmistetaan, etteivät toiset käyttäjät pysty vaikuttamaan palvelun toimivuuteen.

Fyysisen koskemattomuuden lisäksi tietojen luottamuksellisuus ja eheys tulee myös taata. Koska tietoja muokataan ja tallennetaan verkon yli, oli salaus ainoa mahdollisuus tietoturvan takaamiseksi. Tietoturvaratkaisuksi valittiin yleinen SSL-tekniikkaan perustuva sertifikaattisuojaus. Käyttäjä tarvitsee sovellukseen päästäkseen julkisen avaimen, joka tulee aina erikseen asentaa käytettävälle koneelle. Käyttäjän vastuulle jää kuitenkin huolehtiminen, ettei sertifikaatti joudu vääriin käsiin, sillä oletuksena sertifikaatti jää käytetyn selaimen asetuksiin, ellei sitä manuaalisesti poisteta. SSL-salatulla sivulla tiedot liikkuvat salattuina, joten tietojen kaappaus ei ole mahdollista. SSL-suojauksen lisäksi sovelluksessa on käytössä yksinkertainen

PHP-pohjainen käyttäjän perustodennus.

6.6 Lopputuote ja tulevaisuus

Go On henkilöstöhallintaohjelma on ollut yrityksen käytössä nyt lähes vuoden. Toistaiseksi suurimmilta ongelmilta on vältytty, vaikka kehitystyö kohti oikeaa sovellusta on vasta alkutekijöissä.

Sovelluksen suunnittelun kannalta lopputuote on vastannut melko hyvin sille asetettuja tavoitteita. Joitain alun perin suunniteltuja ominaisuuksia ei vielä ole toteutettu, mutta puolestaan kokonaan uusia piirteitä on saatu ohjelmaan mukaan.

Tietokannan toimivuuden kannalta sovellus on onnistunut mielestäni varsin mainiosti. Sille asetetut tavoitteet joustavuudesta, nopeudesta ja helposta laajennettavuudesta ovat toteutuneet kiitettävästi. Teknisen toteutuksen osalta PHP:n olio-ominaisuuksien käyttö oli tekijälle täysin uutta, eikä lopputuote ole rakenteeltaan aivan niin mallikelpoinen kuin suunnitelmissa oli tarkoitettu. Siksi sillä saralla kehitettävää löytyy vielä paljon. Lopputyön anti tuleekin esille nimenomaan tulevia koodaustoimenpiteitä suunniteltaessa.

7 YHTEENVETO

WWW-sovellukset ovat viime vuosina tulleet jokapäiväiseen käyttöön, samalla useat tavalliset staattiset sivut ovat saaneet dynaamista sisältöä. Sovellukset laajenevat entisestään, ja niiden toteutus vaati yhä enemmän tietoa ja taitoa paitsi teknisellä puolella, myös projektinhallinnan osa-alueella. Jo nyt keskisuuretkin verkkosovellukset alkavat olla yhden ihmisen tehtäväksi turhan laajoja. Projektin hallinnan, sovelluksen suunnittelun, graafisen ilmeen ja teknisen toteutuksen ollessa yksissä käsissä kokonaisuus kärsii väistämättä.

Verkkosovellusten toteuttamiseen on tapoja lähes yhtä monta kun on tekijöitäkin. Tekniikat ovat periaatteeltaan hyvin samanlaisia, yleensä tekijän oma kokemus ohjelmointitekniikoista määrittelee käytettävän kehitysympäristön. Aloittelijat lähtevät liikkeelle yksinkertaisesta ja laajasti tuetusta PHP:stä, kun taas kokeneemmat ohjelmoijat turvautuvat monimutkaisempiin ASP- ja JSP-tekniikoihin.

Web-projekteihin lähdetään yleensä sen kummemmin suunnittelematta. Kunnon suunnitelman puute syö inspiraatiota ja projektin tehokkuutta, viimeistään ensimmäisten vastoinkäymisten kohdalla. Tämän työn yksi tarkoitus on antaa eväitä näiden karikoiden välttämiseen jo ennakkoon, ja tarjota monien vaihtoehtojen joukosta kullekin tekijälle sopiva projektin onnelliseen läpiviemiseen.

Tässä mielessä työ toteutti hyvin sille asetetut niin teoreettiset kuin konkreettisetkin tavoitteet. Lopputuotteena saatiin aikataulun puitteissa asiakkaan mieleinen sovellus, jota on helppo kehittää ja laajentaa myöhemminkin. Työ antoi tekijälleen myös hyvän kokonaiskuvan laajan web-projektin läpiviemisestä.

LÄHTEET

Ahonen, T. & Hämeen-Anttila, T., 2004. JSP-ohjelmointi. Docendo Finland Oy

Allen, Julia 2002. CERT: verkkotietoturvan hallinta. Oy Edita Ab

Arvidsson, S. & Ek, J. 2000.Tietokantojen käyttö Internetissä, ASP 3.0. Schildts Kustannus Oy

DeCenzo, D. & Robbins, S. 1999. Human Resource Management, sixth edition. John Wiley & Sons

Eriksson, H.-E. & Penker, M. 2000. UML. Oy Edita Ab

Tietoturvallisuuden perusteet [verkkodokumentti]. Viestintävirasto [viitattu 2.4.2006]. Saatavissa:
http://www.ficora.fi/suomi/tietoturva/salausmenetelmat.htm

Goto, K. & Cotler, E. 2002. Verkkopalveluprojekti. Edita Publishing Oy

Heinisuo, Rami 2001. PHP ja MySQL. Talentum Media Oy

Henkilötietolaki 22.4.1999/523, 5 §

Henttunen, Tommi: WWW-palvelimien ohjelmointitekniikat [verkkodokumentti]. Oulun yliopisto, 2001 [viitattu 24.3.2006]. Saatavissa:
http://www.student.oulu.fi/~thenttun/ITV/seminaari.html

ITviikko: Apache valtaa markkinoita Microsoftilta [verkkodokumentti]. Startel, 19.11. 2003 [viitattu 26.2.2006]. Saatavissa:
http://www.itviikko.fi/uutiset/uutinen.asp?UutisID=58244

Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet [verkkodokumentti]. Juhta, 15.6.2005 [viitattu 15.3.2006]. Saatavissa:
http://www.jhs-suositukset.fi/intermin/hankkeet/jhs/home.nsf/files/JHS129/$file/JHS129.pdf

Järvinen, Asko 1996. Henkilöstö voimavarana. Oy Edita Ab

Järvinen, Petteri 2003. Salausmenetelmät. Docendo Finland Oy

Kauhanen, Juhani 2000. Henkilöstövoimavarojen johtaminen, 3. pianos.

WSOY

Kerttula, Esa 2000. Tietoverkkojen tietoturva. Oy Edita Ab

Kinnunen, Erja: Tietoturva - katoava luonnonvara? [verkkodokumentti] [viitattu 30.3.2006] Saatavissa: http://www.csc.fi/lehdet/atcsc/atcsc3-98/erja.html

KK Mediat 2000a. ASP [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/asp/ [viitattu 30.3.2006]

KK Mediat 2000b. PHP [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/php/ [viitattu 30.3.2006]

KK Mediat 2000c. ASP.NET [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/dotnet/aspnet1.asp [viitattu 28.3.2006]

KK Mediat 2000d. SSI [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/dhtml/ssi.asp [viitattu 28.3.2006]

KK Mediat 2000e. ColdFusion [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/internetohjelmointi/coldfusion.asp [viitattu 28.3.2006]

KK Mediat 2000f. Apache [verkkodokumentti]. Saatavuus: http://www.2kmediat.com/apache/ [viitattu 11.4.2006]

Kontio, M., Vierimaa, K. & Niskanen, P 2001. WWW-ohjelmointi Trainer Kit. Oy Edita Ab

Laakso, Sari 2002. Käyttöliittymät II luento 2 [verkkodokumentti]. Saatavuus: http://www.cs.helsinki.fi/u/salaakso/kl2-2002/luennot/KaliII-luento2.pdf [viitattu 27.3.2006]

Lahtonen, Tommi 2002. SQL. Docendo Finland Oy

Meloni, Julie 2003. MySQL Trainer Kit. Edita Publishing Oy

Metsämäki, Markku 2000. Verkkopalvelun suunnittelu. Oy Edita Ab

Mielonen, S. & Hintikka, K. 1998. Web-palveluiden käytettävyys ja tuotanto [verkkodokumentti]. Saatavuus: http://www2.uiah.fi/mediastudio/survey4/index.html [viitattu 26.3.2006]

Heikniemi, Jouni. Mitä web-palvelimet ovat? [verkkodokumentti]. Saatavuus

http://www.heikniemi.net/svwww-vukk/k35.html [viitattu 11.4.2006]

Nachimovsky, A. & Myers, T 2002. Inside Java ja XML. Edida Publishing Oy

Netcraft [verkkodokumentti]. March 2006 Web Server Survey, 6.3.2006 [viitattu 2.4.2006]. Saatavissa:
http://news.netcraft.com/archives/2006/03/06/march_2006_web_server_survey.html

Nykänen, Ossi [verkkodokumentti]. W3C ja verkkopalvelujen laatu, 1.10.2003 [viitattu 2.4.2006]. Saatavissa:
http://www.w3c.tut.fi/reports/2003/1001quality/index.html

Parkkinen, Jarmo 2002. Hyvään verkkopalveluun. Inforviestintä Oy

Peltomäki, J., Inkinen, V. & Rantala, A. 2000. CGI- ja ASP-ohjelmoint. Docendo Oy Finland Oy

Rantala, Ari 2002. PHP. Docendo Finland Oy

Sähköisen kaupankäynnin aapinen [verkkodokumentti]. Tietoyhteiskunnan kehittämiskeskus Ry, 1999. Päivitetty 10/2003 [viitattu 28.3.2006]. Saatavissa:
http://www.tieke.fi/mp/db/file_library/x/IMG/12422/file/Sahkoisenkaupankaynninaapinen.pdf

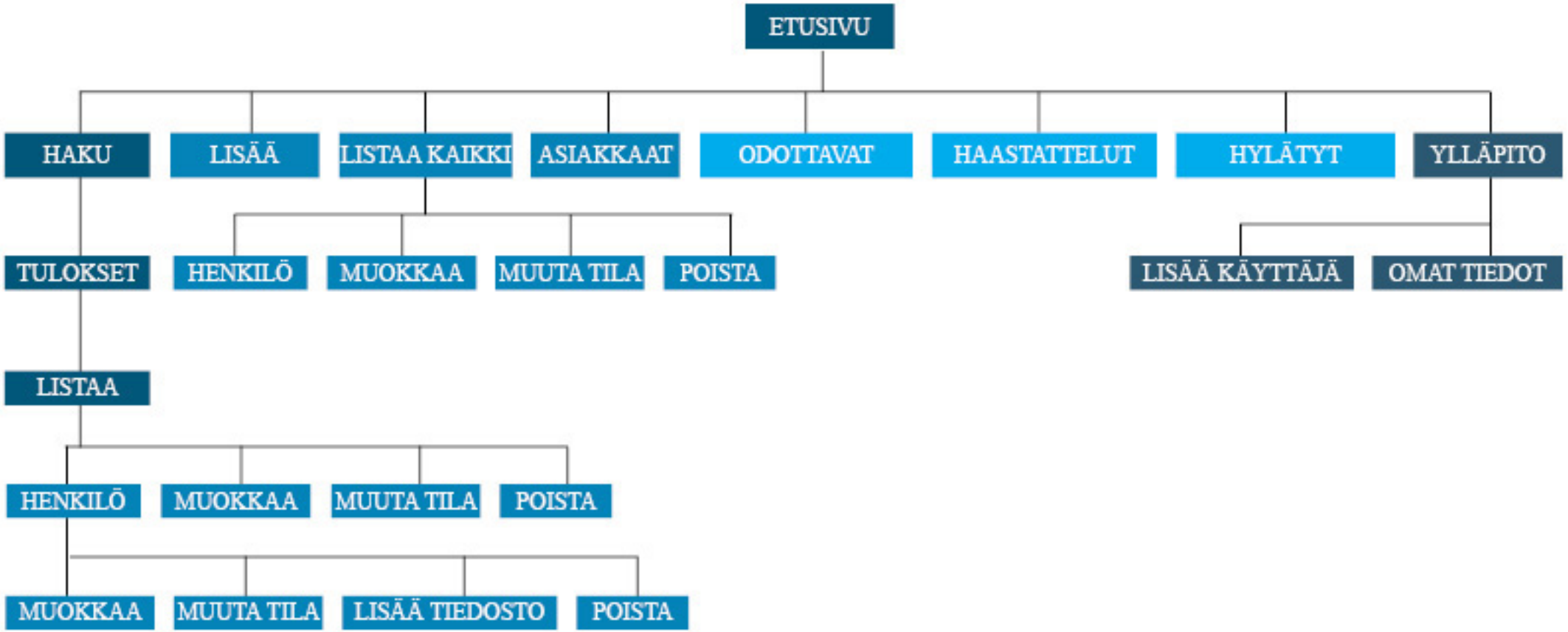
Tietoturva [verkkodokumentti]. Tietoyhteiskunnan kehittämiskeskus Ry [viitattu 27.3.2006]. Saatavissa:
http://palvelut.tieke.fi/arkisto/tiveke/turva/turva-1.htm

Zandstra, Matt 2001. PHP Trainer kit. Oy Edita Ab

LIITTEET

Liite 1. Sivukartta: Henkilöstöhallintajärjestelmän sivurakenne
Liite 2. Rakennekaavio: Henkilöstöhallintajärjestelmän tietokannan rakenne

Liite 1. Sivukartta: Henkilöstöhallintajärjestelmän sivurakenne



Liite 2. Rakennekaavio: Henkilöstöhallintajärjestelmän tietokannan rakenne