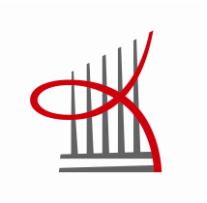


TIETOPANKKISOVELLUS TILITOIMISTOLLE

CASE: Näsitilit Oy

PAULI RAUHANEN

Opinnäytetyö
Tammikuu 2010
Tietojenkäsittelyn koulutusohjelma
Digimedian suuntautumisvaihtoehto
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

RAUHANEN, PAULI: Tietopankkisovellus tilitoimistolle
CASE: Näsitilit Oy

Opinnäytetyö 47 s, liitteet 3 s
Toukokuu 2011

Tässä opinnäytetyössä tutustutaan tilitoimistokäyttöön tarkoitetun tietopankkisovelluksen suunnitteluun ja toteutukseen. Opinnäytetyön toimeksiantajana toimii tamperelainen tilitoimisto Näsitilit Oy. Tavoitteena oli luoda tietopankkisovellus, joka vastaa parhaiten tilitoimiston sille asettamia vaatimuksia.

Idea tietopankista sai alkunsa ollessani toimeksiantajallani työharjoittelussa kesällä 2009. Tilitoimistoalalla tulee esille hyvin usein ajankohtaisia ja tärkeitä asioita, joiden tiedottaminen koko henkilöstölle ei aina ole helppoa. Tilitoimistolla ei myöskään ollut yhtenäistä tapaa tiedottaa asioista, vaan ne kulkeutuivat henkilöstölle hyvin vaihtelevilla tavoilla. Lisäksi uudet asiat eivät olleet aina helposti henkilöstön löydettävissä. Vaikka asiat olivat tällä tavalla sujuneetkin, oli tiedottamiselle mielestäni myös parempia ratkaisuja. Näin syntyi idea verkkosovelluksesta, johon henkilöstön olisi helppo tallentaa uusi asia tai tiedote. Verkkosovelluksesta myös tietojen hakeminen olisi selkeämpää. Kesän 2009 jälkeen jäin kehittelemään ideaa tietopankista, josta lopulta muodostui aihe opinnäytetyölleni keväällä 2010.

Avainsanat: verkkosovellus, tietopankki, intranet, tilitoimisto.

ABSTRACT

Tampere University of Applied Sciences

Business Information Systems

RAUHANEN PAULI: Databank application for an accounting company

CASE: Näsitilit Oy

Bachelor's thesis 47 s, enclosures 3 s

May 2011

This thesis describes producing and planning of databank application for use in the accounting company environment. The commissioner of the thesis is Näsitilit Oy, which is a accounting company from Tampere. My goal was to create a databank, which would respond to all requirements of a accounting company.

An idea of a databank was born in summer 2009 when I was working as a trainee for the company. In a accounting company, there are a lot of current and important daily matters, and it is sometimes difficult to inform an all employees. In my opinion, there was no a solid way to inform things and the old information was sometimes hard to find. Although things worked out in some way, I thought there were better ways to inform. So, I came up with an idea of a databank, where employees could save new information and bulletins. After my training period I started to develop my idea, which finally became a subject of my thesis.

Keywords: net-application, databank, intranet, accounting company.

Sisältö

1 JOHDANTO

Kirjoittaja on ollut työharjoittelussa tilitoimisto Näsitilit Oy:ssä kesällä 2009 ja 2010 yhteensä 5 kuukautta. Työharjoittelussani tein erilaisia ATK-töitä, kuten verkon ylläpitoa, tietokoneiden asennusta, ohjelmistojen ylläpitoa ja päivityksiä, Internet-sivujen rakentamista ja päivittämistä. Näiden lisäksi töihini kuului perinteisiä tilitoimistotöitä kuten kirjanpidot, tilinpäätökset ja veroilmoitukset.

Tilitoimisto Näsitilit on keskisuuri tilitoimisto Tampereen keskustassa. Tilitoimiston on perustanut nykyinen toimitusjohtaja Satu Sullanmaa vuonna 1994. Tilitoimistossa on tällä hetkellä 9 työntekijää. Toimisto on hiljalleen kasvattanut kokoaan, sillä vuonna 2005 työntekijöitä oli 5. Tätä nykyä tilitoimisto kuuluu koko Suomen laajuiseen tilitoimistoketjuun. Nykyään tilitoimistoala perustuu hyvin pitkälti tietotekniikkaan ja työt tehdään miltei pelkästään tietokoneella.

Tilitoimistoalalla työntekijän tulee hallita monia erilaisia tiedonaloja. On erilaisia lakeja ja asetuksia, kuten esimerkiksi verolait, kirjanpitolaki ja asetus, lait erilaisille yhtiömuodoille, sekä tilintarkastuslaki. Monesti tilitoimisto hoitaa myös asiakkaan palkanlaskennan, jolloin vastaan tulee kyseisen alan työehtosopimuksen opettelu. Nykyään tilitoimistoilla, kuten myös toimeksiantajallani, on tapana jakaa vastuualueet eri työntekijöille. Tästä huolimatta jokaisella työntekijällä on hyvä olla perustiedot mahdollisimman monelta tietoalueelta. Lisäksi jotkin tietoalueet ovat niin laajoja, että yhden ihmisen on niitä vaikea kokonaan ulkoa omaksua.

Kun alalla tuli uusi lakimuutos, pidettiin siitä työharjoittelupaikassani palaveri, jossa asia käsiteltiin. Toinen vaihtoehto oli, että uusi asia välitettiin muille työntekijöille sähköpostitse. Kun tilitoimiston koko on suurempi, voivat nämä molemmat tavat olla hieman ongelmallisia. Kun työntekijöitä on enemmän, eivät kaikki välttämättä ole samaan aikaan paikalla. Sähköpostilla lähetetyt viestit tavoittavat kohteensa, mutta tällöin yksi tieto on tallennettuna moneen kertaan. Tärkeintä kuitenkin olisi, että työntekijöillä olisi yksi tietty paikka, mistä he voisivat ongelmallisen tilanteissa tietoa hakea. Työharjoittelun aikana tuli puheeksi

ATK-vastaavan kanssa tämä asia, ja tästä syntyi idea tilitoimiston sisäisestä tietopankista.

Opinnäytetyön toteutusosan tarkoituksena oli luoda tilitoimistopohjainen tietopankki. Tietopankki olisi WWW-sovellus, joka olisi vain tilitoimiston henkilökunnan käytössä, eli toimisi kuten intranet. Kun uusi ja ajankohtainen asia ilmaantuu, voisivat työntekijät lisätä sen tietopankkiin kaikkien nähtäville. Toimeksiantajallani ei ollut entuudestaan minkäänlaista tietopankkia tai intranetiä, joten minun on aloitettava työni aivan alusta.

Tietopankki verkkosovelluksena toimii toimeksiantajan ympäristössä parhaiten, sillä tällöin joka työpisteelle ei tarvitse tehdä omia asennuksia. WWW-sovelluksen ohjelmointikielenä käytin PHP:tä ja relaatiotietokannan hallintaan MySQL:ää, sillä ne tukevat hyvin opiskeluitani ja minulla on niistä entuudestaan jo kokemusta. Minulle uutena tekniikkana käytän työssäni JavaScriptia.

Tämän opinnäytteen tarkoituksena on kertoa tietopankin suunnittelusta sekä toteuttamisen eri vaiheista. Opinnäytetyö esittelee myös sovelluksen lukijalle ja kertoo, kuinka sitä käytetään. Opinnäytetyöstä voi olla hyötyä myös henkilölle, joka on luomassa tai kehittämässä sisäistä verkkoa tilitoimistolle tai vastaavalle yhteisölle.

Lähteiksi olen pyrkinyt löytämään ensisijaisesti kirjoja, mutta joissakin asioissa, kuten käytetyissä tekniikoissa, olen käyttänyt lähteenä näiden tekniikoiden kotisivuja. Koska sovellukseni on verkkosovellus, löytyy siinä käytetyistä tekniikoista paljon luotettavaakin tietoa Internetistä. Pidän kaikkia lähteitäni luotettavina, koska pyrin käyttämään lähteinäni pääasiassa kirjoja. Kaikki Internet-lähteeni ovat ohjelmistokehittäjien omia sivuja, joiden luotettavuutta en epäile. Itse sovellusta rakentaessa olen etsinyt erilaisia ratkaisuja ja toimintojen toteuttamiseen liittyviä kysymyksiä Internetistä. Näiden lähteiden luotettavuus näkyy itse ohjelmointikoodin toimivuudessa.

Alussa pohjustetaan työn lähtötietoja, minkä jälkeen määritellään sovellukselle vaatimuksia. Vaatimusten määrittelyn jälkeen kerrotaan suunnittelusta ja käydään läpi sovelluksen rakentamisessa käytetyt tekniikat.

Itse sovelluksesta kerrotaan ensiksi tietokannasta ja rakenteista, jonka jälkeen esitellään sovelluksen toiminnot. Lopuksi käydään läpi sovelluksen jatkokehitysmahdollisuudet ja parannusehdotukset, sekä yhteenvedossa arvioidaan muun muassa opinnäytetyön onnistumista.

2 TAUSTATIETOA

Tässä luvussa kerrotaan joitakin lähtötietoja, jotka vaikuttavat sovellukseen ja sen rakenteeseen.

2.1 Miksi WWW-sovellus?

Päädyin tekemään tietopankista WWW-sovelluksen, koska mielestäni tässä tapauksessa se toimii näin parhaiten. Jokaisella työntekijällä on käytössään tietokone, jossa on jonkinlainen WWW-selain, useimmilla Internet Explorer. Sovelluksen käyttöönotto ei näin vaatisi erityisiä asennuksia, koska se toimisi työkoneilla jo olevan selaimen avulla. Selaimen avulla toimiva WWW-sovellus olisi mahdollisesti helposti käytettävissä myös työpaikan ulkopuolella. Lisäksi kirjoittaja on suuntautunut opiskeluissaan WWW-puolelle.

2.2 Ympäristö

Sovelluksen ympäristönä toimii toimisto, jossa jokaisella työntekijällä on oma tietokone. Toimiston koneet ovat yhteydessä pääkoneeseen eli palvelimeen. Palvelin koneella sijaitsee toimiston yhteiset tiedostot, joihin on kaikkien työntekijöiden mahdollista päästä käsiksi omilta koneiltaan. Lisäksi palvelimelle on asennettu toimistossa käytetyt ohjelmat. Tietopankki-sovellus on tarkoitus asentaa palvelimelle, josta työntekijät pääsevät siihen käsiksi omilta koneiltaan.

2.2 Intranet

Tietopankki-sovellus toimii kuten intranet eli sovellus on vain työntekijöiden käytössä. Tarkoitus on luoda sovelluksesta sellainen, että sitä olisi mahdollista

laajentaa, tai se voitaisiin liittää osaksi suurempaa kokonaisuutta. Tulevaisuudessa tietopankki voisi siis olla osa suurempaa kokonaisuutta.

Intranet on tietyn ryhmän käytössä oleva sisäinen verkko, joka parhaimmillaan pystyy toimimaan esim. jonkun yrityksen tai organisaation informaatiorakenteen perustana. Intranetin avulla voidaan hoitaa yrityksen sisäinen viestintä ilman ylimääräisten sähköpostien lähetystä. Yrityksen sisällä intranet mahdollistaa tiedon hankkimisen toiselta yksiköltä ilman henkilöstön vaivaamista (Nielsen 2000, 276–278).

Tietopankki-sovelluksen avulla tavoitellaan juuri samoja etuja ja mahdollisuuksia. Henkilöstöllä menee päivittäin paljon aikaa tiedon etsimiseen, koska ei tiedetä, mistä tietoa kannattaisi ensiksi etsiä. Lisäksi tietoa joudutaan joskus lähettämään sähköpostitse, jolloin se tallentuu moneen paikkaan, niin lähettäjälle kuin vastaanottajille. Tietopankki-sovelluksessa tieto olisi yhdessä paikassa, eikä sitä tarvitsisi välittää sähköpostilla.

3 TARPEIDEN MÄÄRITTELY JA SUUNNITTELU

Tässä luvussa käydään ensiksi läpi sovellukselle havaittuja ja asetettuja vaatimuksia, jonka jälkeen suunnitellaan sovelluksen rakennetta enemmänkin teoreettisesta näkökulmasta. Suunnittelussa verrataan sovellusta intranetiin, koska Tietopankki-sovellus tulee käytännössä toimimaan kuin intranet.

3.1 Tarpeiden määrittely

Tietopankki-sovelluksen tarpeiden ja vaatimusten määritys tapahtui työharjoittelussa tehtyjen havaintojen sekä työntekijöiden kanssa käytyjen keskustelujen perusteella. Havaintojen perusteella tulin siihen tulokseen, että sovelluksessa tulisi olla kaksi ominaisuutta ylitse muiden. Tietojen tallentaminen tulee olla yksinkertaista ja niiden hakeminen tulee olla helppoa.

Sovelluksen toimivuus työympäristössä edellyttää, että sen käyttäjät lisäävät uusia tietoja kantaan niiden ilmentyessä. Jotta tietoja lisätään, niiden lisääminen on oltava yksinkertaista. Jos tiedon lisääminen koetaan haastavaksi, jätetään ne lisäämättä ja jatketaan mieluummin vanhoilla menetelmillä.

Tiedon löytyminen tietokannasta edellyttää sitä, että tieto on ensinnäkin lisätty kantaan. Kun tieto on kannassa, on seuraava edellytys sen löytymiselle hakutoimintojen toimivuus. Tietopankin päätehtävänä on mielestäni tiedon säilyttäminen. Ja jotta tiedon säilyttämisessä olisi järkeä, tulisi sen myös löytyä helposti. Olettamus on se, että tietokannassa tulee olemaan paljon tietoa eri alueilta ja aloilta. Eri alojen tiedot pitää siis olla hyvin ryhmitelty. Kun tiedot ovat hyvin ryhmitelty ja ryhmien nimet ovat osuvat, on tiettyä tietoa helpompi etsiä suuresta massasta.

Tilitoimistoalalla tietoon ja tiedotteisiin voi tulla ajan myötä muutoksia. Jos jo tietopankissa olevaan tietoon tulee laissa pieni muutos, ei ole järkevää, että muutosta varten luodaan kokonaan uusi tieto. Olemassa olevia tietoja pitäisi

siis pystyä muokkaamaan. Lisäksi kokonaan vanhentuneita tietoja pitää pystyä poistamaan sovelluksen avulla, jotteivät ne tuota lukijoilleen väärinkäsityksiä.

Tietopankki-sovelluksen yleisilme ei saa olla liian monimutkainen. Toimintoja tulee olla vain sen verran kuin on välttämätöntä. Ulkoasun ei tarvitse välttämättä olla kovin erikoinen, kunhan tarvittavat toiminnot ovat selkeästi esillä. Sovellukselle tulee olla myös helposti ymmärrettävät käyttöohjeet. Mitä helpompi sovellus on käyttää, sitä nopeammin työntekijät tottuvat käyttämään sovellusta muilta kiireiltään.

3.2 Suunnittelu

Intranetin suunnittelu ei juuri eroa tavallisten Internet-sivujen suunnittelusta. Intranetin suunnittelua on kuitenkin lähestyttävä hieman eri lähtökohdista, kuin Internet-sivujen suunnittelua. Internet- ja intranetsivuilla on kuitenkin eri päämäärät ja käyttäjäkuntansa. Internet-sivut tehdään yleensä asiakkaille, kun taas intranet työntekijöille. Tietopankin päämäärä on tiedon tallettaminen ja jakaa sitä edelleen tietoa tarvitseville (Nielsen 2000, 263–264).

Intranetin käytettävyys heijastuu suoraan työntekijöiden tehokkuuteen ja sitä kautta yrityksen tulokseen, koska intranetin tarkoituksena on yleensä auttaa työntekijöitä tekemään työnsä tehokkaammin. Intranetin suunnittelussa ja toteutuksessa tärkeimpiä tavoitteita käytettävyyden kannalta ovat käytön tehokkuus, virheiden minimointi ja helposti muistettavat toiminnot (Nielsen 2000, 274).

Tietopankki-sovellus on tehokas silloin, kun tietojen tallentaminen sekä etsiminen että lukeminen on helppoa. Koska toimeksiantajalla ei ole aikaisemmin ollut minkäänlaista tietopankkia, jo yksinkertainenkin tietojen tallentamisjärjestelmä tekisi työntekijöiden työstä tehokkaampaa. Yksinkertainen järjestelmä ei kuitenkaan kannustaisi työntekijöitä käyttämään sitä, vaan suunnittelussa ja

toteutuksessa pitää ottaa huomioon myös käytettävyys. Virheet saadaan sovelluksesta minimoitua toimintojen selkeydellä. Kun toimintoja on vähän, mutta riittävästi, ovat ne helpommin muistettavissa. Tämän takia pyrin rakentamaan sovellukseen vain välttämättömät toiminnot ja pyrin tekemään niistä selkeät.

Intranetissä on kolme tärkeää elementtiä: hakemisto, hakutoiminto ja uutiset. Näistä ehkä tärkein on hierarkkinen hakemistorakenne, jonka avulla intranetin sisältö katetaan. Hakemisto kootaan aina jokaisen yrityksen intranetin sisällön mukaisesti ja sen avulla pystyy sisältöä selaamaan. Intranetin etusivulla tulisi olla aina myös hakukone, jolla käyttäjä pystyy hakemaan intranetistä haluamaansa sisältöä. Hakukoneen tulee kattaa kaikki intranetiin liittyvä sisältö. Lisäksi intranetin etusivulla olisi hyvä olla yritykseen liittyviä uutisia, jotka olisivat käyttäjien, eli yleensä työntekijöiden, kannalta mielenkiintoisia. Nämä kolme seikkaa tukevat sitä, että työntekijä saavat kaipaamansa tiedot intranetistä, eivätkä käytä muita kannattamattomampia kanavia tiedonjakoon (Nielsen 2000, 279–280).

Tietopankki-sovelluksessa tärkeitä toimintoja ovat varsinkin hakukone ja hakemisto sisällöstä, jotka olisivat hyvä olla aina käyttäjän saatavilla. Myös virhemahdollisuuksien minimointi on erityisen tärkeää, sillä tällöin tietopankin käyttö olisi mahdollisimman tehokasta. Käyttäjillä ei välttämättä ole paljon kokemusta tietotekniikasta ja tällöin virheen tullessa joudutaan kääntymään asiantuntijan puoleen. Ja tämä laskee tehokkuutta.

4 KÄYTETYT TEKNIIKAT

Tässä luvussa käydään läpi sovelluksen rakentamisessa käytettyjä tekniikoita.

4.1 (X)HTML

HTML (HyperText Markup Language) on tietotekninen merkkauskieli, joka toimii perustana WWW:lle eli World Wide Web:lle. HTML on yksinkertainen sivunkuvaus kieli, jota käytetään Internet-sivujen luontiin (Anderson & King 2002, 1).

HTML-kieli perustuu merkkaus-tageihin eli elementteihin, joita käytetään HTML-sivun sisällön kuvaamiseen. HTML-tagi muodostuu sen nimestä, joka on ympäröity kulmasulkeilla, kuten . Melkein kaikki HTML-elementit vaativat sekä aloitus-, että lopetus tagin. HTML-tageista ja tavallisesta tekstistä muodostuu sitten HTML-dokumentteja, jotka kuvaavat verkkosivuja (W3schools 2010a).

HTML-dokumentteja voidaan luoda perinteisellä tekstieditorilla. HTML-dokumentin tunnistaa .html-päätteestä. Selainten, kuten Internet Explorer tai Mozilla Firefox, tehtävänä on lukea näitä HTML-tiedostoja ja näyttävät tiedostot käyttäjälle verkkosivuna. Jotta selain pystyy lukemaan tiedostoa, on HTML-tiedostoille olemassa yhtenäiset määritykset (Anderson 2002, 3).

Esimerkkejä HTML-kielestä:
 Tämä on yksi kappale, jossa p on tagin nimi ja on lyhenne sanasta paragraph.
<h1>Tämä on otsikko, jossa h tarkoittaa header.</h1>

Yksittäinen Internetin käyttäjä pystyy myös näkemään jokaisen verkkosivun lähdekoodin, eli käytännössä tämän HTML-tiedoston (Anderson 2002, 3). Mozilla Firefoxissa tämä onnistuu painamalla hiiren oikeaa painiketta verkkosivun

päällä ja valitsemalla tämän jälkeen ruudulle ilmestyvästä valikosta *Näytä sivun lähdekoodi*.

XHTML (*eXtensible Hypertext Markup Language)* on uudistettu versio HTML:stä. XHTML käyttää XML-kielen syntaksia. Suurin osa XHTML:n elementeistä ja attribuuteista ovat samoja kuin HTML:ssä, mutta XHTML:n säännöt määritykset ovat tiukemmat. (Duckett 2005, 39).

HTML:ää pidetään nykyään huonona merkkauskielenä, vaikka se toimiikin monella selaimella ihan hyvin. Nykypäivänä selaimia on hyvin moneen tarkoitukseen. Jotkin selaimet ovat tehty pöytäkoneille, toiset puhelimille tai muille pienemmille laitteille. Kaikilla selaimilla ei ole kykyä tulkata käyttäjälle vähän sinnepäin olevaa HTML-koodia. Tästä syystä merkkauskieleksi suositellaan XHTML:ää, jota pidetään käytännöllisenä nyt ja tulevaisuudessa (W3Schools 2010b).

Verkkosivujen tekijä voi varmistu siitä, että hänen dokumenttinsa noudattavat XHTML-säädöksiä, vahvistamalla ne W3C validaattorilla. Validaattori löytyy esimerkiksi sivulta http://www.w3schools.com/site/site_validate.asp (W3Schools 2010c).

4.2 CSS

CSS (Cascading Style Sheets) on kieli, jonka avulla voidaan määritellä, miltä kukin HTML-elementti WWW-sivulla näyttää, ja miten se sivulla esiintyy. CSS:n avulla voit vaikuttaa mm. verkkosivun fontteihin, tekstin sijaintiin, väreihin ja moniin muihin seikkoihin (Duckett 2005, 87).

Tyyliarkit (Style Sheets) voidaan tallentaa omaan tiedostoon, jonka tiedostopääte on .css. Näin voidaan säästää paljon vaivaa, sillä muokkaamalla yhtä CSS-tiedostoa, voidaan vaikuttaa kaikkiin verkkosivuihin, joissa CSS-tiedosto on linkitettynä (W3schools 2010d).

4.3 Tietokanta

Tietokanta on kokoelma erilaisista yksittäisistä tiedoista, joilla on kuitenkin jokin yhdistävä tekijä. Tietokannan yleinen määritelmä pitää olla laaja, koska on paljon ohjelmistoja, jotka tarjoavat hyvin erilaisia tietokantaratkaisuja (Oppel, 2004, 23).

Alkujaan tietokannat olivat vain sellaisten tutkijoiden käytössä, jotka pyrkivät tekemään niiden käytöstä hyödyllistä ja kannattavaa. Nykyään tietokantoja esiintyy kaikkialla ja ne ovat yksi tärkeimpiä tietotekniikan osia teollisuudessa ja taloudessa yleensä (Oppel, 2004, 21).

4.3.1 Relaatiotietokanta

Relaatiotietokanta on tietokanta, jonka tietotaulujen välille voidaan luoda relaatioita eli yhteyksiä. Näin voidaan yhdistää tietoa monesta eri taulusta, ja sen avulla taas pystytään luomaan monipuolisempia tulosteita, sekä raportteja erilaisista näkökulmista (Vawani 2005, 6).

4.3.3 Taulut

Relaatiotietokannassa tiedon ensisijainen tallennusyksikkö on taulu. Taulu koostuu kahdesta eri ulottuvuudesta; riveistä ja sarakkeista. Jokainen rivi on taulua edustava itsenäinen kokonaisuus, ja jokainen sarake edustaa ominaisuutta tai määritettä tälle itsenäiselle kokonaisuudelle (Oppel, 2004, 33). Kuvassa 1 on esimerkki taulusta. Esimerkkitaulun nimi on työntekijä. Taulun riveillä on eri työntekijöitä ja sarakkeilla niiden attribuutit eli ominaisuudet.

Työntekijä

Nimi	Ikä	Asema	Kuukausipalkka
Mäkinen	32	Johtaja	3 000 €
Lehikoinen	30	Varastomies	2 500 €
Nykänen	45	Asentaja	2 500 €

Kuva 1. Excelillä tehty työntekijätaulu esimerkiksi

4.3.2 SQL

SQL on maailmanlaajuisesti käytetty kieli relaatiotietokantojen hallinnassa. SQL:n suosio perustuu siihen, että se toimii kielenä useimmissa relaatiotietokantojen hallintatyökaluissa (Oppel, 2004, 89).

4.3.1 MySQL

MySQL on tietokanta sekä sen hallintatyökalu, joka on keskittynyt relaatiotietokantoihin. MySQL:n saa ilmaiseksi omalla lisenssillään, mutta jos levittää ohjelmistoa kaupallisesti, pitää olla maksullinen lisenssi. MySQL sopii hyvin myös pienempien sovellusten tietokannaksi, sillä se ei vaadi paljon ylläpitoa ja se on helppo asentaa. (Heinisuo & Ranta 2007, 37–38).

MySQL toimii monilla sovellusalustoilla, joista tunnetuimpia ovat Windows, Linux ja Mac Os. Järjestelmän suosiosta kertoo paljon se, että sitä käyttävät suuret yritykset kuten Yahoo!, Google, Nokia ja Youtube (MySQL 2010).

4.4 PHP

PHP (HyperText Preptocessor) on avoimen lähdekoodin (Open Source) skriptikieli, joka sopeutuu parhaiten palvelinpuolen ohjelmointiin. PHP vaatii toimiakseen WWW-palvelimelle asennettavan PHP-ohjelmiston, joka generoi tuloksen

käyttäjän selaimelle näytettäväksi. PHP suosion kasvettua 2000-luvun alun jälkeen, ovat sen laajuus ja ominaisuudet lisääntyneet huomattavasti. PHP-tiedostot voidaan luoda esimerkiksi tekstieditorilla ja ne tunnistaa .php-päätteestä. PHP-koodi sijoitetaan tagien <? ja ?> väliin, joilla sitä voi sijoittaa myös HTML-koodin sekaan (Kolehmainen 2006, 3–4).

PHP:ssa on tuki MySQL-tietokantaohjelmistolle, joten MySQL:n käsittely onnistuu PHP:lla. PHP:llä on omat funktiot, joilla voi luoda yhteyden tietokantaan, sekä valita tietokannan. PHP:n avulla voidaan mm. myös lisätä tietokantaan tietoja sekä tehdä erilaisia kyselyitä tietokannasta (Kolehmainen 2006, 281, 295–296).

4.5 JavaScript

Javascript on suosittu skriptikieli, jolla pystyy luomaan HTML-sivuista näyttävämpiä ja lisäämään niille vuorovaikutteisuutta. Javascript toimii usealla selaimella, eikä sen käyttö vaadi lisenssejä JavaScript on melko yksinkertainen skriptikieli ja näin sitä on helppo lisätä HTML- sivuille pieninä osina (W3schools 2010e).

Skriptikielellä tarkoitetaan ohjelmointikieltä, jolla voi muokata, käsitellä tai automatisoida toimivaa järjestelmää. Skriptikieli liitetään itse ohjelmaan, jolloin sillä pystytään muokkaamaan ja laajentamaan itse sovelluksen toimintoja. Skriptikielien vaatimukset eivät aina ole yhtä tarkat kuin sovelluskehityskielissä, kuten esim. C++ (Peltomäki 2006, 7 & 9).

5 TIETOKANNAN TOTEUTUS

Luvussa kerrotaan ensiksi sovelluksen tietokantatauluista. Tämän jälkeen käydään läpi tietotauluissa käytetyt tietotyypit ja attribuutit. Viimeiseksi selvitetään, kuinka sovelluksen tietokanta on rakennettu.

5.1 Tietopankki-sovelluksen tietokantataulut

Tietokantataulujen huolellinen suunnitteleminen vaikuttaa sovelluksen toimivuuteen monella tapaa myöhemmin. Jos tauluja ei suunnittele hyvin, voi se näkyä jopa sovelluksen suorittamisessa tai ylläpitämisessä. Lisäksi hyvä suunnittelu johtaa siihen, että tiedot on tallennettu järkevästi, eivätkä vie turhaa tilaa. (Gilmore 2005, 565).

Tietopankki-sovelluksen tietokannassa on yhteensä neljä taulua. Tietokannassa on kaksi erilaista kategoria taulua, koska mielestäni tietojen lajitteleminen on erittäin tärkeää tietopankissa, johon on tarkoitus tallettaa tietoja paljon monelta eri alueelta. Näiden lisäksi on yksi taulu itse tiedoille sekä yksi taulu, johon talletetaan ylläpitäjien tiedot. Kategoriataulut sekä tietotaulu ovat yhteydessä toisiinsa.

Pää- ja alakategoriat olisi voinut mahdollisesti tallentaa myös samaan tauluun, mutta mielestäni selkeyden vuoksi oli hyvä luoda molemmille omansa. Näin tietoja voi hakea ja lajitella helposti niin pää- kuin alakategoriankin mukaan.

5.1.1 Pääkategoriat

Pääkategoriat talletetaan Tietopankki-sovelluksessa tauluun, jonka nimi on categories. Categories-taululla on atribuutteina cat_id, cat_name ja cat_info. Cat_id on kategorian id-numero, cat_name on kategorian nimi ja cat_info on kategorian selite.

Sovelluksessa ei ole mahdollisuutta lisätä, poistaa tai muokata pääkategorioita. Jos pääkategorioihin tulee muutoksia, on ne muutettava suoraan tietokantaan. Olen syöttänyt pääkategoriat jo valmiiksi tietokantaan. Sovelluksesta löytyvät pääkategoriat käytiin toimeksiantajan kanssa läpi, ja uskon että, ne kattavat

riittävän hyvin taloushallinnon eri toimialueet. Lisäksi sovellus menisi liian sekavaksi, jos sen avulla olisi mahdollista muokata, poistaa tai lisätä myös pääkategorioita. Esimerkiksi, jos ylläpitäjä poistaa yhden pääkategorian, pitäisi sen alla olevat kategoriat ja tiedot ohjata samalla johonkin muuhun kategoriaan. Jos yhdeltä tiedolta poistaa pääkategorian, jää se ikään kuin ilmaan kellumaan, ja sitä on tämän jälkeen vaikea löytää sovelluksen avulla. Kuvassa 2 on esitetty categories-taulun rakenne ja kuvassa 3 categories-taulun sisältö.

categories			
PK	**cat id**	**INT(7)**	**auto_increment**
	cat_name cat_info	**VARCHAR(50)** VARCHAR(500)	

Kuva 2. Kuvassa categories-taulun rakenne

cat_id	cat_name	cat_info
1	Verot	Kaikki veroihin liittyvä
2	Kirjanpito	Kaikki kirjanpidosta, kuten lait ja säädökset
3	Palkanlaskenta	Kaikki palkanlaskennasta, kuten työehtosopimuksiin liittyvät asiat
4	Tilintarkastus	Kaikki tilintarkastukseen liittyvä, kuten lait ja käytännöt
5	Uutiset	Talon sisäiset asiat
6	Muut	Jos jotain muuta, niin tänne

Kuva 3. Kuvassa categories-taulun sisältö 14.12.2010

5.1.2 Alakategoriat

Alakategorioiden taulun nimi tietokannassa on sub_categories. Taulussa on neljä attribuuttia; sub_cat_id, cat_id, sub_cat_name sekä sub_cat_info. Attribuutit määräytyvät muuten samalla tavalla kuten categories-taulussakin, paitsi että sub_categories-taulussa on myös cat_id attribuutti. Attribuutin avulla alakategorioita viitataan johonkin pääkategoriaan. Alakategoriat voivat siis kuulua johonkin pääkategoriaan. Esimerkiksi, jos sub_categories-tauluun perustetaan

uusi tietue nimeltä *Arvonlisäverot*, olisi tämän arvo kohdassa cat_id 1, eli alakategoria arvonlisäverot kuuluisi pääkategoriaan verot. Kuvassa 4 esitellään sub_categories-taulun rakenne.

sub_categories			
PK	**sub_cat_id**	**INT(7)**	auto_increment
	cat_id	INT(7)	
	sub_cat_name	**VARCHAR(50)**	
	sub_cat_info	VARCHAR(500)	

Kuva 4. Kuvassa sub_categories-taulun rakenne

5.1.3 Tietotaulu

Taulu, johon itse tietueet tallennetaan, on nimeltään data. Data-taulussa on useampia attribuutteja. Kuvassa 5 on tarkennettu data-taulun attribuuttien tarkoituksia. Kuvassa 6 on esitelty data-taulun rakenne.

data	
data_id	Tiedon tunniste numero
cat_id	Kategorian numero, mihinkä tieto kuuluu
sub_cat_id	Alakategorian numero, mihinkä tieto kuuluu
data_name	Tiedon nimi
data_intro	Tiedon selite
data_text	Itse tieto
data_add_date	Tiedon lisäyspäivämäärä
data_last_edit_date	Tiedon muokkauspäivämäärä
data_add_name	Tiedon lisääjän nimi
data_add_mail	Tiedon lisääjän sähköposti

Kuva 5. Data-taulun attribuutit ja niiden tarkoitukset

data			
PK	**data_id**	**INT(7)**	auto_increment
	cat_id	INT(7)	
	sub_cat_id	INT(7)	
	data_name	VARCHAR(100)	
	data_intro	VARCHAR(255)	
	data_text	TEXT	
	data_add_date	timestamp	CURRENT_TIMESTAMP
	data_lastedit_date	datetime	
	data_add_name	VARCHAR(50)	
	data_add_mail	VARCHAR(70)	

Kuva 6. Data-taulun rakenne

5.1.4 Ylläpidon taulu

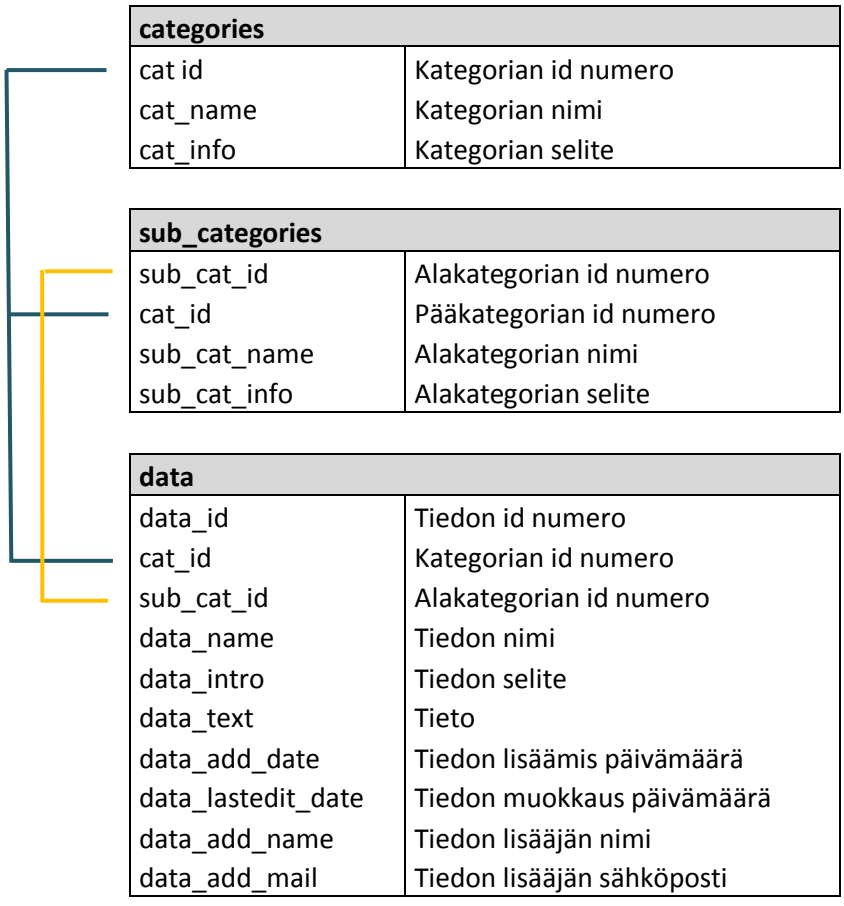
Koska sovelluksessa on oma ylläpitopuoli, on ylläpidolle luotu oma taulu, jonka nimi on user. User-tauluun tallennetaan kaikkien ylläpitäjien tiedot, kuten esim. käyttäjätunnukset ja salasanat. Kun ylläpito puolelle kirjaudutaan sisään, tarkastaa sovellus syötetyn käyttäjätunnuksen ja salasanan user-taulusta. Kuvassa 7 esitellään user-taulun rakenne.

user			
PK	**user_id**	**INT(7)**	auto_increment
	user_name	VARCHAR(50)	
	user_pass	VARCHAR(50)	
	user_realname	VARCHAR(50)	
	user_mail	VARCHAR(50)	
	user_phone	VARCHAR(20)	

Kuva 7. user-taulun rakenne

5.1.5 Taulujen relaatiot

Molemmat kategoriataulut ovat yhteydessä data-tauluun niin, että data-tauluun tallentuu tiedon pää- ja alakategorioiden id-numerot. Tämä mahdollistaa sen että data-taulun tietoja voi lajitella ja hakea kategorioiden mukaan. Lisäksi sub_categories taulussa on jokaisella alakategorialla sen pääkategorian id-numero, johon se kuuluu. Näin alakategoriat ovat siis jonkun pääkategorian alaisuudessa. User-taulusta ei ole relaatiota muihin tauluihin, koska pääkäyttäjien tiedot eivät liity mitenkään Tietopankkiin tallennettaviin tietoihin. Kuvassa 8 esitetään taulujen relaatiot.



Kuva 8. Taulujen relaatiot

5.2 Tietotauluissa käytetyt tietotyypit ja attribuutit

Tässä alaluvussa esitellään Tietopankki-sovelluksen tietokantatauluissa käytetyt tietotyypit sekä attribuutit. Tietotyyppejä ja attribuutteja on olemassa paljon enemmän, mutta niihin ei tässä sen tarkemmin perehdytä. Kappaleen avulla pyritään selvittämään lukijalle kappaleessa 4.1 esiteltyjä tietokanta rakenteita.

5.2.1 Tietotyypit

Tietokantataulujen sarakkeisiin tallennetaan tietoja, jotka voivat olla tietotyypiltään hyvin erilaisia. Tallennettavia muotoja voivat olla esimerkiksi merkkijonot, numeeriset, tai vaikka päivämäärä ja aika-tyyppiset tallenteet. Tämä johdosta MySQL:ssä on mahdollisuus antaa jokaiselle sarakkeelle oma tietotyyppi, joka määrittelee sarakkeeseen tallennettavan tiedon ja antaa sille tietyt raamit (Gilmore 2005, 572).

DATE-tietotyypille tallennetaan päivämäärätietoja. Tietotyypille tallennetaan tietoja muodossa YYYY-MM-DD, eli esimerkiksi päivämäärä 21.12.2010 tallennettaisiin taulun päivämääräsarakkeeseen muodossa 2010-12-21. MySQL mahdollistaa tiedon tallentamisen DATE-tietotyypille myös merkkijonona, eli esim. muodossa 20101221 (Gilmore 2005, 572).

TIMESTAMP-tietotyyppi on aikaan ja päivämäärään liittyvä tyyppi. MySQL päivittää sen aina, kun siihen tulee lisäys tai päivitys. TIMESTAMP-tietotyyppiin tallentuu lisäyksen tai päivityksen yhteydessä päivämäärä ja aika, joko numeroina tai merkkijonona, kuten DATE-tietotyypissä (Gilmore 2005, 573).

INT-tietotyyppi on numeerinen, eli se mahdollistaa numeerisen datan tallentamisen. INT-tietotyypillä on mahdollista rajoittaa talletettavan luvun enimmäiskokoa. INT-tietotyypissä määrittely voidaan tehdä asettamalla haluttu numero sulkeiden sisään tietotyypin nimen jälkeen. Esimerkiksi asettamalla jonkin sa-

rakkeen tietotyypiksi INT(7), sallitaan sarakkeeseen tallennettavan vain kokonaislukuja, joissa on enintään seitsemän numeroa (Gilmore 2005, 574).

CHAR-tietotyyppi mahdollistaa merkkijonojen tallentamisen kyseiselle sarakkeelle. CHAR-tietotyypille pystytään myös määrittelemään pituus asettamalla haluttu lukuarvo sulkeisiin tietotyypin nimen perään. VARCHAR-tietotyyppi tukee enintään 255 merkkiä (Gilmore 2005, 576).

TEXT-tietottyyppi mahdollistaa myös merkkijonojen tallentamisen sarakkeelle, mutta tukee enintään 65 535 merkkiä (Gilmore 2005, 577).

5.2.2 Attribuutit

Tietotyyppien toimintaa voi edelleen tarkentaa, lisäämällä sarakkeille attribuutteja. (Gilmore 2005, 572).

AUTO_INCREMENT-attribuuttia käytetään yleensä ID-numero sarakkeessa. Kun sarakkeen attribuuttina auto_increment, lisää MySQL sarakkeen edelliseen arvoon luvun yksi ja tallentaa luvun kyseiselle riville. Näin pystytään luomaan helposti kaikille tauluun syötetyille tiedoille eri ID-numero, jolloin jokaisesta eri tiedosta tulee uniikki ja ne ovat eroteltavissa toisistaan. Yhdessä taulussa voi olla vain yksi auto_increment-sarake. (Gilmore 2005, 577).

PRIMARY KEY-attribuutilla taataan taulun jokaisen rivin ainutkertaisuus. Kun sarakkeen attribuuttina on primary key, mikään sen sarakkeen riveillä olevista arvoista ei saa olla sama. Yleensä käytetään auto_increment-attribuuttia sarakkeessa, joka on määritteeltään PRIMARY KEY (Gilmore 2005, 580).

5.3 Tietokannan ja taulujen toteutus

Jotta tietokantaan voisi lisätä haluamiaan tietotauluja, on ensiksi luotava itse tietokanta. Tietokanta luodaan MySQL:ssä komennolla: ”*CREATE DATABASE tietokannan nimi;”* (Gilmore 2005, 582). Tietopankki-sovelluksessa tietokannan nimenä on tietopankki.

Kun tietokanta on luotu, pitää käyttäjän valita tietokanta päästäkseen luomaan sinne tietotauluja. Tietokantaan siirtyminen tapahtuu komennolla: ”*USE tietokannan nimi;”*. Kun tietokanta on luotu ja sinne on siirrytty, voidaan kantaan luoda tauluja. Taulun luonti tapahtuu yleisesti *CREATE TABLE*-komennolla. Lisättyyn tauluun saadaan sisältöä INSERT INTO -komennolla. (Gilmore 2005, 582–583 & 628).

Liitteestä 1 löytyvät komennot, joilla on luotu Tietopankki-sovelluksen tietokanta ja taulut. Liitteestä löytyvät myös komennot, joilla lisätään categories-taulun sisältö sekä ensimmäisen ylläpitäjän tiedot, kuten käyttäjätunnus ja salasana.

6 SOVELLUKSEN TOIMINNOT JA NIIDEN TOTEUTUS

Tässä luvussa esitellään Tietopankki-sovelluksen toiminnot ja kerrotaan, kuinka niitä tulisi käyttää.

6.1 Tietopankki

Tietopankki-sovelluksen etusivu on toteutettu noudattaen Jacob Nielsenin tapaa rakentaa Intranet-portaalin etusivu, koska myös itse pidän näitä ominaisuuksia tärkeinä juuri Tietopankki-sovelluksessa. Nielsenin (2000, 279–280) mukaan Intranetin etusivulla tulee olla hierarkkinen hakemistorakenne, hakukenttä, sekä ajankohtaiset uutiset.

Tietopankki-sovelluksen etusivun vasemmassa laidassa on esillä kategoriat tietopankin sisällöstä. Tämän valikon nimi on *Kategoriat*. *Kategoriat*-valikko toimii sivuston hierarkkisena hakemistorakenteena, sillä sen avulla näkee kaikki kategoriat ja niiden alakategoriat, joita Tietopankista löytyy. *Kategoriat*-valikon alla on hakukenttä, jonka avulla käyttäjä voi hakea tietoa Tietopankista hakusanan perusteella. *Kategoriat*-valikko, sekä hakukenttä näkyvät käyttäjälle aina sivun vasemmassa reunassa, liikkuivatpa he missä päin sivustoa tahansa. Näin tietojen hakeminen on tehty mahdollisimman helpoksi.

Käyttäjälle näkyy aina myös sivun yläosa, eli niin kutsuttu header. Headerissa on tietopankin ”logo” sekä linkit etusivulle, ohjeisiin, tiedon lisäämiseen sekä ylläpidon puolelle. Sivun keskellä on tila tulosteelle, johon sovellus tulostaa tietoa käyttäjän valintojen mukaisesti. Kutsun aluetta tekstissäni *Info*-alueeksi. Kun Tietopankki-sovellus avataan, *Info*-alueelle tulostuu tervehdysteksti, sekä Tietopankkiin viimeisimpinä lisätyt tietueet kategoriasta *Uutiset*. *Uutiset*-kategoriaan on käyttäjien tai ylläpitäjän tarkoitus lisätä lähinnä talon sisäisiä asioita.

Kuvassa 9 on kuvankaappaus sovelluksen etusivusta. Kuvassa hiiri on kohdistettu kategorian *Verot* päälle, jolloin sovellus tuo esille kyseisen kategorian alakategoriat.

Kuva 9. Tietopankki-sovelluksen etusivu

6.1.1 Tietojen selaaminen Kategoriat-valikon avulla

Sovelluksen vasemmassa reunassa sijaitsee siis *Kategoriat*-valikko. Valikon avulla käyttäjä pääsee selaamaan tietoja kategorioiden, sekä alakategorioiden mukaan. Kun käyttäjä vie hiiren jonkun kategorian päälle, näyttää sovellus automaattisesti sen kategorian alakategoriat, jonka päällä hiiri kulloinkin on. Tämän kaltainen valikko oli suunnitteluvaiheessa mielestäni elinehto toimivalle tietojen selaamiselle. Koska tietoja on paljon ja ne ovat hyvinkin eri aloilta, on kategorioilla oltava myös alakategoriat. Muuten yhden kategorian alla olisi liian paljon tietoa, ja tiettyä asiaa olisi miltei mahdoton löytää. Nyt valikon avulla

pystyy selaamaan myös alakategorioita samalla sivulla. Kun käyttäjä klikkaa tiettyä kategoriaa valikosta, tulostaa sovellus *Info*-alueelle kaikki tietueet, jotka ovat kyseisen kategorian alla. Jos käyttäjä haluaa sovelluksen tulostavan vain tietyn alakategorian tietueet, täytyy hänen klikata haluamaansa alakategoriaa.

Info-alueelle sovellus tulostaa tiedot siinä järjestyksessä, missä ne ovat lisätty niin, että viimeiseksi tietokantaan lisätty tieto tulostuu ensimmäisenä. Tiedosta ensimmäisenä ylimpänä näkyy kategoria, jonka jälkeen tiedon nimi sekä oikeassa reunassa päivämäärä, jolloin tieto on lisätty. Nimen ja päivämäärän jälkeen näkyy tiedon kuvaus. Tiedon kuvauksella on tarkoitus kuvata tietoa muutamalla sanalla. Tiedon kuvauksen tarkoituksena on helpotta tiedonhakua. Kuvauksen jälkeen sovellus tulostaa itse tiedon, jonka jälkeen tulee lisääjän nimi ja sähköposti.

6.1.2 Haku

Sivun vasemmassa reunassa, *Kategoriat*-valikon alapuolella, sijaitsee pikahakutoiminto. Käytän toiminnosta nimitystä pikahaku, sillä tähän hakuun ei ole mahdollista asettaa ehtoja. Pikahakutoiminto on myös näkyvissä käyttäjälle aina riippumatta siitä, missä hän sivustolla liikkuikin. Pikahaun avulla sovellus hakee hakukenttään kirjoitettua hakusanaa kaikkien tietueiden nimestä tai selitteestä. Hakusanana voi olla myös jonkin haettavan sanan osa. Jos haun suorittaa esim. sanalla ”a”, tulostuu hakutulokseksi kaikki ne tietueet, joiden nimessä tai selitteessä on a-kirjain. Hakutulokset tulostuvat päivämääräjärjestyksessä niin, että viimeiseksi lisätty tietue tulostuu ensimmäisenä. Tarkennettuun hakuun pääsee klikkaamalla yläreunasta linkkiä *Haku*. Kun linkkiä on painettu, kysyy sovellus ensimmäisenä käyttäjältä pääkategoriaa pudotusvalikolla. Pudotusvalikossa on kaikki pääkategoriat sekä vaihtoehto, jolla voi hakea kaikista kategorioista.

Kun kategoria on valittu, tulostaa sovellus käyttäjälle muut mahdolliset hakutarkennukset. Käyttäjällä on mahdollisuus valita, mistä tietueiden osista hän ha-

kusanalla hakee. Vaihtoehtoina ovat tietueen nimi, kuvaus ja teksti. Lisäksi on mahdollista hakea lisääjän nimen perusteella. Valinnat tehdään lisäämällä ruksi haluttuun tai haluttuihin hakualueisiin. Jos käyttäjä ei laita ruksia mihinkään vaihtoehdoista, suoritetaan haku niistä kaikista.

Mikäli käyttäjä on valinnut hakualueeksi jonkin kategorian, voi hän tarkentaa hakua myös tämän kategorian alakategorioihin. Käyttäjällä on mahdollisuus valita joko yksi alakategoria tai kaikki alakategoriat, mistä haku suoritetaan.

Kun hakusana on kirjoitettu kenttään ja hakuehdot ovat mieleiset, voidaan haku suorittaa painamalla *Hae*-painiketta. Tämän jälkeen sovellus tulostaa hakutulokset käyttäjälle niin, että viimeiseksi lisätty on ensimmäisenä. Jos haulla ei löydy tulostettavaa, ilmoittaa sovellus sen käyttäjälle.

6.1.3 Tietojen lisääminen

Sivulle, josta voi lisätä tietoa tietokantaan, pääsee käyttäjä kahdella eri tapaa. Suoraan tiedonlisäämissivulle käyttäjä pääsee sivun yläreunassa eli headerissa olevasta linkistä *Lisää*. Toinen reitti tietojen lisäämissivulle löytyy kategorioita selaillessa. Aina kun käyttäjä valitsee jonkin kategorian tietueet tulostettavaksi sivun vasemmasta reunasta, ilmestyy tulosteiden yläpuolelle linkki *Lisää tietue*.

Kun käyttäjä siirtyy tietueen lisäämiseen kategorioiden selaamisen kautta, valitsee sovellus automaattisesti lisättävän tiedon pääkategoriaksi sen kategorian, jonka tietoja käyttäjä oli selaamassa. Jos taas tietojen lisäämiseen siirrytään headerissa olevan linkin kautta, kysyy sovellus ensimmäisen lisättävän tiedon pääkategoriaa.

Kun pääkategoria on tavalla tai toisella valittu, tulostaa sovellus käyttäjälle kentät, joihin voi lisättävään tietueen tiedot syöttää. Ensimmäinen kenttä on pudotusvalikko, josta valitaan tietueelle alakategoria. Sovellus hakee auto-

maattisesti tietokannasta valitun kategorian alakategoriat pudotusvalikkoon. Jos käyttäjä ei valitse alakategoriaa, löytyy tietue vain sille valitun pääkategorian alta.

Tekstikenttiä on viisi, joista neljä on pakollisia. Tietueen lisääminen on pyritty tekemään mahdollisimman yksinkertaisesti, ja siksi kenttiä on vähän. Koska kenttiä on vähän, ovat ne kaikki taas pakollisia, jotta tietueiden lukija saisi tiedoista haluamansa irti riittävän helposti. Kaikilla kentillä on siis tarkoin suunniteltu merkitys. Tietueen nimi- ja itse tekstikenttä ovat itsestään selvyyksiä, kun taas kuvaus ja lisääjän tiedot eivät sitä välttämättä ole. *Kuvaus*-kentällä pyritään helpottamaan tietojen hakua. Kenttään on tarkoitus kirjoittaa muutama tietuetta kuvaava sana perusmuodossa, esim. pilkuilla toisistaan eroteltuna. Kun käyttäjän hakee näin haluamaansa tietoa hakusanalla, löytyy se tietokannasta helpommin. Pikahakutoiminto etsii hakusanaa tietueiden nimestä ja kuvauksesta. Lisääjän tiedot ovat oleellisia, koska on oletettavaa, että käyttäjälle, joka selaa tietueita, voi syntyä jostain asiasta lisäkysymyksiä. Näin lukijalle selviää tietuetta lukiessaan, kuka on sen lisännyt, ja kenellä siitä voisi olla enemmän tietoa. Tiedon lisääjä siis ilmoittaa itsensä samalla tavallaan tietyn tiedon asiantuntijaksi.

Ainoastaan tietueen lisääjän sähköpostia ei vaadita, koska toimiston sisällä pystytään olemaan yhteydessä muihin työntekijöihin jo pelkästään nimen perusteella. Jos käyttäjä kuitenkin lisää kenttään sähköpostinsa, tarkastaa sovellus lähetettäessä tietuetta sähköpostin oikeellisuuden. Sovellus tarkastaa myös että kaikkiin muihin tekstikenttiin on syötetty jotain. Muihin tekstikenttiin ei ole asetettu aluksi muuta vaatimusta, kuin että ne eivät saa olla tyhjiä lähetettäessä. Jos jokin kenttä ei ole sovelluksen mielestä kunnossa, ei tietuetta tallenneta, ja sovellus ympäröi virheellisen kentän punaisella reunuksella ja näyttää käyttäjälle virheilmoituksen. Kun virheet on korjattu, antaa sovellus käyttäjän tallentaa tiedot. Kun tiedot on tallennettu onnistuneesti, tulostaa sovellus ne vielä käyttäjälle. Tämän jälkeen tietue löytyy tietokannasta. Järjestelmässä ei ole mahdollisuutta liittää liitteitä tietueiden mukaan, vaan teksti pitää kopioida tai kirjoittaa erikseen *Asia*-kenttään, johon siis itse teksti tai asia tulee tallettaa.

Kuvassa 10 on kuvankaappaus tietueen lisäämisestä, jossa pääkategoriaksi on valittu kirjanpito.

Kuva 10. Tietueiden lisääminen

6.1.4 Käyttöohjeet

Sivuston käyttöohjeet löytyvät sivun yläreunassa olevan linkin *Ohjeet* takaa. Ennen sovelluksen käyttöä suositellaan kaikille käyttöohjeiden lukemista, jotta sovelluksen käyttö ja toiminta sujuisi mahdollisimman hyvin, eikä ongelmia il-

menisi. Käyttöohjeissa pyritään esittelemään sovelluksen käyttö mahdollisimman selkeästi. Vaikka sovelluksen toiminnat on pyritty tekemään yksinkertaisiksi, ovat käyttöohjeet mielestäni silti välttämättömät.

6.2 Ylläpitäjän sivut

Sovelluksen ylläpitopuolelle pääsee sivun yläreunassa olevasta linkistä *Ylläpito*. Käyttäjätunnus ja salasana kirjoitetaan sovelluksen tulostamille kentille, jonka jälkeen sisään kirjaudutaan *Kirjaudu*-painiketta klikkaamalla. Ylläpitopuolella ylläpitäjän on mahdollista muokata ja poistaa lisättyjä tietueita, sekä lisätä, poistaa ja muokata alakategorioita. Ylläpitäjälle näkyy myös samat asiat kuin peruskäyttäjällekin.

6.2.1 Tietueiden poistaminen ja muokkaaminen

Kun sovelluksen ylläpitäjä on kirjautunut sisään, näkyy hänelle jokaisen tietueen alareunassa linkit *Muokkaa* ja *Poista*. Nämä linkit siis näkyvät vain ylläpitäjälle, kun hän on kirjautunut sisään.

Kun ylläpitäjä klikkaa *Muokkaa*-painiketta, tulostuu hänelle tietueen tiedot kenttiin, joihin hän pystyy tekemään muutoksia. Muokkaajalle tulostuu myös kaksi kenttää, jotka eivät tietueita luettaessa näy. Näihin kenttiin on tarkoitus syöttää muokkaajan nimi ja viimeinen muokkauspäivämäärä. Näiden kenttien tarkoituksena on se, että ylläpitäjät pääsevät seuraamaan tekemiään muutoksia ja näin voidaan välttää päällekkäisiä muokkauksia. Jos ylläpitäjä haluaa näyttää muokkauksen kaikille, voi hän esimerkiksi lisätä muokatun tekstin perään lauseen ”Muokattu 18.12.2010”. Kun ylläpitäjä on korjannut muutokset kenttiin tallentaakseen muutokset, tulee hänen painaa Tallenna-painiketta. *Tallenna*-painikkeen vieressä on *Peruuta*-linkki, josta ylläpitäjä voi peruuttaa muokkauk-

sen ja palata edelliselle sivulle. Onnistuneen muokkauksen jälkeen sovellus ilmoittaa ylläpitäjälle, että tiedot on muokattu onnistuneesti.

Ylläpitäjän klikatessa tietueen alareunassa olevaa *Poista*-painiketta, siirtyy hän sivulle, jossa sovellus kysyy, haluaako ylläpitäjä varmasti poistaa kyseisen tietueen. Vastaamalla edelleen poista, poistuu tietue tietokannasta. Varmistussivulla on myös *Peruuta*-linkki, josta ylläpitäjä voi peruuttaa poiston ja palata edelliselle sivulle.

Kuvassa 11 on kuvankaappaus ylläpidon etusivulta. Etusivun tietueen alapuolella näkyy *Muokkaa*- ja *Poista*-painikkeet.

Kuva 11. Ylläpidon etusivu

6.2.2 Alakategoriat

Ylläpitäjä voi sisään kirjautuneena lisätä, poistaa tai muokata alakategorioita. Alakategorioita pääsee muokkaamaan sivun ylälaidassa ylläpitotilassa näkyvästä linkistä *Kategoriat*. Kun ylläpitäjä klikkaa *Kategoriat*-linkkiä, aukeaa sivulle pudotusvalikko, josta valitaan pääkategoria. Kun pääkategoria on valittu, tulostuu sivulle näkymä pääkategorian alakategorioista. Valitun alakategorian

nimen oikealla puolella on linkki alakategorian lisäämiseen. Jos pääkategorian alla on alakategorioita, tulostuvat ne näkyviin listana allekkain, kuten tietueetkin. Alakategorioista ylläpitäjälle näkyy tässä vaiheessa nimi ja selite. Jokaisen alakategorian alla on linkit *Muokkaa* ja *Poista*.

Alakategorioita voidaan siis lisätä siirtymällä *Lisää kategoria*-linkistä eteenpäin, sen jälkeen, kun pääkategoria on valittu. Tällöin alakategoria lisätään siihen pääkategoriaan, jonka sisältöä sillä hetkellä tarkkaillaan. Alakategoriaa lisätessä ei tarvitse täyttää kuin kentät alakategorian nimi ja kuvaus. *Kuvaus*-kenttään voi kirjoittaa tarkempaa selitystä, mitä tietoja alakategoria pitää sisällään. Vaikka *Selite*-kenttä ei näy kuin ylläpitäjälle, on sen täyttö pakollista. Kun kentät on täytetty ja alakategoria halutaan lisätä tietokantaan, tulee painaa lisää painiketta. Mikäli lisäys onnistuu, ilmoitetaan siitä käyttäjälle. *Lisää*-painikkeen vieressä on *Peruuta*-painike, josta lisäys voidaan peruuttaa ja siirrytään edelliselle sivulle.

Alakategorioiden muokkaus ei ole sen kummempaa kuin tietueidenkaan. Kun muokattavan alakategorian alla olevasta linkistä on siirrytty muokkaustilaan, tulostaa sovellus alakategorian tiedot käyttäjälle kentissä, joista hän pääsee tietoja muokkaamaan. Muokkaukset tallennetaan *Tallenna*-painikkeesta ja muokkaus voidaan peruuttaa *Peruuta*-linkistä.

Alakategorian poistaminen onnistuu *Muokkaa*-linkin viereisestä *Poista*-linkistä. Alakategorian poistaminen tapahtuu muutoin samalla tavalla kuin tietueenkin. Alakategorian poistamisessa on kuitenkin syytä muistaa se, että jos alakategorian alla on tietueita, on niillä poistamisen jälkeenkin sama alakategoria id. Tämä tarkoittaa sitä, että myös tietueiden alakategorioita on erikseen käytävä muokkaamassa, jotta ne näkyvät sovelluksessa oikein.

Kuvassa 12 on kuvankaappaus alakategorioiden muokkaus-sivusta, jossa pääkategoriaksi on valittu verot.

Kuva 12. Alakategorioiden muokkaus sivu

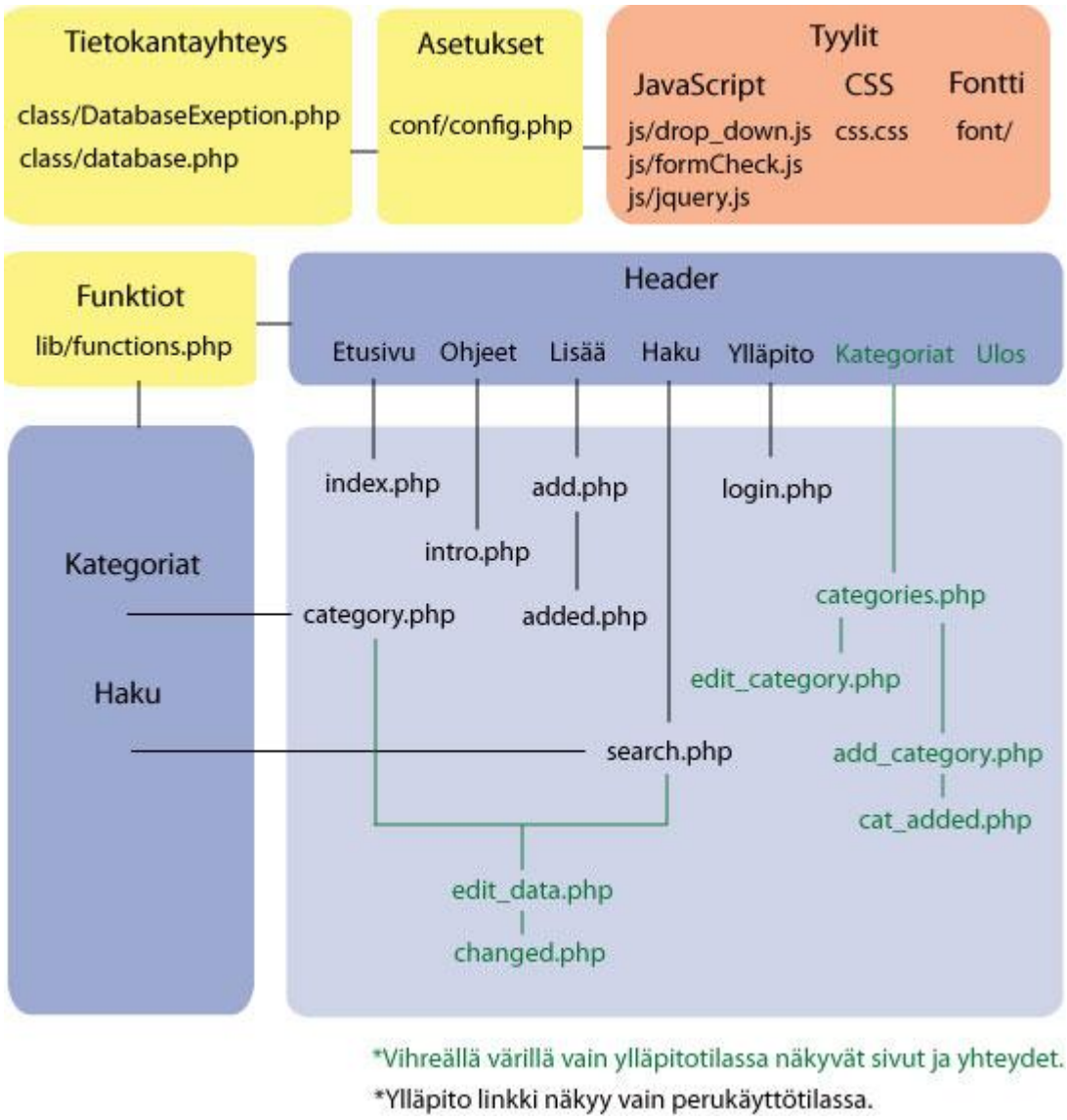
6.2.3 Uloskirjautuminen

Kun ylläpitotoimet halutaan lopettaa, pitää ylläpitäjän muistaa kirjautua ulos. Uloskirjautuminen tapahtuu ylläpitotilassa sivun oikeassa yläreunassa näkyvästä linkistä *Ulos*.

6.3 Tekniikka

Tässä alaluvussa esitellään Tietopankki-sovelluksen tekniikkaa ja kerrotaan hieman siitä, miten sovelluksessa nähdyt toiminnot ovat toteutettu.

Sovellus koostuu monesta eri tiedostosta, jotka ovat yhteydessä toisiinsa. Kuvassa 13 esitellään Tietopankki-sovelluksen rakenne tiedostojen ja niiden yhteyksien avulla.



Kuva 13. Tietopankki-sovelluksen rakenne

6.3.1 PHP:n Funktiot

Tietokoneohjelmissa, kuten verkkosovellus, on useasti toistuvia prosesseja. Näitä prosesseja voidaan ilmentää PHP:ssä ns. nimetyiksi käsitteiksi, joita oh-

jelmoija voi tarvittaessa kutsua, kun ohjelma tätä prosessia vaatii. PHP:n standardipakettiin kuuluu yli 1000 perusfunktiota, joutuu ohjelmoija useasti myös luomaan omia uusia funktioita. Seuraavassa määritellään yksinkertainen funktio, joka tulostaa tekstin ”tulostusteksti” (Gilmore 2005, 93).

```
function tulosta () {
        echo ”tulostusteksti”;
}
```

Koodiesimerkki 1.

Funktioon voi myös välittää arvoja parametrien avulla. Seuraavassa määritellään funktio, joka vastaanottaa parametrit luku1 ja $luku2. Esimerkki funktio laskee sille annetut parametrit yhteen ja tulostaa summan (Gilmore 2005, 94–95).

```
function yhteenlasku ($luku1, $luku2) {
        $summa = $luku1 + $luku2;
        echo $tulostus;
}
```

Koodiesimerkki 2.

Kun funktio on luotu, voidaan sitä kutsua. Seuraava funktio tulostaa arvon 6 (Gilmore 2005, 95).

```
yhteenlasku (1, 5);
```

Koodiesimerkki 3.

Tietopankki-sovelluksessa yleisesti käytetyt funktiot on koottu tiedostoon functions.php, joka sijaitsee alihakemiston lib\ alla. Functions.php sisältää mm. funktiot, joiden avulla sovellukseen tulostetaan sivun yläosan linkit, sekä sivuston vasemmassa reunassa oleva kategoriat valikko. Nämä toiminnot esiin-

tyvät lähes jokaisessa sovelluksen sivussa. Jos esim. kategoriat valikkoa pitää muuttaa, täytyy muutokset tehdä vain tähän kyseiseen funktioon, joka sijaitsee functions.php:ssa. Jos näin ei olisi, joutuisi muutokset tekemään jokaiselle sivulle, jossa kyseinen valikko esiintyy.

Tietopankkisovelluksessa jotkin funktiot sijaitsevat siis ainoastaan yhdessä tiedostossa. PHP mahdollistaa eri tiedostojen sisällyttämisen toiseen tiedostoon omalla komennollaan, joka mahdollistaa esim. funktioiden käytön toisesta tiedostosta. PHP tarjoaa neljä eri tiedoston sisällyttämislausetta, joista tietopankkisovelluksessa on käytetty *require_once()*-funktiota. *Require_once()*-funktion avulla varmistetaan, että lisäystiedosto lisätään skriptissä vain kerran, toisin kuin esimerkiksi *require()*-funktiota käytettäessä. Koodiesimerkissä 4 sisällytetään tiedostoon functions.php (Gilmore 2005, 89–92).

```
require_once("lib/functions.php");
```

Koodiesimerkki 4.

Aina kun jokin tietopankkisovelluksen tiedosto käyttää jotain funktiota functions.php:stä, pitää tiedoston yläreunassa olla yllä oleva sisällyttämislause.

6.3.2 Tietokantayhteys

Vaikka PHP-skriptikieli ja MySQL-tietokantapalvelin ovat kaksi erillistä teknologiaa, ovat molemmat kuitenkin tiiviisti yhteydessä toisiinsa. PHP:n MySQL-rajapinta tarjoaa yli 45 valmista funktiota, joiden avulla pystyy suorittamaan monenlaisia tehtäviä. PHP-kielen avulla on mahdollista mm. lisätä, noutaa, muokata tai poistaa MySQL-dataa. Suurin osa Tietopankki-sovelluksen toiminnoista perustuu yhteistyöhön tietokannan kanssa. (Gilmore 2005, 617–618).

Ennen kuin tietokannan kanssa voi tehdä mitään, on sinne avattava yhteys. Tietokanta yhteyden luomiseen tarjotaan useampaa funktiota, mutta Tieto-

pankki-sovelluksessa on käytetty funktiota *mysql_connect( )* (Gilmore 2005, 618). Koodiesimerkissä 5 luodaan tietopankkisovelluksen tietokantayhteys.

```
mysql_connect ($this->server, $this->login, $this->password);
```

Koodiesimerkki 5

Mysql_connect()-funktioon välitetään tyypillisesti kolme parametria; palvelimen nimi, käyttäjätunnus ja salasana. Käyttäjätunnuksen ja salasanan tulee olla sama kuin mitkä ovat määritelty MySQL-palvelimelle. Tietopankkisovelluksessa on nämä vaadittavat parametrit määritelty eri tiedostoon, josta sovellus hakee erikseen. Tämän takia ne eivät suoraan näy esimerkissä. Kun tietokantapalvelimeen on muodostettu yhteys onnistuneesti, valitaan vielä tietokanta, jota halutaan käyttää (Gilmore 2005, 618–621). Koodiesimerkissä 6 valitaan tietopankkisovelluksen tietokanta.

```
mysql_select_db($this->dbname, $conn)
```

Koodiesimerkki 6. Tietokannan valitseminen

Tietokannanvalintafunktioon välitetään yksi pakollinen parametri, joka on tietokannan nimi. Toinen parametri pitää välittää, jos yhteyksiä on auki useita. Tietopankkisovelluksessa tietokannan nimi on määritelty samassa tiedostossa kuin parametrit yhteyden luomiseen. Sovelluksessa on määritelty myös toinenkin parametri. Muuttujaksi *$conn* on määritelty funktio tietokantayhteyden luomisesta, joka käy ilmi koodiesimerkistä 2 (Gilmore 2005, 621).

Tietopankki-sovelluksessa tietokantayhteyden luominen on tehty omaan tiedostoonsa database.php, joka sijaitsee alihakemistossa class. Tietokantayhteyden luomiseen ja tietokannan valitsemiseen tarkoitetut parametrit on tallennettu conf/config.php-tiedostoon.

6.3.3 Kyselyn suorittaminen MySQL-palvelimeen, sekä datan käsitteleminen

Kun yhteys tietokantaan on luotu, voidaan sinne suorittaa kyselyitä. Kysely suoritetaan funktiolla *mysql_query(),* johon parametriksi annetaan itse kysely (Gilmore 2005, 622).

Kun tietokantakysely on suoritettu, tulee kyselystä palautunut tieto vielä noutaa. Tietopankkisovelluksessa on käytetty suurimmaksi osaksi *mysql_fetch_array()*-funktiota. Funktio noutaa kokonaisen datarivin ja sijoittaa arvot taulukkoon. Taulukosta voi tämän jälkeen helposti poimia halutut arvot ja esimerkiksi tulostaa ne näkyviin (Gilmore 2005, 623–625).

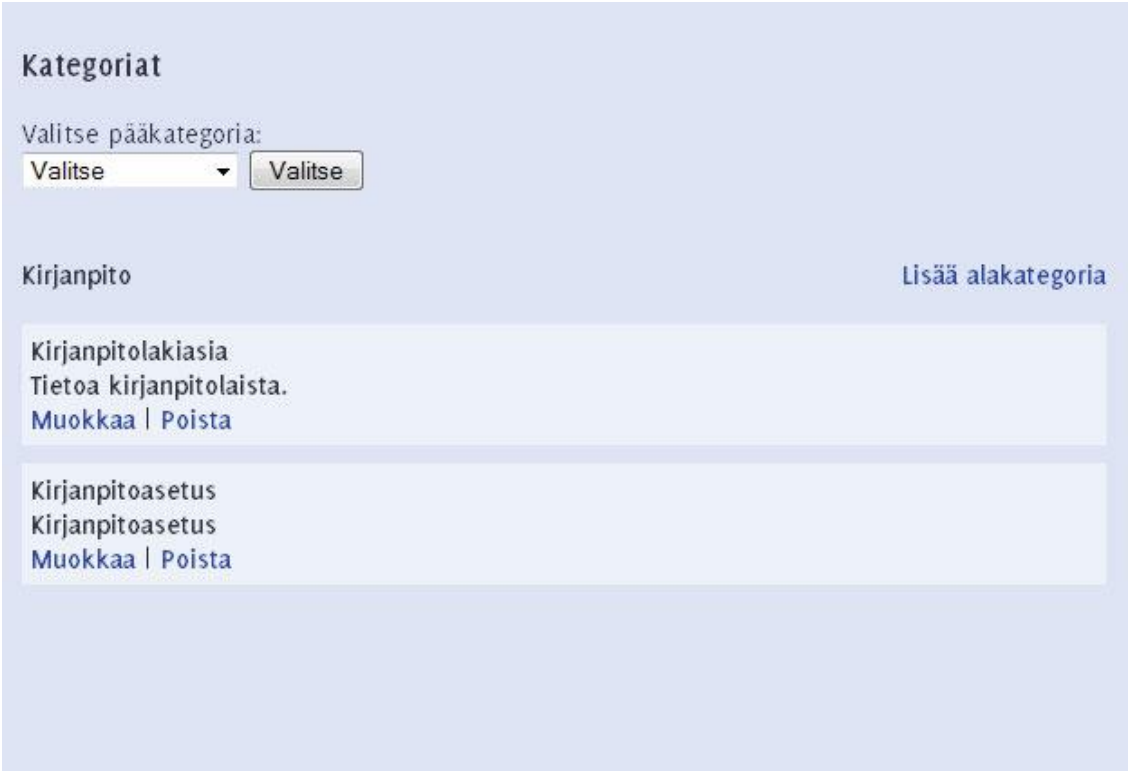
Koodiesimerkissä 7 esitellään tietopankkisovelluksen funktio, joka hakee ja tulostaa alakategorian tiedot. Funktion parametriksi *$result* määritellään SQL-kyselylauseke. Funktio tulostaa valitun kategorian kaikki alakategoriat tietoineen omille riveille, sekä jokaisen löytyneen alakategorian alle Muokkaa- ja Poista painikkeet. Tietopankki-sovelluksessa funktiota käytetään ylläpidon puolella, jossa ylläpitäjällä on mahdollisuus muokata tai poistaa alakategorioita.

```
function printCategoryInfo($result)
{
        $db = new Database();
        $db -> connect();

        $query = $db -> execute($result);

        while($row = mysql_fetch_array($query, MYSQL_ASSOC))
        {

        print '<div id ="data">';
        print '<b>'.$row["sub_cat_name"]. '</b> ';
         print '<b><br />'.$row["sub_cat_info"]. '</b><br /> ';
        print '<a class ="edit_data" href="categories.php?action=1&sub_cat_id=' .
        $row["sub_cat_id"] . '">Muokkaa</a> | ';
```

```
        print '<a class ="delete_data" href="categories.php?action=2&sub_cat_id=' .
        $row["sub_cat_id"] . '">Poista</a></p>';

         print '</div>';
        }
}
```

Koodiesimerkki 7: *printCategoryInfo($result)*-funktio

Kuvassa 14 näkyy *printCategoryInfo($result)*-funktion käyttäjälle näkyvä tuloste, kun hän on valinnut pudotusvalikosta pääkategoriaksi kirjanpidon.



Kuva 14: Funktion printCategoryInfo($result) tulostama näkymä, kun pääkategoriaksi on valittu kirjanpito.

6.4 Sovelluksen testaus

Sovellusta ei ole testattu käyttäen virallisia ohjelmistotestauksen keinoja. Sovelluksen toimintoja testattiin ensiksi niin, että pyrittiin käyttämään toimintoa niin, kuten sitä kuuluisikin käyttää. Jos toiminnoista löytyi virheitä, ne korjattiin. Tätä jatkettiin kunnes toiminnon avulla pystyttiin suorittamaan se tehtävä, mikä sen avulla oli tarkoitus pystyä suorittamaan. Tämän jälkeen toimintoja testattiin käyttämällä niitä ohjeiden vastaisesti. Kaikki tätä kautta havaitut virheet korjattiin. Sovelluksesta pyrittiin tekemään hyvin selkeä, ettei käyttäjällä olisi mahdollisuutta käyttää sitä väärin. Tarkoilla käyttöohjeilla pyritään myös tukemaan tätä.

6.5 Sovelluksen käyttö ja testaus mahdollisuus

Loppusyksystä 2010 Näsitilit Oy yhdistyi suurempaan tilitoimistoon ja tällä tilitoimistolla on käytössä hyvin laaja intranet, joka sisältää ainakin osin Tietopankki-sovelluksen toimintoja. Toimeksiantajani oli kuitenkin sitä mieltä, että teen sovelluksen loppuun niin kuin olen sen alun perin suunnitellut. Kun yritysten yhdistyminen on viety loppuun, pystytään vasta katsomaan, tuoko Tietopankki-sovellus merkittävää hyötyä yritykselle sellaisenaan. On myös mahdollista että tulevaisuudessa Tietopankki-sovelluksen joitain osia integroida jo olemassa olevaan intranetiin.

7 YHTEENVETO

Tässä luvussa kerrotaan, miten sovellusta voisi tulevaisuudessa kehittää, sekä tehdään yhteenveto sovelluksen tekoprosessista ja tuloksista.

7.1 Kehitettävää

Koska Tietopankki-sovellus on toiminnaltaan kuin intranet, on siihen mahdollista luoda paljon muitakin ominaisuuksia. Sovellusta luodessa oli myös tarkoitus rakentaa se niin, että siihen pystyy myöhemmin lisäämään toimintoja. Toimintojen lisääminen sovellukseen kävisi periaatteessa niin, että jokaista toimintoa varten luotaisiin yläpalkkiin uusi linkki, josta toimintoon pääsisi käsiksi. Mutta koska työni tavoite oli luoda tietopankki, keskityn tässä luvussa vain niihin ominaisuuksiin, jotka voisivat vielä kehittää itse tietopankkia. Jos tietopankkia joskus vielä kehitetään, tulee minulle mieleen neljä seikkaa, joita voisi ensimmäisenä parantaa. Seuraavassa on esiteltynä nämä neljä ominaisuutta.

Alunperin oli tarkoitus luoda sovellukseen mahdollisuus kommentoida lisättyjä tietueita. Jokaisen tietueen alla oli tarkoitus näkyä *Kommentoi*- ja *Näytä kommentit*- linkit. Kommenttien avulla olisivat käyttäjät voineet myöhemmin asettaa tietueille esimerkiksi kysymyksiä tai lisätä kommentin kautta niihin tarpeellisia tietoja niin, ettei koko tietuetta tarvitsisi muokata. Kommenttien avulla olisi käyttäjien ollut myös helppo seurata tietueiden muutosta ja kehitystä. Esimerkiksi, jos jonkin tietueen sisältö muuttuu laissa, voisi ylläpitäjä muokata tietuetta ja lisätä siihen kommentin, että tietuetta on muokattu. Tämä kiinnittäisi helpommin käyttäjän huomion, ja näin hän varmistuisi, että vuosi sitten lisätty tietue on ajan tasalla. Kommentointi jäi kuitenkin sovelluksesta pois silkan ajanpuutteen vuoksi.

Toinen toiminto, jonka aluksi suunnittelin sovellukseen, oli mahdollisuus lisätä liitteitä tai kuvia tietueisiin. Sovellusta rakentaessani päätin jättää liitteet tietu-

eista pois, koska tarkoitus oli rakentaa sovelluksesta selkeä ja yksinkertainen käyttää. Mielestäni tietueiden lisääminen olisi ollut liian monimutkaista ja vaivalloista, jos siinä olisi ollut vielä mahdollisuus lisätä liitteitä. Päätin, että liitteet voisi jättää omaksi toiminnokseen sovelluksessa, jos sovellusta joskus vielä kehitetään. Sovelluksessa olisi ollut siis oma dokumenttipankki, johon olisi voinut tallentaa toimistossa yleisesti käytettäviä dokumentteja. Hakutoiminnossa voisi sitten olla oma rajaus, että haetaan pelkästään dokumenteista.

Sovelluksen ylläpitoa helpottaisi, jos ylläpitäjällä olisi mahdollisuus lisätä, muokata tai poistaa ylläpito-oikeuksia. Tällä hetkellä ylläpitotunnuksia ei ole kuin yhdet ja uusien tunnuksien lisääminen edellyttää niiden syöttämistä tietokantatauluun jonkin tietokannan hallintatyökalun avulla. Tämän lisäksi sovelluksessa voisi olla mahdollisuus siihen, että sen käyttö edellyttäisi sisään kirjautumista ihan alusta asti. Tämä on ensisijaisen tärkeää, jos sovellus joskus tulisi olemaan verkkopalvelimella. Koska alun perin suunniteltiin, että sovellus sijoitettaisiin toimiston omalle palvelimelle, jätin tämän toiminnon pois jo suunnitteluvaiheessa.

Sovelluksessa on myös muutama toiminnallinen puute tai heikkous. Sovelluksen avulla voi tietokantaan lisätä samannimisiä tietueita ja alakategorioita. Lisäksi, jos on juuri lisännyt tietueen ja klikkailee selaimen *taaksepäin*- ja *eteenpäin*-painikkeita vuorotellen, lisää sovellus saman tietueen aina uudestaan ja uudestaan.

7.2 Yhteenveto

Laajan WWW-sovelluksen rakentamista yksin alusta alkaen pidin itselleni suurena, mutta mielenkiintoisena haasteena. Halusin kuitenkin toteuttaa sovelluksen nimenomaan yksin, sillä näin koin saavani parhaan mahdollisen hyödyn itselleni ja oppisin verkkosovelluksen ohjelmointia parhaiten.

Kokonaisuudessa onnistuin tavoitteissani mielestäni kuitenkin hyvin. Sovelluksessa on kaikki tarpeelliset toiminnot ja se on yksinkertainen käyttää. Ennen kaikkea sovellus on toimiva tilitoimistoympäristössä johon se on tarkoitettu. Nopealla testauksella vastaan ei ole tullut toiminta virheitä. Sovelluksesta tuli lähestulkoon sen näköinen ja oloinen kuin, mitä ennen työn aloittamista ajattelin. Koska projektissa oli haastetta itselleni hyvin ja onnistuin mielestäni lopputuloksessa, olen tulokseen erittäin tyytyväinen.

Henkilökohtaisena tavoitteena minulla oli oppia käyttämään verkkosovelluksen rakentamiseen käytettäviä tekniikoita, kuten PHP:tä MySQL:ää, syventävästi. Siinä tavoitteenani oli tutustua minulle uuteen JavaScript-kieleen. Nyt kun olen laajan sovelluksen luonut yksin, koen että olen näissä tekniikoissa kehittynyt paljon, verrattuna lähtötilanteeseen. Myös projektinhallintataitoni ovat varmasti karttuneet. Koen nyt olevani valmis myös tulevaisuudessa hoitamaan esimerkiksi työkseni vastaavanlaisia projekteja.

Vaikka sovellus ei vielä keväällä 2011 ollut oikeassa käytössä, takaavat sen rakentamisesta karttuneet kokemukset ja lopputulos sen, että olen työhöni ja sen tulokseen tyytyväinen.

LÄHDELUETTELO

Anderson, Andy & King, Konrad. 2002. HTML & Web Design Tips & Techniques. McGraw-Hill Companies, The.
Saatavilla myös:
http://site.ebrary.com.elib.tamk.fi/lib/tamperepoly/docDetail.action?docID=10043852&p00=html%20basics

Duckett, Jon. 2005. Accessible XHTML and CSS Web Sites : Problem - Design – Solution. John Wiley & Sons, Incorporated.
Saatavilla myös:
http://site.ebrary.com.elib.tamk.fi/lib/tamperepoly/docDetail.action?docID=10114213&p00=html%20xhtml

Gilmore, W Jason. 2005. PHP & MySQL – Tehokas hallinta. Helsinki: Readme.fi.

Heinisuo, Rami & Rauta, Ilkka. 2007. PHP & MySQL Tietokantapohjaiset verkkopalvelut. Valikko-sarja. Helsinki: Talentum Media Oy.

Kolehmainen, Kauko. 2006. PHP & MySQL Teoriasta käytäntöön. Jyväskylä. Helsinki: Readme.fi.

MySQL 2010. Why MySQL?, [online] [tulostettu 17.9.2010] Saatavissa: http://www.mysql.com/why-mysql/

Nielsen, Jacob. 2000. WWW-suunnittelu. Oy Edita Ab.

Oppel, Andrew J. 2004. Databases Demystified. McGraw-Hill Professional Publishing.

Peltomäki, Juha. JavaScript e-kirja Toolkit. Jyväskylä: Docendo Finland Oy.

Vawani, Vikram 2005. How to Do Everything with PHP and MySQL. McGraw-Hill Professional Publishing.

W3schools HTML 2010a. HTML Introduction
[online] [tulostettu 17.9.2010]
Saatavissa: http://www.w3schools.com/html/html_intro.asp

W3schools 2010b. XHTML – Why?.
[online] [tulostettu 17.9.2010]
Saatavissa: http://www.w3schools.com/xhtml/xhtml_why.asp

W3schools 2010c. XHTML Validation
[online] [tulostettu 17.9.2010]
Saatavissa: http://www.w3schools.com/xhtml/xhtml_validate.asp

W3schools 2010d. CSS Introduction
[online] [tulostettu 17.9.2010]
Saatavissa: http://www.w3schools.com/css/css_intro.asp

W3schools 2010e. JavaScript Introduction
[online] [tulostettu 17.9.2010]
Saatavissa: http://www.w3schools.com/js/js_intro.asp

LIITTEET

Liitteeltä 1 löytyy komennot, joilla on luotu Tietopankki-sovelluksen tietokanta ja taulut. Liitteestä löytyy myös komennot, joilla lisätään categories-taulun sisältö, sekä ensimmäisen ylläpitäjän tiedot, kuten käyttäjätunnus ja salasana.

```
-- phpMyAdmin SQL Dump
-- version 3.2.0.1
-- http://www.phpmyadmin.net
--
-- Palvelin: localhost
-- Luontiaika: 16.04.2011 klo 13:09
-- Palvelimen versio: 5.1.36
-- PHP:n versio: 5.3.0

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";

--
-- Tietokanta: `tietopankki`
--

-- --------------------------------------------------------

--
-- Rakenne taululle `categories`
--

CREATE TABLE IF NOT EXISTS `categories` (
  `cat_id` int(7) NOT NULL AUTO_INCREMENT,
  `cat_name` varchar(50) NOT NULL,
  `cat_info` varchar(500) NOT NULL,
  PRIMARY KEY (`cat_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

--
-- Vedos taulusta `categories`
--

INSERT INTO `categories` (`cat_id`, `cat_name`, `cat_info`) VALUES
(1, 'Verot', 'Kaikki veroihin liittyvä.'),
(2, 'Kirjanpito', 'Kaikki kirjanpidosta, kuten lait ja säädökset.'),
(3, 'Palkanlaskenta', 'Kaikki palkanlaskennasta, kuten työehtosopimuksiin liittyvät asiat.'),
(4, 'Tilintarkastus', 'Kaikki tilintarkastukseen liittyvä, kuten lait ja käytännöt.'),
```

```
(5, 'Uutiset', 'Talon sisäiset asiat'),
(6, 'Muut', 'Jos tulee jotain muuta niin tänne.');

-- --------------------------------------------------------

--
-- Rakenne taululle `data`
--

CREATE TABLE IF NOT EXISTS `data` (
  `data_id` int(7) NOT NULL AUTO_INCREMENT,
  `cat_id` int(7) NOT NULL,
  `sub_cat_id` int(7) NOT NULL,
  `data_name` varchar(100) NOT NULL,
  `data_intro` varchar(255) NOT NULL,
  `data_text` text NOT NULL,
  `data_add_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `data_lastedit_date` date NOT NULL,
  `data_lastedit_name` varchar(40) NOT NULL,
  `data_add_name` varchar(50) NOT NULL,
  `data_add_mail` varchar(70) NOT NULL,
  PRIMARY KEY (`data_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=39 ;


--
-- Rakenne taululle `sub_categories`
--

CREATE TABLE IF NOT EXISTS `sub_categories` (
  `sub_cat_id` int(7) NOT NULL AUTO_INCREMENT,
  `cat_id` int(7) NOT NULL,
  `sub_cat_name` varchar(50) NOT NULL,
  `sub_cat_info` varchar(500) NOT NULL,
  PRIMARY KEY (`sub_cat_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=24 ;

--
-- Vedos taulusta `sub_categories`
--

INSERT INTO `sub_categories` (`sub_cat_id`, `cat_id`, `sub_cat_name`,
`sub_cat_info`) VALUES
(11, 5, 'Tampereen Tilitoimisto', 'TT:n uutiset.'),
(4, 1, 'ArvonlisÃ¤vero', 'Kaikki arvonlisÃ¤verosta.'),
(12, 5, 'NÃ¤sitilit', 'NÃ¤sitilien uutiset.'),
(13, 4, 'Tilintarkastuslaki', 'Tilintarkastuslaki'),
(14, 2, 'Kirjanpitolakiasia', 'Tietoa kirjanpitolaista.'),
(15, 2, 'Kirjanpitoasetus', 'Kirjanpitoasetus'),
(22, 1, 'HenkilÃ¶verot', 'henkilÃ¶t');
```

```
-- --------------------------------------------------------

--
-- Rakenne taululle `user`
--

CREATE TABLE IF NOT EXISTS `user` (
  `user_id` int(7) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(50) NOT NULL,
  `user_pass` varchar(50) NOT NULL,
  `user_realname` varchar(50) NOT NULL,
  `user_mail` varchar(50) NOT NULL,
  `user_phone` varchar(20) NOT NULL,
  PRIMARY KEY (`user_id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

--
-- Vedos taulusta `user`
--

INSERT INTO `user` (`user_id`, `user_name`, `user_pass`, `user_realname`, `user_mail`, `user_phone`) VALUES
(2, 'seppo', 'hovi', 'Pauli Rauhanen', 'pauli.rauhanen@cs.tamk.fi', '050 3461909');
```