OULUN SEUDUN
AMMATTIKORKEAKOULU

Antti Karjaluoto

KALAPAIKKASIVUSTON TOTEUTUS

KALAPAIKKASIVUSTON TOTEUTUS

Antti Karjaluoto
Opinnäytetyö
Kevät 2012
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistojen kehitys

Tekijä(t): Antti Karjaluoto
Opinnäytetyön nimi: Kalapaikkasivuston toteutus
Työn ohjaaja(t): Veijo Väisänen
Työn valmistumislukukausi ja -vuosi: kevät 2012 Sivumäärä: 36 + 5 liitettä

Tämän opinnäytetyön tavoitteena oli luoda nettiselaimella käytettävä kalapaikkatietokanta. Kalapaikkojen selaamisen ja lisäämisen tuli tapahtua helposti, mutta sen tulee vaatia rekisteröityminen kalapaikkasivustolle. Kalapaikasta tulee saada myös helposti luettava raportti. Sivuston tulee lähtökohtaisesti soveltua myös mobiilikäyttöön.

Työ aloitettiin tietokantasuunnittelulla, jonka pohjalta toteutettiin sovelluksen käyttämä MySQL-tietokanta. Sovelluksen toiminnallisuus toteutettiin PHP-ohjelmointikielellä, ja käyttöliittymänä toimii PHP:llä toteutettu dynaaminen HTML-sivusto.

Työn tuloksena valmistui kalapaikkasovellus, joka täyttää vaaditut perustoiminnallisuudet ja toimii pohjana jatkokehitystä ajatellen. Sovelluksen toteutuksessa kiinnitettiin huomiota sen laajennettavuuteen.

Asiasanat:
PHP, HTML, MySQL, tietokannat

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Information Technology, Software Developement

---

Author(s): Antti Karjaluoto
Title of thesis: Kalapaikkasivuston toteutus
Supervisor(s): Veijo Väisänen
Term and year when the thesis was submitted: Spring 2012 Pages: 36 + 5 appendices

---

The purpose of the thesis was to create a fishing-place database that was usable with internet browsers. The adding and viewing of the fishing-places was supposed to happen easily, but it demanded subscribing to the site. It was also necessary to get easily readable reports from the fishing places. The website should be practically usable with mobile devices.

The thesis was started with database-planning from within the MySQL database used by the solution was produced. The functionality of the solution was made with the PHP-programming language, and as an interface worked a dynamic HTML-website produced with PHP.

The result of the thesis was a fishing-place-solution, that has all the demanded functionality and which works as a base start for further development in the future. Expandability was taken care of while making the solution.

Keywords:
PHP, HTML, MySQL, database

SISÄLLYS

LYHENTEET

CSS Cascading Style Sheets on erityisesti www-dokumenteille kehitetty tyyliohjeiden laji. CSS:ssä dokumentille voi määritellä useita tyyliohjeita, jotka yhdistetään tietyllä tavalla yhdeksi säännöstöksi.

HTML Hypertext Markup Language on nettisivujen toteuttamiseen käytettävä kuvauskieli, jolla määritetään sivujen rakenteet.

HTTP Hypertext Transfer Protocol on protokolla, jota selaimet ja www-palvelimet käyttävät tiedonsiirtoon.

LAMP Linux, Apache, MySQL, PHP. Kehitysympäristö, jossa on yhdessä paketissa web-, SQL- ja PHP-palvelin Linux-käyttöjärjestelmälle.

MySQL Ilmainen relaatiotietokantamoottori useille eri käyttöjärjestelmille.

PHP PHP: Hypertext Preprocessor on ohjelmointikieli, jota käytetään erityisesti web-palvelinympäristöissä dynaamisten web-sivujen luonnissa.

SQL Structured Query Language on IBM:n kehittämä kyselykieli, jolla voi ohjata relaatiotietokantoja.

.

1 JOHDANTO

Tämän työn tavoitteena oli luoda kalapaikkatietokanta ja sen käyttämiseen helppokäyttöinen käyttöliittymä, joka tarjoaa käyttäjille sekä helpon tiedon hakemisen että lisäämisen ja muokkaamisen. Työn tilaajana toimi Pohjois-Pohjanmaan Vapaa-ajankalastajapiiri. Tietoa on tarjolla varsin kattavasti ja kalapaikkoihin voi lisätä esimerkiksi niin kalastuslupien tarpeellisuuden kuin myös vene- ja laavupaikkojen sijainnit.

Kalapaikkasivuston tarkoituksena on helpottaa kalastajia ja muita löytämään enemmän ja tarkempaa tietoa eri kalapaikoista. Sivustolle voi rekisteröityä kuka tahansa käyttäjä ja lisätä omat salaisimmatkin kalapaikkansa, joista ei välttämättä muualla internetissä ole paljon tietoa. Kalapaikkasivustoa pystyy käyttämään sekä tietokoneiden että mobiililaitteiden selaimilla.

Työn lopputuloksena tilaaja sai toimivan kalapaikkatietokannan ja sen päällä käyttöliittymänä toimivan nettisivuston. Sivusto tulee toimimaan käyttäjien avulla, jotka hoitavat tiedon lisäämisen ja muokkaamisen tietokantaan. Tulevaisuudessa palvelua on tarkoitus kehittää toimimaan suoraan jollakin mobiilialustalla.

2 KÄYTETYT TEKNOLOGIAT

Ohjelmointikieleksi kalapaikkasovelluksen käyttöliittymälle valittiin PHP, joka on tunnettu ja ilmainen web-ohjelmointikieli. Tietokannan toteutukseen valittiin MySQL-tietokantamoottori. MySQL on myös ilmainen tietokantamoottori, joka toimii hyvin erilaisissa käyttöjärjestelmissä. PHP ja MySQL ovat erittäin suosittuja nettisovelluksien kehittämiseen käytettyjä työkaluja.

2.1 PHP-ohjelmointikieli

PHP on serveripuolen skriptauskieli, joka on suunniteltu erityisesti web-kehitystä varten. HTML-sivun sisään voidaan lisätä PHP-koodia, joka suoritetaan joka kerta kun sivulla käydään. PHP-koodi tulkataan web-palvelimella ja se generoi HTML:ää tai jotain muuta tulostettavaa, jonka käyttäjä näkee. PHP on suunniteltu vuonna 1994 ja on alun perin yhden miehen työtä, Rasmus Lerdorfin. Myös moni muu ihminen on kehittänyt PHP:tä ja se on käynyt läpi neljä isoa uudelleenkirjoitusta, ennen kuin se saatiin siihen muotoon, mitä PHP on nykyään. (1.)

PHP on avoimen lähdekoodin projekti, joka tarkoittaa sitä, että kaikilla on pääsy lähdekoodiin ja sitä voidaan käyttää, muuttaa tai jakaa ilman maksua. Nimi PHP oli alun perin Personal Home Page, mutta nimi on sittemmin vaihdettu (PHP Hypertext Preprocessor). Nykyinen pääversio on PHP 5. Tähän versioon on toteutettu Zend-moottorin täydellinen uudelleenkirjoitus, joka tukee olio-ohjelmointia ja sisältää sisäänrakennetun tietokantamoottorin (SQLiten) ja eräitä muita suuria parannuksia kieleen. (1.)

2.2 HTML-merkintäkieli

HTML on www-sivuilla käytettävä merkintäkieli, jolla määritetään suurimmaksi osaksi www-sivujen rakenteet, jotka selainohjelma sitten näyttää käyttäjälle. Elementit ovat HTML-kielen perusrakenne ja niillä voidaan kuvata esimerkiksi www-sivuilla taulukoita, linkkejä tai muita HTML-elementtejä. HTML-

elementtien perusominaisuudet ovat attribuutit (ominaisuudet) ja sisältö. HTML-sivun aloitus- ja lopetuselementti voi olla lyhkäisyydessään esimerkiksi <html> </html>, ja merkkien väliin tulevat sisältö ja muut elementit. HTML:ää ei pidetä varsinaisena ohjelmointikielenä. (2.)

2.3 MySQL relaatiotietokantaohjelmisto

MySQL on erittäin nopea ja luotettava relaatiotietokantojen ohjausjärjestelmä. Se mahdollistaa tietojen tehokkaan tallentamisen, etsimisen, järjestämisen ja palauttamisen. MySQL-serveri ohjaa pääsyä tietoihin ja varmistaa, että käyttäjät voivat työskennellä sen kanssa samanaikaisesti, tarjoaa nopean sisäänpääsyn ja varmistaa, että vain valtuutetuilla on oikeudet käyttää sitä. Tästä johtuen MySQL on monen käyttäjän monisäikeinen palvelin. Se käyttää Structured Query Languagea (SQL), standardoitua kyselykieltä. MySQL on ollut avoimesti tarjolla vuodesta 1996, mutta sen kehittämishistoria ulottuu ainakin vuoteen 1979. Se on yksi maailman suosituimpia avoimen lähdekoodin tietokantoja, joka on voittanut ”Linux Journal Readers Choice”-palkinnon useita kertoja. (3.)

MySQL on saatavilla kahdella erilaisella lupajärjestelmällä. Sitä voi käyttää avoimen lähdekoodin lisenssillä ilmaiseksi, niin kauan kuin noudattaa sen lisenssin ehtoja. MySQL:stä voi myös halutessaan ostaa kaupallisen lisenssin version. (3.)

2.4 SQL-kyselykieli

Structured Query Language (SQL) on IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä. Käytännössä kaikki relaatiotietokannat ymmärtävät SQL-kieltä. Tärkeimmät käyttökäskyt ovat SELECT, UPDATE, INSERT ja DELETE. Tärkeimmät tiedon määrittelykäskyt ovat CREATE TABLE ja CREATE VIEW, joilla luodaan uusia tietokantaobjekteja. (4.)

2.5 LAMP

LAMP on Linuxille tehty kokoelma avoimen lähdekoodin ohjelmia, jotka yhdessä muodostavat www-palvelimen, jonka alla voidaan suorittaa dynaamisia websivuja. LAMP:iin sisältyy Apache-web-palvelin, MySQL-tietokantapalvelin ja PHP-palvelin. LAMP on helppo ja nopea asentaa sekä sen asetusten muokkaaminen onnistuu helposti myöhemminkin asennuspaketin avulla. Asetukset voi määrittää myös asennusvaiheessa. Ohjelmistoa voi asentamisen jälkeen käyttää erilaisilla graafisilla käyttöliittymillä, joilla onnistuu myös PHP-moduulien käyttöönotto ja poistaminen. (5.)

2.6 CSS-tyylikieli

CSS on erityisesti www-dokumenteille kehitetty tyyliohjeiden laji. Siinä dokumenteille voi määrittää useita erilaisia tyyliohjeita, joita yhdistetään yhdeksi säännöstöksi. Säännöt eivät ole ehdottomia, vaan ne voi halutessaan kiertää ja jotkin säännöt saattavat korvata toisten tyyliohjeiden vastaavat säännöt. (6.)

CSS:llä voidaan kuvata monipuolisesti sekä nähtävää että kuultavaa esitystapaa. Äänisyntetisaattoreita varten on määritelty muun muassa äänen korkeutta, painotusta ja äänenväriä säädeleviä ominaisuuksia. Visuaalisen esitystavan perusta on ns. laatikkomalli (box-model). Jokainen dokumentin elementti käsitetään mallissa laatikoksi, joka sijoitetaan ympäröivän elementin laatikon sisään muiden saman tason elementtien vierelle. Kullakin elementillä on neljä sisäkkäistä laatikkoa, joista uloin on marginaali; sen sisällä on reunus, joka rajoittuu sisäpuolelta sisältöä ympäröivään täytteeseen. Sisältölaatikossa on kuvattavan elementin sisältö. Sen perussyntaksi muodostuu valitsimesta, ominaisuudesta ja arvosta. Esimerkiksi: Valitsin {Ominaisuus: Arvo; }. (6.)

2.7 Apache palvelinohjelma

Apache on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma. Se on saatavilla Unixille, Windowsille ja useille muille käyttöjärjestelmille. Pelkkä Apache tukee ainoastaan staattisten tiedostojen jakamista HTTP-protokollan yli,

mutta Apachen ydintä voidaan täydentää useilla moduuleilla, jotka mahdollistavat palvelimen räätälöinnin omia tarpeita vastaavaksi. Osa moduuleista tulee ohjelmiston mukana kuten esimerkiksi mod_cgi, jolla voidaan ajaa ulkoisia ohjelmia CGI-ympäristössä. Ulkopuolisia moduuleita ovat muun muassa mod_perl, joka integroi Perl-tulkin palvelimeen ja mod_log_mysql, joka tallentaa lokit MySQL-tietokantaan. (7.)

2.8 Tietokannat

Tietokantojen suunnittelussa tärkeitä asioita ovat tietokannan ylläpidettävyys, rakenne ja tehokkuus. Muita hyvän tietokannan ominaisuuksia ovat ainakin tiedon eheys, tietokantarakenteiden helppo muokkaaminen ja ylläpito, tietojen helppo hakeminen ja muokkaaminen. Hyvä tietokanta tukee tietojen hakemista ja erikoishakuja, ja tietokanta on alustavasti toteutettu käytettäväksi myös tulevaisuudessa. (8.)

2.8.1 Tietokantayhteydet

Kahden tietokantataulun välille luotu liitos tunnetaan yhteytenä. Yhteys on olemassa, kun kaksi taulua on yhdistetty pääavaimen avulla tai linkitetty yhteen kolmannen taulun avulla, josta käytetään nimitystä linkitystaulu. Yhteydet ovat datan eheyden kannalta tärkeitä, koska ne auttavat vähentämään ylimääräistä ja kaksinkertaista dataa. Taulujen välisiä yhteystyyppejä on kolme: yhden suhde moneen -yhteydellä päästään toteuttamaan tietokantojen normalisointia eli tietojen toistamisen minimointia. Kaksi harvinaisempaa ja vähemmän käytettyä yhteystyyppiä ovat yhden suhde yhteen -yhteys ja monen suhde moneen -yhteys. Kahden viimeksi mainitun yhteystyypin käyttöä pyritään välttämään tietokantojen suunnittelussa. (8.)

Kahden taulun välillä on olemassa yhden suhde moneen -yhteys, jos yksittäinen ensimmäisen taulun tietue voi liittyä yhteen tai useampaan toisen taulun tietueeseen, mutta toisen taulun yksittäinen tietue voi liittyä vain yhteen ensimmäisen taulun tietueeseen. Kahden taulun välillä on olemassa yhden suhde

yhteen -yhteys, jos ensimmäisen taulun yksittäinen tietue liittyy vain yhteen toisen taulun tietueeseen ja toisen taulun yksittäinen tietue liittyy vain yhteen ensimmäisen taulun tietueeseen. Kahden taulun välillä on monen suhde moneen -yhteys, jos ensimmäisen taulun yksittäinen tietue voi liittyä yhteen tai useampaan toisen taulun tietueeseen ja toisen taulun yksittäinen tietue voi liittyä yhteen tai useampaan ensimmäisen taulun tietueeseen. (8.)

2.8.2 Indeksit

Indeksejä käytetään löytämään nopeasti rivejä, joilla on tietyt etukäteen määritellyt arvot. Ilman indeksejä MySQL:n täytyy aloittaa etsiminen ensimmäiseltä riviltä ja sitten tutkia koko taulu läpi löytääkseen oikeat rivit. Mitä laajempi tietokantataulu on, sitä kauemmin tämä vie aikaa. Kun taulussa on indeksit kyseessä oleviin kolumneihin, MySQL voi nopeasti päätellä paikan, mistä etsiä tietoa keskeltä datatiedosta, eikä sen tarvitse käydä läpi koko taulun dataa. Jos esimerkiksi taulussa on 1000 riviä, tämä tekniikka on ainakin 100 kertaa nopeampaa kuin lukea koko tiedosto pala kerrallaan. Kun taas on tarvetta päästä kerralla käsiksi useimpiin riveihin kerralla, on nopeampaa lukea koko tiedosto osa kerrallaan. (16.)

2.8.3 Normalisointi

Tietokannan normalisointi on vaiheittainen malli, jota seuraamalla saadaan relaatiotietokannan rakenne parhaiten tukemaan tietojen ehjää tallennusta ja tiedon tehokasta saatavuutta. Vaiheet vähentävät tiedon redundanssia (samaa tietoa tallennettaisiin useaan kertaan) ja parantavat tallennetun tiedon eheyden (keskinäisen konsistenssin) säilymistä. (10.)

Kuitenkin monista relaatiotietokannoista puuttuu puhdas erottelu tietokannan loogisen rakenteen ja tiedon fyysisen tallennuksen toteutustavan välillä, jolloin kyselyt täydellisesti normalisoituun tietokantaan voivat olla suoritukseltaan hitaita. Siinä tapauksessa denormalisointia voidaan käyttää tehokkuuden paranta-

miseen – tällöin saavutetun tehokkuuden hintana on tiedon eheyden hallinnan vaikeutuminen. (10.)

Normalisointi kannattaa toteuttaa ympäristölle, jossa tietojen päivittymistä tapahtuu paljon. Normalisoinnissa on useita sääntöjä, joista eniten käytettyjä ovat kolme ensimmäistä. (10.)

Ensimmäinen normaalimuoto (1NF) vaatii, että tietokannan jokaisen taulun sarakkeiden eli relaatioiden attribuuttien arvot ovat atomisia. Toisin sanoen moniarvoiset attribuutit on poistettava. Esimerkiksi numeroarvo on atominen, kun taas lista numeroita ei ole. Normalisointi tehdään siirtämällä moniarvoiset attribuutit omiin erillisiin tauluihinsa. Toinen normaalimuoto (2NF) kieltää ei-avainattribuuttien ei-triviaalit funktionaaliset riippuvuudet avainehdokkaan osaan. Jos jokaisen taulun avain koostuu vain yhdestä attribuutista, tietokanta käytännössä täyttää suoraan toisen normaalimuodon. Jos kantaan kuuluu tauluja, joiden avainehdokas koostuu monesta attribuutista, on niiden osalta tarkistettava, että mikään attribuutti, joka ei ole avain, ei saa olla osittain funktionaalisesti riippuva mistään avainehdokkaasta. Jos attribuutti on riippuvainen koko avaimesta, ei siis pelkästään osa-avaimesta, se saa sijaita taulussa (2NF) mukaisesti. (10.)

Kolmas normaalimuoto (3NF) kieltää attribuuteilta, jotka eivät ole avaimia, ei-triviaalit toiminnalliset riippuvuudet muihin kuin avainehdokkaiden superjoukkoon. Tämä tarkoittaa sitä, että taulun ei-avainkenttien pitää riippua avainkentistä. (10.)

2.8.4 Taulutyypit

MySQL tukee useita erilaisia taulutyyppejä tai niin kutsuttuja storage engineitä, jotka mahdollistavat tietokannan optimointia. Taulutyypit, jotka ovat mahdollisia MySQL:ssä, ovat ISAM, MyISAM, InnoDB, BerkleyDB, MERGE ja HEAP. Tärkein ominaisuus, joka erottaa kaikki taulutyypit, on mahdollisuus transaktiotur-

vallisuuteen. Pelkästään InnoDB ja BerkleyDB ovat transaktioturvallisia ja vain MyISAM tukee FULLTEXT INDEX:n luomista. Työssä käytetään INNODB-tietokantataulutyyppiä. (11.)

InnoDB-taulut ovat transaktioturvallisia rivitason lukituksella ja niistä löytyy vierasavaintuki. InnoDB-tietokanta on helposti siirrettävissä järjestelmästä toiseen. InnoDB:n heikkous verrattuna MyIsamiin on, että se vie enemmän kovalevytilaa. (11.)

2.8.5 Transaktiot

Transaktio on joukko tapahtumia, jotka suoritetaan peräkkäin, aivan kuin kyseessä olisi vain yksi ainoa tapahtuma. Jos jokin tapahtuma epäonnistuu, silloin koko transaktio perutaan. (9.)

Transaktion ominaisuudet ovat Atomicity, Consistency, Isolation ja Durability (ACID). Atomicity tarkoittaa jakamattomuutta, eli kaikkien tapahtumien on onnistuttava tai muuten tapahtuma perutaan. Consistency tarkoittaa eheyttä, joka varmistaa, että tietokanta vaihtaa tilaa transaktion onnistuessa. Isolation eli eristyvyys tarkoittaa, että transaktiot toimivat toisistaan riippumatta. Durability tarkoittaa pysyvyyttä eli loppuun saatettu transaktio säilyy, vaikka järjestelmä kaatuisi. MySQL:n transaktiot alkavat lauseella START TRANSACTION (BEGIN tai BEGIN WORK voi myös käyttää) ja loppuvat joko COMMIT- tai ROLLBACK-lauseeseen. (9.)

2.8.6 Viite-eheys

Viite-eheyksien avulla voidaan varmistua siitä, että tietokannan sisältämä tieto on yhtenäinen. Viite-eheydet luodaan taulujen luontivaiheessa. Viite-eheyksillä voidaan varmistaa, ettei isäntätauluun jää viittauksettomia tietueita lapsitaulua muokattaessa. Viite-eheyksien avulla voidaan myös automaattisesti päivittää lapsitaulun tietueet, mikäli isäntätaulun arvot muuttuvat. Lapsitaulun viiteavaimelle on löydyttävä sopiva vastine isäntätaulusta. (15.)

3 KALAPAIKKASIVUSTON TIETOKANTOJEN SUUNNITTELU JA TOTEUTUS

Opinnäytetyötä aloittaessani oli yksi ensimmäisistä tehtävistäni päättää, mitä työkaluja haluan käyttää opinnäytetyöhöni kuuluvassa tietokannan ja käyttöliittymän tekemisessä. Ainoana vaatimuksena oli, että tietokannan tulisi mieluiten olla avoimen lähdekoodin tietokanta, ja lisäetuna avoimen koodin työkaluissa on, että ne myös ovat yleensä ilmaisia. Päädyin käyttämään työssäni MySQL-tietokantaa, joka on yksi maailman suosituimpia avoimen lähdekoodin tietokantoja. Pääsin siis kehittämään tietokannan aivan alusta ja suunnittelemaan sen rakenteet siten, että se olisi mahdollisimman helppo käyttää ja sitä olisi tarvittaessa mukava ja helppo jatkokehittää myöhemmin. Päädyin käyttämään opinnäytetyössäni InnoDB-taulutyyppejä niissä olevan hyvän transaktioiden ja viiteavaimien tukemisen takia. Alempana kerron tarkemmin eri vaiheista toteuttamisen ja suunnittelun parissa. (3.)

3.1 Tietokantataulujen suunnittelu ja luonti

Tietokannan suunnitteluun sain opinnäytetyössäni melko vapaat kädet. Sain pienimuotoisen määrittelylistan (liite 1), jossa näkyy, mitä kenttiä työn tilaajat vähintään halusivat tietokantaan. Sain kuitenkin vapaat kädet suunnitella ja toteuttaa itseäni suunnittelemalla taulu- ja kenttärakenteita: miten eri tietokantataulut tultiin yhdistämään toisiinsa, miten kaikki erilainen tieto saataisiin lisättyä helposti ja vaivattomasti ja miten tieto saataisiin helposti luettua tietokannasta.

Lyhyen suunnittelun jälkeen päädyin jakamaan tauluja niin, että niitä oli helppo yhdistellä toisiin tauluihin linkkitaulujen ja viiteavaimien kautta ja samoja tietoja ei tarvitsisi kirjoittaa tietokantaan useita kertoja. Esimerkiksi tietokannassa olevaan kala-tauluun on toteutettu kalojen tyypit, jotka sitten yhdistetään sieltä viiteavaimen kautta kalapaikan viiteavainta vastaavaan kalapaikan nimeen. Näin vältytään esimerkiksi siltä, että kalat pitäisi kirjoittaa joka kalapaikkaan erikseen sen sijaan, että ne vain yhdistettäisiin jo kertaalleen tiedot sisältävästä kala-

taulusta. Minun tuli rakentaa tietokantataulu myös käyttäjätilejä varten ja niiden yhteydet ja käyttöoikeudet muihin tauluihin ja tehtäviin tuli ottaa huomioon.

Suunnittelun jälkeen päätin toteuttaa kalapuikko-tietokannan, johon luotiin kahdeksan taulua, jotka olivat paikka, tiedot, kalat, kalastusmuoto, käyttäjä,counter_count, counter_iipeet ja kuva. Tämän lisäksi luotiin kaksi linkkitaulua, jotka olivat kala_linkki ja kalastusmuoto_linkki, jotka oli helppo yhdistellä aina tarvittaessa muihin tauluihin, eikä samaa tietoa tarvinnut kirjoittaa useaan kertaan.

Tietokantatauluihin tiedot, paikka, kalat, kalastusmuoto, käyttäjä ja kuva tuli kaikkiin tunnistuskentäksi ID-kenttä, jossa oli juokseva numerointi (AUTO INCREMENT), johon voi viitata muista alitauluista vierasavaimella.

Ensimmäisenä tauluna toteutettiin käyttäjä-taulu, johon kuului kentät id, käyttäjänimi, salasana, etunimi, sukunimi, session_id, ipCheck, email ja access. Access-kentässä määritetään käyttäjän oikeudet sivustolla, iPCheck-kenttään tallennetaan käyttäjän ip-osoite ja session_id-kenttää käytetään käyttäjän istuntoon liittyvissä tarkasteluissa. Tällä taululla pidetään huolta käyttäjän oikeuksista kirjautumiseen ja eri tehtävien suorittamiseen sekä seurataan käyttäjän tekemiä tiedon lisäyksiä tietokantaan.

Paikka-tauluun toteutettiin kentät id, kunta, vesinimi, vesikoko, vesisyvyys, yleiskuvaus, linkki ja posterid. Tauluun voidaan tallentaa tiedoksi hieman tarkempaa kuvausta paikasta ja posterid-kentästä on viittaus käyttäjä-tauluun, josta nähdään käyttäjä, joka on lisännyt kyseisen paikan. Läheisesti paikka-tauluun liittyvään tiedot-tauluun voi tallentaa lisäksi tietoa venepaikoista, laavuja tulipaikoista ja kalastusluvista. Tiedot-tauluun on toteutettu viitteet, posterid ja refid, joista selviää käyttäjä, joka on tallentanut lisätiedot kalapaikasta ja mihin kalapaikkaan kyseiset tiedot viittaavat.

Kalastusmuoto- ja kala-tauluihin toteutettiin id:n ja viittauskentän lisäksi vain yhdet kentät. Kala-tauluun tehtiin kenttä kalatyyppi, joka sisälsi tiedon kalan tyypistä, esimerkiksi ahven, kun taas kalastusmuoto-tauluun tuli tieto kalastusmuodosta (esimerkiksi onginta).

Linkkitaulut kalamuoto_linkki ja kala_linkki pitivät huolta, että esimerkiksi kalojen lajityyppiä ei tarvinnut kirjoittaa joka paikalle aina erikseen, vaan ne saatiin yhdistettyä oikeaan paikkaan viitteitä käyttäen. Kalastusmuoto-taulu toimi samalla periaatteella.

Kuva-tauluun toteutettiin kuva-kenttä, joka sisältää kuvan nimen, ja viittauskentät sekä posterid että refid, joilla kuva yhdistettiin oikeaan kalapaikkaan ja käyttäjään. Kuville on määritetty maksimikoko ja -määrä kalapaikkaa kohden.

Counter_count- ja counter_iipeet-taulujen avulla toteutettiin sivustolle kävijälaskuri. Laskuri toimii siten, että se rekisteröi aina yhden käyttäjän ip-osoitetta kohden ja lisää sen arvoa yhdellä. 24 tunnin kuluttua samasta ip-osoitteesta sivulle tultaessa se uusii toimintansa. Laskuri huomioi jo pelkästään etusivulle tulleet käyttäjät.

3.2 Tietotyypit

Kaikissa tauluissa on perusavaimena ID-kenttä, joka on määritetty kaikkiin tauluihin samanlaiseksi. ID-kentissä kokona on käytetty INT(11) arvoa ja kentissä on myös käytössä (AUTO_INCREMENT) eli juokseva numerointi, jonka tietokantapalvelin toteuttaa. Kaikki numerot ovat positiivisia ja niille tulevat arvot antaa tietokantapalvelin itsestään, eikä arvoja pysty itse muuttamaan. Käyttäjien tiedot on tallennettu käyttäjä-tauluun ja heidän käyttöoikeutensa on tallennettu tarkkaan numeroarvoon, joten tarvittaessa käyttäjille voi helposti muokata erilaisia käyttöoikeuksia viittaamalla oikeudet suoraan tähän tiettyyn numeroarvoon.

Numeraalisina arvoina olevia kenttiä on lisäksi käyttäjän tunnistamisessa käytetyt kentät posterid ja refid, joilla viitataan suoraan käyttäjiin ja saadaan selville helposti, kuka käyttäjä on lisännyt mitäkin tietoja tietokantaan. Myös vesistön koko ja syvyys on merkattu tietokantaan numerolukuina.

Tietokannan on opinnäytetyössäni otettava vastaan myös paljon tekstimuodossa tulevaa dataa. Tekstimuodossa olevan datan tallentamiseen on käytetty VARCHAR-tietotyyppejä, jotka ovat varta vasten tarkoitettuja eripituisten merkkijonojen tallentamiseen. VARCHAR-tyypin kentille määritetään maksimipituudet. Esimerkiksi etunimeen on työssäni käytetty arvoa VARCHAR(20) ja sukunimeen arvoa VARCHAR(25). Varsinaisen datan lisäksi VARCHAR arvoihin tallennetaan 1–2 tavun etuliite, joka kertoo merkkijonojen pituuden. Tämän menettelyn ansiosta säästetään tilaa, jos suurin osa tallennettavasta datasta ei täytä määriteltyjä maksimipituuksia. Etuliitteen pituuden määrää kentän maksimipituus. Jos kentän maksimipituus on enintään 255 merkkiä, riittää etuliitteeksi yksi tavu, kun taas jos kentän maksimipituus ylittää 255 merkkiä, tarvitaan kahden tavun etuliite. VARCHAR-tietotyypin maksimiarvo on MySQL:ssä 65535 tavua ja tämä jaetaan kesken kaikkien taulun muiden VARCHAR-tyypin kenttien kanssa.

3.3 Tietokannan hallinta

Tietokannan hallintaa varten toteutettiin databaseinterface-luokka, joka käyttää hyväkseen PHP:n mysqli-MySQL-rajapintaa. Luokkaan on toteutettu kaikki kalapaikkasovelluksessa tarvittavat tietokantaoperaatiot ja tietokantakyselyt. Databaseinterface-luokka pitää huolen myös tietokantayhteyksistä. Databaseinterface-luokka toimii siis rajapintana tietokantaoperaatioille ja operaatioita ei tarvitse uudelleen määritellä missään sovelluksen eri paikoissa. Databaseinterface-luokasta löytyy esimerkki funktioita liitteestä 2.

Lisäksi sovellukseen on itseensä koodattu käyttäjälle mahdollisuus muuttaa muutamia omia asetuksiaan sovelluksen Asetukset-ikkunassa. Kentät, joita

normaali käyttäjä pystyy muuttamaan, ovat email ja salasana. Pääkäyttäjät pystyvät muuttamaan keiden tahansa käyttäjien tietoja. Kentät, joita pääkäyttäjät voivat muokata, ovat email, etunimi, sukunimi, käyttäjänimi, access ja salasana. Tämä mahdollistaa esimerkiksi sen, että pääkäyttäjä saa väärinkäyttäytyvältä normaalikäyttäjältä vaihdettua käyttöoikeudet nopeasti ilman, että käyttäjä pitäisi poistaa suoraan tietokannasta. Kuvassa 1 on kuva pääkäyttäjän Asetukset-ikkunasta.

KUVA 1. Adminin eli pääkäyttäjän Asetukset-ikkuna

3.4 Tietokannan ja tietokantakyselyiden optimointi

Ensimmäinen asia, joka on hyvä ottaa huomioon tietokannan optimoinnissa, on suunnitella tietokannan rakenne hyvin perusteellisesti. Työssäni pyrin suunnittelemaan tietokannan taulut niin, että muuttujat olivat suurin piirtein oikean kokoisia ja eivät näin vieneet turhaa tilaa. Esimerkkinä kerrottakoon, että käyttäjätaulussa oleva etunimi-kenttä on määritelty ottamaan vastaan enintään 20 merkkiä VARCHAR(20). Pyrin myös ottamaan huomioon, että samaa dataa ei tallennettaisi tietokantaan useita kertoja, vaan jo tietokannassa olemassa olevaa tietoa voitiin käyttää helposti hyväksi myös useiden eri kalapaikkojen kanssa.

Myös tietokantaan tallennettavan tiedon tyypin tarkistaminen ennen tallentamista on hyvä suorittaa. Tämän voi tarkistaa esimerkiksi ohjelmallisesti. Tallennettavan tiedon tyypin täsmätessä sen tietokannan kentän kanssa, joka ottaa tietoa vastaan, ei tietokantamoottorissa aiheudu turhaa tehtävää eli prosessointia, eikä se tallenna vääränlaista tietoa.

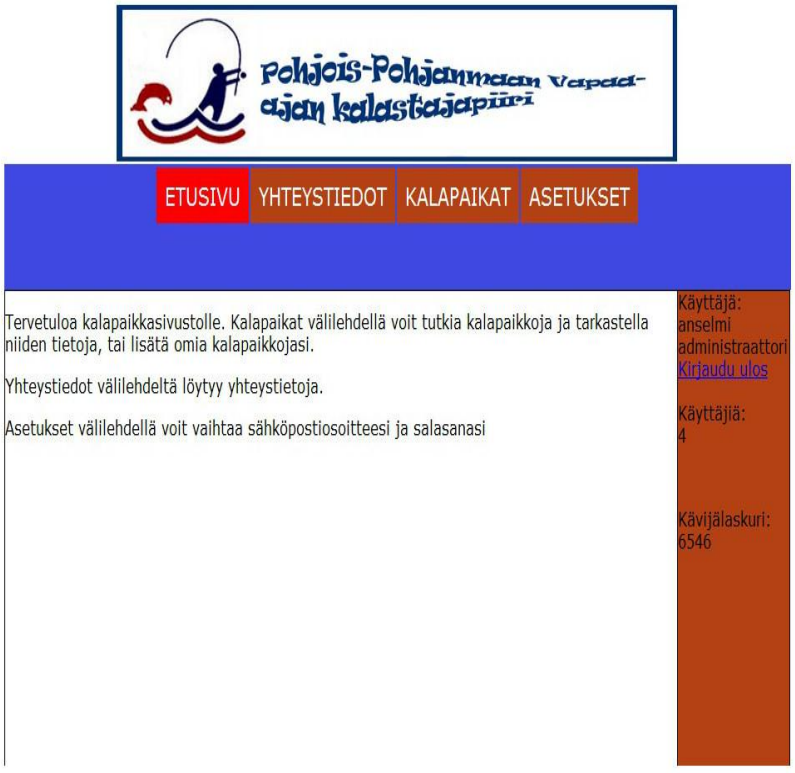
Tehtäessä kyselyitä tietokannasta on hyvä tietää, mitä tietoa hakee. Vie paljon resursseja hakea turhaa tietoa, joka ei ole välttämätöntä tarpeellisen tiedon hakemisessa. Myös esimerkiksi minun työssäni ei ole järkevää hakea tietokannasta suoraan kaikkien kalapaikkojen kaikkia tietoja, vaan pilkkoa haku niin, että ensin haetaan nimet ja näistä saa sitten hakemalla lisätietoa. Hakukyselyissä voi kriteereinä käyttää indeksoituja kenttiä, joiden tiedon hakeminen on nopeampaa. Indeksejä ei yleensä ole kuin yksi taulua kohden. On suositeltavaa välttää koko taulun tietojen hakemista tietokannasta kerralla.

Työssäni voi Etsi-toiminnolla tehdä kyselyjä, joissa yhdistetään paikan, vesistön, kalatyypin ja kalastusmuodon yhteishaku. Kuvassa 2 on kuva kalapaikkojen Selaa-näkymän Etsi-toiminnosta.

KUVA 2. Sivuston kalapaikkojen Selaa-näkymä ja Etsi-toiminto

4 KALAPAIKKASIVUSTON KÄYTTÖLIITTYMÄ

Tietokannan muokkaamiseen tarvittiin selkeä ja helposti käytettävä käyttöliittymä. Käyttöliittymän avulla tietojen lisäämisen, muokkaamisen ja näyttämisen tulee onnistua helposti. Käyttöliittymänä toimiva web-pohjainen sovellus on toteutettu PHP-ohjelmointikielellä. Käyttöliittymän ulkoasu on tehty alusta alkaen selkeäksi, jotta se mahdollistaisi helpon käytön myös mobiililaitteissa olevilla selainohjelmilla. Sivusto on toteutettu toimimaan lähes saman oloisessa näkymässä vain sisällön muuttuessa Sisältö-ikkunassa ja linkkien muuttuessa valikko-näkymässä. Kuvassa 3 on sivuston Etusivu-näkymä.



KUVA 3. Sivuston Etusivu-näkymä

4.1 Käyttöliittymän suunnittelu

Käyttöliittymän toteuttamista varten saatiin melko vapaat kädet, sillä ainoana vaatimuksena oli sivuston toimiminen lähtökohtaisesti myös mobiilikäytössä. Otin tavoitteeksi käyttöliittymälle, että tietoa olisi helppo lisätä tietokantaan ja lisäämisen jälkeen tiedon muokkaaminen ja näyttäminen onnistuisi erittäin helposti.

Käyttöliittymää, sen toiminnallisuutta ja ulkoasua suunnitellessani aloin pala palalta saada mieleeni ajatuksia, miten haluaisin sen toteuttaa, mitä kaikkia toimintoja käyttöliittymässä tulisi olla ja miten toiminnot osioitaisiin niin, että ne pysyisivät mahdollisimman helppoina käyttää. Sovelluksen toiminta vaatii kirjautumisen, joten ei-kirjautuneelle käyttäjälle suunniteltiin näytettäväksi vain kaksi valikko-linkkiä: etusivu ja rekisteröidy. Rekisteröitymisen jälkeen käyttäjän tulee saada täysi näkymä, eli mukaan tulevat valikko-linkit ovat kalapaikat, yhteystiedot ja asetukset. Sovellukseen olisi helppo lisätä toiminnallisuutta myös ei-kirjautuneille käyttäjille kuin myös lisää toiminnallisuutta kirjautuneille käyttäjille. Nämä tulevat olemaan sovelluksen päänäkymät ja näistä tulee lisätietoa toteutuksessa. Käyttöliittymän toiminnot ovat listattuna seuraavaksi:

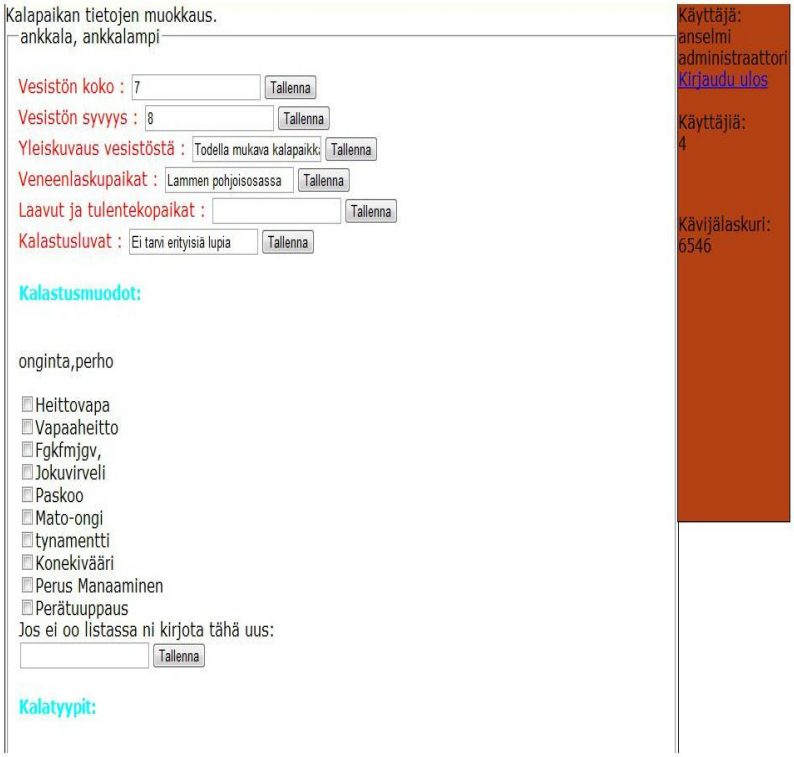
- rekisteröityminen sivulle
- sisään- ja uloskirjautuminen sivulle
- käyttäjälaskuri
- kävijälaskuri
- kalapaikkojen lisääminen
- kalapaikkojen muokkaaminen
- kalapaikkojen haku
- kalapaikkojen lukeminen
- kuvan lisääminen
- käyttäjän asetusten muokkaaminen
- pääkäyttäjällä mahdollisuus muokata muita käyttäjiä
- raportti kalapaikasta.

4.2 Käyttöliittymän toteutus

Ensimmäisenä käyttöliittymään toteutettiin yksinkertainen ulkoasupohja, josta kerrotaan lisää Käyttöliittymän ulkoasu alaluvussa. Tämän jälkeen sovellukseen toteutettiin käyttäjät ja otettiin huomioon kirjautuneiden ja ei-kirjautuneiden käyttäjien eri näkymät. On helpointa toteuttaa käyttäjien oikeuksien tarkistaminen sovelluksen alkuvaiheessa kuin vasta myöhemmässä vaiheessa varsinkin sovelluksissa, joissa vaaditaan käyttäjiä kirjautumaan sisään palveluun sen täyden toiminnallisuuden saamiseksi. Tämän jälkeen on helpompi toteuttaa muita tulevia toimintoja, kun voi verrata aina suoraan, onko käyttäjällä oikeutta tällaiseen toimintoon. Käyttäjien käyttöoikeuksia vertaillaan tietokannan käyttäjä-taulussa olevaan access-kentän arvoon määriteltäessä toimintojen käyttämisoikeuksia.

Seuraavaksi toteutettiin sovelluksen perusvaatimuksiin kuuluva kalapaikan lisääminen tietokantaan -toiminto. Kalapaikan lisääminen toteutettiin myös mahdollisimman yksinkertaisesti niin, että aluksi siihen vaadittavia tietoja ovat kalapaikan nimi, vesistön nimi ja kalat, joita vesistöstä löytyy. Kalapaikkaan pystytään lisäämään lisätietoa muokkausruudussa, mutta lisäämisestä ei haluttu tehdä suoraan liian pitkää. Kalapaikasta tuli saada ulos myös yksinkertainen raportti, joka näyttää kaikki tarpeelliset tiedot kalapaikasta. Raportti aukeaa suoraan painamalla halutun kalapaikan linkkiä.

Sovelluksen käyttäjien kaikki pyynnöt keskitetään main.php-tiedostolle URL-parametreina. Sivun nimiä ei näytetä selaimien osoitekentässä, vaan tämän sijaan se näyttää siltä, että oltaisiin main-sivulla, jossa vain parametrin arvo muuttuu. Osoiterivillä näkyy esimerkiksi ”main.php?page=2”. Kuvassa 4 on kalapaikan Muokkaus-näkymä ja kuvassa 5 on kuva kalapaikan Raportti-näkymästä.



KUVA 4. Kalapaikkojen Muokkaus-näkymä

KUVA 5. Kalapaikan Raportti-näkymä

4.3 Käyttöliittymän ulkoasu

Ulkoasuun ei ollut mahdollisuutta keskittyä tarpeeksi, vaan suurin painoarvo oli teknisissä asioissa. Käyttöliittymään toteutettiin niin kutsuttu sisältö-sivu, jonka yläpuolella sijoitettiin valikko, viereen oikea sivupalkki ja valikon päälle oli asetettu yhdistyksen logo. Pyrin tekemään käyttöliittymän niin, että kaikki vaihtuva tieto käsiteltiin sisältö-sivulla ja näin ollen web-sivusto toimii samanlaisissa kehyksissä koko ajan. Valikko-ikkunassa on muutamia vaihtuvia linkki-painikkeita, jotka muuttuvat käyttäjän käyttöoikeuksien mukaan, ja oikeassa sivupalkissa näytetään kirjautuneen henkilön tiedot ja kaikkien käyttäjien yhteislukumäärä.

Ulkoasun muokkaaminen onnistuu kuhunkin lohkoon varsin vaivattomasti ja sen voi toteuttaa suoraan sovellukselle määritellyssä CSS-tiedostossa. CSS-tiedosto sisältää muutenkin kaikki sovelluksen perustyylimäärittelyt, kuten esimerkiksi fontin tyypin, värin, mahdollisen taustakuvan tai taustavärin. Liitteessä 3 on esimerkki työssä käytetystä CSS-koodista.

4.4 Tietokanta-luokka

Tiedon tallentaminen ja muokkaaminen tapahtuu sovellukseen tehdyissä HTML-formeissa. HTML-formi tarkoittaa kenttää tai kenttiä nettisivulla, johon käyttäjä voi laittaa tietoa, joka lähetetään sitten serverille prosessoitavaksi. Liitteessä 4 on yksi esimerkki työssä käytetystä HTML-formista.

Tietokanta-luokkaa käytetään sovelluksessa kaikkeen toimintaan, jossa haetaan tietoa tietokannasta tai laitetaan sitä sinne lisää. Luokkaan on toteutettu kaikki tarvittavat funktiot, joita sovelluksessa voidaan tarvita. Luokka on määritelty mukaan sovellukseen main.php-sivulla, joten se on käytettävissä suoraan sovelluksen eri osissa. Kun tietokantaa halutaan käyttää, luodaan tietokanta-luokasta olio, jolla toteutetaan sitten haluttu tehtävä.

4.5 Mobiilikäyttö

Sivuston tuli toimia kohtuullisesti myös mobiililaitteissa ja niiden selainohjelmissa. Siksi sivustosta tehtiin lähtökohtaisesti skaalautuva mobiilikäyttöön. Linkkikuvakkeet on muotoiltu hieman suuremmiksi ja sopeutuvaksi pienemmällekin resoluutiolle. Toteutuksessa huomioon otettiin myös, ettei liikkuvan datan määrä kasva liian suureksi ja näin ollen hidasta sivuston selaamista mobiililaitteella.

5 TIETOTURVA

Tietoturva on yksi kaikkein tärkeimpiä asioita, joka tulee ottaa vakavasti kehitettäessä palveluita. Monet kehittäjät luottavat liikaa ihmisiin ja kuvittelevat, että he eivät edes kiusallaankaan kokeile, miten jokin komponentti esimerkiksi toimisi laittamalla sinne ei-sopivia arvoja tai merkkejä. Tämän jälkeen on otettava huomioon vielä ihmiset, jotka haluavat tahallaan väärinkäyttää palveluita (esimerkkinä hakkerit). Hyvää tietoturvaa kehitettäessä on syytä myös antaa testaamiselle painoarvoa eikä vain uskoa, että kaikki toimii suoraan niin kuin pitäisi.

5.1 SQL-injektio

SQL-injektio on tekniikka tietoturva-aukkojen hyödyntämiseksi järjestelmiin tunkeutumisessa. Niitä esiintyy tietokantapohjaisissa sovelluksissa. Ne ovat varsin yleisiä www-pohjaisissa sovelluksissa, joissa käyttäjät käyttävät tietokantaa www-rajapinnan yli, mutta SQL-injektiot eivät sinällään ole www-sidonnaisia. SQL-injektiossa hyökkääjä antaa tietokantapalvelimelle SQL-komentoja, joita hänen ei pitäisi pystyä tekemään. Tämänkaltainen hyökkäys tapahtuu useimmiten puuttuvan tai väärin toteutetun syöttötiedon tarkistuksen kautta ja joissain tapauksissa tietokantarajapinnassa tapahtuvan tiedon väärästä käsittelystä. SQL-injektioiden torjumiseksi on hyvä tarkistaa kaiken käyttäjältä tulevan tiedon oikeellisuus ja estää erikoismerkkien lisääminen vääriin kohtiin. Työssä olen käyttänyt tähän PHP:ssä olevia funktioita mysql_real_escape_string() ja htmlspecialchars(). Toinen asia, joka hankaloittaa hakkerin tietomurto yrityksiä, on antaa maksimipituudet kentille, koska tällöin niissä ei saa käytettyä liikaa merkkejä. (12.)

5.2 Huomautukset ja virheilmoitukset

Huomautuksista ja virheilmoituksista on hyötyä enimmäkseen sovelluksen kehittämisen aikana. On hyvä kertoa virheilmoituksessa kohta, jossa virheen voi käydä korjaamassa. On olemassa erilaisia tapoja ja asetuksia, miten virheet halutaan tuoda esille. Esimerkiksi koodinpätkä error_reporting(E_ERROR) tarkoittaa, että näytetään vain virheet ja huomautuksia ei tule ollenkaan. Sovelluksen valmistuessa ne yleensä otetaan pois käytöstä jo visuaalisen puolen kannalta, etteivät virheet näy käyttäjille, mutta myös sen takia, että ne voivat sisältää tietorakenteita, joita käyttäjien ei tarkoituksellisesti haluta näkevän.

5.3 Salasanat

Salasanoja ei kannata tietokantaan tallentaa suoraan pelkkänä merkkijonona. Jos tunkeutuja pääsee käsiksi tietokantaan ja salasanoja ei ole suojattu millään tavalla, on hän saanut haltuunsa paljon helppoja käyttäjätietoja. Työssäni olen suojannut salasanakentän MySQL:n omalla password() funktiolla, joka sitten generoi hashi-suojauksen käyttäjän salasanasta.

5.4 XSS

Cross Site Scripting eli XSS on tietoturva-aukko, jota esiintyy yleensä www-sovelluksissa. XSS mahdollistaa sellaisen koodin ja skriptien syöttämisen sivustolle, jotka eivät sinne kuulu. Tämä mahdollistaa hyökkääjän tunkeutumisen sivustolle. Ei-pysyvissä tai väliaikaisissa XSS-aukoissa haitallisesti muotoiltu koodi ei tallennu sivustoa ylläpitävälle palvelimelle, mutta toimii käyttäjän selaimessa siten kuin sen alkuperä olisi sivusto itsessään. Haittakoodi ujutetaan sivustolle osoitteen avulla. Käyttäjä täytyy huijata klikkaamaan vahingollisesti muotoiltua linkkiä esimerkiksi sähköpostissa. Tällä tavalla hyökkääjä voi saada käyttäjien tiedot haltuunsa. Paras keino puolustautua XSS-hyökkäyksiä vastaan on käyttäjiltä tulevan tiedon oikeellisuuden varmistaminen ja haitallisen tiedon pois suodattaminen. (13.)

5.5 Sessiot

Session hallinta on tekniikka, jonka avulla on tarkoitus säilyttää turvallisesti tietoa yhden session ajan eli siihen asti, kunnes käyttäjä lopettaa session tai sulkee selaimen. Kun sessio käynnistetään, sille luodaan yksilöllinen sessiotunniste. Istunnon kaappauksen täydellinen estäminen on mahdotonta ja siksi siihen ei tule tallentaa tärkeää tietoa, mikäli se on mahdollista.

Työssä sessioihin on tallennettu vähemmän tärkeitä tietoja käyttäjistä, kuten session_id ja ipCheck. Työssä sessio luodaan kirjautumisen jälkeen ja sitä tarkistetaan aina sivulta toiselle siirtyessä. (14.)

6 TESTAUS

Testaaminen on yksi ohjelmistojen kehitykseen kuuluvista tärkeistä vaiheista. Työssäni en suunnitellut mitään järjestelmällistä testaussuunnitelmaa, enkä saanut mitään ohjeita tämän suorittamiseen, vaan päädyin testaamaan sovelluksen toimintoja aina niiden valmistuessa. Testasin eri toimintojen toimivuutta ja tietoturvaa.

Sivuston ulkoasun testaaminen oli testaamisen helpoin osa-alue. Suurin syy tähän oli, että HTML-koodin tulokset näkyvät välittömästi sivulla. Käytin testaamiseen useita suosituimpia selaimia, kuten Operaa, Chromea, Firefoxia ja Internet Exploreria. Tarkoituksena oli saada sivusto toimimaan samannäköisenä ja samalla lailla kaikissa näissä selaimissa. Pienimuotoisen CSS-säätelyn jälkeen ulkoasu saatiin toimimaan samalla lailla kaikissa selaimissa.

Useissa eri toiminnoissa sovelluksen aikana käyttäjä lähettää tietoja tietokantaan. Nämä toiminnot olen testannut antamalle sovellukselle tahallaan väärän muotoista tietoa. Käyttäjän syöttämän tiedon tietokantaan tallentamisen parissa testattiin sovelluksen haavoittuvuutta myös yksinkertaisille SQL-injektioille. SQL-injektio tarkoittaa, että sovellukselle syötetään SQL-koodia sisältävää dataa. Lisäksi testattiin, ettei käyttäjä pääse suorittamaan sivustolla omaa koodiaan, esimerkiksi käyttäjän tietokantaan syöttämää tietoa tulostettaessa. Kaikki testeissä havaitut ongelmat korjattiin.

7 POHDINTA

Työn tavoitteena oli kehittää kalapaikkatietokanta ja siihen päälle käyttöliittymänä toimiva www-sivusto. Sivustolle piti luoda kaikki tarpeelliset toiminnot tiedon tallentamiseen, muokkaamiseen ja näyttämiseen tietokannasta. Sivustossa on myös toimiva hakujärjestelmä, jolla kalapaikkaa voidaan etsiä ja kalapaikasta saa myös tekstipohjaisen raportin. Käyttöliittymä toteutettiin lähtökohtaisesti sopimaan myös mobiilikäyttöön.

7.1 Jatkokehitys

Ohjelmiston toiminnallisen puolen valmistuessa sivuston ulkoisia ominaisuuksia voisi tehdä edustavammiksi. Lisäksi jollekin mobiilialustalle on tulossa sovellus, joka toimii yhteistyössä jo olemassa olevan tietokannan kanssa. Sovelluksen tarkoituksena on pystyä löytämään käyttäjän sijainnin perusteella tietoja vesistön kalastosta ja kalastuslupien saatavuudesta.

7.2 Opinnäytetyön kirjallinen osuus

Kirjallisen osuuden kirjoitus opinnäytetyöhön ei ollut aina helppoa. Useita kertoja sai miettiä tarkkaan, mistä asioista haluaisi kirjoittaa ja kuinka tarkasti. Teoreettinen osuus oli huomattavasti helpompi kirjoittaa kuin käytännön osuus. Tähän vaikutti varmasti se, etten tehnyt tarkempia muistiinpanoja itselle opinnäytetyön edetessä. Kuvien ja liitteiden valitseminen työhön vaati myös hieman enemmän harkintaa, että osaisi löytää oleellisimpia asioita.

7.3 Kokemuksia opinnäytetyöstä

Aluksi opinnäytetyön onnistuminen annetussa aikataulussa tuntui haastavalta, sillä aloitin työn tekemisen vasta helmikuussa. Tulisin siis kohtaamaan sekä mahdollisia aikatauluongelmia että PHP-ohjelmoinnin kokemuksen puutteesta johtuvia ongelmia. Tiedossa oli siis paljon haasteita ja uuden tiedon opettelua. Opinnäytetyö lähti hyvin käyntiin ja sain aikaan paljonkin asioita lyhyessä ajassa. Vastaan tuli myös asioita, jotka veivät huomattavasti enemmän aikaa kun

niille oli itse alun perin suunnitellut. Työn etenemistä voisi kuvailla myös suureksi oppimisprosessiksi.

Sain suunnitella ja toteuttaa opinnäytetyöni lähes täysin itse ja sain vain karkeasti määritellyt tarpeelliset tiedot, miten sovelluksen tulee toimia. Koulussa opitut perusteet projektien suunnitteluun, toteutukseen ja testaukseen olivat hyödyllisiä, joskin monia asioita joutui kertaamaan jonkin verran.

Suoritin opinnäytetyön etätyönä, eikä minulla olisi ollut mahdollista tehdä työtä paikan päällä tilaajien tiloissa. Opinnäytetyön tekeminen etätyönä sopi minulle erittäin hyvin ja se oli helpompi sovittaa muiden aikataulujen kanssa yhteen, koska pystyin tekemään sitä aina halutessani.

Opinnäytetyö pysyi mielenkiintoisena koko projektin keston ajan. Työtä tehdessä jouduin opiskelemaan valtavasti uusia asioita, mutta siitä saatu kokemus ja uuden oppiminen pitivät homman mielekkäänä. Opinnäytetyö oli laajuudeltaan ja vaativuudeltaan suurin projekti, missä minä olen ollut mukana.

LÄHTEET

1. PHP. 2012. Saatavissa: http://fi.wikipedia.org/wiki/PHP. Hakupäivä 16.4.2012.

2. HTML. 2012. Saatavissa: http://fi.wikipedia.org/wiki/HTML. Hakupäivä 16.4.2012.

3. MySQL. 2012. Saatavissa: http://fi.wikipedia.org/wiki/MySQL. Hakupäivä 16.4.2012.

4. SQL. 2012. Saatavissa: http://fi.wikipedia.org/wiki/SQL. Hakupäivä 16.4.2012.

5. LAMP. 2012. Saatavissa: http://fi.wikipedia.org/wiki/LAMP_(tietotekniikka). Hakupäivä 16.4.2012.

6. CSS. 2012. Saatavissa: http://fi.wikipedia.org/wiki/CSS. Hakupäivä 16.4.2012.

7. Apache. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Apache_(palvelinohjelma). Hakupäivä 16.4.2012.

8. Tietokannat.2012. Saatavissa: http://materiaali.osao.fi/kaul/verkko-ope-tus/datanomi/tietojarjestelmien_kaytto_ja_kehittaminen/tietokantasuunnittelun_perusteet/tietokannan_suunnittelu/suunnittelu_asia.htm. Hakupäivä 23.4.2012.

9. Transaktio. 2012. Saatavissa: http://www.ratol.fi/opensource/mysql/transaktio.htm. Hakupäivä 23.4.2012.

10. Normalisointi. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Tietokannan_normalisointi. Hakupäivä 23.4.2012.

11. Taulutyypit. 2012. Saatavissa: http://www.mysqltutorial.org/understand-mysql-table-types-innodb-myisam.aspx. Hakupäivä 23.4.2012.

12. SQL injektio. 2012. Saatavissa: http://fi.wikipedia.org/wiki/SQL-injektio. Hakupäivä 23.4.2012.

13. XSS. 2012. Saatavissa: http://fi.wikipedia.org/wiki/Cross_site_scripting. Hakupäivä 23.4.2012.

14. Sessiot. 2012 Saatavissa: http://users.jyu.fi/~kolli/ITK215_05/php/?sivu=sessiot. Hakupäivä 9.5.2012

15. Viite-eheys. 2012 Saatavissa: http://www.oamk.fi/sbc/www/mysql.php#viiteeheys. Hakupäivä 9.5.2012

16. Indeksointi. 2012 Saatavissa: http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=mysqlphp10. Hakupäivä 9.5.2012

TIETOKANNAN MÄÄRITTELY LIITE 1

Avainkenttä numerona (int PIILOKENTTÄ)
Kunta, jossa vesistö sijaitsee (String kenttä)
Avainkenttänumeroa vastaava Vesistön nimi (kenttä vesistö: String)
Vesistön koko (int ha – tästä lasketaan säde oletuksella, että vesistö on pyöreä (ellei parempaa tapaa))
Vesistön keskisyvyys (float m)
Kalastusmuodot (String kenttä)
Kalat (String kenttä)
Yleiskuvaus (String kenttä)
Veneenlaskupaikat (String kenttä)
Laavut ja tulipaikat (String kenttä)
Kalastusluvat (String kenttä)
Linkki (URL, String mahdolliseen lisätietosivustoon, voi olla myös tyhjä)
Kuva (jpg = tiedostonnimi ja sijainti, saa olla myös tyhjä)
Author (kuka lisännyt ko. tiedot = käyttäjätunnus, PIILOKENTTÄ)

```
public function getLinesAssocArray()
        {
                $array=array();
                $resultRows=mysql_num_rows($this->result);
                for ($i=0;$i<$resultRows;$i++){
                $array[]=mysql_fetch_array($this->result, MYSQL_ASSOC);

                }
                return $array;
        }

 public function rollback()
        {
                $this->result = mysql_query("ROLLBACK") or die('Query failed:
' . mysql_error());
                $this->transaction_active = FALSE;
        }
public function query($query)
        {       $this->query = $query;
                $this->result = mysql_query($this->query);
                if ($this->result === FALSE) die(mysql_error());
```

```
 else return $this->result;

        }

public function selectDatabase($db_name)

        {

                $this->$db_name = $db_name;

                mysql_select_db($db_name) or die('Could not select database');

                if ($this->echo) print("selected database ".$db_name." succesful-
ly<BR>");

         }

public function connectToDatabase($db_address, $db_username, $db_password)

        {

                $this->db_address = $db_address;

                $this->db_username = $db_username;

                $this->db_password = $db_password;

                $this->db_link = mysql_connect($this->db_address,$this-
>db_username,$this->db_password)or die('Could not connect: ' . mysql_error());

                if ($this->echo) print("connected to database succesfully<BR>");

        }
```

CSS-esimerkki content-nimiselle diville LIITE 3

```
div.content
{
position:absolute;
margin-top:0px;
margin-left: 15%;
width: 70%;
height: auto;
min-height: 60%;
overflow: none;
border: 1px solid;
padding-bottom: 50px;
    }
```

Esimerkki työssä käytetystä formista LIITE 4

```
<form action="?page='.GET_PAGE.'&subPage=rekisteroidy" meth-
od="post"><table>

<tr><td>Käyttäjänimi:</td><td><input class="login" name="username"
maxlength="20 type="text" /></td></tr>

<tr><td>Salasana:    </td><td><input class="login" name="password"
maxlength="20 type="password" /></td></tr>

<tr><td>Etunimi:     </td><td><input class="login" name="etunimi"
maxlength="20 type="text" /></td></tr>

<tr><td>Sukunimi:    </td><td><input class="login" name="sukunimi"
maxlength="20 type="text" /></td></tr>

<tr><td>Email:       </td><td><input class="login" name="email" maxlength="40
type="text" /></td></tr>

<tr><td></td><td><input type="submit" value="Rekisteröidy"></table>';
```