Helsinki Metropolia University of Applied Sciences
Degree Programme in Information Technology

Kirsi-Marja Waltzer

Bird Observation Application for Mobile Devices

Bachelor's Thesis. 11. April 2010

Supervisor: Jaana Holvikivi, Principal Lecturer
Language Advisor: Taru Sotavalta, Senior Lecturer

Helsinki Metropolia University of Applied Sciences **Abstract**

Author	Kirsi-Marja Waltzer
Title	Bird Observation Application for Mobile Devices
Number of Pages	45
Date	11 April 2010
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Instructor Supervisor	Jaana Holvikivi, Principal Lecturer

The goal of this project was to create a bird observation gathering site designed to be used by mobile devices. The purpose of this simple dynamic Internet application was to study how to create mobile compatible Internet sites.

The project was realized with the use of MySQL database, PHP scripting language and XHTML markup language. Helsinki Metropolia University of Applied Sciences provided the existing MySQL database and a web server supporting PHP scripting.

The completed application included two script files and three XHTML pages: a script for retrieving observation information from the database, a script for inserting data into the database, an index, an instruction and a form page. The application worked with both Internet browsers and with mobile devices.

The application is too simple for serious hobbyists and it lacks all features for data verification. However, it can provide a good way to quickly share information with a large peer base but its usefulness depends solely on the integrity of the data provided by users.

Keywords	MySQL, PHP, XHTML, WAP

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä Otsikko Sivumäärä Aika	Kirsi-Marja Waltzer Lintuhavaintosovellus liikuteltaville laitteille 45 sivua 21.4.2010
Koulutusohjelma	Information Technology
Tutkinto	insinööri (AMK)
Ohjaaja Ohjaava opettaja	 yliopettaja Jaana Holvikivi

Projektin tavoitteena oli toteuttaa liikkuvilla laitteilla toimiva internetsivusto lintuhavaintojen keräämiseen. Matkapuhelinten kanssa yhteensopivien internetsivustojen suunnittelun ja toteutuksen opiskelu oli tämän vuorovaikutteisen, mutta yksinkertaisen, web-sovelluksen päämääränä.

Projekti toteutettiin käyttämällä MySQL-tietokantaa, PHP-komentosarjakieltä ja XHTML-merkintäkieltä. Metropolia Ammattikorkeakoulu tarjosi pääsyn sekä olemassa olevaan MySQL-palvelimeen että PHP-kieltä tukevaan web-palvelimeen.

Valmis sovellus sisälsi kaksi komentotiedostoa ja kolme XHTML-sivua: komentotiedoston havaintotietojen noutamiseen tietokannasta, komentotiedoston tietojen lisäämiseksi tietokantaan, linkkiluettelon sisältävän pääsivun, ohjesivun sekä lomakesivun. Sovellus toimi testeissä sekä normaalilla internetselaimella että matkapuhelimilla.

Sovellus on liian yksinkertainen todella vakavasti otettaville harrastajille. Siitä puuttuvat myös kaikki toiminnot, joita käytetään tiedon varmistuksessa. Sovelluksen hyödyllisyys riippuu täysin käyttäjien toimittamien tietojen rehellisyydestä ja tarkkuudesta, mutta se tarjoaa hyvän keinon jakaa tietoa nopeasti laajan vertaisryhmän kesken.

Hakusanat	MySQL, PHP, XHTML, WAP

Contents

Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheet
DB	Database
IE	Microsoft Internet Explorer
PC	Personal computer
PHP	Hypertext Preprocessor, a server side scripting language
SMS	Short Message Service, a mobile text message service component
XHTML	Extensible HyperText Markup Language
XHTML-MP	XHTML Mobile Profile
XML	Extensible Markup Language
WAP	Wireless Application Protocol

1 Introduction

I have always been interested in dynamic Internet applications and mobile technology, but I am not good at creating content to warrant such an undertaking. When this subject was suggested to me I was happy to accept it, although I was a little insecure as I had not used database or scripts before.

The goal of this project is to design a dynamic Internet application which could be used by a mobile phone to gather bird sighting information. This will be achieved by creating Wireless Application Protocol (WAP) compatible Internet pages which will be connected to the MySQL database through Hypertext Preprocessor (PHP) scripting. As this is an application just to collect information from several people into one place, no identification or search process will be provided.

Many people want to use or share bird sighting information, but most have different motives and needs. This causes problems when designing an application to the still undefined user group. Rather than predefining a target group, it was decided to gather only the basic information, which hopefully might be enough for most users.

A mobile environment creates a few challenges as the screen size is small, data processing capabilities significantly lower than with personal computers (PC) and the data connection bit rate might be slow especially during heavy traffic in the mobile network. The main weapon to combat all these problems is simplicity. Everything shown in the mobile end of the application is very simple and designed to fit the mobile screen. All scripting is in php files which will be interpreted and generated into a web page by a web server, so only a simple web page will be shown to the mobile device.

2 Theoretical background

2.1 Bird observation on the Internet

There is no right way to pursuit bird observation. There are almost as many motivations as there are hobbyists. Some people might be interested in helping conserve species and gain more information about nature, and others may just want to be able to get "points" according to how many rare species they have been able to identify. There is one thing in common among all these groups: almost everyone writes down, and often shares, detailed information about their sightings.

A twitcher follows information from a sighting someone else has done and tries to positively identify that same bird. In the twitcher community, fast, reliable and up-to-date information is essential. Some twitcher organisations maintain information sharing services about rare sightings, but these services are usually limited to their own members. The sharing method in these services is usually the Internet, e-mail or short message service (SMS)-text message.[1]

The website Lintutiedotus provides its members an access to their extensive observation database (DB), a personal bird information diary as well as alerts via e-mails or SMS text messages. A member can leave sighting information by an SMS message or through the Internet.[1]

Hatikka is an observation DB which collects information about all plant, fungus, animal or bird sightings. It is maintained by the Finnish museum of natural history and it offers a personal observation-diary to all registered users.[2]

Tiira is a bird information DB maintained by BirdLife Suomi ry. BirdLife Suomi is a roof organization of most bird interest associations and that gives Tiira DB a somewhat

official standing. Tiira offers a personal bird information diary and a chance to search through the DB to its registered users.[3]

2.2 WAP 2.0

WAP is a protocol developed for handheld wireless applications such as mobile phones or communicators. The WAP 2.0 specification was released in 2002 by the WAP Forum. It introduced several changes to the previous versions such as support for standard Internet protocols. One of these changes was to have the markup language of WAP 2.0 to follow the Extensible HyperText Markup Language (XHTML) architecture. The XHTML version specified in the standard is the XHTML Mobile Profile (XHTML-MP) which is an extended version of XHTML Basic.[4;5,97-98] This brought WAP and normal Internet applications closer together as now same pages could be used in both cases.

2.3 XHTML

XHTML is one of the extensible markup (XML) languages that have been specified for the use of web community. It follows the strict rules required by XML but defines the same elements as HyperText Markup Language (HTML). XHTML is so close to the HTML syntax that most browsers are able to parse it just like HTML, even though most browsers are not XML aware. XHTML is also an easily extendable language as it can be extended with different modules.[5,97-98;6,24-28]

XHTML 1.0 is based on HTML version 4.0. It uses cascading style sheets (CSS) to modify the layout, which allows several pages to use the same layout just by connecting to one style sheet.

There are several different XHTML versions, but XHTML Basic and XHTML-MP are designed to be used with mobile devices. XHTML Basic includes most commands from XHTML 1.0. Only commands which do not work well with a small screen have been discarded. XHTML-MP is quite similar to XHTML Basic; it just includes a small

number of commands and attributes which are considered to be useful while creating content for mobile devices.[5,97-98]

2.4 MySQL

My SQL is an open source relational database management system. A relational database is a database where a group of joined tables are related to each other through specified columns.[7,8] A new server process is started every time MySQL opens a new connection as it does not share any processes. This means that even if one process ends prematurely, all other processes should stay open. MySQL provides a fast, powerful, efficient, reliable, compatible, portable and in most cases free database to build on applications. The roots of the MySQL's popularity lay in its open source code. [7,8-13]

To be able to keep up with the changing circumstances and technologies, MySQL includes a feature called a storage engine. Different storage engines are optimized for different situations, so depending on the used storage engine, the storing method and the location of stored tables may change. For example some storage engines are designed to use a disk space and some others use only random access memory.[9] MySQL offers several native storage engines as well as several community or partner developed ones. A few of the native storage engines are MyISAM, Cluster, Memory and Archive. MyISAM is usually the default storage engine for MySQL. One of the widely used partner developed storage engines is InnoDB. It was developed under GNU General Public Licence by a subsidiary of Oracle, Innobase Oy. InnoDB includes several benefits compared to MyISAM, including a support for transactions and a possibility to make foreign keys.[8,693-705] In figure 1 you the basic structure of MySQL server and different storage engines can be seen.

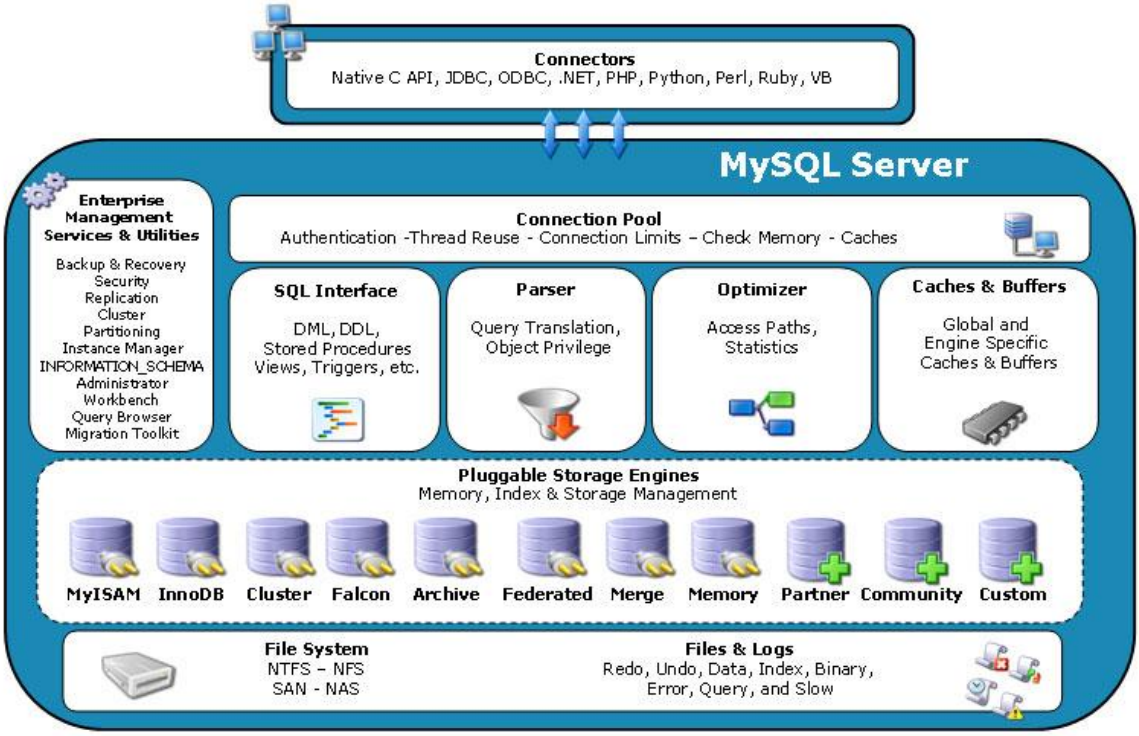

Figure 1. [10]MySQL server

As can be seen from the figure 1, MySQL is very versatile. It offers several options according to the needs of the user.

2.5 PHP

The history of PHP started in 1995 when Rasmus Lerdorf developed a collection of simple Perl scripts and named them as Personal Home page tools. He later wrote and released a larger C-based version, which also included support for communication with databases and named this as PHP/FI (Personal Home Page/ Forms Interpreter).[8,2]

PHP3 was released in 1998 and included a solid infrastructure for a wide range of databases, protocols and Application Programming Interfaces (API) as well as object-oriented syntax support and extensive extensibility features. PHP4 was released in 2000. It was developed to run complex applications efficiently as PHP3 was unable to. Several more features were introduced in this release which included features such as better support for web servers and added security features. PHP5 was released in 2005

and it introduced more new features such as improved object-oriented capabilities, exception handling and improved XML and web services support. [8,2-5]

PHP is a server side scripting language which means all PHP code will be read and executed within a server. After completing the script the server will send resulting file to the browser that will show a printing of it. To be able to combine HTML with PHP, the script will have to be separated from the HTML code. This separation is done by enclosing the script into <?php (script) ?> tag. [7,294-295] This kind of combination can be achieved also by printing out parts of the HTML code during the script.

PHP is considered as an open source software and it can be used free of charge. There have never been any restrictions on usage, modification or redistribution.[8,9]

3 Creating mobile compatible websites with XHTML

3.1 Starting point

The main goal of the application is to be able successfully send and retrieve information from the DB. The solution is to divide this problem into two parts, a connection from a mobile to the Internet site and using a server side scripting language for the connection between the web server and DB. This division is illustrated in figure 2.

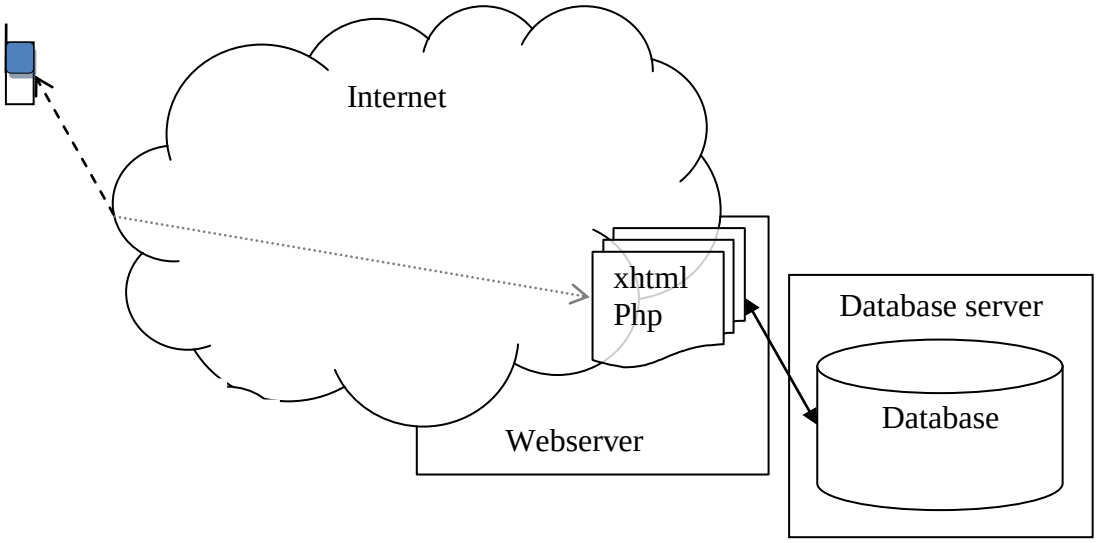

Figure 2. Application overview

The first hurdle is to be able to design Internet pages which will be compatible with programs currently included in mobile phones. This means making an index, instruction and form page compatible with mobile devices.

3.2 Submit form

As the application was developed for mobile use, it was decided to follow the WAP2.0 standard in the use of XHTML. The supported XHTML version for WAP2.0 is XHTML-MP.[2] To increase compatibility for older mobile phones, it was decided to use an older Mobile Profile version which led to Mobile Profile 1.0 to be used.

According to instructions for the Tiira-bird information service, the minimum requirements for bird observations are the date of the observation, species, number, sex, age, dress and state of the birds.[3] To make this application even slightly practical to the target group, all these requirements were incorporated into the submit form. The form consists of five text fields for name, place, date, species and quantity, two radio fields for choosing the error margin and the sex, four dropdown menus for age, dress, state and compass direction and a checkbox for choosing if the observed bird is travelling. To make it easier to navigate the "state" dropdown menu, it was divided into three separate parts: a 'state' which consists of the basic state options, 'migrate' which allows to choose if the bird is currently migrating and a 'direction', which contains all compass points. Only the field name (Nimi), place (Paikka) and species (Laji) are mandatory as those are the very basic information which everyone should be able to fill in without any previous experience.

The form was created by using the notepad text editor provided by Windows. Figure 3 shows a screenshot taken from the Internet, and figure 4 shows the same page from a mobile device. These figures show the basic layout of the form page.

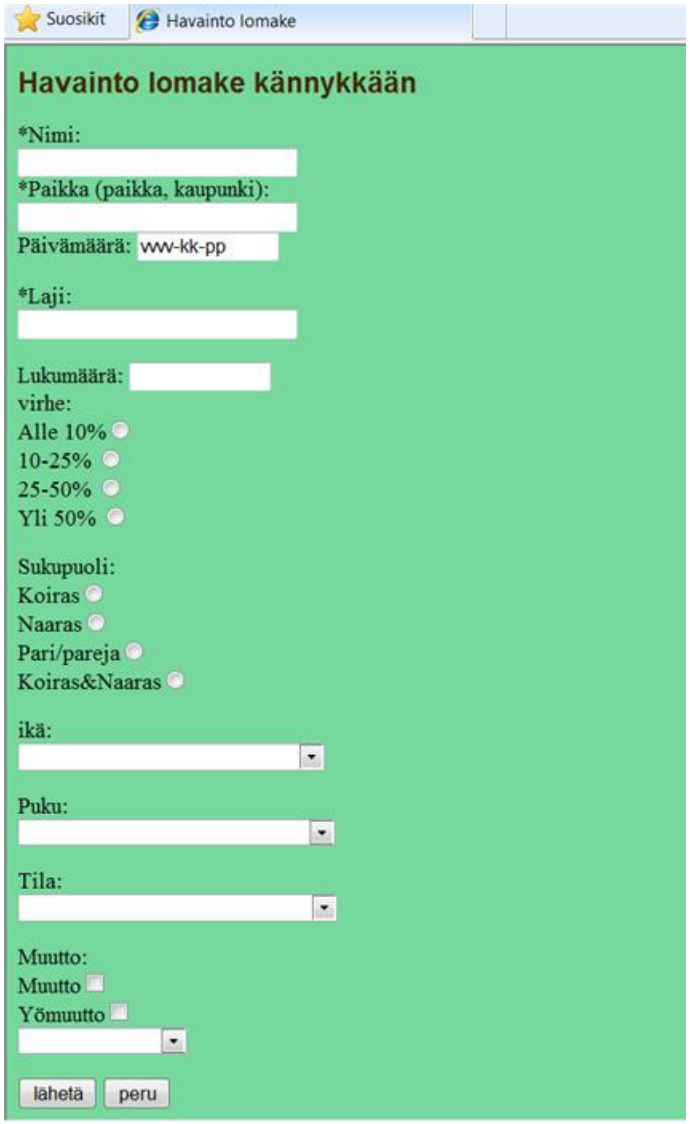

Figure 3. Screenshot from form page

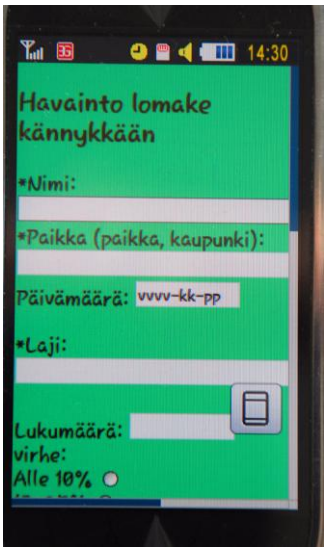

Figure 4. Mobile form screenshot

As seen in figure 4, the layout is not perfect for mobile use but the use of dropdown menus in the form causes the form to be wide. Dropdown menus are used in places where an inexperienced user might not know what to enter.

3.3 Index and instruction sites

The notepad was used to create fully functional pages with XHTML. As this is a mobile application, its index page was designed to be very minimalistic as shown in figure 5. The index contains only three links, one to the instruction page (seen in figure 6), one to the form and one to a list where all previously submitted observations will be shown.

Figure 5. Index page

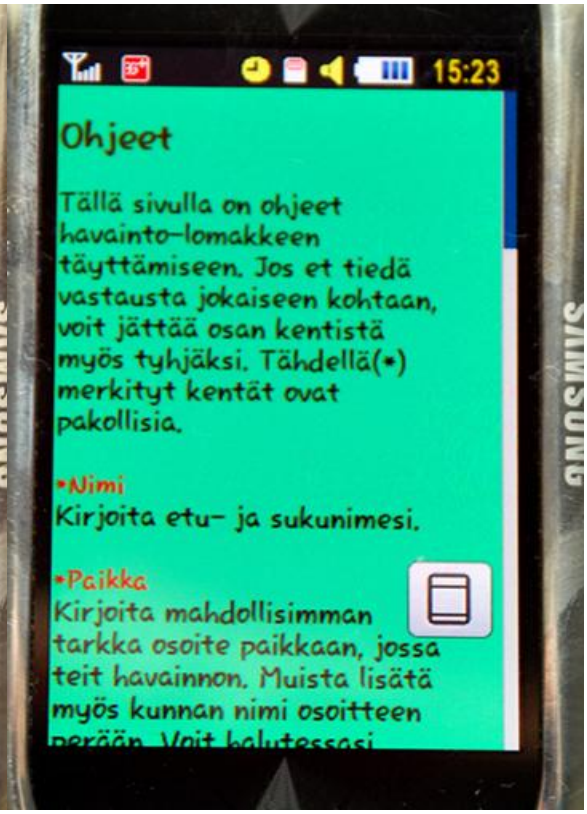

Figure 6. Instruction page

The index page works well in a small screen of a mobile phone. Even though the page is minimalistic, it does not seem too bare in mobile use.

The instruction page is also very simple in design. It contains a short explanation of each field on the form. It should give the user a better understanding of what information should be included.

4 MySQL database for the application

Helsinki Metropolia University of Applied Sciences provides an installed MySQL database and phpMyAdmin, so there was no need to install any new software. As phpMyAdmin has a graphic interface, there was no need to create tables with sql sentences though the command line interface.

4.1 Structure of the database

The database consists of 12 tables from which 11 are related to each other through a table called “Havainto”. The 12th table is a “storage” for XHTML codes that are used inside the PHP script to generate a fully working XHTML page to the user. More detailed description of tables is shown in appendix 1.

The Havainto table collects all the information together, concerning each observation. It includes only three original fields for id, date and number of birds; everything else is an id number pointing to the other tables. The same information can be used in several observations, but it will be recorded to the other tables only once, which makes the relationship between the tables one for many. Figure 7 shows how all the tables are linked together.

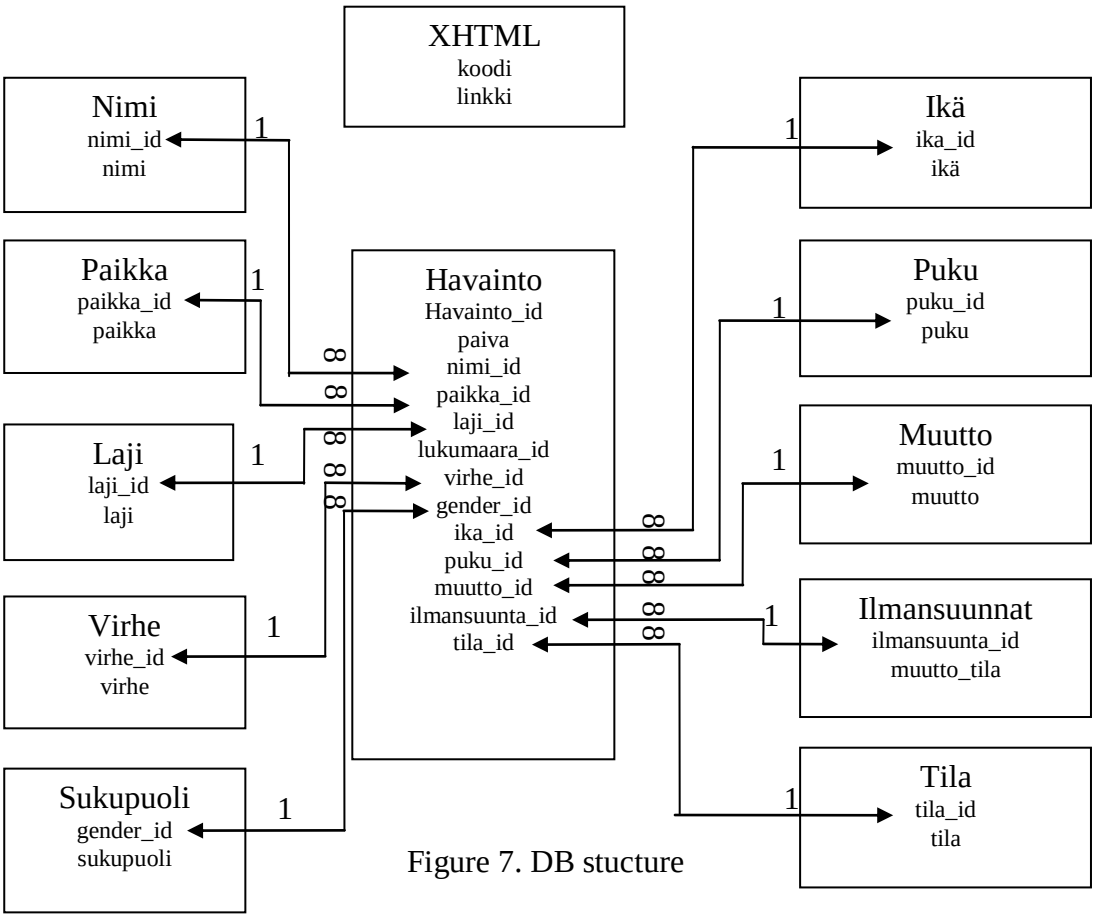

Figure 7. DB stucture

As from figure 7 can be seen, most tables contain just two fields. These are an id field which connects the table to the “havainto” table and the information field.

4.2 Keys and queries

The MySQL relational database management system supports three types of keys: primary keys, unique keys and foreign keys. Primary and unique keys are used to create unique, indexed fields which cannot get a NULL value. There can be only one primary key in the table, but several unique keys can be specified. Primary keys are usually used to create identification fields for the table.[11,286-287] Foreign keys are used to create relational links between parent and child tables. To be able to use foreign keys, the tables must be defined to use the InnoDB storage engine. A foreign key connects the primary key of one table to the indexed field from another table and defines consequent action in case the referred field is updated or deleted. [12,45-46]

In this application all table_id fields were assigned as primary keys. Most id-fields were defined to use auto-incrementing to assure the field will continue to grow in a controlled way without a possibility of duplicate entries. Havainto_id is the only primary key without an auto-incrementing feature, so it can be completely deleted several times without losing use of several id numbers.

There were also three unique keys: Nimi.nimi, Paikka.paikka and Laji.laji. This is to prevent any duplicate entries as there might be several entries from the same user, from the same address or sightings of the same species of bird.

All information was added to the DB with simple INSERT commands embedded in the output.php file. An example of the insert structure used is:

INSERT INTO table VALUES (“value”,”value”); .[11,36]

Data was retrieved from the DB by using SELECT queries. A simple select query follows the following format:

SELECT * FROM table;.

A ‘*’ character represents all/everything in a select sentence, but also the name of the field(s) the query is directed to can be in its place. This type of select structure can also be broadened by adding more conditions after the table name, for example

SELECT * FROM table WHERE something = something; or

SELECT * FROM table ORDER BY field DESC;.[11,40-46]

5 PHP scripts

5.1 PHP between the form and the DB

A PHP script between the submit form and database, seen in appendix 2, was designed to have three main parts during the execution. First it will assign variables to the data read from the form and check that all three mandatory fields contain information. If no information is found, the script will end and show an error message shown by a screenshot below. The second part is to insert all obtained information to the DB, and the third part is to write down a list where the latest 30 currently existing observations are listed. The observations will be listed in descending order, so the latest observations can be seen first. These three parts can be seen in figure 8, and more detailed architecture is shown in figure 9.

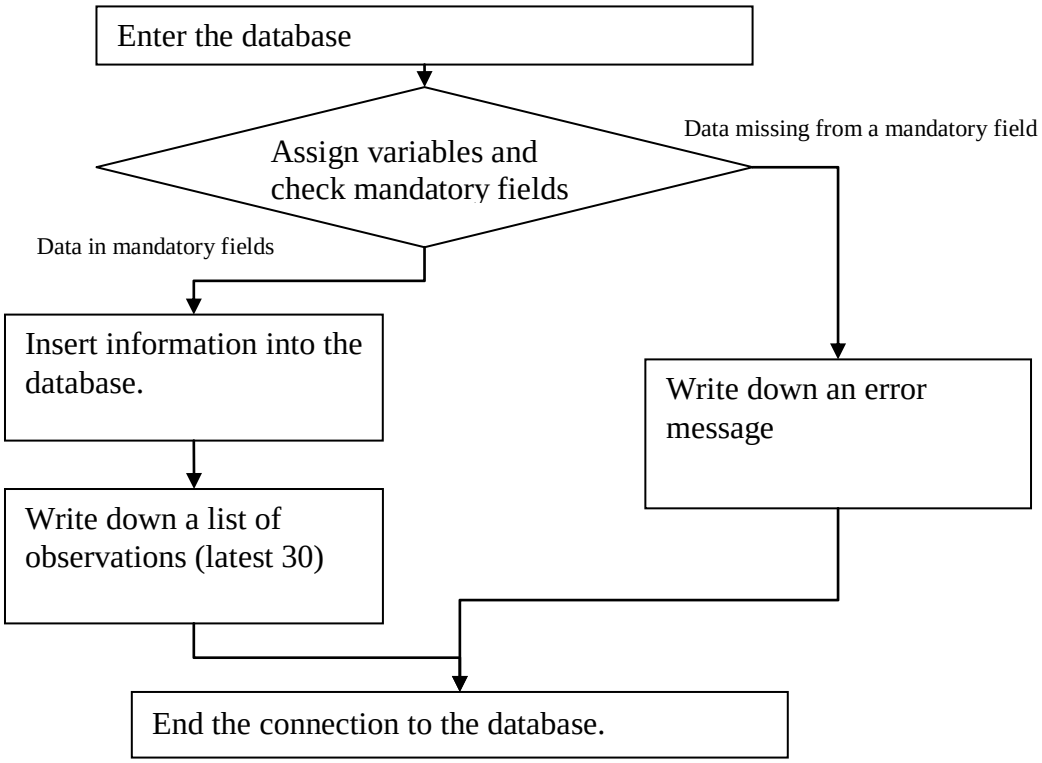

Figure 8. output.php flowchart1

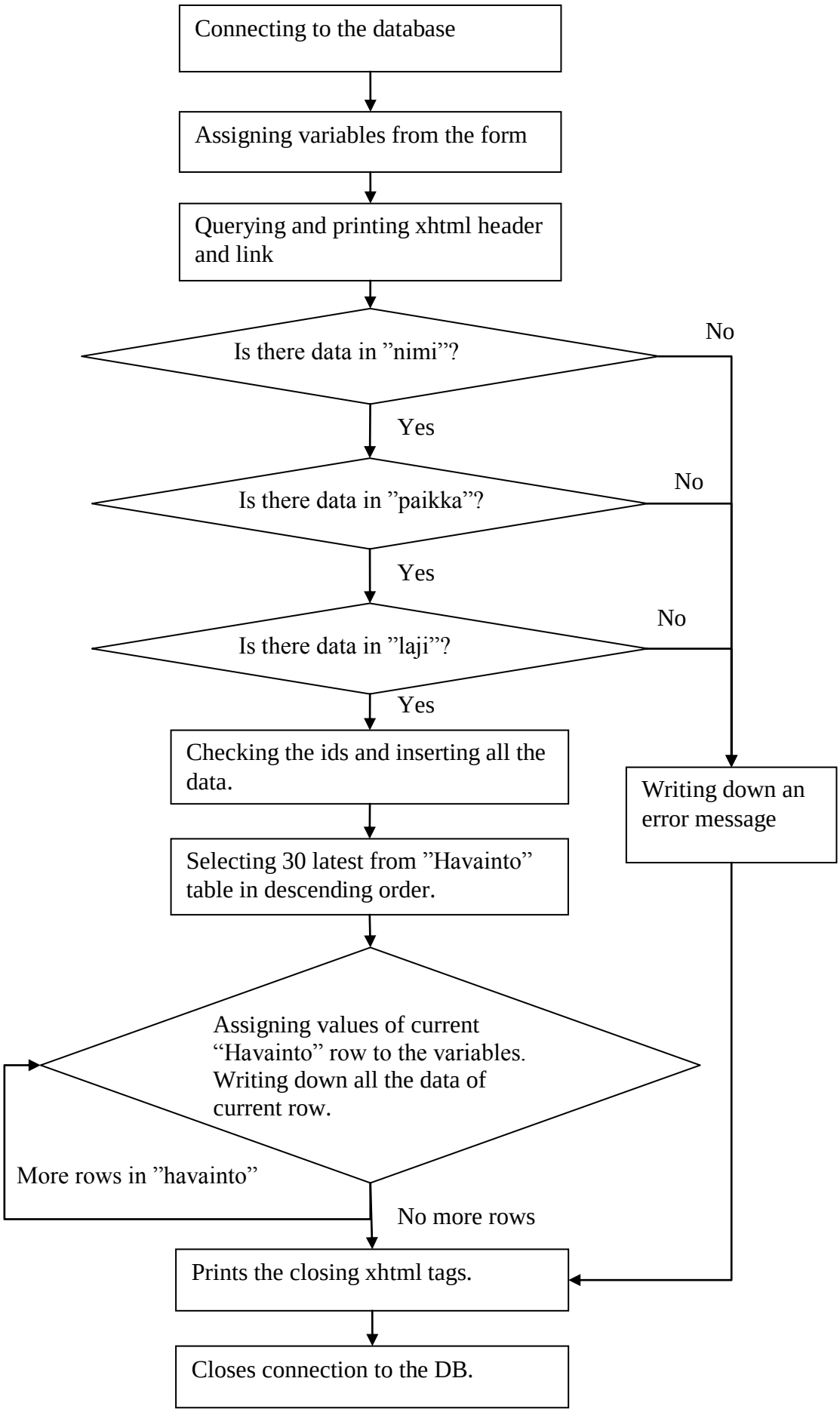

Figure 9. output.php flowchart2

As seen from the figures 8 and 9, the XHTML information will be included as long as the connection to the database is established. Depending on whether mandatory fields are filled or not, there will be two different outputs for this script as long as the database connection is working. Both possible results can be seen in figure 10 and in figure 11.

Figure 10. output.php with no errors

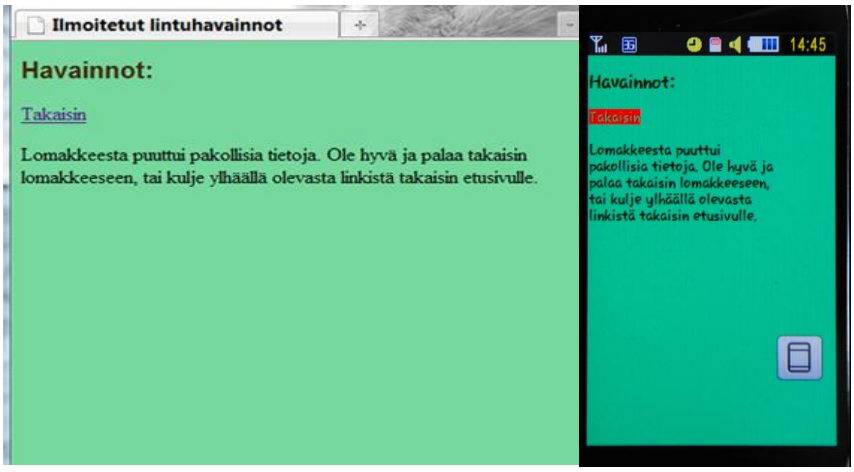

Figure 11. output.php with a mandatory field error

5.2 PHP queries from the DB

A website (havainto.php) listing all the observations submitted to the database was made, using a PHP script embedded with the XHTML Mobile Profile code, as seen in appendix 3. XHTML Mobile Profile headers were stored to the DB and written down during the execution of the script. The script writes down all rows in the Havainto table in descending order, forming a list of observations, as seen in figure 12.

Figure 12. Result of havainto.php

As can be seen from figure 12, havainto.php may give a similar result as output.php if no error occurs during execution. In this function the scripts are nearly identical, havainto.php will give a list of every observation and output.php will give a list of the last 30 observations. The main difference between the scripts is that output.php handles also the data submitted in the form and only then writes down the changed listing.

6 Testing the system

6.1 Testing with the PC

The testing of the PHP scripts were mainly done with Firefox 3.6 browser in the Windows Vista environment. After concluding the first tests, secondary tests were performed with the Windows Internet Explorer 8 browser in the Windows Vista environment. In both cases the testing consisted of submitting several observation entries, while varying the data submitted. As the application was done in Finnish, both English and non-English characters were used while testing. The application worked well in the Firefox3.6 environment, but the form page caused some trouble with the IE8 browser. IE8 refused to open the form correctly and it tried to force users to download the file instead. This problem was solved after changing the filename from lomake3.xhtml to lomake3.html.

All pages included in the application were validated with the W3C Markup Validation Service [13]. After a few minor errors were corrected, everything was validated as correct XHTML-MP.

6.2 Testing with mobile phones

The secondary testing was conducted by using Nokia N71, Nokia 6220 Classic and Samsung M7600 BEAT DJ mobile phones. Several entries were made with these three phones, but no exact procedure was followed.

During the testing all three mobile phones showed the application without problems and let the user send the form and showed the resulted information. In a few cases the pages seemed to take a long time to load, but this might be due to increased traffic in the mobile network. The form page is the only one which in some cases did not fit fully to the mobile screen.

The final testing was conducted with the Nokia 6220 Classic mobile phone. It was conducted in a much more organized fashion and consisted of 20 unique test entries. To

simulate normal circumstances some entries were left incomplete and some were made using both English and non-English characters.

During the testing both date and time of the test entry were recorded as well as estimation about usability, problems which might have occurred, whether non-English characters were used, and information whether the form was left incomplete or not. The final testing results were derived after looking through all the entries recorded during the testing.

The testing was completed successfully and the application was according to the guidelines and non-English characters were shown correctly. There are a few potential problems which should be taken into consideration if the application is developed further.

The test results indicate a small usability problem with the form page. Some fields seem to be harder to access directly than others. This can be helped with some rearrangements on the layout. On the whole the form page fit well to the screen. If the date is not entered correctly, it might cause incorrect results. The database accepts only correct dates or full zeroes, so dates such as 2010-02-31 or 2010-04-31 were changed into the default value of 0000-00-00 after the form was submitted.

7 Discussion

7.1 Drawbacks

The application does not provide any real advance compared to the existing Internet sites. It has less features and the database does not provide scientifically useful data to any organization. Users are for the most part accountable for the data provided, as there is no verification process built in. This means the application will not be very useful if users do not leave accurate information.

As long as both DB and web servers work, the application should work reliably, but if one server fails, so does the whole site. If the application should become popular and receive substantial amount of submissions, its current way of printing out all information will become a problem. A continuous list of hundreds or thousands of observations will be hard to read, and the application will become much slower as the amount of data transferred during the use will increase.

7.2 Benefits

Most existing bird information gathering sites require a registration, or in some cases even a membership of some organization, but do not offer any truly mobile way to follow all that gathered information. This application is limited in its details but it provides a way to share information nearly instantaneously with anyone interested.

The economical value of the application by itself is nonexistent, but as both MySQL and PHP are open-source programs, it will not cost anything but a little inconvenience to add this type of feature to the existing server. Even if only a few people are interested, it might still generate more traffic to the original site.

7.3 Summary

During the project I had some problems with the scripting and database design as I had before actively used neither. Because of this, everything was created to be as straightforward as possible. Maybe some aspects of the application became over simplified but the main emphasis was to create, for mobile use, a simple, dynamic site for gathering bird sighting information. The resulting application meets the requirements set on this project, but I believe it could be improved in the future if someone actually wants to use it.

After completing this project it seems that it should be quite easy to provide simple dynamic Internet applications for latest mobile phone models. If one has a basic understanding of the PHP scripting language and database management, it will be easy to make small message boards for mobile use. It seems the biggest difference between Internet and mobile application comes with usability. A small screen size and a ‘keyboard’ hard to use do not prevent most features, but make them cumbersome to use.

8 Conclusions

The goal of the project was to design a dynamic Internet application which can be used by a mobile phone for gathering bird observation information. The application works verifiably with new mobile devices, but the value of the gathered data depends on the users. As no features were implemented to verify any of the data submitted through the form, a moderator with administrative rights to the database should be appointed to remove any inappropriate submissions.

This application did not have many features, so security was not taken into consideration, but in many cases it would be one of the top priorities. If someone wanted to expand this application, security features might need to be implemented. In the future this application might be expanded to include a simple search function and registering to the site, so a registered user would be able to follow a personal observation list and add private notes to the application. If username and passwords will be implemented to provide private services, some security measures should be automatically taken. The connection to the database should be done with lesser rights and all passwords in the database should be in encrypted form. All servers should always have well-working antivirus software in use, as well as solid firewalls to prevent any outside interference. As this application is located in the database server and the web server maintained by the Helsinki Metropolia University of Applied Sciences, I have trusted the basic security features to be implemented on the server side.

If one does not just emphasize the fact that this is a mobile application, it should be considered far inferior to the “official” online bird observation information gathering sites, such as www.tiira.fi or www.hatikka.fi in included features, accuracy and purpose. The Hatikka database is maintained by the Finnish museum of natural history, and the Tiira database is used by the Finnish ornithological societies, but both sites are used to keep track of bird populations and to help conserve species. For these reasons the target group for this mobile application should be twitchers or amateurs who are interested in

observing birds and sharing that information with their peers, but not in a serious fashion.

References

1 Bongariliitto ry. Birdfiles.com –lintupäiväkirja ja rariteettihälyytykset matkapuhelimessa [online]. Suomi, Bongariliitto ry; 23.3.2006. URL:http://www.santamargarita.fi/lintutiedotus/info1.asp. Accessed 2.4.2010.

2 Luonnontieteellinen keskusmuseo. Hatikka- luontohavaintoja ja oma havaintopäiväkirja [online]. Suomi, Luonnontieteellinen keskusmuseo; 23.10.2007. URL:http://www.hatikka.fi/. Accessed 3.4.2010.

3 BirdLife Suomi ry. Tiira – Lintutietopalvelu –Opas [online]. Suomi, BirdLife Suomi ry; 1.4.2009. URL:http://www.birdlife.fi/ohjeet/tiira_ohje_fi.pdf. Accessed 3.4.2010

4 The WAP Forum Ltd. WAP 2.0 Technical White Paper [online]. France, The WAP Forum Ltd.; 18.1.2002. URL:http://www.wapforum.org/what/WAPWhite_Paper1.pdf. Accessed 5.4.2010.

5 Kontio, Tervo, Jääskeläinen, Arokoski, Vierimaa, Raatikainen, Köykkä. Mobiiliteknologiat. Helsinki, Edita Publishing Oy; 2002.

6 Graham I.S. XHTML 1.0 Web Development Sourcebook. New York, John Wiley & Sons, Inc.; 2000.

7 Meloni J.C. MySQL Trainer Kit. Helsinki, Edita Publishing Oy; 2003.

8 Gilmore W.J. Beginning PHP and MySQL: From Novice to Professional, Third Edition. Berkley, Apress; 2008.

9 Lentz A. MySQL Storage Engine [online]. Sweden, MySQL AB; 2.4.2004. URL:http://dev.mysql.com/tech-resources/articles/storage-engine/part_1.html. Accessed 28.3.2010.

10 MySQL AB. PSEA_diagram [online]. Sweden, MySQL AB; 21.4.2006. URL:http://solutions.mysql.com/common/images/PSEA_diagram.jpg. Accessed 30.3.2010

11 Reese G., Yarger R.J., King T., Williams H.E. Managing and Using MySQL, 2nd Ed. Sebastopol, O'Reilly & Associates, Inc.; 2002

12 Widenius M, Axmark D, MySQL AB. MySQL Reference Manual. Sebastopol, O'Reilly & Associates, Inc.; 2002

13 W3C. The W3C Markup Validation Service [online]. USA,W3C; 1.3.2010. URL:http://validator.w3.org/. Accessed 3.4.2010.

Appendices

Appendix 1 Detailed DB table information

Havainto				
Field	**type**	**null**	**default**	**extra**
havainto_id	int(11)	no	none	
paiva	date	no	0000-00-00	
nimi_id	int(11)	no	none	
paikka_id	int(11)	no	none	
laji_id	int(11)	no	none	
lukumaara_id	varchar(7)	no	1	
virhe_id	int(11)	no	1	
gender_id	int(11)	no	none	
ika_id	int(11)	no	none	
puku_id	int(11)	no	none	
muutto_id	int(11)	yes	NULL	
ilmansuunta_id	int(11)	yes	NULL	
tila_id	int(11)	no	none	

havainto_id = An id field which works as a primary key and indicates the order of submitted observations. The results will be shown according to this field in reverse order.

paiva = A field which shows the date the user specified in the form while submitting the observation. The order of the information in this field is yyyy-mm-dd and if left empty while submitting it will give out 0000-00-00 as a result.

nimi_id = An id field which is tied to the name of the observer from the Nimi table.

paikka_id = An id field which is tied to the place of the observation from the Paikka table.

laji_id = An id field which is tied to the species of the observation from the Laji table.

Appendix 1 Detailed DB table information

lukumaara_id = Shows the number(s) which the user specified in the form or shows "1" if the place in the form was left blank.
virhe_id = An id field which is tied to the error margin information from the Virhe table. If no value is submitted this field will show "1" as a default.
gender_id = An id field which is tied to the gender information from the Sukupuoli table.
ika_id = An id field which is tied to the age information from the Ikä table.
puku_id = An id field which is tied to the dress information from the Puku table.
muutto_id = An id field which is tied to the moving information from the Muutto table. If this field is left blank it will give NULL as a default value.
ilmansuunta_id = An id field which is tied to the moving direction information from the Ilmansuunta table. If this field is left blank it will give NULL as a default value.
tila_id = An id field which is tied to the state information from the Tila table.

Nimi				
Field	**type**	**null**	**default**	**extra**
nimi_id	int(11)	no	none	auto_increment
nimi	varchar(50)	no	none	

nimi_id = An id field which works as a primary key and is automatically incemented.
nimi = A name field which behaves as a unique key. In these fields names of the observers are stored.

Paikka				
Field	**type**	**null**	**default**	**extra**
paikka_id	int(11)	no	none	auto_increment
paikka	varchar(50)	no	none	

paikka_id = An id field which works as a primary key and is automatically incemented.
paikka = A location field which behaves as a unique key. In these fields observation locations are stored.

Appendix 1 Detailed DB table information

Laji				
Field	**type**	**null**	**default**	**extra**
laji_id	int(11)	no	none	auto_increment
laji	varchar(50)	no	none	

laji_id = An id field which works as a primary key and is automatically incemented.
laji = A species field which behaves as a unique key. In these fields all submitted species information are stored.

Virhe				
Field	**type**	**null**	**default**	**extra**
virhe_id	int(11)	no	none	auto_increment
virhe	varchar(3)	no	none	

virhe_id = An id field which works as a primary key and is automatically incemented.
virhe = These fields stores all alternative error margin abbreviations.

Sukupuoli				
Field	**type**	**null**	**default**	**extra**
gender_id	int(11)	no	none	auto_increment
sukupuoli	varchar(13)	no	none	

gender_id = An id field which works as a primary key and is automatically incemented.
sukupuoli = These fields stores all used gender options.

Puku				
Field	**type**	**null**	**default**	**extra**
puku_id	int(11)	no	none	auto_increment
puku	varchar(5)	no	none	

puku_id = An id field which works as a primary key and is automatically incemented.
puku = These fields stores all abbreviations for dress options.

Appendix 1 Detailed DB table information

Ikä				
Field	**type**	**null**	**default**	**extra**
ika_id	int(11)	no	none	auto_increment
ikä	varchar(4)	no	none	

ika_id = An id field which works as a primary key and is automatically incemented.
ikä = These fields stores all abbreviations for different age options.

Tila				
Field	**type**	**null**	**default**	**extra**
tila_id	int(11)	no	none	auto_increment
tila	varchar(7)	no	none	

tila_id = An id field which works as a primary key and is automatically incemented.
tila = These fields stores all abbreviations for different state options.

Muutto				
Field	**type**	**null**	**default**	**extra**
muutto_id	int(11)	no	none	auto_increment
muutto	varchar(9)	no	none	

muutto_id = An id field which works as a primary key and is automatically incemented.
muutto = Stores options for regular move and nightmove.

Ilmansuunnat				
Field	**type**	**null**	**default**	**extra**
ilmansuunta_id	int(8)	no	none	auto_increment
muutto_tila	varchar(3)	no	none	

ilmansuunta_id = An id field which works as a primary key and is automatically incemented.
muutto_tila = Stores all possible moving directions.

Appendix 1 Detailed DB table information

xhtml				
Field	**type**	**null**	**default**	**extra**
koodi	text	no	none	
linkki	text	no	none	

This is an extra table which stores parts of xhtml code used in the application. As the same code appears in both php files, it seemed more practical to white it once to the DB and then call it just like all the other information in this application.

koodi = All xhtml header information for this application. From doctype declaration to <body> tag.
linkki = Link back to the index page.

Appendix 2 output.php

```
<?php

   // Connecting to database server

   $hd = mysql_connect("mysql.metropolia.fi", "username", "password")
        or die ("Unable to connect");



   // Selecting database
   mysql_select_db ("kirsimw", $hd)
        or die ("Unable to select database");

   // Assigning form fields into variables
  $paiva = $_POST['Paiva'];
  $nimi = $_POST['nimi'];
  $paikka = $_POST['paikka'];
  $laji = $_POST['laji'];
  $maara = $_POST['lukumaara'];
  $virhe = $_POST['virhe'];
  $sex = $_POST['sukupuoli'];
  $ika = $_POST['ika'];
  $puku = $_POST['puku'];
  $tila = $_POST['tila'];
  $muutto = $_POST['muutto'];
  $suunta = $_POST['ilmansuunta'];



   //Fetching the xhtml-page header information from the database and adding return
link
   $alku = mysql_query("SELECT * FROM xhtml", $hd)
```

Appendix 2 output.php

```
      or die ("Query execution failed");
   $koodi = mysql_fetch_row($alku);
   echo "$koodi[0] <p> $koodi[1]</p>";

   //Making sure that there will be data at required fields 'nimi', 'paikka' and 'laji'.
  if (empty($nimi)) {
     echo "<p>Lomakkeesta puuttui pakollisia tietoja. Ole hyv&#228; ja palaa takaisin
lomakkeeseen, tai kulje ylh&#228;&#228;ll&#228; olevasta linkist&#228; takaisin
etusivulle.</p>" ;

  } elseif (empty($paikka)) {
     echo "<p>Lomakkeesta puuttui pakollisia tietoja. Ole hyv&#228; ja palaa takaisin
lomakkeeseen, tai kulje ylh&#228;&#228;ll&#228; olevasta linkist&#228; takaisin
etusivulle.</p>";

  } elseif (empty($laji)) {
     echo "<p>Lomakkeesta puuttui pakollisia tietoja. Ole hyv&#228; ja palaa takaisin
lomakkeeseen, tai kulje ylh&#228;&#228;ll&#228; olevasta linkist&#228; takaisin
etusivulle.</p>";
}
     else {

  //Looking through for the highest havainto_id number
  $old = mysql_query("SELECT * FROM Havainto ORDER BY havainto_id DESC",
$hd)
      or die ("Query execution failed");
  $oldrow = mysql_fetch_row($old);
  $oldnum = $oldrow[0];
  $hid = $oldnum + 1;
```

Appendix 2 output.php

```
    //Inserting laji, paikka and nimi values to the database
    $ndquery = "INSERT INTO Laji VALUES (', '$laji')";
    mysql_query($ndquery, $hd);

    $rdquery = "INSERT INTO Paikka VALUES (', '$paikka')";
    mysql_query($rdquery, $hd);

    $thquery = "INSERT INTO Nimi VALUES (', '$nimi')";
    mysql_query($thquery, $hd);

    //Looking through tables Nimi, Laji and Paikka to see the correct id
                $nid = "SELECT nimi_id FROM Nimi WHERE nimi = '$nimi'";
                $niid = mysql_query($nid, $hd);
                $temp = mysql_fetch_row($niid);
                $nimiid = $temp[0];

                $lid = "SELECT laji_id FROM Laji WHERE laji = '$laji'";
                $laid = mysql_query($lid, $hd);
                $temp = mysql_fetch_row($laid);
                $lajiid = $temp[0];

                $pid = "SELECT paikka_id FROM Paikka WHERE paikka = '$paikka'";
                $paid = mysql_query($pid, $hd);
                $temp = mysql_fetch_row($paid);
                $paikkaid = $temp[0];

    //Inserting all the values to the Havainto table
    $query = "INSERT INTO Havainto VALUES ('$hid', '$paiva','$nimiid','$paikkaid',
'$lajiid','$maara','$virhe','$sex','$ika','$puku','$muutto','$suunta','$tila')";
    mysql_query($query, $hd);
```

Appendix 2 output.php

```
    //Querying the Havainto table to get last 30 submitted havainto_id numbers
    $res = mysql_query("SELECT * FROM Havainto ORDER BY havainto_id DESC
LIMIT 0, 30"", $hd)
        or die ("Query execution failed");

    //Looping through all data in Havainto table
    while(1)

    {

        // Fetching one row
        $row = mysql_fetch_row($res);

        // If no data row found, exit loop
        if (!$row) break;



            // Assigns value for date
            $val2 = $row[1];

                    // Assings value for name
                    $nim = "SELECT nimi FROM Nimi WHERE nimi_id = $row[2]";
                    $nim1 = mysql_query($nim, $hd);
                    $temp = mysql_fetch_row($nim1);
                    $val3 = $temp[0];

                    // Assings value for place
                    $pai = "SELECT paikka FROM Paikka WHERE paikka_id = $row[3]";
                    $pai1 = mysql_query($pai, $hd);
```

Appendix 2 output.php

```
                    $temp = mysql_fetch_row($pai1);
```

```
$val4 = $temp[0];

// Assings value for species
$laj = "SELECT laji FROM Laji WHERE laji_id = $row[4]";
$laj1 = mysql_query($laj, $hd);
$temp = mysql_fetch_row($laj1);
$val5 = $temp[0];

// Assings value for number
$val6 = $row[5];

// Assings value for error margin
$vir = "select virhe from Virhe where virhe_id = $row[6]";
$vir1 = mysql_query($vir, $hd);
$temp = mysql_fetch_row($vir1);
$val7 = $temp[0];

// Assings value for sex
$suk = "select sukupuoli from Sukupuoli where gender_id = $row[7]";
$suk1 = mysql_query($suk, $hd);
$temp = mysql_fetch_row($suk1);
$val8 = $temp[0];

// Assings value for age
$ik = "select ikä from Ikä where ika_id = $row[8]";
$ik1 = mysql_query($ik, $hd);
$temp = mysql_fetch_row($ik1);
$val9 = $temp[0];
```

Appendix 2 output.php

```
// Assings value for dress
$puk = "select puku from Puku where puku_id = $row[9]";
```

```
                $puk1 = mysql_query($puk, $hd);
                $temp = mysql_fetch_row($puk1);
                $val10 = $temp[0];

                // Assings value for move
                $muu = "select muutto from Muutto where muutto_id = $row[10]";
                $muu1 = mysql_query($muu, $hd);
                $temp = mysql_fetch_row($muu1);
                $val11 = $temp[0];

                // Assings value for direction
                $ilm = "select muutto_tila from Ilmansuunnat where ilmansuunta_id =
$row[11]";
                $ilm1 = mysql_query($ilm, $hd);
                $temp = mysql_fetch_row($ilm1);
                $val12 = $temp[0];

                // Assings value for state
                $til = "select tila from Tila where tila_id = $row[12]";
                $til1 = mysql_query($til, $hd);
                $temp = mysql_fetch_row($til1);
                $val13 = $temp[0];



                // Writes down all the data from current row
                echo "<p> $val2 , Nimi: $val3 <br />
                        Paikka: $val4 <br />
                        Laji: $val5, $val6 $val7 kpl $val8 <br />
```

Appendix 2 output.php

```
                        Puku: $val10 Tila: $val13, $val11 $val12
                        </p>";
    }
```

```
    }

    //Writes down the closing html tags
    echo "</body></html>";

    // Closes the connection
    mysql_close($hd);
?>
```

Appendix 3 havainto.php

```
<?php

   // Connect to database server
   $hd = mysql_connect("mysql.metropolia.fi", "username", "password")
        or die ("Unable to connect");

   // Select database
   mysql_select_db ("kirsimw", $hd)
        or die ("Unable to select database");

   //Querying the Havainto table in order of decending havainto_id number
   $res = mysql_query("SELECT * FROM Havainto ORDER BY havainto_id DESC",
$hd)
       or die ("Query execution failed");

   //Fetching the xhtml-page header information from the database and adding return
link
    $alku = mysql_query("SELECT * FROM xhtml", $hd)
      or die ("Query execution failed");
    $koodi = mysql_fetch_row($alku);
    echo "$koodi[0] <p>$koodi[1]</p>";

   //Looping through all data in Havainto table
   while(1)

   {

      // Fetch one row
      $row = mysql_fetch_row($res);
```

Appendix 3 havainto.php

```
// If no data row found, exit loop
if (!$row) break;

   // Assigns value for date
   $val2 = $row[1];

            // Assings value for name
            $nim = "SELECT nimi FROM Nimi WHERE nimi_id = $row[2]";
            $nim1 = mysql_query($nim, $hd);
            $temp = mysql_fetch_row($nim1);
            $val3 = $temp[0];

            // Assings value for place
            $pai = "SELECT paikka FROM Paikka WHERE paikka_id = $row[3]";
            $pai1 = mysql_query($pai, $hd);
            $temp = mysql_fetch_row($pai1);
            $val4 = $temp[0];

            // Assings value for species
            $laj = "SELECT laji FROM Laji WHERE laji_id = $row[4]";
            $laj1 = mysql_query($laj, $hd);
            $temp = mysql_fetch_row($laj1);
            $val5 = $temp[0];

   // Assings value for number of birds
   $val6 = $row[5];

            // Assings value for error margin
            $vir = "select virhe from Virhe where virhe_id = $row[6]";
            $vir1 = mysql_query($vir, $hd);
```

Appendix 3 havainto.php

```
$temp = mysql_fetch_row($vir1);
$val7 = $temp[0];

// Assings value for gender
$suk = "select sukupuoli from Sukupuoli where gender_id = $row[7]";
$suk1 = mysql_query($suk, $hd);
$temp = mysql_fetch_row($suk1);
$val8 = $temp[0];

// Assings value for age
$ik = "select ikä from Ikä where ika_id = $row[8]";
$ik1 = mysql_query($ik, $hd);
$temp = mysql_fetch_row($ik1);
$val9 = $temp[0];

// Assings value for dress
$puk = "select puku from Puku where puku_id = $row[9]";
$puk1 = mysql_query($puk, $hd);
$temp = mysql_fetch_row($puk1);
$val10 = $temp[0];

// Assings value for moving
$muu = "select muutto from Muutto where muutto_id = $row[10]";
$muu1 = mysql_query($muu, $hd);
$temp = mysql_fetch_row($muu1);
$val11 = $temp[0];

// Assings value for compass direction
$ilm = "select muutto_tila from Ilmansuunnat where ilmansuunta_id =
$row[11]";
```

Appendix 3 havainto.php

```
                $ilm1 = mysql_query($ilm, $hd);
                $temp = mysql_fetch_row($ilm1);
                $val12 = $temp[0];

                // Assings value for state
                $til = "select tila from Tila where tila_id = $row[12]";
                $til1 = mysql_query($til, $hd);
                $temp = mysql_fetch_row($til1);
                $val13 = $temp[0];

                // Writes down all the data from current row
              echo "<p> $val2 , Nimi: $val3 <br />
                        Paikka: $val4 <br />
                        Laji: $val5, $val6 $val7 kpl $val8 <br />
                        Puku: $val10 Tila: $val13, $val11 $val12
                        </p>";

   }

 //Closes xhtml tags
   echo "</body></html>";

//Close the connection to the database
   mysql_close($hd);

?>
```