HAAGA-HELIA
University of Applied Sciences

Veterinary work management application - MatVet

Matti Nurmikari

Bachelor's Thesis

Degree Programme in Business information technology

2012

Authors Matti Nurmikari	**Group**
The title of your thesis Veterinary work management application - MatVet	**Number of pages and appendices** 27 + 1
Supervisors Jyri Partanen	

The veterinary sector in finnish business is among the smallest. Only a handful of veterinarians graduate each year. All these vets need a computer application to manage their work; clients, patients, medication, timetables, receipts, et cetera. Currently they have to choose a client management program between old 1990's style applications that have (or in the worst case, haven't) been patched over and over again for 20 years, and a heavy duty online application, that's mainly built for multimillion turnover veterinary practices. There's no publically available software for this purpose for small, one or two vet clinics. MatVet is here to fix that deficiency.

MatVet is a web application that helps to store and manage information and procedures that vets have to deal with every day, and keep track of the patients. The pages are made by HTML, the database is handled by MySQL, the application logic is controlled by PHP.

Key words
Vet, Veterinary, work management, PHP, web

Vocabulary

Element

Refers to a HTML document element, like <table>.

Open source

Allows free distribution of the application and its source code.

Owner

In MatVet, owner is the person, who owns a patient.

Patient

In MatVet patient is the animal being treated by the user and owned by owner.

Patient card

List that shows patient details, like age, weight, etc.

SQL injection

SQL injection is a technique that exploits web application security flaws. By injecting malicious code, the attacker can perform operations on the database that hasn't been intended.

Security through obscurity

A way of securing an application by hiding or obscuring some details of it. Not considered as a valid security feature on itself.

Abbreviations

AJAX	Asynchronous JavaScript and XML
CSS	Cascading style sheets
DOM	Document object model
HTML	Hypertext markup language
IDE	Integrated development environment
JSON	JavaScript object notation
MVC	Model-View-Controller
PHP	PHP hypertext preprocessor
WAMP	Apache MySQL PHP for Windows
XSS	Cross site scripting

Table of contents

1 Introduction

The goal of this bachelor's thesis was to create a tool for my sponsor for managing the work flow of a veterinarian. There was a need for a light weight application that would keep track of patients, appointments, procedures and medication and I was assigned to make this happen. Secondary goal was to teach myself a bit more about CodeIgniter PHP framework. MatVet is a web based application, the main reason for this approach was that this was the field of programming that I was the most familiar with. This also has other advantages, such as the application can be used anywhere, on almost any system having a web browser, including PDA's and mobile phones.

The sponsor is Eläinlääkäri Satu Nurmikari, a vet. Sponsor's contact person is Satu Nurmikari. It's a private company or *toiminimi* working in the area of veterinary. It's a small scale smaal animal practice without a real office or business premises. The sponsor works mostly in dog shows as a vet, and does private house calls. Most common procedures during dog shows are curing different sorts of accident victims e.g. bite wounds, checking limping dogs for their entry fee refund and making sure the participants are healthy and vaccinated according to regulations. House calls are usually for smaller procedures, such as vaccinations, eye- and ear infections and some orthopedic consultations. Sometimes house calls might also include home euthanasias and helping dogs suffering from dystocia. When done, more complicated operations requiring anesthesia are done at a clinic.

In the second chapter we go through the methods, tools and techniques that were used developing MatVet.

The third chapter lists all the main features of the application and explains their functionality in more detail.

Fourth chapter tells about the planning of the project and includes diagrams, use case scenarios and lists that help the reader to get a better understanding of how the application works.

In fifth chapter we go through the security features of MatVet and information security in general.

In sixth chapter I've listed the important files of the project, and explained what they do.

In seventh chapter the reader will find the summary, results, conclusions and analysis of the project, and the plans for future development of MatVet.

2 Methods and tools

The application is built using mostly open source development tools for budget reasons. Following tools and methods were used to create MatVet.

WampServer is a package of programs made for web development. It allows the user to create web applications with Apache2, PHP and MySQL database. If you would to install all of the featured applications separately, it would be hard and time consuming, with WampServer environment, everything's done with a single click. Also configuring and managing the installed programs is very easy.

Apache2 is the most popular http server since April 1996 developed by Apache software foundation (Apache Software foundation).

PHP is a general purpose scripting language, originally designed for the web. PHP code can be embedded directly in HTML pages (Welling & Thomson 2009, 2). From there (or from external PHP file) the web server parses the code before sending the page to the browser, so the user doesn't actually see any of the code. PHP is used as the application backend, providing connection to the database, handling the database queries and formatting data before it's sent back to JavaScript to be presented.

MatVet uses MySQL (pronounced my es q ell) as its DBMS. MySQL is the most used open source database management system in the world. It is developed, distributed and supported by Oracle Corporation. MySQL is a RDBMS. Relational database stores data in tables, which consist of rows and columns, rather than storing every bit of information in one place. The SQL part in MySQL is an abbreviation for structured query language. It is a common query language used to access databases, it's standardized and defined by the ANSI/ISO SQL standard. (MySQL 5.1 Reference manual.) It was chosen for the project because it is open source, easy to use, requires almost no configuration to start up, and because I am already familiar with it.

Aptana studio 3 is an open source IDE made specifically for web developers. Its most important extra features for this project are project management, code assist and highlighting for PHP, CSS, HTML and JavaScript.

phpMyAdmin is a software tool written in PHP used in administering MySQL databases over the web. The main uses for it is to browse and manage databases, tables, fields, relations, execute SQL commands and import and export data. (phpMyAdmin.)

MySQL Workbench was used to design the diagrams of the database.

HTML is a markup language that is used to create text documents that are used by web browsers to present text and graphics as a web page. A HTML document consists of a set of tags, or elements, like <p> for paragraph, and <h1> for header, that define the structure of the document (Ragget 2005). In MatVet project, I will use HTML5 syntax, although I'm not incorporating any of its new features.

Earlier in the history of web, before CSS, the appearance of a HTML document was defined embedded in html code with tags like <bold> for bold styled text, and for centered elements, and so on. Later with invention and standardization of CSS this method has become obsolete. With CSS included in a web page, it is possible to separate the document structure from its presentational data. The example styles would been denoted in CSS as element {font-weight: bold;}, and another element {text-align: center;}.

The CSS syntax for styling an element consists of a selector (element, another element), and one or more declarations, of which each has a property (font-weight, text-align), and a value (bold, center) (Powell 2010, 430-433). A CSS declaration ends with a semicolon.

CSS styles can be included in a web page with 3 different methods: inline, external and embedded. Inline style is embedded in HTML tags, for example <p color="red">, and it's the least favorable approach, because the styles aren't really separated from the structural markup, which is the whole point of CSS in the first place. Second way of adding CSS to web pages is to put it in an external file. This requires a <link> tag in the head section of the HTML document, and it contains a reference to the CSS file, which in turn, consists only of CSS notation. The last method, embedded, the CSS styles are placed within a <style> tag in head section of the document (Powell 2010, 434).

The front end magic of the application is done by JavaScript. JavaScript is the most used programming language in web pages (Duckett 2010, 481). It gives developers the ability to manipulate, move and read elements or text in documents, it performs mathematical calculations like every other programming language, and it can react to events, like a mouse click, on a web page.

Not really a programming language or syntax, like all the above, AJAX is a name given for a group of existing technologies are made work together to take the user experience of a web page visitor to a whole new level. It allows the sending and receiving of data from server, without (re-)loading the page.

CodeIgniter is an application development framework, a toolkit for developers who use PHP to build sites. It provides libraries and helpers for commonly used tasks, making the coding process faster. CodeIgniter is based on MVC architecture. MVC is an approach where application presentation is separated from its logic (CodeIgniter User Guide Version 2.1.0.) CodeIgniter has quite loose approach on MVC, since Models aren't really required.

The main reason I included CodeIgniter in the project was to make all the database queries more readable and more easier editable, because there's really a lot of them. I also wanted to deepen my knowledge of the framework in real web project.

jQuery is by definition a light weight, cross browser JavaScript library, which will simplify the client side scripting process in HTML pages (jQuery Project).

3 Application overview

3.1 Features

MatVet is a web site, but it's not your everyday page. First of all, there's only a one HTML page, acting as a canvas, against what all the content is displayed at. AJAX is widely used, almost every feature uses AJAX in some way.

At this point MatVet has four distinctive main features, each having one to many sub-functions, plus user management and authentication. The main features are search, time booking, the ability to print receipts and invoices, and content management. All of the features are made according to usability first–approach, meaning that everyone should be able to use the application without, or with as few instructions as possible.

3.1.1 Search (haku)

This view lets the user search database for patients' patient cards or visit details and lets him change the information in them. The user can also add or remove medication to patient. There are 9 different search parameters and their search types listed below:

- owner – text and select
- patient – text and select
- breed – text
- patient min age – text
- patient max age – text
- species – select

Whichever search method the user uses, all the other search parameters will be sent with the query as well.

Using the patient select-search also brings up the patients patient card, which lists all the information about the patient. These can be edited here. Selecting the visit date from "Käynnit" list brings up the details of the visit for that date. Also these can be edited and saved.

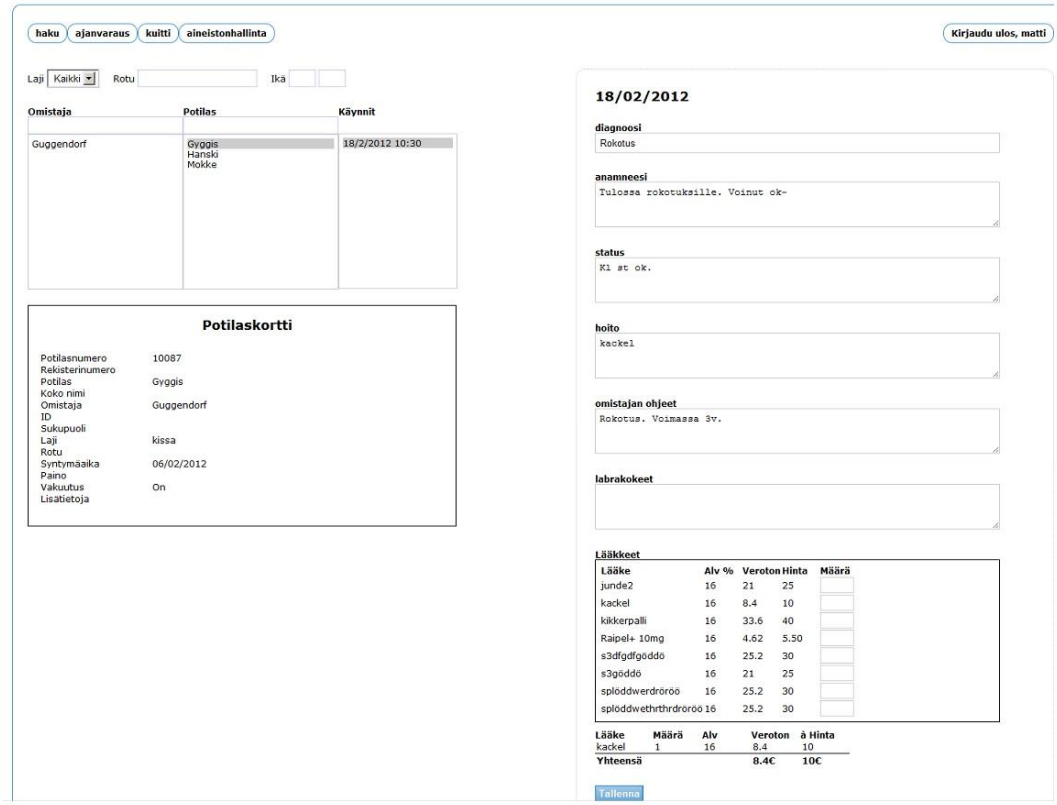

Figure 1. Screen capture of the search feature with patient card on lower left, and visit details on right.

3.1.2 Time booking (ajanvaraus)

This is the heart and soul of MatVet. Here the user can book a time for an appointment, add a patient, and remove an appointment.

Booking a time for an appointment is done by selecting a date from the calendar, (shows current month and 4 next month in future), then selecting a free time from the list that shows the appointments for that day, selecting an operation, or use custom operation name and end time input fields. Then the user can change the operation durations if needed, from "muuta"-link. The user cannot book an appointment to a time where there is already a booked time. From here, the user can also add procedures, patients and owners to the system. These are available from "lisää uusi" buttons. From add patient and owner view, the user can enter details of the patient. Required fields are owner name, address, and phone number for owner, and name and species for the patient. This information is the base for a patient card, listed in search view, the user can update the information from there after saving the patient/owner

info here. The add appointment view also has a search text input field, to limit the listed patients for those that match the search string.

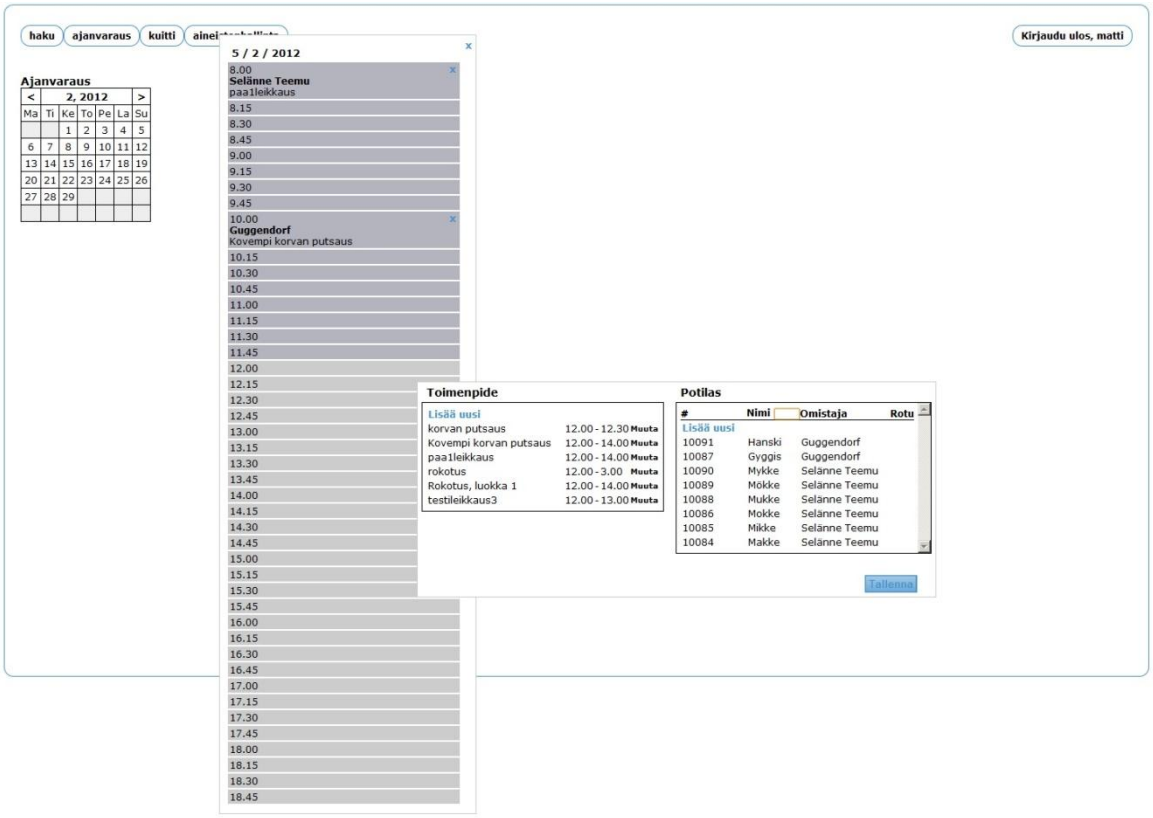

Figure 2. Screen capture of the time booking feature with 2 booked times for selected day, and add new appointment view.

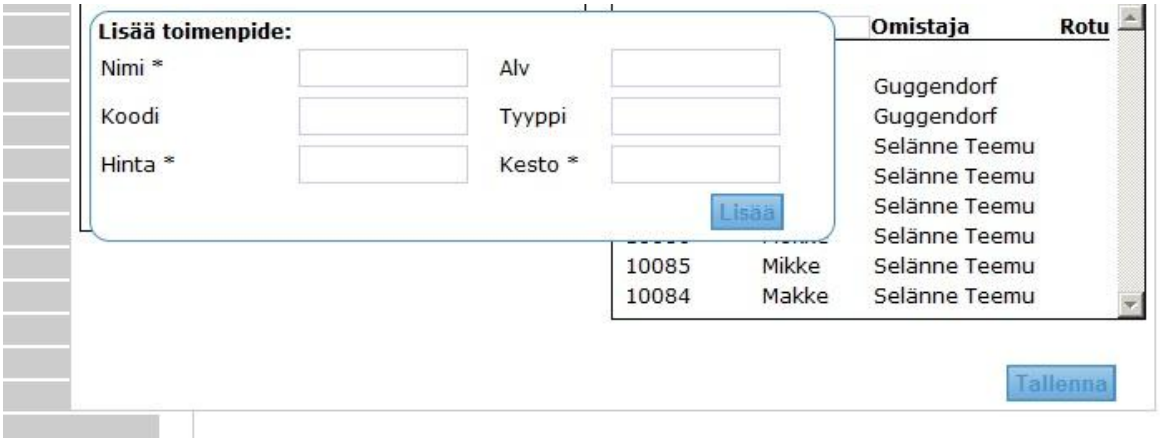

Figure 3. Screen capture of the add procedure -view of the time booking feature. Required fields are marked with an asterisk, and the procedure cannot be added if those fields aren't filled.

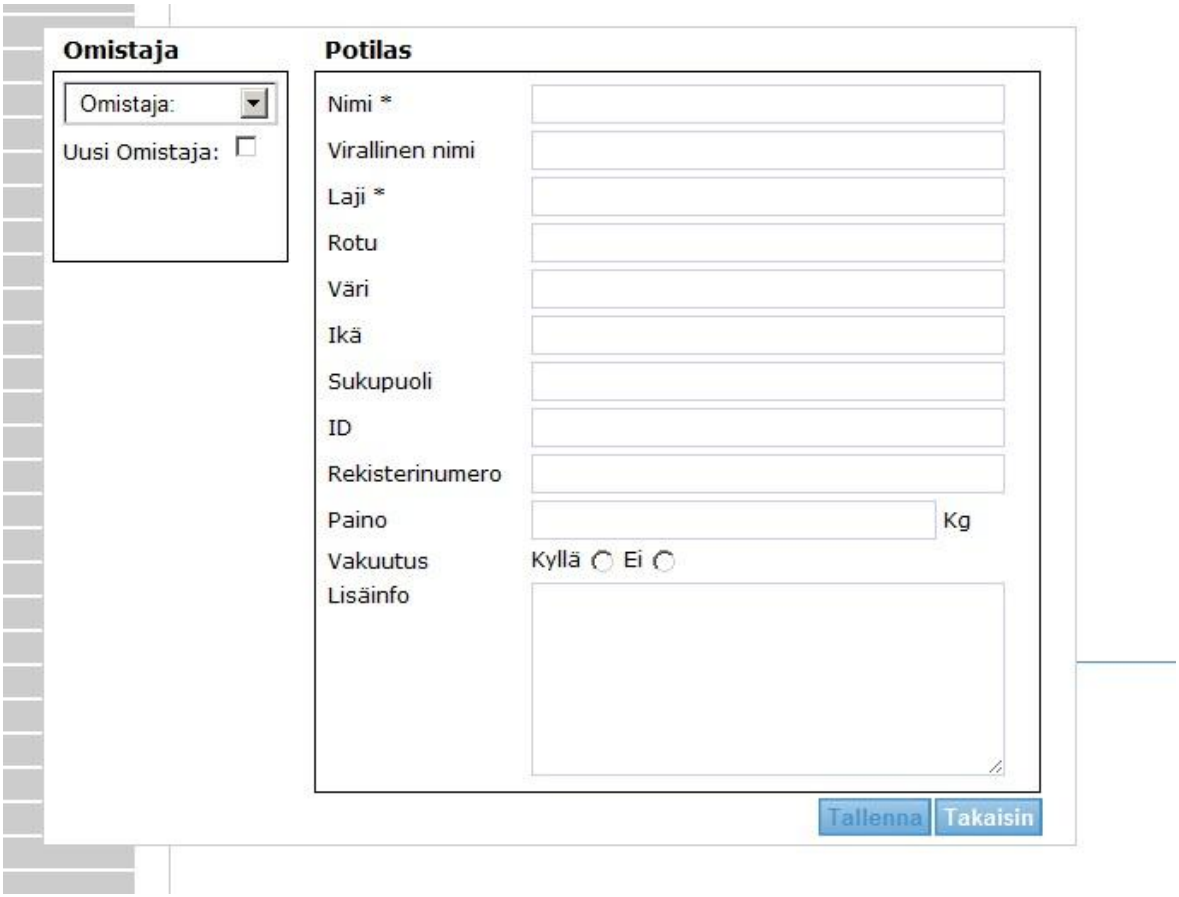

Figure 4. Screen capture of add patient and owner. The owner info will be pre-entered if an existing owner is selected from dropdown list.

3.1.3 Print receipt / invoice (kuitti)

Here the user can browse all the visits saved in database. Listed are patient number, name, owner, and date. When a visit is selected by mouse click, the user sees a receipt on the right, which can be printed to the customer.

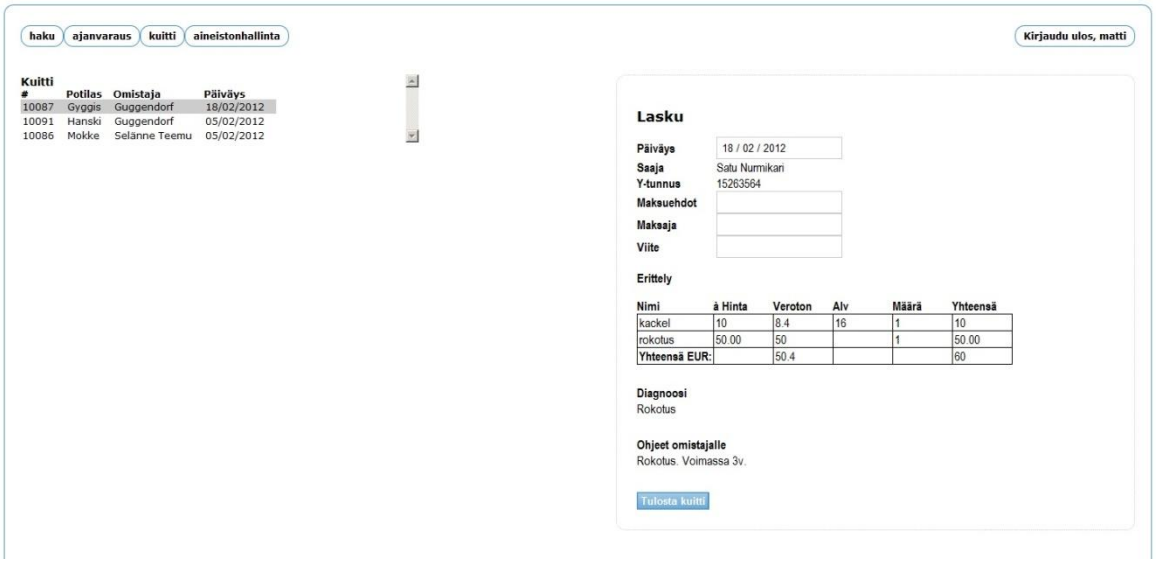

Figure 5. Screen capture of receipt / invoice.

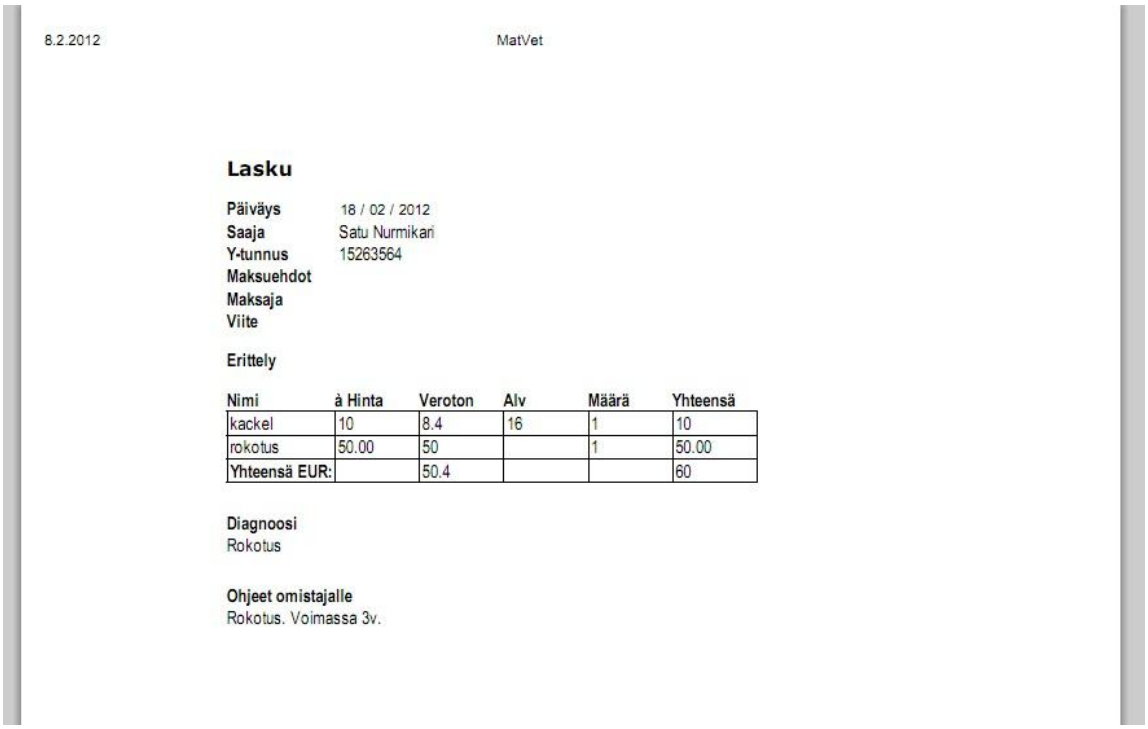

Figure 6. A ready to be printed invoice looks like this. The grey borders are from browsers print preview.

3.1.4 Content management (aineistonhallinta)

The user can manage, that is add and remove medication and procedures that are shown in menus and lists everywhere in the application. When deleting content, a prompt is shown to confirm the deletion, to prevent accidental removal of content.

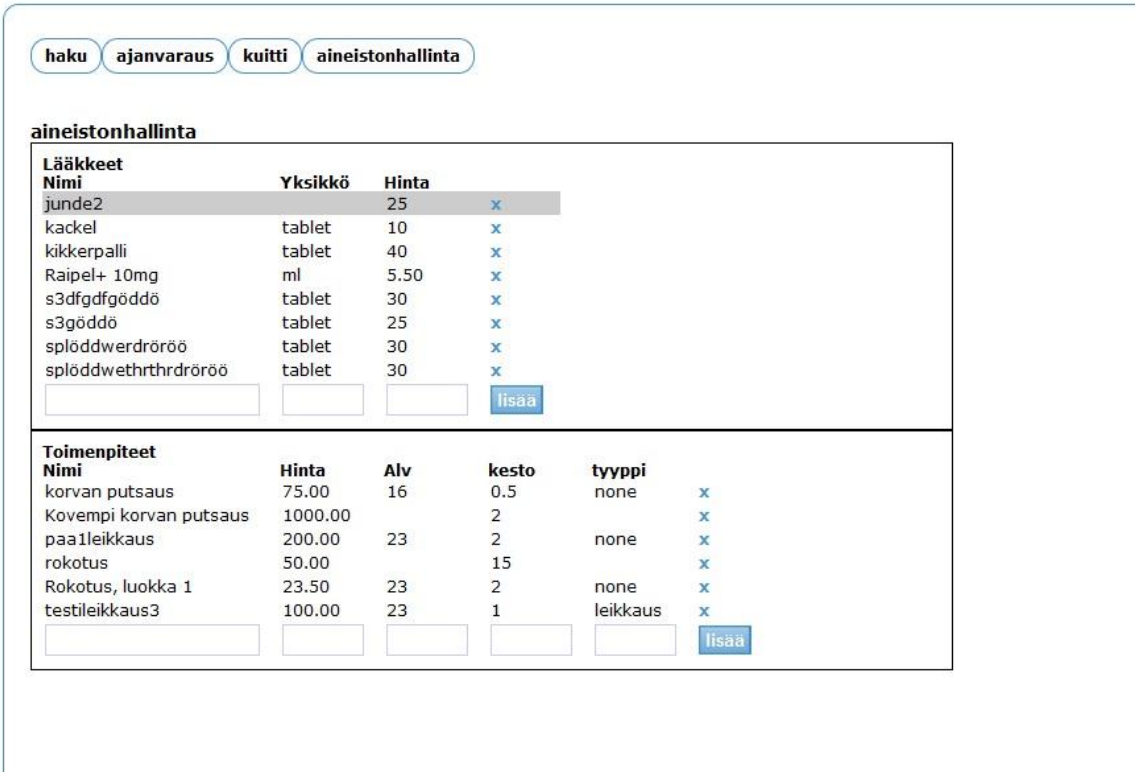

Figure 7. Screen capture of the content management feature.

4 Definition and planning

The application requirements were quite loosely defined. The sponsor wanted a light program, which would be easy to use, and that would require minimum help from outside to set up and to use. All the communication between the developer and the sponsor was done orally, and there is no documentation of any kind of these conversations aside from this document.

The original required features were patient card view with search, time booking and the ability to print receipts. Later we agreed to add content management and some extra search options for the search feature. User login system was included for information security reasons, even though it wasn't explicitly required by the sponsor.

Because MatVet is so light code wise, the only real planning I did was the database structure, the application and its logic was planned on the fly. This was the last time I will code anything without planning it thoroughly in advance. First of all, I had to remake the whole application not once, but twice, because I didn't have any plans to follow, and some features just couldn't be created with the approach I chose. Secondly, it's unbelievably hard to try to make different main features to work together, if it hasn't been planned ahead.

4.1 Use case diagram

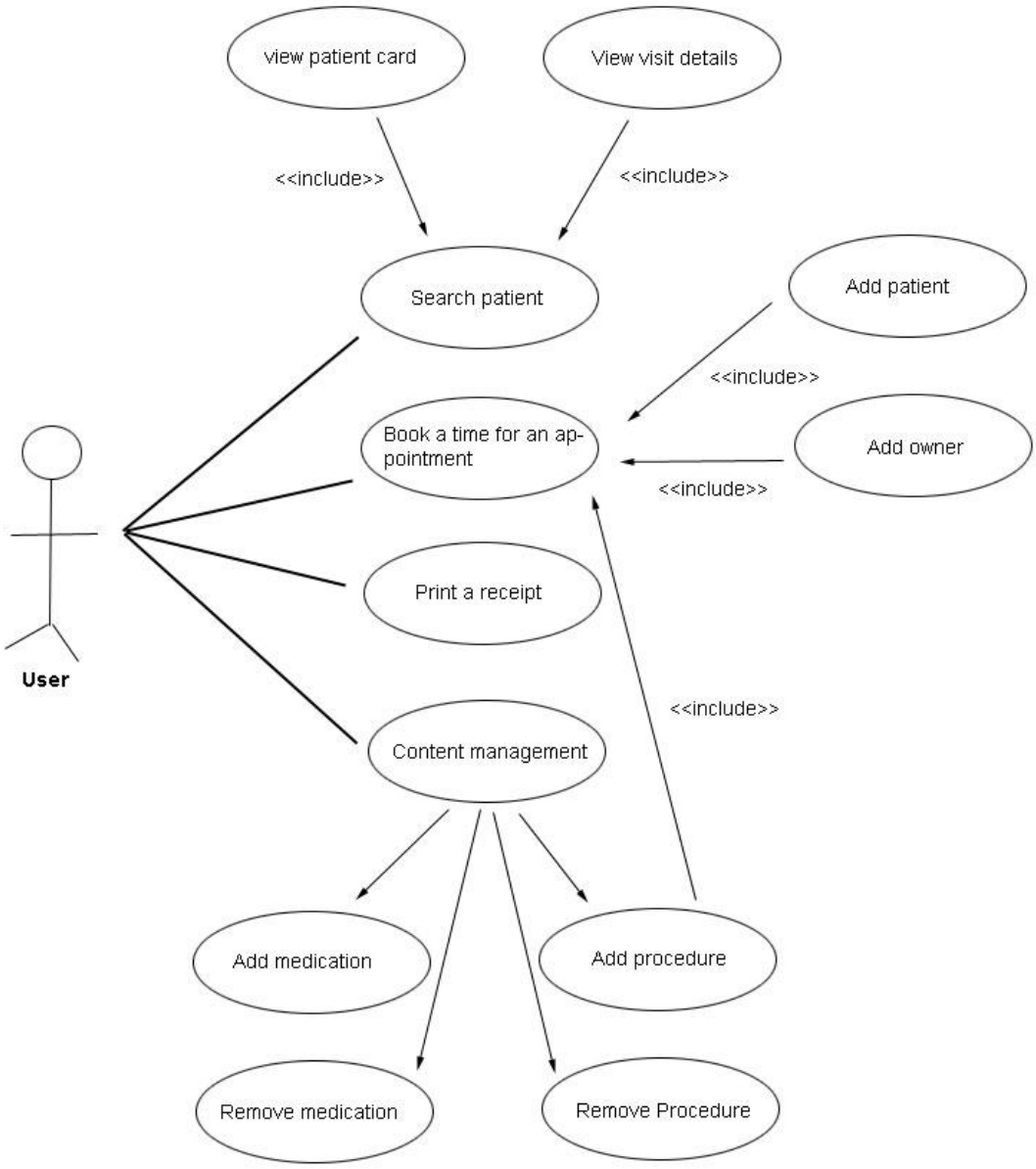

Figure 8. A use case diagram.

4.2 Use cases

4.2.1 Use case: search for a patient

Actor:	User
preconditions:	The user has to be logged in.
Goal:	The user is shown the list of patients matching the search criteria.
Exceptions	If there's no matches, no results text will be shown.
Includes	This use case is a prerequisite for View patient card and view visit details use cases.

Actor actions:

- The actor uses the input fields to do a search. One to eight of the search criteria must be used.

4.2.2 Use case: View patient card

Actor:	User
preconditions:	The user has to be logged in, and a search for patients has to be done.
Goal:	The user is shown the right patient card.

Actor actions:

- The actor selects a patient by name from the "potilas"-list.

4.2.3 Use case: View visit details

Actor:	User
preconditions:	The user has to be logged in, and a search for patients has to be done.
Goal:	The user is shown the details of a visit.

Actor actions:

- The actor selects a visit entry by date from "käynnit"-list.

4.2.4 Use case: Book a time for an appointment

Actor:	User
preconditions:	The user has to be logged in, and on time booking view. Selected time cannot be reserved already.
Goal:	An appointment is booked for a wanted time.
Exceptions	If the user tries to select a time that intersects earlier booked time, the save button stays disabled and time cannot be booked.

Actor actions:

- The actor selects a day from calendar.
- The actor selects an hour from the list that is populated.
- The actor selects a procedure from the predefined list or enters a new procedure from "lisää uusi"-button.
- The actor selects a time
- The actor selects a patient from the list or adds a patient to the system from "lisää uusi"-button.
- The actor clicks save, and the appointment is booked.

4.2.5 Use case: Print a receipt for a customer

Actor:	User
preconditions:	The user has to be logged in, and on receipt view.
Goal:	A receipt is printed for a customer.

Actor actions:

- The actor selects the wanted visit from the list.
- The actor enters missing details to the receipt
- The actor prints the receipt

4.2.6 Use case: Remove medication or procedure from the application

Actor:	User
preconditions:	The user has to be logged in, and on content management view.
Goal:	Medication is removed from the lists in MatVet. The medication will not be removed from databases.

Actor actions:

- The actor selects the wanted medication from the list and clicks x
- The medication will be marked as inactive, and it will not show in the application any more.

4.2.7 Use case: Add medication or a procedure to application

Actor:	User
preconditions:	The user has to be logged in, and on content management view.
Goal:	Medication or procedure is added to the system.
Exceptions	If the user tries to add an item that is already in the system, an error message will be shown.

Actor actions:

- The actor enters the medication or procedure information to the text boxes.
- The actor clicks add-button
- If all the information is correct, the medication or procedure is added to database and can be used in the application.

4.3 Database planning

The MatVet database was originally planned and designed using pen and paper, and the final form was constructed with MySQL Workbench and phpMyAdmin. The application uses one database called matvet, which has 8 tables. I've listed the tables, their descriptions and an EER-diagram of the database below:

Table name	**Table description**
userx	Database users information
owner	Owner information
patient	Patient information
medication	List of all the medication that is in use
givenmedication	List of medication given to patients
procedures	List of all prestored procedures
performedprocedures	List of all procedures performed on patients
visits	List of all visits and their details

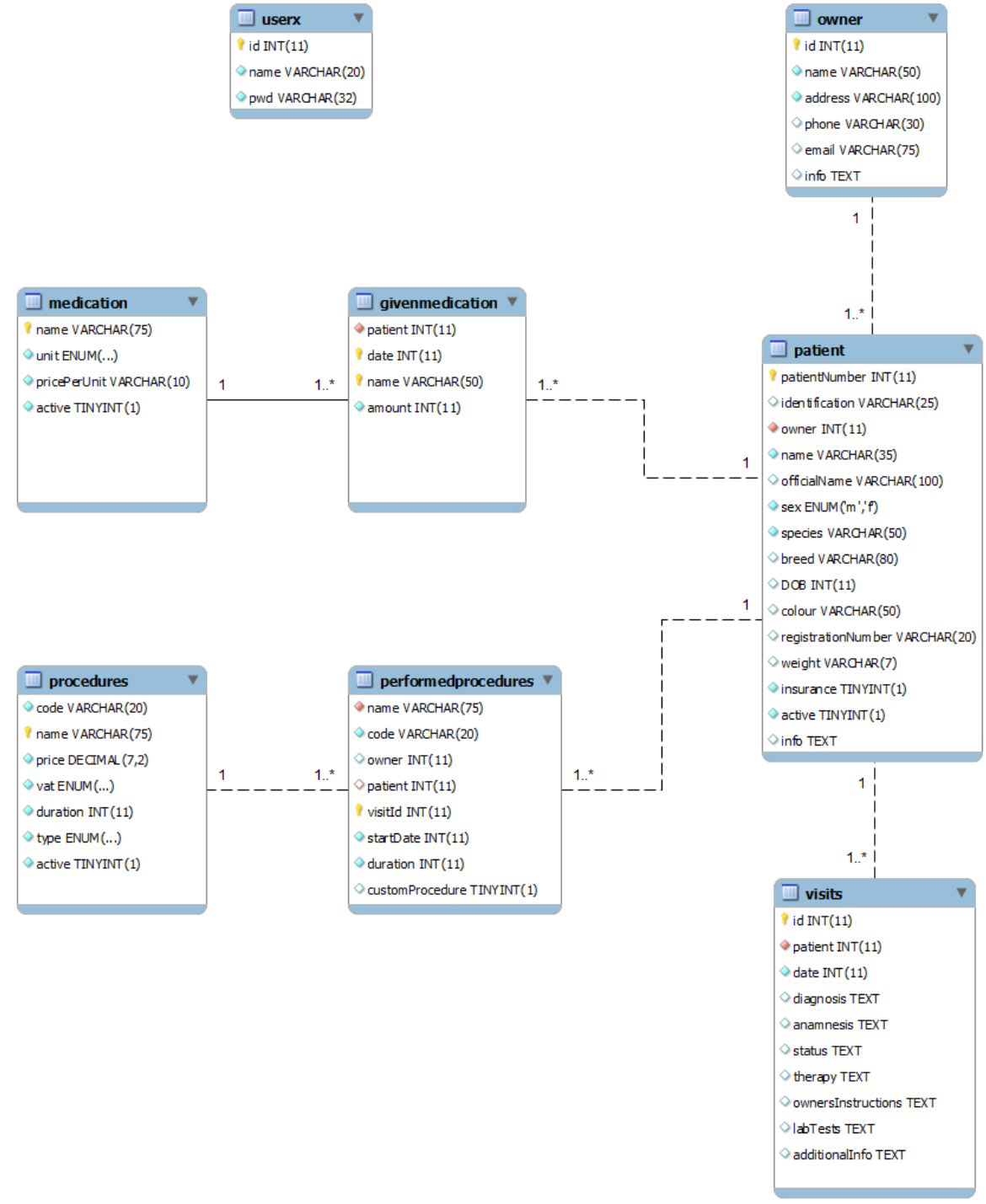

Figure 9. EER diagram of the MatVet database.

4.4 Database tables

Owner

This table holds information about owners. Indexed fields are 'id' for Primary Key, and name and address for unique constraint, so same owner can't be added more than once.

Field	Type	Extra	Description
id	int(11)	PK, AI	Identifying field for owner row. For MatVet internal use
name	varchar(50)		Owner name
address	varchar(100)		Owner home address
phone	varchar(30)		Owner phone number
email	varchar(75)		Owner E-mail address
info	text		Additional info about the owner

List 1. Structure of owner table

Patient

This could be considered as the main table in the system, since most of the tables use this one as a reference. The table holds information about patients. Primary key is patientNumber and foreign key is owner, which references to owner table id field.

Field	Type	Extra	Description
patientNumber	int(11)	PK, AI	Identifying field
identification	varchar(25)		kysy
owner	int(11)	FK (owner.id)	Owner id
name	varchar(35)		Patient name
officialName	varchar(100)		Patient official name
sex	enum('m','f')		Patient gender
species	varchar(50)		Patient species
breed	varchar(80)		Breed of the patient
DOB	int(11)		Patient date of birth in Unix time stamp
colour	varchar(50)		Patient colour
registrationNumber	varchar(20)		Patient registration number
weight	varchar(7)		Patient Weight in kilos
insurance	tinyint(1)		1 if patient has insurance 0 if patient doesn't have insurance
active	tinyint(1)		1 deceased 0 alive
info	text		Additional info about the patient

List 2. Structure of patient table

Visits

This table stores the visit details. The Primary key is field 'id', foreign key is patient field, and it references to patient table, patientNumber field. Date field has a unique constraint.

Field	Type	Extra	Description
id	int(11)	PK, AI	Identifying field
patient	int(11)	FK (patient.patientNumber)	Patient number
date	int(11)		Unix timestamp for start date of a visit
diagnosis	text		What is wrong with the patient.
anamnesis	text		The reason the patient is brought to the vet.
status	text		The current clinical status of the patient.
therapy	text		Given medication, procedures and plans for the treatment.
ownersInstructions	text		Instructions given to owner in appointment
labTests	text		Done laboratory tests
additionalnfo	text		Additional info about the visit for the user

List 3. Structure of visits table.

Userx

Table contains the user authentication information.

Field	Type	Extra	Description
id	int(11)	PK, AI	Identifying field, user number
name	varchar(20)		Username
pwd	varchar(32)		User password

List 4. Structure of userx table.

Medication

Medication is the main database for all the medication the user has prescribed. The only index is the name as Primary Key.

Field	Type	Extra	
name	varchar(75)	PK	Medication name
unit	enum('tablet','package','ml')		The unit that is used in measuring this medication
pricePerUnit	varchar(10)		Price for medication
active	tinyint(1)		1 Active, in use 0 Not active, not used in application

List 5. Structure of medication table.

Givenmedication

Givenmedication table is a junction table. It is a list of all medication that has been prescribed to patients. Indexes are date + name as a Primary Key (composite key), date as Foreign key referencing a visits date, name also as Foreign Key referencing medication name.

Field	Type	Extra	Description
patient	int(11)	FK (patient.patientNumber)	Patient number, target of the medication given
date	int(11)	PK, FK (visits.date) Date & name form a composite Primary key	Unix timestamp when this medication was given or prescribed
name	varchar(50)	PK, FK (medication.name)	Medication name
amount	int(11)		Amount of the medicine prescribed.

List 6. Structure of givenmedication table.

Procedures

List of all procedures the user will do. The only index is name as a Primary Key.

Field	Type	Extra	Description
code	varchar(20)		Code for the procedure. Shown in lists. Not in use yet.
name	varchar(75)	PK	Procedure name
price	decimal(7,2)		Price for the procedure in euros
vat	enum('23','16','8','0')		Value added tax for the operation
duration	int(11)		Duration of the operation
type	enum('leikkaus','none')		Type of the operation. Is used to differentiate operations in time booking view.
active	tinyint(1)		1 = in use 0 = not in use

List 7. Structure of procedures table

Performedprocedures

This table is like givenmedication, but for procedures. All done procedures are listed here. Indexes are name as Primary Key, name, owner, patient and visitId as foreign keys.

Field	Type	Extra	
name	varchar(75)	PK, FK (procedures.name)	Procedure name
code	varchar(20)		Procedure code (not in use yet)
owner	int(11)	FK (owner.id)	Owner id
patient	int(11)	FK (patient.patientNumber)	Patient number
visitId	int(11)	FK (visits.id)	Visit id
startDate	int(11)		Procedure start date as a unix timestamp
duration	int(11)		Procedure duration

List 8. Structure of performedprocedures table.

5 MatVet Information Security

5.1 Introduction

Information security is a vital part of every software application. The value of security is multiplied when it's about a web service, since the user base isn't limited to a certain computers' users, as is the case with regular desktop software. Information security isn't just a feature among others, it has to be part of the development process from beginning to start (Thomson & Wellington 2009, 362).

5.2 Security through obscurity

Although considered as an invalid technique on its own, security through obscurity has its uses when applied with other methods. The application itself is hidden from the most popular search engines by having a file robots.txt in the root folder with content "User-agent: * Disallow: /". The user-agent part states that it concerns all robots, and disallow / means that no page on the site should be indexed (The Web Robots Pages). Robots are not forced to obey the rule, but some of the most used search engines (Google, Yahoo) still will.

5.3 CodeIgniter's internal security features

CodeIgniter comes with a few automatic features that help the developer to boost the applications security (CodeIgniter User Guide Version 2.1.0).

- URI security. CodeIgniter filters the URI's passed to the application and accepts only certain characters. By default, these characters are: any alphanumeric, tilde, period, colon, underscore and dash. (~.:_-).
- Register_globals. During system initialization all global variables are unset, except those found in $_GET, $_POST, $_COOKIE arrays.
- Error reporting. Stops PHP's errors from rendering as output to prevent possible sensitive information from showing.
- Magic_quotes_runtime. Sets magic_quotes_runtime to false.
- Database queries are automatically escaped.

There are also security features that are not automatic, but I have set them on for the project in CodeIgniter configuration. The first one is XSS filtering, which looks for the most used techniques to execute cross site scripting, and if malicious code is found, it's rendered safe by replacing the illegal characters with character entities. The second one is cookie encryption, which encrypts the session cookies, so they cannot be tampered with. The last one is input sanitation through $_GET and $_POST arrays via CodeIgniter's input method. This checks for the input variables, and filters all non-alphanumeric characters. (CodeIgniter User Guide Version 2.1.0).

5.4 Authentication

The first line of defense is authentication. The user needs to be authorized to be able to use it. If correct user name and password are not entered, the user is shortly noted for failure in logging in. Currently MatVet only supports one concurrent user, meaning that one user at the time can use the application. Passwords are stored in the database as md5 hashes, so they are not visible to anyone as plain text. User sessions are managed by CodeIgniter's Session class. Sessions are made to last for 2 hours before they time out, and the session is terminated. After this, the user has to log in again to be able to use the application.

5.5 Database backup

Not really a security issue, but important one nonetheless. So far there isn't any automatic backing up of the MatVet database. The user has to manually take care for it.

6 File structure

6.1 Directory structure

The files in CodeIgniter projects are organized in folders. The directory structure of MatVet is following:

/matvet3	the main directory, contains all the project files and directories		
	/application	Application directory, contains all application files	
		/config	configuration files
		/controller	Controller files
		/models	Model files
		/views	View files
	/css	CSS style files	
	/js	JavaScript files	
	/system	CodeIgniter system directory, contains core files for CI to work	
	/user_guide	User guide files	

List 9. MatVet directory structure.

6.2 Files of importance

The application files in a CodeIgniter project are of three different types, models, views, or controllers. Views contain the actual HTML code and possibly some PHP to help iterate through the data provided by a model, Models handle the access to data sources and controllers act as an intermediary between the models, the views and the user. Model and controller files contain one model or controller class. MatVet project has two models, five controllers and a number of views.

Models	
User	The user authentication is handled by User model
Dbqueries	All the rest of the database activity is provided by this model

List 10. MatVet model files.

Controllers	
Main	All information that's needed from the database goes through this controller.
Login	Handles the login process.
Navigation	Every view is loaded via AJAX, this handles the top level navigation.
Ajax_navigation	All the minor views are loaded by this.
Error	Handles error messaging.

List 11. MatVet Controller files.

Views	
content	The basic HTML content structure of MatVet.
navi	Top level navigation.
template	The HTML document template including header and footer.
addpatient	Add owner & patient view.
addprocedure	Add an appointment view.
contentmanagement	Content management main view (aineistonhallinta).
login	The login view.
patientCard	Patient card view found in search.
receipt	The receipt (kuitti) main view.
receiptDetails	The receipt details view. This is what the printed receipt contains.
schedule	A schedule view for a certain day.
search	Search (haku) main view.
timebooking	Time booking (ajanvaraus) main view.
visits	Visit details view in search (haku).
error	The error and notice message view

List 12. MatVet View files.

7 Summary

7.1 Conclusions

The end product hasn't yet been delivered to the sponsor, so it's hard to say what the exact effects of using an improved work management tool are. The first reaction of the client to the ready application was positive. My projection is that if the current work management tool is pen and paper, the most important change is in time saved in micromanaging time booking and patients' details and if the tool now used is a corporate level application, the money saved in software licenses is the most important factor.

The application is not a hundred percent ready product, but due to the time limitation, I have to release it with some features still missing from what was planned. There might be, and there probably is, some minor bugs as well, for the same reason. MatVet has been accepted by my sponsor, and it will be maintained and updated to a proper tool for any veterinarian, once the developer team has more resources available.

On personal level the whole process has been very rewarding. It has been a pleasure to note how much more there was, and still is, to learn about the methods of my choice. As previously stated in chapter four, I have had to remake the application from the beginning two times, and if I had to start over once again, it would again look totally different from this version, just because for the things I've learned on my way to this point.

7.2 Analysis

The chosen methods and tools were noted to be sufficient in projects like this. In the future I predict a rise in web based applications in general, because of the portable nature of the technology.

7.3 Further development

After the first iteration, due to a tight schedule the application has only the minimum features required to make it usable. In future, more features will be applied to MatVet.

7.3.1 Next iteration, during spring 2012

Moving fixed variables like calendar dates range (4 months to future now), schedule hours range (a static 8-19 at the moment) and possibly error messages to an external file, which can then be included in the application, and can be easily edited. I will possibly replace my own scheduling feature and start using Google calendar for better usability and more flexible features. Also adding an edit feature to MatVet content management, giving an ability to add procedures or medication by hand in print receipt view, and making it possible to remove patients from the system to extend the user control over the content even more better.

7.3.2 Third iteration, late 2012

Adding multiuser support and user management view, with the possibility to create, remove and edit user accounts and their account details. Will add real time updates in the system: Views refresh automatically every couple of minutes and the user can see changes to content made by other users. Automatic database backup will be added. Also planning on extending the browser support.

Bibliography

CodeIgniter User Guide. Welcome to CodeIgniter – User guide version 2.1.0 URL: http://codeigniter.com/user_guide/ . Accessed 12.1.2012.

Duckett, J., Beginning HTML, XHTML, CSS, and Javascript. Wiley Publishing, Inc.

Holdener, A., T., III 2008. Ajax, The Definitive Guide. O'Reilly Media, Inc.

jQuery Project, the. jQuery: The write less – do more, Javascript library. URL: jquery.com . Accessed 12.1.2012.

MySQL Reference Manual. What is MySQL. URL: http://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html . Accessed 12.1.2012.

Raggett, D. 2005. Getting started with HTML. URL: http://www.w3.org/MarkUp/Guide/ . Accessed 10.1.2012.

phpMyAdmin. About. URL: http://www.phpmyadmin.net/home_page/index.php . Accessed 18.1.2012.

Powell, T. A. 2010. HTML & CSS: The Complete Reference. 5th Edition. The McGraw-Hill Companies.

The Web Robots Pages. About /robots.txt. URL: http://www.robotstxt.org/robotstxt.html . Accessed 25.1.2012.

Welling, L., Thomson, L. 2009. PHP and MySQL Web Development. 4th Edition. Pearson Educational, Inc.